OpenStack

Heat documentation

(August 5, 2016)

docs.openstack.org

OpenStack Heat documentation

Copyright © 2010-2016 OpenStack Foundation All rights reserved.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

Welcome to the Heat documentation!

Heat is a service to orchestrate composite cloud applications using a declarative template format through an OpenStack-native REST API.

Heat’s purpose and vision

- Heat provides a template based orchestration for describing a cloud application by executing appropriate *OpenStack* API calls to generate running cloud applications.
- A Heat template describes the infrastructure for a cloud application in text files which are readable and writable by humans, and can be managed by version control tools.
- Templates specify the relationships between resources (e.g. this volume is connected to this server). This enables Heat to call out to the OpenStack APIs to create all of your infrastructure in the correct order to completely launch your application.
- The software integrates other components of OpenStack. The templates allow creation of most OpenStack resource types (such as instances, floating ips, volumes, security groups, users, etc), as well as some more advanced functionality such as instance high availability, instance autoscaling, and nested stacks.
- Heat primarily manages infrastructure, but the templates integrate well with software configuration management tools such as Puppet and Ansible.
- Operators can customise the capabilities of Heat by installing plugins.

This documentation offers information aimed at end-users, operators and developers of Heat.

Using Heat

- Creating your first stack
- Template Guide
- Example HOT Templates
- Example CFN Templates
- Glossary

Operating Heat

- Installing OpenStack and Heat on RHEL/Fedora/CentOS
- Installing Heat on Ubuntu
- Scaling a Deployment
- Man pages for services and utilities

Developing Heat

- Heat Contribution Guidelines
- Heat and DevStack
- Heat architecture
- Heat Resource Plug-in Development Guide
- Heat Stack Lifecycle Scheduler Hints
- Guru Meditation Reports
- Heat Support Status usage Guide
- Using Rally on Heat gates

API Documentation

- Heat REST API Reference (OpenStack API Complete Reference - Orchestration)

Code Documentation

- The heat.api.aws.ec2token Module
- The heat.api.aws.exception Module
- The heat.api.aws.utils Module
- The heat.api.cfn.v1.signal Module
- The heat.api.cfn.v1.stacks Module
- The heat.api.cfn.versions Module
- The heat.api.cloudwatch.watch Module
- The heat.api.middleware.fault Module
- The heat.api.middleware.ssl Module
- The heat.api.middleware.version_negotiation Module
- The heat.api.openstack.v1.actions Module
- The heat.api.openstack.v1.build_info Module
- The heat.api.openstack.v1.events Module
- The heat.api.openstack.v1.resources Module
- The heat.api.openstack.v1.services Module
- The heat.api.openstack.v1.software_configs Module
- The heat.api.openstack.v1.software_deployments Module
- The heat.api.openstack.v1.stacks Module
- The heat.api.openstack.v1.util Module
- The heat.api.openstack.v1.views.stacks_view Module
- The heat.api.openstack.v1.views.views_common Module
- The heat.api.openstack.versions Module
- The heat.api.versions Module
- The heat.engine.api Module
- The heat.engine.attributes Module
- The heat.engine.cfn.functions Module
- The heat.engine.cfn.template Module
- The heat.engine.check_resource Module
- The heat.engine.clients.client_plugin Module
- The heat.engine.clients.os.aodh Module
- The heat.engine.clients.os.barbican Module
- The heat.engine.clients.os.ceilometer Module
- The heat.engine.clients.os.cinder Module
- The heat.engine.clients.os.designate Module
- The heat.engine.clients.os.glance Module
- The heat.engine.clients.os.heat_plugin Module

- The heat.engine.clients.os.keystone.heat_keystoneclient Module
- The heat.engine.clients.os.keystone.keystone_constraints Module
- The heat.engine.clients.os.magnum Module
- The heat.engine.clients.os.manila Module
- The heat.engine.clients.os.mistral Module
- The heat.engine.clients.os.monasca Module
- The heat.engine.clients.os.neutron.lbaas_constraints Module
- The heat.engine.clients.os.neutron.neutron_constraints Module
- The heat.engine.clients.os.nova Module
- The heat.engine.clients.os.sahara Module
- The heat.engine.clients.os.senlin Module
- The heat.engine.clients.os.swift Module
- The heat.engine.clients.os.trove Module
- The heat.engine.clients.os.zaqar Module
- The heat.engine.clients.progress Module
- The heat.engine.constraint.common_constraints Module
- The heat.engine.constraint.heat_constraints Module
- The heat.engine.constraints Module
- The heat.engine.dependencies Module
- The heat.engine.environment Module
- The heat.engine.event Module
- The heat.engine.function Module
- The heat.engine.hot.functions Module
- The heat.engine.hot.parameters Module
- The heat.engine.hot.template Module
- The heat.engine.lifecycle_plugin Module
- The heat.engine.notification.autoscaling Module
- The heat.engine.notification.stack Module
- The heat.engine.parameter_groups Module
- The heat.engine.parameters Module
- The heat.engine.plugin_manager Module
- The heat.engine.properties Module
- The heat.engine.resource Module
- The heat.engine.rsrc_defn Module
- The heat.engine.scheduler Module
- The heat.engine.service Module
- The heat.engine.service_software_config Module
- The heat.engine.service_stack_watch Module
- The heat.engine.stack Module

- The heat.engine.stack_lock Module
- The heat.engine.support Module
- The heat.engine.sync_point Module
- The heat.engine.template Module
- The heat.engine.template_files Module
- The heat.engine.timestamp Module
- The heat.engine.translation Module
- The heat.engine.update Module
- The heat.engine.watchrule Module
- The heat.engine.worker Module
- The heat.hacking.checks Module
- The heat.httpd.heat_api Module
- The heat.httpd.heat_api_cfn Module
- The heat.httpd.heat_api_cloudwatch Module
- The heat.objects.base Module
- The heat.objects.event Module
- The heat.objects.fields Module
- The heat.objects.raw_template Module
- The heat.objects.raw_template_files Module
- The heat.objects.resource Module
- The heat.objects.resource_data Module
- The heat.objects.service Module
- The heat.objects.snapshot Module
- The heat.objects.software_config Module
- The heat.objects.software_deployment Module
- The heat.objects.stack Module
- The heat.objects.stack_lock Module
- The heat.objects.stack_tag Module
- The heat.objects.sync_point Module
- The heat.objects.user_creds Module
- The heat.objects.watch_data Module
- The heat.objects.watch_rule Module
- The heat.rpc.api Module
- The heat.rpc.client Module
- The heat.rpc.listener_client Module
- The heat.rpc.worker_api Module
- The heat.rpc.worker_client Module
- The heat.scaling.cooldown Module
- The heat.scaling.lbutils Module

- The heat.scaling.rolling_update Module
- The heat.scaling.scalingutil Module
- The heat.scaling.template Module
- The heat.version Module

Indices and tables

- *Index*
- *Module Index*

Creating your first stack

Confirming you can access a Heat endpoint

Before any Heat commands can be run, your cloud credentials need to be sourced:

```
$ source openrc
```

You can confirm that Heat is available with this command:

```
$ heat stack-list
```

This should return an empty line

Preparing to create a stack

Your cloud will have different flavors and images available for launching instances, you can discover what is available by running:

```
$ openstack flavor list
$ openstack image list
```

To allow you to SSH into instances launched by Heat, a keypair will be generated:

```
$ openstack keypair create heat_key > heat_key.priv
$ chmod 600 heat_key.priv
```

Launching a stack

Now lets launch a stack, using an example template from the heat-templates repository:

```
$ heat stack-create -u http://git.openstack.org/cgit/openstack/heat-
templates/plain/hot/F20/WordPress_Native.yaml -P key_name=heat_key -P
image_id=my-fedora-image -P instance_type=m1.small teststack
```

Which will respond:

```
+--------------------------------------+-----------+--------------------+----------------------+
| ID                                   | Name      | Status             | Created              |
+--------------------------------------+-----------+--------------------+----------------------+
| (uuid)                               | teststack | CREATE_IN_PROGRESS | (timestamp)          |
+--------------------------------------+-----------+--------------------+----------------------+
```

> **Note**
>
> Link on Heat template presented in command above should reference on RAW template. In case if it be a “html” page with template, Heat will return an error.

List stacks

List the stacks in your tenant:

```
$ heat stack-list
```

List stack events

List the events related to a particular stack:

```
$ heat event-list teststack
```

Describe the wordpress stack

Show detailed state of a stack:

```
$ heat stack-show teststack
```

Note: After a few seconds, the stack_status should change from `IN_PROGRESS` to `CREATE_COMPLETE`.

Verify instance creation

Because the software takes some time to install from the repository, it may be a few minutes before the Wordpress instance is in a running state.

Point a web browser at the location given by the `WebsiteURL` output as shown by `heat output-show`:

```
$ WebsiteURL=$(heat output-show --format raw teststack WebsiteURL)
$ curl $WebsiteURL
```

Delete the instance when done

Note: The list operation will show no running stack.:

```
$ heat stack-delete teststack
$ heat stack-list
```

You can explore other heat commands by referring to the Heat chapter of the OpenStack Command-Line Interface Reference then read the *Template Guide* and start authoring your own templates.

Template Guide

- Heat Orchestration Template (HOT) Guide
  - Status
  - Writing a hello world HOT template
- Writing a hello world HOT template
  - A most basic template
  - Input parameters
  - Template outputs
- Heat Orchestration Template (HOT) specification
  - Status
  - Template structure
  - Heat template version
  - Parameter groups section
  - Parameters section
  - Resources section
  - Outputs section
  - Conditions section
  - Intrinsic functions

- Instances
  - Manage instances
  - Manage networks
  - Manage volumes
- Software configuration
  - Image building
  - User-data boot scripts and cloud-init
  - Software deployment resources
- Environments
  - Environment file format
  - Global and effective environments
  - Usage examples
- Template composition
  - Use the template filename as type
  - Define a new resource type
  - Get access to nested attributes
  - OpenStack Resource Types
    - OS::Aodh::Alarm
    - OS::Aodh::GnocchiAggregationByMetricsAlarm
    - OS::Aodh::GnocchiAggregationByResourcesAlarm
    - OS::Aodh::GnocchiResourcesAlarm
    - OS::Barbican::CertificateContainer
    - OS::Barbican::GenericContainer
    - OS::Barbican::Order
    - OS::Barbican::RSAContainer
    - OS::Barbican::Secret
    - OS::Cinder::EncryptedVolumeType
    - OS::Cinder::Volume
    - OS::Cinder::VolumeAttachment
    - OS::Cinder::VolumeType
    - OS::Designate::Domain
    - OS::Designate::Record
    - OS::Glance::Image
    - OS::Heat::AccessPolicy
    - OS::Heat::AutoScalingGroup
    - OS::Heat::CloudConfig
    - OS::Heat::InstanceGroup
    - OS::Heat::MultipartMime
    - OS::Heat::None

- OS::Heat::RandomString
- OS::Heat::ResourceChain
- OS::Heat::ResourceGroup
- OS::Heat::ScalingPolicy
- OS::Heat::SoftwareComponent
- OS::Heat::SoftwareConfig
- OS::Heat::SoftwareDeployment
- OS::Heat::SoftwareDeploymentGroup
- OS::Heat::Stack
- OS::Heat::StructuredConfig
- OS::Heat::StructuredDeployment
- OS::Heat::StructuredDeploymentGroup
- OS::Heat::SwiftSignal
- OS::Heat::SwiftSignalHandle
- OS::Heat::TestResource
- OS::Heat::UpdateWaitConditionHandle
- OS::Heat::WaitCondition
- OS::Heat::WaitConditionHandle
- OS::Keystone::Endpoint
- OS::Keystone::Group
- OS::Keystone::GroupRoleAssignment
- OS::Keystone::Project
- OS::Keystone::Region
- OS::Keystone::Role
- OS::Keystone::Service
- OS::Keystone::User
- OS::Keystone::UserRoleAssignment
- OS::Magnum::Bay
- OS::Magnum::BayModel
- OS::Manila::SecurityService
- OS::Manila::Share
- OS::Manila::ShareNetwork
- OS::Manila::ShareType
- OS::Mistral::CronTrigger
- OS::Mistral::Workflow
- OS::Monasca::AlarmDefinition
- OS::Monasca::Notification
- OS::Neutron::AddressScope
- OS::Neutron::Firewall

- OS::Neutron::FirewallPolicy
- OS::Neutron::FirewallRule
- OS::Neutron::FloatingIP
- OS::Neutron::FloatingIPAssociation
- OS::Neutron::IKEPolicy
- OS::Neutron::IPsecPolicy
- OS::Neutron::IPsecSiteConnection
- OS::Neutron::LBaaS::HealthMonitor
- OS::Neutron::LBaaS::L7Policy
- OS::Neutron::LBaaS::Listener
- OS::Neutron::LBaaS::LoadBalancer
- OS::Neutron::LBaaS::Pool
- OS::Neutron::LBaaS::PoolMember
- OS::Neutron::MeteringLabel
- OS::Neutron::MeteringRule
- OS::Neutron::Net
- OS::Neutron::NetworkGateway
- OS::Neutron::Port
- OS::Neutron::ProviderNet
- OS::Neutron::QoSBandwidthLimitRule
- OS::Neutron::QoSDscpMarkingRule
- OS::Neutron::QoSPolicy
- OS::Neutron::RBACPolicy
- OS::Neutron::Router
- OS::Neutron::RouterInterface
- OS::Neutron::SecurityGroup
- OS::Neutron::SecurityGroupRule
- OS::Neutron::Subnet
- OS::Neutron::SubnetPool
- OS::Neutron::VPNService
- OS::Nova::Flavor
- OS::Nova::FloatingIP
- OS::Nova::FloatingIPAssociation
- OS::Nova::HostAggregate
- OS::Nova::KeyPair
- OS::Nova::Server
- OS::Nova::ServerGroup
- OS::Sahara::Cluster
- OS::Sahara::ClusterTemplate

        - OS::Sahara::DataSource
        - OS::Sahara::ImageRegistry
        - OS::Sahara::JobBinary
        - OS::Sahara::NodeGroupTemplate
        - OS::Senlin::Cluster
        - OS::Senlin::Node
        - OS::Senlin::Policy
        - OS::Senlin::Profile
        - OS::Senlin::Receiver
        - OS::Swift::Container
        - OS::Trove::Cluster
        - OS::Trove::Instance
        - OS::Zaqar::Queue
    - CloudFormation Compatible Resource Types
        - AWS::AutoScaling::AutoScalingGroup
        - AWS::AutoScaling::LaunchConfiguration
        - AWS::AutoScaling::ScalingPolicy
        - AWS::CloudFormation::Stack
        - AWS::CloudFormation::WaitCondition
        - AWS::CloudFormation::WaitConditionHandle
        - AWS::EC2::EIP
        - AWS::EC2::EIPAssociation
        - AWS::EC2::Instance
        - AWS::EC2::InternetGateway
        - AWS::EC2::NetworkInterface
        - AWS::EC2::RouteTable
        - AWS::EC2::SecurityGroup
        - AWS::EC2::Subnet
        - AWS::EC2::SubnetRouteTableAssociation
        - AWS::EC2::VPC
        - AWS::EC2::VPCGatewayAttachment
        - AWS::EC2::Volume
        - AWS::EC2::VolumeAttachment
        - AWS::ElasticLoadBalancing::LoadBalancer
        - AWS::IAM::AccessKey
        - AWS::IAM::User
        - AWS::S3::Bucket
    - Unsupported Heat Resource Types
        - OS::Aodh::CombinationAlarm

        - OS::Heat::HARestarter
        - OS::Neutron::ExtraRoute
        - OS::Neutron::HealthMonitor
        - OS::Neutron::LoadBalancer
        - OS::Neutron::Pool
        - OS::Neutron::PoolMember
        - OS::Neutron::PortPair
    - Contributed Heat Resource Types
        - OS::Nova::Server
        - Rackspace Cloud Resource Types
        - DockerInc Resource
    - CloudFormation Compatible Functions
        - Ref
        - Fn::Base64
        - Fn::FindInMap
        - Fn::GetAtt
        - Fn::GetAZs
        - Fn::Join
        - Fn::Select
        - Fn::Split
        - Fn::Replace
        - Fn::ResourceFacade
        - Fn::MemberListToMap
        - Fn::Equals

Heat Orchestration Template (HOT) Guide

HOT is a new template format meant to replace the Heat CloudFormation-compatible format (CFN) as the native format supported by the Heat over time. This guide is targeted towards template authors and explains how to write HOT templates based on examples. A detailed specification of HOT can be found at *Heat Orchestration Template (HOT) specification*.

Status

HOT support is still under development and needs more work to provide access to all functionality currently available via the CFN compatible template interface.

This guide will be updated periodically whenever new features get implemented for HOT.

Writing a hello world HOT template

This section gives an introduction on how to write HOT templates, starting from very basic steps and then going into more and more detail by means of examples.

A most basic template

The most basic template you can think of may contain only a single resource definition using only predefined properties (along with the mandatory Heat template version tag). For example, the template below could be used to simply deploy a single compute instance.

```
heat_template_version: 2015-04-30

description: Simple template to deploy a single compute instance

resources:
  my_instance:
    type: OS::Nova::Server
    properties:
      key_name: my_key
      image: F18-x86_64-cfntools
      flavor: m1.small
```

Each HOT template has to include the *heat_template_version* key with a valid version of HOT, e.g. 2015-10-15 (see *Heat template version* for a list of all versions). While the *description* is optional, it is good practice to include some useful text that describes what users can do with the template. In case you want to provide a longer description that does not fit on a single line, you can provide multi-line text in YAML, for example:

```
description: >
  This is how you can provide a longer description
  of your template that goes over several lines.
```

The *resources* section is required and must contain at least one resource definition. In the example above, a compute instance is defined with fixed values for the 'key_name', 'image' and 'flavor' parameters.

Note that all those elements, i.e. a key-pair with the given name, the image and the flavor have to exist in the OpenStack environment where the template is used. Typically a template is made more easily reusable, though, by defining a set of *input parameters* instead of hard-coding such values.

Template input parameters

Input parameters defined in the *parameters* section of a HOT template (see also *Parameters section*) allow users to customize a template during deployment. For example, this allows for providing custom key-pair names or image IDs to be used for a deployment. From a template author's perspective, this helps to make a template more easily reusable by avoiding hardcoded assumptions.

Sticking to the example used above, it makes sense to allow users to provide their custom key-pairs, provide their own image, and to select a flavor for the compute instance. This can be achieved by extending the initial template as follows:

```
heat_template_version: 2015-04-30

description: Simple template to deploy a single compute instance

parameters:
  key_name:
    type: string
    label: Key Name
    description: Name of key-pair to be used for compute instance
  image_id:
    type: string
    label: Image ID
    description: Image to be used for compute instance
  instance_type:
```

```
    type: string
    label: Instance Type
    description: Type of instance (flavor) to be used

resources:
  my_instance:
    type: OS::Nova::Server
    properties:
      key_name: { get_param: key_name }
      image: { get_param: image_id }
      flavor: { get_param: instance_type }
```

In the example above, three input parameters have been defined that have to be provided by the user upon deployment. The fixed values for the respective resource properties have been replaced by references to the corresponding input parameters by means of the *get_param* function (see also *Intrinsic functions*).

You can also define default values for input parameters which will be used in case the user does not provide the respective parameter during deployment. For example, the following definition for the *instance_type* parameter would select the 'm1.small' flavor unless specified otherwise by the user.

```
parameters:
  instance_type:
    type: string
    label: Instance Type
    description: Type of instance (flavor) to be used
    default: m1.small
```

Another option that can be specified for a parameter is to hide its value when users request information about a stack deployed from a template. This is achieved by the *hidden* attribute and useful, for example when requesting passwords as user input:

```
parameters:
  database_password:
    type: string
    label: Database Password
    description: Password to be used for database
    hidden: true
```

Restricting user input

In some cases you might want to restrict the values of input parameters that users can supply. For example, you might know that the software running in a compute instance needs a certain amount of resources so you might want to restrict the *instance_type* parameter introduced above. Parameters in HOT templates can be

restricted by adding a *constraints* section (see also *Parameter Constraints*). For example, the following would allow only three values to be provided as input for the *instance_type* parameter:

```
parameters:
  instance_type:
    type: string
    label: Instance Type
    description: Type of instance (flavor) to be used
    constraints:
      - allowed_values: [ m1.medium, m1.large, m1.xlarge ]
        description: Value must be one of m1.medium, m1.large or
m1.xlarge.
```

The *constraints* section allows for defining a list of constraints that must all be fulfilled by user input. For example, the following list of constraints could be used to clearly specify format requirements on a password to be provided by users:

```
parameters:
  database_password:
    type: string
    label: Database Password
    description: Password to be used for database
    hidden: true
    constraints:
      - length: { min: 6, max: 8 }
        description: Password length must be between 6 and 8 characters.
      - allowed_pattern: "[a-zA-Z0-9]+"
        description: Password must consist of characters and numbers only.
      - allowed_pattern: "[A-Z]+[a-zA-Z0-9]*"
        description: Password must start with an uppercase character.
```

Note that you can define multiple constraints of the same type. Especially in the case of allowed patterns this not only allows for keeping regular expressions simple and maintainable, but also for keeping error messages to be presented to users precise.

Providing template outputs

In addition to template customization through input parameters, you will typically want to provide outputs to users, which can be done in the *outputs* section of a template (see also *Outputs section*). For example, the IP address by which the instance defined in the example above can be accessed should be provided to users. Otherwise, users would have to look it up themselves. The definition for providing the IP address of the compute instance as an output is shown in the

following snippet:

```
outputs:
  instance_ip:
    description: The IP address of the deployed instance
    value: { get_attr: [my_instance, first_address] }
```

Output values are typically resolved using intrinsic function such as the *get_attr* function in the example above (see also *Intrinsic functions*).

Writing a hello world HOT template

HOT is a new template format meant to replace the CloudFormation-compatible format (CFN) as the native format supported by the Orchestration module over time. This guide is targeted towards template authors and explains how to write HOT templates based on examples. A detailed specification of HOT can be found at *Heat Orchestration Template (HOT) specification*.

This section gives an introduction on how to write HOT templates, starting from very basic steps and then going into more and more detail by means of examples.

A most basic template

The most basic template you can think of contains only a single resource definition using only predefined properties. For example, the template below could be used to deploy a single compute instance:

```
heat_template_version: 2015-04-30

description: Simple template to deploy a single compute instance

resources:
  my_instance:
    type: OS::Nova::Server
    properties:
      key_name: my_key
      image: ubuntu-trusty-x86_64
      flavor: m1.small
```

Each HOT template must include the `heat_template_version` key with the HOT version value, for example, `2013-05-23`. A list of HOT template versions can be found at Heat Template Version file

The `description` key is optional, however it is good practice to include some useful

text that describes what users can do with the template. In case you want to provide a longer description that does not fit on a single line, you can provide multi-line text in YAML, for example:

```
description: >
  This is how you can provide a longer description
  of your template that goes over several lines.
```

The `resources` section is required and must contain at least one resource definition. In the above example, a compute instance is defined with fixed values for the `key_name`, `image` and `flavor` properties.

> **Note**
>
> All the defined elements (key pair, image, flavor) have to exist in the OpenStack environment where the template is used.

Input parameters

Input parameters defined in the `parameters` section of a template allow users to customize a template during deployment. For example, this allows for providing custom key pair names or image IDs to be used for a deployment. From a template author's perspective, this helps to make a template more easily reusable by avoiding hardcoded assumptions.

The following example extends the previous template to provide parameters for the key pair, image and flavor properties of the resource:

```
heat_template_version: 2015-04-30

description: Simple template to deploy a single compute instance

parameters:
  key_name:
    type: string
    label: Key Name
    description: Name of key-pair to be used for compute instance
  image_id:
    type: string
    label: Image ID
    description: Image to be used for compute instance
  flavor:
    type: string
    label: Instance Type
```

```
    description: Type of instance (flavor) to be used

resources:
  my_instance:
    type: OS::Nova::Server
    properties:
      key_name: { get_param: key_name }
      image: { get_param: image_id }
      flavor: { get_param: flavor }
```

Values for the three parameters must be defined by the template user during the deployment of a stack. The `get_param` intrinsic function retrieves a user-specified value for a given parameter and uses this value for the associated resource property.

For more information about intrinsic functions, see *Intrinsic functions*.

Providing default values

You can provide default values for parameters. If a user doesn't define a value for a parameter, the default value is used during the stack deployment. The following example defines a default value `m1.small` for the `flavor` property:

```
parameters:
  flavor:
    type: string
    label: Instance Type
    description: Flavor to be used
   default: m1.small
```

> **Note**
>
> If a template doesn't define a default value for a parameter, then the user must define the value, otherwise the stack creation will fail.

Hiding parameters values

The values that a user provides when deploying a stack are available in the stack details and can be accessed by any user in the same tenant. To hide the value of a parameter, use the `hidden` boolean attribute of the parameter:

```
parameters:
  database_password:
    type: string
    label: Database Password
    description: Password to be used for database
    hidden: true
```

Restricting user input

You can restrict the values of an input parameter to make sure that the user defines valid data for this parameter. The `constraints` property of an input parameter defines a list of constraints to apply for the parameter. The following example restricts the `flavor` parameter to a list of three possible values:

```
parameters:
  flavor:
    type: string
    label: Instance Type
    description: Type of instance (flavor) to be used
    constraints:
      - allowed_values: [ m1.medium, m1.large, m1.xlarge ]
        description: Value must be one of m1.medium, m1.large or
m1.xlarge.
```

The following example defines multiple constraints for a password definition:

```
parameters:
  database_password:
    type: string
    label: Database Password
    description: Password to be used for database
    hidden: true
    constraints:
      - length: { min: 6, max: 8 }
        description: Password length must be between 6 and 8 characters.
      - allowed_pattern: "[a-zA-Z0-9]+"
        description: Password must consist of characters and numbers only.
      - allowed_pattern: "[A-Z]+[a-zA-Z0-9]*"
        description: Password must start with an uppercase character.
```

The list of supported constraints is available in the *Parameter Constraints* section.

> **Note**
>
> You can define multiple constraints of the same type. Especially in the case of allowed patterns this not only allows for keeping regular expressions simple and maintainable, but also for keeping error messages to be presented to users precise.

Template outputs

In addition to template customization through input parameters, you can provide information about the resources created during the stack deployment to the users in the `outputs` section of a template. In the following example the output section provides the IP address of the `my_instance` resource:

```
outputs:
  instance_ip:
    description: The IP address of the deployed instance
    value: { get_attr: [my_instance, first_address] }
```

> **Note**
>
> Output values are typically resolved using intrinsic function such as the `get_attr`. See *Intrinsic functions* for more information about intrinsic functions..

See *Outputs section* for more information about the `outputs` section.

Heat Orchestration Template (HOT) specification

HOT is a new template format meant to replace the Heat CloudFormation-compatible format (CFN) as the native format supported by the Heat over time. This specification explains in detail all elements of the HOT template format. An example driven guide to writing HOT templates can be found at *Heat Orchestration Template (HOT) Guide*.

Status

HOT is considered reliable, supported, and standardized as of our Icehouse (April 2014) release. The Heat core team may make improvements to the standard, which very likely would be backward compatible. The template format is also versioned. Since Juno release, Heat supports multiple different versions of the HOT specification.

Template structure

HOT templates are defined in YAML and follow the structure outlined below.

```
heat_template_version: 2016-10-14

description:
  # a description of the template

parameter_groups:
  # a declaration of input parameter groups and order

parameters:
  # declaration of input parameters

resources:
  # declaration of template resources
```

```
outputs:
  # declaration of output parameters

conditions:
  # declaration of conditions
```

heat_template_version
: This key with value `2013-05-23` (or a later date) indicates that the YAML document is a HOT template of the specified version.

description
: This optional key allows for giving a description of the template, or the workload that can be deployed using the template.

parameter_groups
: This section allows for specifying how the input parameters should be grouped and the order to provide the parameters in. This section is optional and can be omitted when necessary.

parameters
: This section allows for specifying input parameters that have to be provided when instantiating the template. The section is optional and can be omitted when no input is required.

resources
: This section contains the declaration of the single resources of the template. This section with at least one resource should be defined in any HOT template, or the template would not really do anything when being instantiated.

outputs
: This section allows for specifying output parameters available to users once the template has been instantiated. This section is optional and can be omitted when no output values are required.

conditions

: This optional section includes statements which can be used to restrict when a resource is created or when a property is defined. They can be associated with resources and resource properties in the `resources` section, also can be associated with outputs in the `outputs` sections of a template.

  Note: Support for this section is added in the Newton version.

Heat template version

The value of `heat_template_version` tells Heat not only the format of the template but also features that will be validated and supported. Beginning with the Newton release, the version can be either the date of the Heat release or the code name of the Heat release. Heat currently supports the following values for the `heat_template_version` key:

2013-05-23

> The key with value `2013-05-23` indicates that the YAML document is a HOT template and it may contain features implemented until the Icehouse release. This version supports the following functions (some are back ported to this version):
>
> ```
> get_attr
> get_file
> get_param
> get_resource
> list_join
> resource_facade
> str_replace
> Fn::Base64
> Fn::GetAZs
> Fn::Join
> Fn::MemberListToMap
> Fn::Replace
> Fn::ResourceFacade
> Fn::Select
> Fn::Split
> Ref
> ```

2014-10-16

> The key with value `2014-10-16` indicates that the YAML document is a HOT template and it may contain features added and/or removed up until the Juno release. This version removes most CFN functions that were supported in the Icehouse release, i.e. the `2013-05-23` version. So the supported functions now are:
>
> ```
> get_attr
> get_file
> get_param
> ```

```
get_resource
list_join
resource_facade
str_replace
Fn::Select
```

2015-04-30

The key with value `2015-04-30` indicates that the YAML document is a HOT template and it may contain features added and/or removed up until the Kilo release. This version adds the `repeat` function. So the complete list of supported functions is:

```
get_attr
get_file
get_param
get_resource
list_join
repeat
digest
resource_facade
str_replace
Fn::Select
```

2015-10-15

The key with value `2015-10-15` indicates that the YAML document is a HOT template and it may contain features added and/or removed up until the Liberty release. This version removes the *Fn::Select* function, path based `get_attr`/`get_param` references should be used instead. Moreover `get_attr` since this version returns dict of all attributes for the given resource excluding *show* attribute, if there's no <attribute name> specified, e.g. `{ get_attr: [<resource name>]}`. This version also adds the str_split function and support for passing multiple lists to the existing list_join function. The complete list of supported functions is:

```
get_attr
get_file
get_param
get_resource
list_join
```

```
repeat
digest
resource_facade
str_replace
str_split
```

2016-04-08

The key with value `2016-04-08` indicates that the YAML document is a HOT template and it may contain features added and/or removed up until the Mitaka release. This version also adds the `map_merge` function which can be used to merge the contents of maps. The complete list of supported functions is:

```
digest
get_attr
get_file
get_param
get_resource
list_join
map_merge
repeat
resource_facade
str_replace
str_split
```

2016-10-14 | newton

The key with value `2016-10-14` or `newton` indicates that the YAML document is a HOT template and it may contain features added and/or removed up until the Newton release. This version adds the `yaql` function which can be used for evaluation of complex expressions, and the `map_replace` function that can do key/value replacements on a mapping. The complete list of supported functions is:

```
digest
get_attr
get_file
get_param
get_resource
list_join
map_merge
```

```
map_replace
repeat
resource_facade
str_replace
str_split
yaql
```

This version also adds `equals` condition function which can be used to compare whether two values are equal. The complete list of supported condition functions is:

```
equals
get_param
```

Parameter groups section

The `parameter_groups` section allows for specifying how the input parameters should be grouped and the order to provide the parameters in. These groups are typically used to describe expected behavior for downstream user interfaces.

These groups are specified in a list with each group containing a list of associated parameters. The lists are used to denote the expected order of the parameters. Each parameter should be associated to a specific group only once using the parameter name to bind it to a defined parameter in the `parameters` section.

```
parameter_groups:
- label: <human-readable label of parameter group>
  description: <description of the parameter group>
  parameters:
  - <param name>
  - <param name>
```

label
: A human-readable label that defines the associated group of parameters.

description
: This attribute allows for giving a human-readable description of the parameter group.

parameters
: A list of parameters associated with this parameter group.

param name
: The name of the parameter that is defined in the associated `parameters` section.

Parameters section

The `parameters` section allows for specifying input parameters that have to be provided when instantiating the template. Such parameters are typically used to customize each deployment (e.g. by setting custom user names or passwords) or for binding to environment-specifics like certain images.

Each parameter is specified in a separated nested block with the name of the parameters defined in the first line and additional attributes such as type or default value defined as nested elements.

```
parameters:
  <param name>:
    type: <string | number | json | comma_delimited_list | boolean>
    label: <human-readable name of the parameter>
    description: <description of the parameter>
    default: <default value for parameter>
    hidden: <true | false>
    constraints:
      <parameter constraints>
    immutable: <true | false>
```

param name
The name of the parameter.

type
The type of the parameter. Supported types are `string`, `number`, `comma_delimited_list`, `json` and `boolean`. This attribute is required.

label
A human readable name for the parameter. This attribute is optional.

description
A human readable description for the parameter. This attribute is optional.

default
A default value for the parameter. This value is used if the user doesn't specify his own value during deployment. This attribute is optional.

hidden

Defines whether the parameters should be hidden when a user requests information about a stack created from the template. This attribute can be used to hide passwords specified as parameters.

This attribute is optional and defaults to `false`.

constraints
A list of constraints to apply. The constraints are validated by the Orchestration engine when a user deploys a stack. The stack creation fails if the parameter

value doesn't comply to the constraints. This attribute is optional.

immutable

Defines whether the parameter is updatable. Stack update fails, if this is set to `true` and the parameter value is changed. This attribute is optional and defaults to `false`.

The table below describes all currently supported types with examples:

Type	Description	Examples
string	A literal string.	"String param"
number	An integer or float.	"2"; "0.2"
comma_delimited_list	An array of literal strings that are separated by commas. The total number of strings should be one more than the total number of commas.	["one", "two"]; "one, two"; Note: "one, two" returns ["one", " two"]
json	A JSON-formatted map or list.	{"key": "value"}
boolean	Boolean type value, which can be equal "t", "true", "on", "y", "yes", or "1" for true value and "f", "false", "off", "n", "no", or "0" for false value.	"on"; "n"

The following example shows a minimalistic definition of two parameters

```
parameters:
  user_name:
    type: string
    label: User Name
    description: User name to be configured for the application
  port_number:
    type: number
    label: Port Number
    description: Port number to be configured for the web server
```

> **Note**
>
> The description and the label are optional, but defining these attributes is good practice to provide useful information about the role of the parameter to the user.

Parameter Constraints

The `constraints` block of a parameter definition defines additional validation constraints that apply to the value of the parameter. The parameter values provided by a user are validated against the constraints at instantiation time. The constraints are defined as a list with the following syntax

```
constraints:
  - <constraint type>: <constraint definition>
    description: <constraint description>
```

constraint type
: Type of constraint to apply. The set of currently supported constraints is given below.

constraint definition
: The actual constraint, depending on the constraint type. The concrete syntax for each constraint type is given below.

description
: A description of the constraint. The text is presented to the user when the value he defines violates the constraint. If omitted, a default validation message is presented to the user. This attribute is optional.

The following example shows the definition of a string parameter with two constraints. Note that while the descriptions for each constraint are optional, it is good practice to provide concrete descriptions to present useful messages to the user at deployment time.

```
parameters:
  user_name:
    type: string
    label: User Name
    description: User name to be configured for the application
    constraints:
      - length: { min: 6, max: 8 }
        description: User name must be between 6 and 8 characters
      - allowed_pattern: "[A-Z]+[a-zA-Z0-9]*"
        description: User name must start with an uppercase character
```

> **Note**
>
> While the descriptions for each constraint are optional, it is good practice to provide concrete descriptions so useful messages can be presented to the user at deployment time.

The following sections list the supported types of parameter constraints, along with the concrete syntax for each type.

length

The `length` constraint applies to parameters of type `string`. It defines a lower and upper limit for the length of the string value.

The syntax of the `length` constraint is

```
length: { min: <lower limit>, max: <upper limit> }
```

It is possible to define a length constraint with only a lower limit or an upper limit. However, at least one of `min` or `max` must be specified.

range

The `range` constraint applies to parameters of type `number`. It defines a lower and upper limit for the numeric value of the parameter.

The syntax of the `range` constraint is

```
range: { min: <lower limit>, max: <upper limit> }
```

It is possible to define a range constraint with only a lower limit or an upper limit. However, at least one of `min` or `max` must be specified.

The minimum and maximum boundaries are included in the range. For example, the following range constraint would allow for all numeric values between 0 and 10

```
range: { min: 0, max: 10 }
```

allowed_values

The `allowed_values` constraint applies to parameters of type `string` or `number`. It specifies a set of possible values for a parameter. At deployment time, the user-provided value for the respective parameter must match one of the elements of the list.

The syntax of the `allowed_values` constraint is

```
allowed_values: [ <value>, <value>, ... ]
```

Alternatively, the following YAML list notation can be used

```
allowed_values:
  - <value>
  - <value>
  - ...
```

For example

```
parameters:
  instance_type:
    type: string
    label: Instance Type
    description: Instance type for compute instances
    constraints:
      - allowed_values:
        - m1.small
        - m1.medium
        - m1.large
```

allowed_pattern

The `allowed_pattern` constraint applies to parameters of type `string`. It specifies a regular expression against which a user-provided parameter value must evaluate at deployment.

The syntax of the `allowed_pattern` constraint is

```
allowed_pattern: <regular expression>
```

For example

```
parameters:
  user_name:
    type: string
    label: User Name
    description: User name to be configured for the application
    constraints:
      - allowed_pattern: "[A-Z]+[a-zA-Z0-9]*"
        description: User name must start with an uppercase character
```

custom_constraint

The `custom_constraint` constraint adds an extra step of validation, generally to check that the specified resource exists in the backend. Custom constraints get implemented by plug-ins and can provide any kind of advanced constraint validation logic.

The syntax of the `custom_constraint` constraint is

```
custom_constraint: <name>
```

The `name` attribute specifies the concrete type of custom constraint. It corresponds to the name under which the respective validation plugin has been registered in the Orchestration engine.

For example

```
parameters:
  key_name
    type: string
    description: SSH key pair
    constraints:
      - custom_constraint: nova.keypair
```

The following section lists the custom constraints and the plug-ins that support them.

Name	Plug-in
barbican.container	heat.engine.clients.os.barbican:ContainerConstraint
barbican.secret	heat.engine.clients.os.barbican:SecretConstraint
cinder.backup	heat.engine.clients.os.cinder:VolumeBackupConstraint
cinder.snapshot	heat.engine.clients.os.cinder:VolumeSnapshotConstraint
cinder.volume	heat.engine.clients.os.cinder:VolumeConstraint
cinder.vtype	heat.engine.clients.os.cinder:VolumeTypeConstraint
cron_expression	heat.engine.constraint.common_constraints:CRONExpressionConstraint
designate.domain	heat.engine.clients.os.designate:DesignateDomainConstraint
dns_domain	heat.engine.constraint.common_constraints:DNSDomainConstraint
dns_name	heat.engine.constraint.common_constraints:DNSNameConstraint
expiration	heat.engine.constraint.common_constraints:ExpirationConstraint
glance.image	heat.engine.clients.os.glance:ImageConstraint
heat.resource_type	heat.engine.constraint.heat_constraints:ResourceTypeConstraint

Name	Plug-in
ip_addr	heat.engine.constraint.common_constraints:IPConstraint
iso_8601	heat.engine.constraint.common_constraints:ISO8601Constraint
keystone.domain	heat.engine.clients.os.keystone.keystone_constraints:KeystoneDomainConstraint
keystone.group	heat.engine.clients.os.keystone.keystone_constraints:KeystoneGroupConstraint
keystone.project	heat.engine.clients.os.keystone.keystone_constraints:KeystoneProjectConstraint
keystone.region	heat.engine.clients.os.keystone.keystone_constraints:KeystoneRegionConstraint
keystone.role	heat.engine.clients.os.keystone.keystone_constraints:KeystoneRoleConstraint
keystone.service	heat.engine.clients.os.keystone.keystone_constraints:KeystoneServiceConstraint
keystone.user	heat.engine.clients.os.keystone.keystone_constraints:KeystoneUserConstraint
mac_addr	heat.engine.constraint.common_constraints:MACConstraint
magnum.baymodel	heat.engine.clients.os.magnum:BaymodelConstraint
manila.share_network	heat.engine.clients.os.manila:ManilaShareNetworkConstraint
manila.share_snapshot	heat.engine.clients.os.manila:ManilaShareSnapshotConstraint
manila.share_type	heat.engine.clients.os.manila:ManilaShareTypeConstraint
mistral.workflow	heat.engine.clients.os.mistral:WorkflowConstraint
monasca.notification	heat.engine.clients.os.monasca:MonascaNotificationConstraint
net_cidr	heat.engine.constraint.common_constraints:CIDRConstraint

Name	Plug-in
neutron.address_scope	heat.engine.clients.os.neutron.neutron_constraints:AddressScopeConstraint
neutron.lb.provider	heat.engine.clients.os.neutron.neutron_constraints:LBaasV1ProviderConstraint
neutron.lbaas.listener	heat.engine.clients.os.neutron.lbaas_constraints:ListenerConstraint
neutron.lbaas.loadbalancer	heat.engine.clients.os.neutron.lbaas_constraints:LoadbalancerConstraint
neutron.lbaas.pool	heat.engine.clients.os.neutron.lbaas_constraints:PoolConstraint
neutron.lbaas.provider	heat.engine.clients.os.neutron.lbaas_constraints:LBaasV2ProviderConstraint
neutron.network	heat.engine.clients.os.neutron.neutron_constraints:NetworkConstraint
neutron.port	heat.engine.clients.os.neutron.neutron_constraints:PortConstraint
neutron.qos_policy	heat.engine.clients.os.neutron.neutron_constraints:QoSPolicyConstraint
neutron.router	heat.engine.clients.os.neutron.neutron_constraints:RouterConstraint
neutron.security_group	heat.engine.clients.os.neutron.neutron_constraints:SecurityGroupConstraint
neutron.subnet	heat.engine.clients.os.neutron.neutron_constraints:SubnetConstraint
neutron.subnetpool	heat.engine.clients.os.neutron.neutron_constraints:SubnetPoolConstraint
nova.flavor	heat.engine.clients.os.nova:FlavorConstraint
nova.host	heat.engine.clients.os.nova:HostConstraint
nova.keypair	heat.engine.clients.os.nova:KeypairConstraint
nova.network	heat.engine.clients.os.nova:NetworkConstraint
nova.server	heat.engine.clients.os.nova:ServerConstraint

Name	Plug-in
rel_dns_name	heat.engine.constraint.common_constraints:RelativeDNSNameConstraint
sahara.image	heat.engine.clients.os.sahara:ImageConstraint
sahara.plugin	heat.engine.clients.os.sahara:PluginConstraint
senlin.cluster	heat.engine.clients.os.senlin:ClusterConstraint
senlin.policy_type	heat.engine.clients.os.senlin:PolicyTypeConstraint
senlin.profile	heat.engine.clients.os.senlin:ProfileConstraint
senlin.profile_type	heat.engine.clients.os.senlin:ProfileTypeConstraint
test_constr	heat.engine.constraint.common_constraints:TestConstraintDelay
timezone	heat.engine.constraint.common_constraints:TimezoneConstraint
trove.flavor	heat.engine.clients.os.trove:FlavorConstraint

Pseudo parameters

In addition to parameters defined by a template author, Heat also creates three parameters for every stack that allow referential access to the stack's name, stack's identifier and project's identifier. These parameters are named `OS::stack_name` for the stack name, `OS::stack_id` for the stack identifier and `OS::project_id` for the project identifier. These values are accessible via the get_param intrinsic function, just like user-defined parameters.

> **Note**
>
> `OS::project_id` is available since 2015.1 (Kilo).

Resources section

The `resources` section defines actual resources that make up a stack deployed from the HOT template (for instance compute instances, networks, storage volumes).

Each resource is defined as a separate block in the `resources` section with the following syntax

```
resources:
  <resource ID>:
    type: <resource type>
    properties:
      <property name>: <property value>
    metadata:
      <resource specific metadata>
    depends_on: <resource ID or list of ID>
    update_policy: <update policy>
    deletion_policy: <deletion policy>
```

resource ID
: A resource ID which must be unique within the `resources` section of the template.

type
: The resource type, such as `OS::Nova::Server` or `OS::Neutron::Port`. This attribute is required.

properties
: A list of resource-specific properties. The property value can be provided in place, or via a function (see *Intrinsic functions*). This section is optional.

metadata
: Resource-specific metadata. This section is optional.

depends_on
: Dependencies of the resource on one or more resources of the template. See *Resource dependencies* for details. This attribute is optional.

update_policy
: Update policy for the resource, in the form of a nested dictionary. Whether update policies are supported and what the exact semantics are depends on the type of the current resource. This attribute is optional.

deletion_policy
: Deletion policy for the resource. The allowed deletion policies are `Delete`, `Retain`, and `Snapshot`. Beginning with `heat_template_version` `2016-10-14`, the lowercase equivalents `delete`, `retain`, and `snapshot` are also allowed. This attribute is optional; the default policy is to delete the physical resource when deleting a resource from the stack.

Depending on the type of resource, the resource block might include more resource specific data.

All resource types that can be used in CFN templates can also be used in HOT templates, adapted to the YAML structure as outlined above.

The following example demonstrates the definition of a simple compute resource with some fixed property values

```
resources:
  my_instance:
    type: OS::Nova::Server
    properties:
      flavor: m1.small
      image: F18-x86_64-cfntools
```

Resource dependencies

The `depends_on` attribute of a resource defines a dependency between this resource and one or more other resources.

If a resource depends on just one other resource, the ID of the other resource is specified as string of the `depends_on` attribute, as shown in the following example

```
resources:
  server1:
    type: OS::Nova::Server
    depends_on: server2

  server2:
    type: OS::Nova::Server
```

If a resource depends on more than one other resources, the value of the `depends_on` attribute is specified as a list of resource IDs, as shown in the following example

```
resources:
  server1:
    type: OS::Nova::Server
    depends_on: [ server2, server3 ]

  server2:
    type: OS::Nova::Server

  server3:
    type: OS::Nova::Server
```

Outputs section

The `outputs` section defines output parameters that should be available to the user after a stack has been created. This would be, for example, parameters such as IP addresses of deployed instances, or URLs of web applications deployed as part of a stack.

Each output parameter is defined as a separate block within the outputs section according to the following syntax

```
outputs:
  <parameter name>:
    description: <description>
    value: <parameter value>
```

parameter name
: The output parameter name, which must be unique within the `outputs` section of a template.

description
: A short description of the output parameter. This attribute is optional.

parameter value
: The value of the output parameter. This value is usually resolved by means of a function. See *Intrinsic functions* for details about the functions. This attribute is required.

The example below shows how the IP address of a compute resource can be defined as an output parameter

```
outputs:
  instance_ip:
    description: IP address of the deployed compute instance
    value: { get_attr: [my_instance, first_address] }
```

Conditions section

The `conditions` section defines one or more conditions which are evaluated based on input parameter values provided when a user creates or updates a stack. The condition can be associated with resources, resource properties and outputs. For example, based on the result of a condition, user can conditionally create resources, user can conditionally set different values of properties, and user can conditionally give outputs of a stack.

The `conditions` section is defined with the following syntax

```
conditions:
  <condition name1>: {expression1}
  <condition name2>: {expression2}
  ...
```

condition name

The condition name, which must be unique within the `conditions` section of a template.

expression

The expression which is expected to return True or False. Usually, the condition functions can be used as expression to define conditions:

```
equals
get_param
```

Note: In condition functions, you can reference a value from an input parameter, but you cannot reference resource or its attribute.

An example of conditions section definition

```
conditions:
  cd1: True
  cd2: {get_param: param1}
  cd3: {equals: [{get_param: param2}, "yes"]}
```

Intrinsic functions

HOT provides a set of intrinsic functions that can be used inside templates to perform specific tasks, such as getting the value of a resource attribute at runtime. The following section describes the role and syntax of the intrinsic functions.

Note: these functions can only be used within the “properties” section of each resource or in the outputs section.

get_attr

The `get_attr` function references an attribute of a resource. The attribute value is resolved at runtime using the resource instance created from the respective resource definition.

Path based attribute referencing using keys or indexes requires `heat_template_version 2014-10-16` or higher.

The syntax of the `get_attr` function is

```
get_attr:
  - <resource name>
  - <attribute name>
  - <key/index 1> (optional)
  - <key/index 2> (optional)
  - ...
```

resource name

The resource name for which the attribute needs to be resolved.

The resource name must exist in the `resources` section of the template.

attribute name

The attribute name to be resolved. If the attribute returns a complex data structure such as a list or a map, then subsequent keys or indexes can be specified. These additional parameters are used to navigate the data structure to return the desired value.

The following example demonstrates how to use the `get_attr` function:

```
resources:
  my_instance:
    type: OS::Nova::Server
    # ...

outputs:
  instance_ip:
    description: IP address of the deployed compute instance
    value: { get_attr: [my_instance, first_address] }
  instance_private_ip:
    description: Private IP address of the deployed compute instance
   value: { get_attr: [my_instance, networks, private, 0] }
```

In this example, if the `networks` attribute contained the following data:

```
{"public": ["2001:0db8:0000:0000:0000:ff00:0042:8329", "1.2.3.4"],
 "private": ["10.0.0.1"]}
```

then the value of `get_attr` function would resolve to `10.0.0.1` (first item of the `private` entry in the `networks` map).

From `heat_template_version`: '2015-10-15' <attribute_name> is optional and if <attribute_name> is not specified, `get_attr` returns dict of all attributes for the given resource excluding *show* attribute. In this case syntax would be next:

```
get_attr:
  - <resource_name>
```

get_file

The `get_file` function returns the content of a file into the template. It is generally used as a file inclusion mechanism for files containing scripts or configuration files.

The syntax of `get_file` function is

```
get_file: <content key>
```

The `content key` is used to look up the `files` dictionary that is provided in the REST API call. The Orchestration client command (`heat`) is `get_file` aware and populates the `files` dictionary with the actual content of fetched paths and URLs. The Orchestration client command supports relative paths and transforms these to the absolute URLs required by the Orchestration API.

> **Note**
>
> The `get_file` argument must be a static path or URL and not rely on intrinsic functions like `get_param`. the Orchestration client does not process intrinsic functions (they are only processed by the Orchestration engine).

The example below demonstrates the `get_file` function usage with both relative and absolute URLs

```
resources:
  my_instance:
    type: OS::Nova::Server
    properties:
      # general properties ...
      user_data:
        get_file: my_instance_user_data.sh
  my_other_instance:
    type: OS::Nova::Server
    properties:
      # general properties ...
      user_data:
        get_file: http://example.com/my_other_instance_user_data.sh
```

The `files` dictionary generated by the Orchestration client during instantiation of the stack would contain the following keys:

- `file:///path/to/my_instance_user_data.sh`

- http://example.com/my_other_instance_user_data.sh

get_param

The get_param function references an input parameter of a template. It resolves to the value provided for this input parameter at runtime.

The syntax of the get_param function is

```
get_param:
 - <parameter name>
 - <key/index 1> (optional)
 - <key/index 2> (optional)
 - ...
```

parameter name

The parameter name to be resolved. If the parameters returns a complex data structure such as a list or a map, then subsequent keys or indexes can be specified. These additional parameters are used to navigate the data structure to return the desired value.

The following example demonstrates the use of the get_param function

```
parameters:
   instance_type:
    type: string
    label: Instance Type
    description: Instance type to be used.
  server_data:
    type: json

resources:
  my_instance:
    type: OS::Nova::Server
    properties:
      flavor: { get_param: instance_type}
      metadata: { get_param: [ server_data, metadata ] }
      key_name: { get_param: [ server_data, keys, 0 ] }
```

In this example, if the instance_type and server_data parameters contained the following data:

```
{"instance_type": "m1.tiny",
{"server_data": {"metadata": {"foo": "bar"},
                 "keys": ["a_key","other_key"]}}}
```

then the value of the property flavor would resolve to m1.tiny, metadata would

resolve to `{"foo": "bar"}` and `key_name` would resolve to `a_key`.

get_resource

The `get_resource` function references another resource within the same template. At runtime, it is resolved to reference the ID of the referenced resource, which is resource type specific. For example, a reference to a floating IP resource returns the respective IP address at runtime. The syntax of the `get_resource` function is

```
get_resource: <resource ID>
```

The resource ID of the referenced resource is given as single parameter to the `get_resource` function.

For example

```
resources:
  instance_port:
    type: OS::Neutron::Port
    properties: ...

  instance:
    type: OS::Nova::Server
    properties:
      ...
      networks:
        port: { get_resource: instance_port }
```

list_join

The `list_join` function joins a list of strings with the given delimiter.

The syntax of the `list_join` function is

```
list_join:
- <delimiter>
- <list to join>
```

For example

```
list_join: [', ', ['one', 'two', 'and three']]
```

This resolve to the string `one, two, and three`.

From HOT version `2015-10-15` you may optionally pass additional lists, which will be appended to the previous lists to join.

For example:

```
list_join: [', ', ['one', 'two'], ['three', 'four']]]
```

This resolve to the string `one, two, three, four`.

From HOT version `2015-10-15` you may optionally also pass non-string list items (e.g json/map/list parameters or attributes) and they will be serialized as json before joining.

digest

The `digest` function allows for performing digest operations on a given value. This function has been introduced in the Kilo release and is usable with HOT versions later than `2015-04-30`.

The syntax of the `digest` function is

```
digest:
  - <algorithm>
  - <value>
```

algorithm
: The digest algorithm. Valid algorithms are the ones provided natively by hashlib (md5, sha1, sha224, sha256, sha384, and sha512) or any one provided by OpenSSL.

value
: The value to digest. This function will resolve to the corresponding hash of the value.

For example

```
# from a user supplied parameter
pwd_hash: { digest: ['sha512', { get_param: raw_password }] }
```

The value of the digest function would resolve to the corresponding hash of the value of `raw_password`.

repeat

The `repeat` function allows for dynamically transforming lists by iterating over the contents of one or more source lists and replacing the list elements into a template. The result of this function is a new list, where the elements are set to the template, rendered for each list item.

The syntax of the `repeat` function is

```
repeat:
  template:
```

```
    <template>
  for_each:
    <var>: <list>
```

template

The `template` argument defines the content generated for each iteration, with placeholders for the elements that need to be replaced at runtime. This argument can be of any supported type.

for_each

The `for_each` argument is a dictionary that defines how to generate the repetitions of the template and perform substitutions. In this dictionary the keys are the placeholder names that will be replaced in the template, and the values are the lists to iterate on. On each iteration, the function will render the template by performing substitution with elements of the given lists. If a single key/value pair is given in this argument, the template will be rendered once for each element in the list. When more than one key/value pairs are given, the iterations will be performed on all the permutations of values between the given lists. The values in this dictionary can be given as functions such as `get_attr` or `get_param`.

The following example shows how a security group resource can be defined to include a list of ports given as a parameter

```
parameters:
  ports:
    type: comma_delimited_list
    label: ports
    default: "80,443,8080"

resources:
  security_group:
    type: OS::Neutron::SecurityGroup
    properties:
      name: web_server_security_group
      rules:
        repeat:
          for_each:
            <%port%>: { get_param: ports }
          template:
            protocol: tcp
            port_range_min: <%port%>
            port_range_max: <%port%>
```

The following example demonstrates how the use of multiple lists enables the

security group to also include parameterized protocols

```
parameters:
  ports:
    type: comma_delimited_list
    label: ports
    default: "80,443,8080"
  protocols:
    type: comma_delimited_list
    label: protocols
    default: "tcp,udp"

resources:
  security_group:
    type: OS::Neutron::SecurityGroup
    properties:
      name: web_server_security_group
      rules:
        repeat:
          for_each:
            <%port%>: { get_param: ports }
            <%protocol%>: { get_param: protocols }
          template:
            protocol: <%protocol%>
            port_range_min: <%port%>
```

Note how multiple entries in the `for_each` argument are equivalent to nested for-loops in most programming languages.

resource_facade

The `resource_facade` function retrieves data in a parent provider template.

A provider template provides a custom definition of a resource, called its facade. For more information about custom templates, see *Template composition*. The syntax of the `resource_facade` function is

```
resource_facade: <data type>
```

`data type` can be one of `metadata`, `deletion_policy` or `update_policy`.

str_replace

The `str_replace` function dynamically constructs strings by providing a template string with placeholders and a list of mappings to assign values to those placeholders at runtime. The placeholders are replaced with mapping values

wherever a mapping key exactly matches a placeholder.

The syntax of the `str_replace` function is

```
str_replace:
  template: <template string>
  params: <parameter mappings>
```

template
: Defines the template string that contains placeholders which will be substituted at runtime.

params
: Provides parameter mappings in the form of dictionary. Each key refers to a placeholder used in the `template` attribute. From HOT version `2015-10-15` you may optionally pass non-string parameter values (e.g json/map/list parameters or attributes) and they will be serialized as json before replacing, prior heat/HOT versions require string values.

The following example shows a simple use of the `str_replace` function in the outputs section of a template to build a URL for logging into a deployed application

```
resources:
  my_instance:
    type: OS::Nova::Server
    # general metadata and properties ...

outputs:
  Login_URL:
    description: The URL to log into the deployed application
    value:
      str_replace:
        template: http://host/MyApplication
        params:
          host: { get_attr: [ my_instance, first_address ] }
```

The following examples show the use of the `str_replace` function to build an instance initialization script

```
parameters:
  DBRootPassword:
    type: string
    label: Database Password
    description: Root password for MySQL
    hidden: true

resources:
```

```
my_instance:
  type: OS::Nova::Server
  properties:
    # general properties ...
    user_data:
      str_replace:
        template: |
          #!/bin/bash
          echo "Hello world"
          echo "Setting MySQL root password"
          mysqladmin -u root password $db_rootpassword
          # do more things ...
        params:
          $db_rootpassword: { get_param: DBRootPassword }
```

In the example above, one can imagine that MySQL is being configured on a compute instance and the root password is going to be set based on a user provided parameter. The script for doing this is provided as userdata to the compute instance, leveraging the `str_replace` function.

str_split

The `str_split` function allows for splitting a string into a list by providing an arbitrary delimiter, the opposite of `list_join`.

The syntax of the `str_split` function is as follows:

```
str_split:
  - ','
  - string,to,split
```

Or:

```
str_split: [',', 'string,to,split']
```

The result of which is:

```
['string', 'to', 'split']
```

Optionally, an index may be provided to select a specific entry from the resulting list, similar to `get_attr`/`get_param`:

```
str_split: [',', 'string,to,split', 0]
```

The result of which is:

```
'string'
```

Note: The index starts at zero, and any value outside the maximum (e.g the length of the list minus one) will cause an error.

map_merge

The `map_merge` function merges maps together. Values in the latter maps override any values in earlier ones. Can be very useful when composing maps that contain configuration data into a single consolidated map.

The syntax of the `map_merge` function is

```
map_merge:
- <map 1>
- <map 2>
- ...
```

For example

```
map_merge: [{'k1': 'v1', 'k2': 'v2'}, {'k1': 'v2'}]
```

This resolves to a map containing `{'k1': 'v2', 'k2': 'v2'}`.

Maps containing no items resolve to `{}`.

map_replace

The `map_replace` function does key/value replacements on an existing mapping. An input mapping is processed by iterating over all keys/values and performing a replacement if an exact match is found in either of the optional keys/values mappings.

The syntax of the `map_replace` function is

```
map_replace:
- <input map>
- keys: <map of key replacements>
  values: <map of value replacements>
```

For example

```
map_replace:
- k1: v1
  k2: v2
- keys:
    k1: K1
  values:
    v2: V2
```

This resolves to a map containing `{'K1': 'v1', 'k2': 'V2'}`.

The keys/values mappings are optional, either or both may be specified.

Note that an error is raised if a replacement defined in “keys” results in a collision with an existing keys in the input or output map.

Also note that while unhashable values (e.g lists) in the input map are valid, they will be ignored by the values replacement, because no key can be defined in the values mapping to define their replacement.

yaql

The `yaql` evaluates yaql expression on a given data.

The syntax of the `yaql` function is

```
yaql:
  expression: <expression>
  data: <data>
```

For example

```
parameters:
  list_param:
    type: comma_delimited_list
    default: [1, 2, 3]

outputs:
  max_elem:
    value:
      yaql:
        expression: $.data.list_param.select(int($)).max()
        data:
          list_param: {get_param: list_param}
```

max_elem output will be evaluated to 3

equals

The `equals` function compares whether two values are equal.

The syntax of the `equals` function is

```
equals: [value_1, value_2]
```

The value can be any type that you want to compare. This function returns true if the two values are equal or false if they aren’t.

For example

```
equals: [{get_param: env_type}, 'prod']
```

If param ‘env_type’ equals to ‘prod’, this function returns true, otherwise returns false.

Instances

Manage instances

Create an instance

Use the *OS::Nova::Server* resource to create a Compute instance. The `flavor` property is the only mandatory one, but you need to define a boot source using one of the `image` or `block_device_mapping` properties.

You also need to define the `networks` property to indicate to which networks your instance must connect if multiple networks are available in your tenant.

The following example creates a simple instance, booted from an image, and connecting to the `private` network:

```
resources:
  instance:
    type: OS::Nova::Server
    properties:
      flavor: m1.small
      image: ubuntu-trusty-x86_64
      networks:
        - network: private
```

Connect an instance to a network

Use the `networks` property of an *OS::Nova::Server* resource to define which networks an instance should connect to. Define each network as a YAML map, containing one of the following keys:

`port`
: The ID of an existing Networking port. You usually create this port in the same template using an *OS::Neutron::Port* resource. You will be able to associate a floating IP to this port, and the port to your Compute instance.

`network`
: The name or ID of an existing network. You don't need to create an *OS::Neutron::Port* resource if you use this property. But you will not be able to use neutron floating IP association for this instance because there will be no specified port for server.

The following example demonstrates the use of the `port` and `network` properties:

```
resources:
  instance_port:
    type: OS::Neutron::Port
    properties:
      network: private
      fixed_ips:
        - subnet_id: "private-subnet"

  instance1:
    type: OS::Nova::Server
    properties:
      flavor: m1.small
      image: ubuntu-trusty-x86_64
      networks:
        - port: { get_resource: instance_port }

  instance2:
    type: OS::Nova::Server
    properties:
      flavor: m1.small
      image: ubuntu-trusty-x86_64
      networks:
        - network: private
```

Create and associate security groups to an instance

Use the *OS::Neutron::SecurityGroup* resource to create security groups.

Define the `security_groups` property of the *OS::Neutron::Port* resource to associate security groups to a port, then associate the port to an instance.

The following example creates a security group allowing inbound connections on ports 80 and 443 (web server) and associates this security group to an instance port:

```
resources:
  web_secgroup:
    type: OS::Neutron::SecurityGroup
    properties:
      rules:
        - protocol: tcp
          remote_ip_prefix: 0.0.0.0/0
          port_range_min: 80
          port_range_max: 80
        - protocol: tcp
          remote_ip_prefix: 0.0.0.0/0
          port_range_min: 443
          port_range_max: 443

  instance_port:
    type: OS::Neutron::Port
    properties:
      network: private
      security_groups:
        - default
        - { get_resource: web_secgroup }
      fixed_ips:
        - subnet_id: private-subnet

  instance:
    type: OS::Nova::Server
    properties:
      flavor: m1.small
      image: ubuntu-trusty-x86_64
      networks:
        - port: { get_resource: instance_port }
```

Create and associate a floating IP to an instance

You can use two sets of resources to create and associate floating IPs to instances.

OS::Nova resources

Use the *OS::Nova::FloatingIP* resource to create a floating IP, and the *OS::Nova::FloatingIPAssociation* resource to associate the floating IP to an instance.

The following example creates an instance and a floating IP, and associate the floating IP to the instance:

```
resources:
  floating_ip:
    type: OS::Nova::FloatingIP
    properties:
      pool: public

  inst1:
    type: OS::Nova::Server
    properties:
      flavor: m1.small
      image: ubuntu-trusty-x86_64
      networks:
        - network: private

  association:
    type: OS::Nova::FloatingIPAssociation
    properties:
      floating_ip: { get_resource: floating_ip }
      server_id: { get_resource: inst1 }
```

OS::Neutron resources

> **Note**
>
> The Networking service (neutron) must be enabled on your OpenStack deployment to use these resources.

Use the *OS::Neutron::FloatingIP* resource to create a floating IP, and the *OS::Neutron::FloatingIPAssociation* resource to associate the floating IP to a port:

```
parameters:
  net:
    description: name of network used to launch instance.
    type: string
```

```
    default: private

resources:
  inst1:
    type: OS::Nova::Server
    properties:
      flavor: m1.small
      image: ubuntu-trusty-x86_64
      networks:
        - network: {get_param: net}

  floating_ip:
    type: OS::Neutron::FloatingIP
    properties:
      floating_network: public

  association:
    type: OS::Neutron::FloatingIPAssociation
    properties:
      floatingip_id: { get_resource: floating_ip }
      port_id: {get_attr: [inst1, addresses, {get_param: net}, 0, port]}
```

You can also create an OS::Neutron::Port and associate that with the server and the floating IP. However the approach mentioned above will work better with stack updates.

```
resources:
  instance_port:
    type: OS::Neutron::Port
    properties:
      network: private
      fixed_ips:
        - subnet_id: "private-subnet"

  floating_ip:
    type: OS::Neutron::FloatingIP
    properties:
      floating_network: public

  association:
    type: OS::Neutron::FloatingIPAssociation
    properties:
      floatingip_id: { get_resource: floating_ip }
      port_id: { get_resource: instance_port }
```

Enable remote access to an instance

The `key_name` attribute of the *OS::Nova::Server* resource defines the key pair to use to enable SSH remote access:

```
resources:
  my_instance:
    type: OS::Nova::Server
    properties:
      flavor: m1.small
      image: ubuntu-trusty-x86_64
      key_name: my_key
```

> **Note**
>
> For more information about key pairs, see Configure access and security for instances.

Create a key pair

You can create new key pairs with the *OS::Nova::KeyPair* resource. Key pairs can be imported or created during the stack creation.

If the `public_key` property is not specified, the Orchestration module creates a new key pair. If the `save_private_key` property is set to `true`, the `private_key` attribute of the resource holds the private key.

The following example creates a new key pair and uses it as authentication key for an instance:

```
resources:
  my_key:
    type: OS::Nova::KeyPair
    properties:
      save_private_key: true
      name: my_key

  my_instance:
    type: OS::Nova::Server
    properties:
      flavor: m1.small
      image: ubuntu-trusty-x86_64
      key_name: { get_resource: my_key }

outputs:
  private_key:
```

```
    description: Private key
    value: { get_attr: [ my_key, private_key ] }
```

Manage networks

Create a network and a subnet

> **Note**
>
> The Networking service (neutron) must be enabled on your OpenStack deployment to create and manage networks and subnets. Networks and subnets cannot be created if your deployment uses legacy networking (nova-network).

Use the *OS::Neutron::Net* resource to create a network, and the *OS::Neutron::Subnet* resource to provide a subnet for this network:

```
resources:
  new_net:
    type: OS::Neutron::Net

  new_subnet:
    type: OS::Neutron::Subnet
    properties:
      network_id: { get_resource: new_net }
      cidr: "10.8.1.0/24"
      dns_nameservers: [ "8.8.8.8", "8.8.4.4" ]
      ip_version: 4
```

Create and manage a router

Use the *OS::Neutron::Router* resource to create a router. You can define its gateway with the `external_gateway_info` property:

```
resources:
  router1:
    type: OS::Neutron::Router
    properties:
      external_gateway_info: { network: public }
```

You can connect subnets to routers with the *OS::Neutron::RouterInterface* resource:

```
resources:
  subnet1_interface:
    type: OS::Neutron::RouterInterface
    properties:
```

```
      router_id: { get_resource: router1 }
      subnet: private-subnet
```

Complete network example

The following example creates a network stack:

- A network and an associated subnet.
- A router with an external gateway.
- An interface to the new subnet for the new router.

In this example, the `public` network is an existing shared network:

```
resources:
  internal_net:
    type: OS::Neutron::Net

  internal_subnet:
    type: OS::Neutron::Subnet
    properties:
      network_id: { get_resource: internal_net }
      cidr: "10.8.1.0/24"
      dns_nameservers: [ "8.8.8.8", "8.8.4.4" ]
      ip_version: 4

  internal_router:
    type: OS::Neutron::Router
    properties:
      external_gateway_info: { network: public }

  internal_interface:
    type: OS::Neutron::RouterInterface
    properties:
      router_id: { get_resource: internal_router }
      subnet: { get_resource: internal_subnet }
```

Manage volumes

Create a volume

Use the *OS::Cinder::Volume* resource to create a new Block Storage volume.

For example:

```
resources:
  my_new_volume:
    type: OS::Cinder::Volume
    properties:
      size: 10
```

The volumes that you create are empty by default. Use the `image` property to create a bootable volume from an existing image:

```
resources:
  my_new_bootable_volume:
    type: OS::Cinder::Volume
    properties:
      size: 10
      image: ubuntu-trusty-x86_64
```

You can also create new volumes from another volume, a volume snapshot, or a volume backup. Use the `source_volid`, `snapshot_id` or `backup_id` properties to create a new volume from an existing source.

For example, to create a new volume from a backup:

```
resources:
  another_volume:
    type: OS::Cinder::Volume
    properties:
      backup_id: 2fff50ab-1a9c-4d45-ae60-1d054d6bc868
```

In this example the `size` property is not defined because the Block Storage service uses the size of the backup to define the size of the new volume.

Attach a volume to an instance

Use the *OS::Cinder::VolumeAttachment* resource to attach a volume to an instance.

The following example creates a volume and an instance, and attaches the volume to the instance:

```
resources:
```

```
  new_volume:
    type: OS::Cinder::Volume
    properties:
      size: 1

  new_instance:
    type: OS::Nova::Server
    properties:
      flavor: m1.small
      image: ubuntu-trusty-x86_64

  volume_attachment:
    type: OS::Cinder::VolumeAttachment
    properties:
      volume_id: { get_resource: new_volume }
      instance_uuid: { get_resource: new_instance }
```

Boot an instance from a volume

Use the `block_device_mapping` property of the *OS::Nova::Server* resource to define a volume used to boot the instance. This property is a list of volumes to attach to the instance before its boot.

The following example creates a bootable volume from an image, and uses it to boot an instance:

```
resources:
  bootable_volume:
    type: OS::Cinder::Volume
    properties:
      size: 10
      image: ubuntu-trusty-x86_64

  instance:
    type: OS::Nova::Server
    properties:
      flavor: m1.small
      networks:
        - network: private
      block_device_mapping:
        - device_name: vda
          volume_id: { get_resource: bootable_volume }
          delete_on_termination: false
```

Software configuration

There are a variety of options to configure the software which runs on the servers in your stack. These can be broadly divided into the following:

- Custom image building
- User-data boot scripts and cloud-init
- Software deployment resources

This section will describe each of these options and provide examples for using them together in your stacks.

Image building

The first opportunity to influence what software is configured on your servers is by booting them with a custom-built image. There are a number of reasons you might want to do this, including:

- **Boot speed** - since the required software is already on the image there is no need to download and install anything at boot time.
- **Boot reliability** - software downloads can fail for a number of reasons including transient network failures and inconsistent software repositories.
- **Test verification** - custom built images can be verified in test environments before being promoted to production.
- **Configuration dependencies** - post-boot configuration may depend on agents already being installed and enabled

A number of tools are available for building custom images, including:

- diskimage-builder image building tools for OpenStack
- imagefactory builds images for a variety of operating system/cloud combinations

Examples in this guide which require custom images will use diskimage-builder.

User-data boot scripts and cloud-init

When booting a server it is possible to specify the contents of the user-data to be passed to that server. This user-data is made available either from configured config-drive or from the Metadata service.

How this user-data is consumed depends on the image being booted, but the most commonly used tool for default cloud images is Cloud-init.

Whether the image is using Cloud-init or not, it should be possible to specify a shell

script in the `user_data` property and have it be executed by the server during boot:

```
resources:

  the_server:
    type: OS::Nova::Server
    properties:
      # flavor, image etc
      user_data: |
        #!/bin/bash
        echo "Running boot script"
        # ...
```

> **Note**
>
> Debugging these scripts it is often useful to view the boot log using `nova console-log <server-id>` to view the progress of boot script execution.

Often there is a need to set variable values based on parameters or resources in the stack. This can be done with the `str_replace` intrinsic function:

```
parameters:
  foo:
    default: bar

resources:

  the_server:
    type: OS::Nova::Server
    properties:
      # flavor, image etc
      user_data:
        str_replace:
          template: |
            #!/bin/bash
            echo "Running boot script with $FOO"
            # ...
          params:
           $FOO: {get_param: foo}
```

> **Warning**
>
> If a stack-update is performed and there are any changes at all to the content of user_data then the server will be replaced (deleted and recreated) so that the modified boot configuration can be run on a new server.

When these scripts grow it can become difficult to maintain them inside the template, so the `get_file` intrinsic function can be used to maintain the script in a separate file:

```
parameters:
  foo:
    default: bar

resources:

  the_server:
    type: OS::Nova::Server
    properties:
      # flavor, image etc
      user_data:
        str_replace:
          template: {get_file: the_server_boot.sh}
          params:
            $FOO: {get_param: foo}
```

> **Note**
>
> `str_replace` can replace any strings, not just strings starting with `$`. However doing this for the above example is useful because the script file can be executed for testing by passing in environment variables.

Choosing the user_data_format

The *OS::Nova::Server* `user_data_format` property determines how the `user_data` should be formatted for the server. For the default value `HEAT_CFNTOOLS`, the `user_data` is bundled as part of the heat-cfntools cloud-init boot configuration data. While `HEAT_CFNTOOLS` is the default for `user_data_format`, it is considered legacy and `RAW` or `SOFTWARE_CONFIG` will generally be more appropriate.

For `RAW` the user_data is passed to Nova unmodified. For a Cloud-init enabled image, the following are both valid `RAW` user-data:

```
resources:

  server_with_boot_script:
    type: OS::Nova::Server
    properties:
      # flavor, image etc
      user_data_format: RAW
      user_data: |
        #!/bin/bash
        echo "Running boot script"
        # ...

  server_with_cloud_config:
    type: OS::Nova::Server
    properties:
      # flavor, image etc
      user_data_format: RAW
      user_data: |
        #cloud-config
        final_message: "The system is finally up, after $UPTIME seconds"
```

For `SOFTWARE_CONFIG` `user_data` is bundled as part of the software config data, and metadata is derived from any associated Software deployment resources.

Signals and wait conditions

Often it is necessary to pause further creation of stack resources until the boot configuration script has notified that it has reached a certain state. This is usually either to notify that a service is now active, or to pass out some generated data which is needed by another resource. The resources *OS::Heat::SwiftSignal* both perform this function using different techniques and tradeoffs.

OS::Heat::WaitCondition is implemented as a call to the Orchestration API resource signal. The token is created using credentials for a user account which is scoped only to the wait condition handle resource. This user is created when the handle is created, and is associated to a project which belongs to the stack, in an identity domain which is dedicated to the orchestration service.

Sending the signal is a simple HTTP request, as with this example using curl:

```
curl -i -X POST -H 'X-Auth-Token: <token>' \
     -H 'Content-Type: application/json' -H 'Accept: application/json' \
     '<wait condition URL>' --data-binary '<json containing signal data>'
```

The JSON containing the signal data is expected to be of the following format:

```
{
  "status": "SUCCESS",
  "reason": "The reason which will appear in the 'heat event-list'
output",
  "data": "Data to be used elsewhere in the template via get_attr",
  "id": "Optional unique ID of signal"
}
```

All of these values are optional, and if not specified will be set to the following defaults:

```
{
  "status": "SUCCESS",
  "reason": "Signal <id> received",
  "data": null,
  "id": "<sequential number starting from 1 for each signal received>"
}
```

If `status` is set to `FAILURE` then the resource (and the stack) will go into a `FAILED` state using the `reason` as failure reason.

The following template example uses the convenience attribute `curl_cli` which builds a curl command with a valid token:

```
resources:
  wait_condition:
    type: OS::Heat::WaitCondition
    properties:
      handle: {get_resource: wait_handle}
      # Note, count of 5 vs 6 is due to duplicate signal ID 5 sent below
      count: 5
      timeout: 300

  wait_handle:
    type: OS::Heat::WaitConditionHandle

  the_server:
    type: OS::Nova::Server
    properties:
      # flavor, image etc
      user_data_format: RAW
      user_data:
        str_replace:
          template: |
            #!/bin/sh
```

```
              # Below are some examples of the various ways signals
              # can be sent to the Handle resource

              # Simple success signal
              wc_notify --data-binary '{"status": "SUCCESS"}'

              # Or you optionally can specify any of the additional fields
              wc_notify --data-binary '{"status": "SUCCESS", "reason":
"signal2"}'
              wc_notify --data-binary '{"status": "SUCCESS", "reason":
"signal3", "data": "data3"}'
              wc_notify --data-binary '{"status": "SUCCESS", "reason":
"signal4", "id": "id4", "data": "data4"}'

              # If you require control of the ID, you can pass it.
              # The ID should be unique, unless you intend for duplicate
              # signals to overwrite each other.  The following two calls
              # do the exact same thing, and will be treated as one signal
              # (You can prove this by changing count above to 7)
              wc_notify --data-binary '{"status": "SUCCESS", "id": "id5"}'
              wc_notify --data-binary '{"status": "SUCCESS", "id": "id5"}'

              # Example of sending a failure signal, optionally
              # reason, id, and data can be specified as above
              # wc_notify --data-binary '{"status": "FAILURE"}'
            params:
              wc_notify: { get_attr: [wait_handle, curl_cli] }

outputs:
  wc_data:
    value: { get_attr: [wait_condition, data] }
    # this would return the following json
    # {"1": null, "2": null, "3": "data3", "id4": "data4", "id5": null}

  wc_data_4:
    value: { 'Fn::Select': ['id4', { get_attr: [wait_condition, data] }] }
    # this would return "data4"
```

OS::Heat::SwiftSignal is implemented by creating an Object Storage API temporary URL which is populated with signal data with an HTTP PUT. The orchestration service will poll this object until the signal data is available. Object versioning is used to store multiple signals.

Sending the signal is a simple HTTP request, as with this example using curl:

```
curl -i -X PUT '<object URL>' --data-binary '<json containing signal
data>'
```

The above template example only needs to have the `type` changed to the swift signal resources:

```
resources:
  signal:
    type: OS::Heat::SwiftSignal
    properties:
      handle: {get_resource: wait_handle}
      timeout: 300

  signal_handle:
    type: OS::Heat::SwiftSignalHandle
  # ...
```

The decision to use *OS::Heat::WaitCondition* or *OS::Heat::SwiftSignal* will depend on a few factors:

- *OS::Heat::SwiftSignal* depends on the availability of an Object Storage API
- *OS::Heat::WaitCondition* depends on whether the orchestration service has been configured with a dedicated stack domain (which may depend on the availability of an Identity V3 API).
- The preference to protect signal URLs with token authentication or a secret webhook URL.

Software config resources

Boot configuration scripts can also be managed as their own resources. This allows configuration to be defined once and run on multiple server resources. These software-config resources are stored and retrieved via dedicated calls to the Orchestration API. It is not possible to modify the contents of an existing software-config resource, so a stack-update which changes any existing software-config resource will result in API calls to create a new config and delete the old one.

The resource *OS::Heat::SoftwareConfig* is used for storing configs represented by text scripts, for example:

```
resources:
  boot_script:
    type: OS::Heat::SoftwareConfig
    properties:
      group: ungrouped
      config: |
        #!/bin/bash
```

```
        echo "Running boot script"
        # ...

  server_with_boot_script:
    type: OS::Nova::Server
    properties:
      # flavor, image etc
      user_data_format: SOFTWARE_CONFIG
      user_data: {get_resource: boot_script}
```

The resource *OS::Heat::CloudConfig* allows Cloud-init cloud-config to be represented as template YAML rather than a block string. This allows intrinsic functions to be included when building the cloud-config. This also ensures that the cloud-config is valid YAML, although no further checks for valid cloud-config are done.

```
parameters:
  file_content:
    type: string
    description: The contents of the file /tmp/file

resources:
  boot_config:
    type: OS::Heat::CloudConfig
    properties:
      cloud_config:
        write_files:
        - path: /tmp/file
          content: {get_param: file_content}

  server_with_cloud_config:
    type: OS::Nova::Server
    properties:
      # flavor, image etc
      user_data_format: SOFTWARE_CONFIG
      user_data: {get_resource: boot_config}
```

The resource *OS::Heat::MultipartMime* allows multiple *OS::Heat::CloudConfig* resources to be combined into a single Cloud-init multi-part message:

```
parameters:
  file_content:
    type: string
    description: The contents of the file /tmp/file

  other_config:
```

```
    type: string
    description: The ID of a software-config resource created elsewhere

resources:
  boot_config:
    type: OS::Heat::CloudConfig
    properties:
      cloud_config:
        write_files:
        - path: /tmp/file
          content: {get_param: file_content}

  boot_script:
    type: OS::Heat::SoftwareConfig
    properties:
      group: ungrouped
      config: |
        #!/bin/bash
        echo "Running boot script"
        # ...

  server_init:
    type: OS::Heat::MultipartMime
    properties:
      parts:
      - config: {get_resource: boot_config}
      - config: {get_resource: boot_script}
      - config: {get_param: other_config}

  server:
    type: OS::Nova::Server
    properties:
      # flavor, image etc
      user_data_format: SOFTWARE_CONFIG
      user_data: {get_resource: server_init}
```

Software deployment resources

There are many situations where it is not desirable to replace the server whenever there is a configuration change. The *OS::Heat::SoftwareDeployment* resource allows any number of software configurations to be added or removed from a server throughout its life-cycle.

Building custom image for software deployments

OS::Heat::SoftwareConfig resources are used to store software configuration, and a *OS::Heat::SoftwareDeployment* resource is used to associate a config resource with one server. The `group` attribute on *OS::Heat::SoftwareConfig* specifies what tool will consume the config content.

OS::Heat::SoftwareConfig has the ability to define a schema of `inputs` and which the configuration script supports. Inputs are mapped to whatever concept the configuration tool has for assigning variables/parameters.

Likewise, `outputs` are mapped to the tool's capability to export structured data after configuration execution. For tools which do not support this, outputs can always be written to a known file path for the hook to read.

The *OS::Heat::SoftwareDeployment* resource allows values to be assigned to the config inputs, and the resource remains in an `IN_PROGRESS` state until the server signals to heat what (if any) output values were generated by the config script.

Custom image script

Each of the following examples requires that the servers be booted with a custom image. The following script uses diskimage-builder to create an image required in later examples:

```
# Clone the required repositories. Some of these are also available
# via pypi or as distro packages.
git clone https://git.openstack.org/openstack/diskimage-builder.git
git clone https://git.openstack.org/openstack/tripleo-image-elements.git
git clone https://git.openstack.org/openstack/heat-templates.git

# Required by diskimage-builder to discover element collections
export ELEMENTS_PATH=tripleo-image-elements/elements:heat-
templates/hot/software-config/elements

# The base operating system element(s) provided by the diskimage-builder
# elements collection. Other values which may work include:
# centos7, debian, opensuse, rhel, rhel7, or ubuntu
export BASE_ELEMENTS="fedora selinux-permissive"
# Install and configure the os-collect-config agent to poll the metadata
# server (heat service or zaqar message queue and so on) for configuration
# changes to execute
export AGENT_ELEMENTS="os-collect-config os-refresh-config os-apply-
config"
```

```
# heat-config installs an os-refresh-config script which will invoke the
# appropriate hook to perform configuration. The element heat-config-
script
# installs a hook to perform configuration with shell scripts
export DEPLOYMENT_BASE_ELEMENTS="heat-config heat-config-script"

# Install a hook for any other chosen configuration tool(s).
# Elements which install hooks include:
# heat-config-cfn-init, heat-config-puppet, or heat-config-salt
export DEPLOYMENT_TOOL=""

# The name of the qcow2 image to create, and the name of the image
# uploaded to the OpenStack image registry.
export IMAGE_NAME=fedora-software-config

# Create the image
diskimage-builder/bin/disk-image-create vm $BASE_ELEMENTS $AGENT_ELEMENTS
\
     $DEPLOYMENT_BASE_ELEMENTS $DEPLOYMENT_TOOL -o $IMAGE_NAME.qcow2

# Upload the image, assuming valid credentials are already sourced
glance image-create --disk-format qcow2 --container-format bare \
    --name $IMAGE_NAME < $IMAGE_NAME.qcow2
```

Configuring with scripts

The Custom image script already includes the `heat-config-script` element so the built image will already have the ability to configure using shell scripts.

Config inputs are mapped to shell environment variables. The script can communicate outputs to heat by writing to the `$heat_outputs_path.`*`output name`* file. See the following example for a script which expects inputs `foo`, `bar` and generates an output `result`.

```
resources:
  config:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      inputs:
      - name: foo
      - name: bar
      outputs:
      - name: result
      config: |
```

```
        #!/bin/sh -x
        echo "Writing to /tmp/$bar"
        echo $foo > /tmp/$bar
        echo -n "The file /tmp/$bar contains `cat /tmp/$bar` for server
$deploy_server_id during $deploy_action" > $heat_outputs_path.result
        echo "Written to /tmp/$bar"
        echo "Output to stderr" 1>&2

  deployment:
    type: OS::Heat::SoftwareDeployment
    properties:
      config:
        get_resource: config
      server:
        get_resource: server
      input_values:
        foo: fooooo
        bar: baaaaa

  server:
    type: OS::Nova::Server
    properties:
      # flavor, image etc
      user_data_format: SOFTWARE_CONFIG

outputs:
  result:
    value:
      get_attr: [deployment, result]
  stdout:
    value:
      get_attr: [deployment, deploy_stdout]
  stderr:
    value:
      get_attr: [deployment, deploy_stderr]
  status_code:
    value:
      get_attr: [deployment, deploy_status_code]
```

> **Note**
>
> A config resource can be associated with multiple deployment resources, and each deployment can specify the same or different values for the `server` and `input_values` properties.

As can be seen in the `outputs` section of the above template, the `result` config output value is available as an attribute on the `deployment` resource. Likewise the captured stdout, stderr and status_code are also available as attributes.

Configuring with os-apply-config

The agent toolchain of `os-collect-config`, `os-refresh-config` and `os-apply-config` can actually be used on their own to inject heat stack configuration data into a server running a custom image.

The custom image needs to have the following to use this approach:

- All software dependencies installed
- os-refresh-config scripts to be executed on configuration changes
- os-apply-config templates to transform the heat-provided config data into service configuration files

The projects tripleo-image-elements and tripleo-heat-templates demonstrate this approach.

Configuring with cfn-init

Likely the only reason to use the `cfn-init` hook is to migrate templates which contain AWS::CloudFormation::Init metadata without needing a complete rewrite of the config metadata. It is included here as it introduces a number of new concepts.

To use the `cfn-init` tool the `heat-config-cfn-init` element is required to be on the built image, so Custom image script needs to be modified with the following:

```
export DEPLOYMENT_TOOL="heat-config-cfn-init"
```

Configuration data which used to be included in the `AWS::CloudFormation::Init` section of resource metadata is instead moved to the `config` property of the config resource, as in the following example:

```
resources:

  config:
    type: OS::Heat::StructuredConfig
    properties:
      group: cfn-init
      inputs:
      - name: bar
      config:
        config:
          files:
            /tmp/foo:
```

```
            content:
              get_input: bar
            mode: '000644'

  deployment:
    type: OS::Heat::StructuredDeployment
    properties:
      name: 10_deployment
      signal_transport: NO_SIGNAL
      config:
        get_resource: config
      server:
        get_resource: server
      input_values:
        bar: baaaaa

  other_deployment:
    type: OS::Heat::StructuredDeployment
    properties:
      name: 20_other_deployment
      signal_transport: NO_SIGNAL
      config:
        get_resource: config
      server:
        get_resource: server
      input_values:
        bar: barmy

  server:
    type: OS::Nova::Server
    properties:
      image: {get_param: image}
      flavor: {get_param: flavor}
      key_name: {get_param: key_name}
      user_data_format: SOFTWARE_CONFIG
```

There are a number of things to note about this template example:

- *OS::Heat::StructuredConfig* is like *OS::Heat::SoftwareConfig* except that the `config` property contains structured YAML instead of text script. This is useful for a number of other configuration tools including ansible, salt and os-apply-config.
- `cfn-init` has no concept of inputs, so `{get_input: bar}` acts as a placeholder which gets replaced with the *OS::Heat::StructuredDeployment*

input_values value when the deployment resource is created.

- cfn-init has no concept of outputs, so specifying signal_transport: NO_SIGNAL will mean that the deployment resource will immediately go into the CREATED state instead of waiting for a completed signal from the server.
- The template has 2 deployment resources deploying the same config with different input_values. The order these are deployed in on the server is determined by sorting the values of the name property for each resource (10_deployment, 20_other_deployment)

Configuring with puppet

The puppet hook makes it possible to write configuration as puppet manifests which are deployed and run in a masterless environment.

To specify configuration as puppet manifests the heat-config-puppet element is required to be on the built image, so Custom image script needs to be modified with the following:

```
export DEPLOYMENT_TOOL="heat-config-puppet"

resources:

  config:
    type: OS::Heat::SoftwareConfig
    properties:
      group: puppet
      inputs:
      - name: foo
      - name: bar
      outputs:
      - name: result
      config:
        get_file: example-puppet-manifest.pp

  deployment:
    type: OS::Heat::SoftwareDeployment
    properties:
      config:
        get_resource: config
      server:
        get_resource: server
      input_values:
        foo: fooooo
        bar: baaaaa
```

```
  server:
    type: OS::Nova::Server
    properties:
      image: {get_param: image}
      flavor: {get_param: flavor}
      key_name: {get_param: key_name}
      user_data_format: SOFTWARE_CONFIG

outputs:
  result:
    value:
      get_attr: [deployment, result]
  stdout:
    value:
get_attr: [deployment, deploy_stdout]
```

This demonstrates the use of the `get_file` function, which will attach the contents of the file `example-puppet-manifest.pp`, containing:

```
file { 'barfile':
    ensure  => file,
    mode    => '0644',
    path    => '/tmp/$::bar',
    content => '$::foo',
}

file { 'output_result':
    ensure  => file,
    path    => '$::heat_outputs_path.result',
    mode    => '0644',
    content => 'The file /tmp/$::bar contains $::foo',
}
```

Environments

The environment affects the runtime behavior of a template. It provides a way to override the resource implementations and a mechanism to place parameters that the service needs.

To fully understand the runtime behavior you have to consider what plug-ins are installed on the cloud you're using.

Environment file format

The environment is a yaml text file that contains two main sections:

parameters
: A list of key/value pairs.

resource_registry
: Definition of custom resources.

Use the *-e* option of the **heat stack-create** command to create a stack using the environment defined in such a file.

You can also provide environment parameters as a list of key/value pairs using the *-P* option of the **heat stack-create** command.

In the following example the environment is read from the `my_env.yaml` file and an extra parameter is provided using the *-P* option:

```
$ heat stack-create my_stack -e my_env.yaml -P "param1=val1;param2=val2"
-f my_tmpl.yaml
```

Global and effective environments

The environment used for a stack is the combination of the environment you use with the template for the stack, and a global environment that is determined by your cloud operator. An entry in the user environment takes precedence over the global environment. OpenStack includes a default global environment, but your cloud operator can add additional environment entries.

The cloud operator can add to the global environment by putting environment files in a configurable directory wherever the Orchestration engine runs. The configuration variable is named `environment_dir` and is found in the `[DEFAULT]` section of `/etc/heat/heat.conf`. The default for that directory is `/etc/heat/environment.d`. Its contents are combined in whatever order the shell delivers them when the service starts up, which is the time when these files are read. If the `my_env.yaml` file from the example above had been put in the `environment_dir` then the user's command line could be this:

```
heat stack-create my_stack -P "some_parm=bla" -f my_tmpl.yaml
```

Global templates

A global template directory allows files to be pre-loaded in the global environment. A global template is determined by your cloud operator. An entry in the user template takes precedence over the global environment. OpenStack includes a

default global template, but your cloud operator can add additional template entries.

The cloud operator can add new global templates by putting template files in a configurable directory wherever the Orchestration engine runs. The configuration variable is named `template_dir` and is found in the `[DEFAULT]` section of `/etc/heat/heat.conf`. The default for that directory is `/etc/heat/templates`. Its contents are combined in whatever order the shell delivers them when the service starts up, which is the time when these files are read. If the `my_tmpl.yaml` file from the example below has been put in the `template_dir`, other templates which we used to create stacks could contain following way to include my_tmpl.yaml in it:

```
resourceA:
  type: {get_file: "my_tmpl.yaml"}
```

Usage examples

Define values for template arguments

You can define values for the template arguments in the `parameters` section of an environment file:

```
parameters:
  KeyName: heat_key
  InstanceType: m1.micro
  ImageId: F18-x86_64-cfntools
```

Define defaults to parameters

You can define default values for all template arguments in the `parameter_defaults` section of an environment file. These defaults are passed into all template resources:

```
parameter_defaults:
  KeyName: heat_key
```

Mapping resources

You can map one resource to another in the `resource_registry` section of an environment file. The resource you provide in this manner must have an identifier, and must reference either another resource's ID or the URL of an existing template file.

The following example maps a new `OS::Networking::FloatingIP` resource to an existing `OS::Nova::FloatingIP` resource:

```
resource_registry:
  "OS::Networking::FloatingIP": "OS::Nova::FloatingIP"
```

You can use wildcards to map multiple resources, for example to map all `OS::Neutron` resources to `OS::Network`:

```
resource_registry:
  "OS::Network*": "OS::Neutron*"
```

Override a resource with a custom resource

To create or override a resource with a custom resource, create a template file to define this resource, and provide the URL to the template file in the environment file:

```
resource_registry:
  "AWS::EC2::Instance": file:///path/to/my_instance.yaml
```

The supported URL schemes are `file`, `http` and `https`.

> **Note**
>
> The template file extension must be `.yaml` or `.template`, or it will not be treated as a custom template resource.

You can limit the usage of a custom resource to a specific resource of the template:

```
resource_registry:
  resources:
    my_db_server:
      "OS::DBInstance": file:///home/mine/all_my_cool_templates/db.yaml
```

Pause stack creation, update or deletion on a given resource

If you want to debug your stack as it's being created, updated or deleted, or if you want to run it in phases, you can set `pre-create`, `pre-update`, `pre-delete`, `post-create`, `post-update` and `post-delete` hooks in the `resources` section of `resource_registry`.

To set a hook, add either `hooks: $hook_name` (for example `hooks: pre-update`) to the resource's dictionary. You can also use a list (`hooks: [pre-create, pre-update]`) to stop on several actions.

You can combine hooks with other `resources` properties such as provider templates

or type mapping:

```
resource_registry:
  resources:
    my_server:
      "OS::DBInstance": file:///home/mine/all_my_cool_templates/db.yaml
      hooks: pre-create
    nested_stack:
      nested_resource:
        hooks: pre-update
      another_resource:
        hooks: [pre-create, pre-update]
```

When heat encounters a resource that has a hook, it pauses the resource action until the hook clears. Any resources that depend on the paused action wait as well. Non-dependent resources are created in parallel unless they have their own hooks.

It is possible to perform a wild card match using an asterisk (*) in the resource name. For example, the following entry pauses while creating `app_server` and `database_server`, but not `server` or `app_network`:

```
resource_registry:
  resources:
    "*_server":
      hooks: pre-create
```

Clear hooks by signaling the resource with `{unset_hook: $hook_name}` (for example `{unset_hook: pre-update}`).

Retrieving events

By default events are stored in the database and can be retrieved via the API. Using the environment, you can register an endpoint which will receive events produced by your stack, so that you don’t have to poll Heat.

You can specify endpoints using the `event_sinks` property:

```
event_sinks:
  - type: zaqar-queue
    target: myqueue
    ttl: 1200
```

Restrict update or replace of a given resource

If you want to restrict update or replace of a resource when your stack is being updated, you can set `restricted_actions` in the `resources` section of `resource_registry`.

To restrict update or replace, add `restricted_actions: update` or `restricted_actions: replace` to the resource dictionary. You can also use `[update, replace]` to restrict both actions.

You can combine restricted actions with other `resources` properties such as provider templates or type mapping or hooks:

```
resource_registry:
  resources:
    my_server:
      "OS::DBInstance": file:///home/mine/all_my_cool_templates/db.yaml
      restricted_actions: replace
      hooks: pre-create
    nested_stack:
      nested_resource:
        restricted_actions: update
      another_resource:
        restricted_actions: [update, replace]
```

It is possible to perform a wild card match using an asterisk (*) in the resource name. For example, the following entry restricts replace for `app_server` and `database_server`, but not `server` or `app_network`:

```
resource_registry:
  resources:
    "*_server":
      restricted_actions: replace
```

Template composition

When writing complex templates you are encouraged to break up your template into separate smaller templates. These can then be brought together using template resources. This is a mechanism to define a resource using a template, thus composing one logical stack with multiple templates.

Template resources provide a feature similar to the *AWS::CloudFormation::Stack* resource, but also provide a way to:

- Define new resource types and build your own resource library.
- Override the default behavior of existing resource types.

To achieve this:

- The Orchestration client gets the associated template files and passes them along in the `files` section of the `POST stacks/` API request.
- The environment in the Orchestration engine manages the mapping of resource type to template creation.
- The Orchestration engine translates template parameters into resource properties.

The following examples illustrate how you can use a custom template to define new types of resources. These examples use a custom template stored in a `my_nova.yaml` file

```
heat_template_version: 2015-04-30

parameters:
  key_name:
    type: string
    description: Name of a KeyPair

resources:
  server:
    type: OS::Nova::Server
    properties:
      key_name: {get_param: key_name}
      flavor: m1.small
      image: ubuntu-trusty-x86_64
```

Use the template filename as type

The following template defines the `my_nova.yaml` file as value for the `type` property of a resource

```
heat_template_version: 2015-04-30

resources:
  my_server:
    type: my_nova.yaml
    properties:
      key_name: my_key
```

The `key_name` argument of the `my_nova.yaml` template gets its value from the `key_name` property of the new template.

> **Note**
>
> The above reference to `my_nova.yaml` assumes it is in the same directory. You can use any of the following forms:
>
> - Relative path (`my_nova.yaml`)
> - Absolute path (`file:///home/user/templates/my_nova.yaml`)
> - Http URL (`http://example.com/templates/my_nova.yaml`)
> - Https URL (`https://example.com/templates/my_nova.yaml`)

To create the stack run:

```
$ heat stack-create -f main.yaml stack1
```

Define a new resource type

You can associate a name to the `my_nova.yaml` template in an environment file. If the name is already known by the Orchestration module then your new resource will override the default one.

In the following example a new `OS::Nova::Server` resource overrides the default resource of the same name.

An `env.yaml` environment file holds the definition of the new resource

```
resource_registry:
  "OS::Nova::Server": my_nova.yaml
```

> **Note**
>
> See *Environments* for more detail about environment files.

You can now use the new `OS::Nova::Server` in your new template

```
heat_template_version: 2015-04-30

resources:
  my_server:
    type: OS::Nova::Server
    properties:
      key_name: my_key
```

To create the stack run:

```
$ heat stack-create -f main.yaml -e env.yaml example-two
```

Get access to nested attributes

There are implicit attributes of a template resource. Accessing nested attributes requires `heat_template_version` 2014-10-16 or higher. These are accessible as follows

```
heat_template_version: 2015-04-30

resources:
  my_server:
    type: my_nova.yaml

outputs:
  test_out:
```

```
      value: {get_attr: my_server, resource.server, first_address}
```

Making your template resource more “transparent”

> **Note**
>
> Available since 2015.1 (Kilo).

If you wish to be able to return the ID of one of the inner resources instead of the nested stack’s identifier, you can add the special reserved output `OS::stack_id` to your template resource

```
heat_template_version: 2015-04-30

resources:
  server:
    type: OS::Nova::Server

outputs:
  OS::stack_id:
    value: {get_resource: server}
```

Now when you use `get_resource` from the outer template heat will use the nova server id and not the template resource identifier.

OpenStack Resource Types

OS::Aodh::Alarm

A resource that implements alarming service of Aodh.

A resource that allows for the setting alarms based on threshold evaluation for a collection of samples. Also, you can define actions to take if state of watched resource will be satisfied specified conditions. For example, it can watch for the memory consumption and when it reaches 70% on a given instance if the instance has been up for more than 10 min, some action will be called.

Required Properties

meter_name

Meter name watched by the alarm.

String value expected.

Updates cause replacement.

threshold

Threshold to evaluate against.

Number value expected.

Can be updated without replacement.

Optional Properties

alarm_actions

A list of URLs (webhooks) to invoke when state transitions to alarm.

List value expected.

Can be updated without replacement.

comparison_operator

Operator used to compare specified statistic with threshold.

String value expected.

Can be updated without replacement.

Allowed values: ge, gt, eq, ne, lt, le

description

Description for the alarm.

String value expected.

Can be updated without replacement.

enabled

True if alarm evaluation/actioning is enabled.

Boolean value expected.

Can be updated without replacement.

Defaults to “true”.

evaluation_periods

Number of periods to evaluate over.

Integer value expected.

Can be updated without replacement.

insufficient_data_actions

A list of URLs (webhooks) to invoke when state transitions to insufficient-data.

List value expected.

Can be updated without replacement.

matching_metadata

Meter should match this resource metadata (key=value) additionally to the meter_name.

Map value expected.

Can be updated without replacement.

Defaults to “{}”.

ok_actions

A list of URLs (webhooks) to invoke when state transitions to ok.

List value expected.

Can be updated without replacement.

period

Period (seconds) to evaluate over.

Integer value expected.

Can be updated without replacement.

query

> **Note**
>
> Available since 2015.1 (Kilo)

A list of query factors, each comparing a Sample attribute with a value. Implicitly combined with matching_metadata, if any.

List value expected.

Can be updated without replacement.

List contents:

*

Map value expected.

Can be updated without replacement.

Map properties:

field

Name of attribute to compare. Names of the form metadata.user_metadata.X or metadata.metering.X are equivalent to what you can address through matching_metadata; the former for Nova meters, the latter for all others. To see the attributes of your Samples, use `ceilometer –debug sample-list`.

String value expected.

Can be updated without replacement.

op

Comparison operator.

String value expected.

Can be updated without replacement.

Allowed values: le, ge, eq, lt, gt, ne

value

String value with which to compare.

String value expected.

Can be updated without replacement.

repeat_actions

False to trigger actions when the threshold is reached AND the alarm’s state has changed. By default, actions are called each time the threshold is reached.

Boolean value expected.

Can be updated without replacement.

Defaults to “true”.

severity

> **Note**
>
> Available since 5.0.0 (Liberty)

Severity of the alarm.

String value expected.

Can be updated without replacement.

Defaults to “low”.

Allowed values: low, moderate, critical

statistic

Meter statistic to evaluate.

String value expected.

Can be updated without replacement.

Allowed values: count, avg, sum, min, max

time_constraints

> **Note**
>
> Available since 5.0.0 (Liberty)

Describe time constraints for the alarm. Only evaluate the alarm if the time at evaluation is within this time constraint. Start point(s) of the constraint are specified with a cron expression, whereas its duration is given in seconds.

List value expected.

Updates cause replacement.

Defaults to “[]”.

List contents:

*

> Map value expected.
>
> Updates cause replacement.
>
> *Map properties:*

description

Description for the time constraint.

String value expected.

Updates cause replacement.

duration

Duration for the time constraint.

Integer value expected.

Updates cause replacement.

The value must be at least 0.

name

Name for the time constraint.

String value expected.

Updates cause replacement.

start

Start time for the time constraint. A CRON expression property.

String value expected.

Updates cause replacement.

Value must be of type cron_expression

timezone

Timezone for the time constraint (eg. ‘Taiwan/Taipei’, ‘Europe/Amsterdam’).

String value expected.

Updates cause replacement.

Value must be of type timezone

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Aodh::Alarm
    properties:
      alarm_actions: [Value, Value, ...]
      comparison_operator: String
      description: String
      enabled: Boolean
      evaluation_periods: Integer
      insufficient_data_actions: [Value, Value, ...]
      matching_metadata: {...}
      meter_name: String
      ok_actions: [Value, Value, ...]
      period: Integer
      query: [{"op": String, "value": String, "field": String}, {"op":
String, "value": String, "field": String}, ...]
      repeat_actions: Boolean
      severity: String
      statistic: String
      threshold: Number
      time_constraints: [{"duration": Integer, "name": String,
"description": String, "timezone": String, "start": String}, {"duration":
Integer, "name": String, "description": String, "timezone": String,
"start": String}, ...]
```

OS::Aodh::GnocchiAggregationByMetrics Alarm

> **Note**
>
> Available since 2015.1 (Kilo)

A resource that implements alarm with specified metrics.

A resource that implements alarm which allows to use specified by user metrics in metrics list.

Required Properties

metrics

A list of metric ids.

List value expected.

Can be updated without replacement.

threshold

Threshold to evaluate against.

Number value expected.

Can be updated without replacement.

Optional Properties

aggregation_method

The aggregation method to compare to the threshold.

String value expected.

Can be updated without replacement.

Allowed values: mean, sum, last, max, min, std, median, first, count

alarm_actions

A list of URLs (webhooks) to invoke when state transitions to alarm.

List value expected.

Can be updated without replacement.

comparison_operator

Operator used to compare specified statistic with threshold.

String value expected.

Can be updated without replacement.

Allowed values: ge, gt, eq, ne, lt, le

description

Description for the alarm.

String value expected.

Can be updated without replacement.

enabled

True if alarm evaluation/actioning is enabled.

Boolean value expected.

Can be updated without replacement.

Defaults to “true”.

evaluation_periods

Number of periods to evaluate over.

Integer value expected.

Can be updated without replacement.

granularity

The time range in seconds.

Integer value expected.

Can be updated without replacement.

insufficient_data_actions

A list of URLs (webhooks) to invoke when state transitions to insufficient-data.

List value expected.

Can be updated without replacement.

ok_actions

A list of URLs (webhooks) to invoke when state transitions to ok.

List value expected.

Can be updated without replacement.

repeat_actions

False to trigger actions when the threshold is reached AND the alarm's state has changed. By default, actions are called each time the threshold is reached.

Boolean value expected.

Can be updated without replacement.

Defaults to "true".

severity

> **Note**
>
> Available since 5.0.0 (Liberty)

Severity of the alarm.

String value expected.

Can be updated without replacement.

Defaults to "low".

Allowed values: low, moderate, critical

time_constraints

> **Note**
>
> Available since 5.0.0 (Liberty)

Describe time constraints for the alarm. Only evaluate the alarm if the time at evaluation is within this time constraint. Start point(s) of the constraint are specified with a cron expression, whereas its duration is given in seconds.

List value expected.

Updates cause replacement.

Defaults to "[]".

List contents:

*

Map value expected.

Updates cause replacement.

Map properties:

description

Description for the time constraint.

String value expected.

Updates cause replacement.

duration

Duration for the time constraint.

Integer value expected.

Updates cause replacement.

The value must be at least 0.

name

Name for the time constraint.

String value expected.

Updates cause replacement.

start

Start time for the time constraint. A CRON expression property.

String value expected.

Updates cause replacement.

Value must be of type cron_expression

timezone

> Timezone for the time constraint (eg. ‘Taiwan/Taipei’, ‘Europe/Amsterdam’).
>
> String value expected.
>
> Updates cause replacement.
>
> Value must be of type timezone

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Aodh::GnocchiAggregationByMetricsAlarm
    properties:
      aggregation_method: String
      alarm_actions: [Value, Value, ...]
      comparison_operator: String
      description: String
      enabled: Boolean
      evaluation_periods: Integer
      granularity: Integer
      insufficient_data_actions: [Value, Value, ...]
      metrics: [Value, Value, ...]
      ok_actions: [Value, Value, ...]
      repeat_actions: Boolean
      severity: String
      threshold: Number
      time_constraints: [{"duration": Integer, "name": String,
"description": String, "timezone": String, "start": String}, {"duration":
Integer, "name": String, "description": String, "timezone": String,
"start": String}, ...]
```

OS::Aodh::GnocchiAggregationByResourcesAlarm

> **Note**
>
> Available since 2015.1 (Kilo)

A resource that implements alarm as an aggregation of resources alarms.

A resource that implements alarm which uses aggregation of resources alarms with some condition. If state of a system is satisfied alarm condition, alarm is activated.

Required Properties

metric

Metric name watched by the alarm.

String value expected.

Can be updated without replacement.

query

The query to filter the metrics.

String value expected.

Can be updated without replacement.

resource_type

Resource type.

String value expected.

Can be updated without replacement.

threshold

Threshold to evaluate against.

Number value expected.

Can be updated without replacement.

Optional Properties

aggregation_method

The aggregation method to compare to the threshold.

String value expected.

Can be updated without replacement.

Allowed values: mean, sum, last, max, min, std, median, first, count

alarm_actions

A list of URLs (webhooks) to invoke when state transitions to alarm.

List value expected.

Can be updated without replacement.

comparison_operator

Operator used to compare specified statistic with threshold.

String value expected.

Can be updated without replacement.

Allowed values: ge, gt, eq, ne, lt, le

description

Description for the alarm.

String value expected.

Can be updated without replacement.

enabled

True if alarm evaluation/actioning is enabled.

Boolean value expected.

Can be updated without replacement.

Defaults to “true”.

evaluation_periods

Number of periods to evaluate over.

Integer value expected.

Can be updated without replacement.

granularity

The time range in seconds.

Integer value expected.

Can be updated without replacement.

insufficient_data_actions

A list of URLs (webhooks) to invoke when state transitions to insufficient-data.

List value expected.

Can be updated without replacement.

ok_actions

A list of URLs (webhooks) to invoke when state transitions to ok.

List value expected.

Can be updated without replacement.

repeat_actions

False to trigger actions when the threshold is reached AND the alarm's state has changed. By default, actions are called each time the threshold is reached.

Boolean value expected.

Can be updated without replacement.

Defaults to "true".

severity

> **Note**
>
> Available since 5.0.0 (Liberty)

Severity of the alarm.

String value expected.

Can be updated without replacement.

Defaults to "low".

Allowed values: low, moderate, critical

time_constraints

> **Note**
>
> Available since 5.0.0 (Liberty)

Describe time constraints for the alarm. Only evaluate the alarm if the time at evaluation is within this time constraint. Start point(s) of the constraint are specified with a cron expression, whereas its duration is given in seconds.

List value expected.

Updates cause replacement.

Defaults to “[]”.

List contents:

*

Map value expected.

Updates cause replacement.

Map properties:

description

Description for the time constraint.

String value expected.

Updates cause replacement.

duration

Duration for the time constraint.

Integer value expected.

Updates cause replacement.

The value must be at least 0.

name

Name for the time constraint.

String value expected.

Updates cause replacement.

start

Start time for the time constraint. A CRON expression property.

String value expected.

Updates cause replacement.

Value must be of type cron_expression

timezone

> Timezone for the time constraint (eg. ‘Taiwan/Taipei’, ‘Europe/Amsterdam’).
>
> String value expected.
>
> Updates cause replacement.
>
> Value must be of type timezone

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Aodh::GnocchiAggregationByResourcesAlarm
    properties:
      aggregation_method: String
      alarm_actions: [Value, Value, ...]
      comparison_operator: String
      description: String
      enabled: Boolean
      evaluation_periods: Integer
      granularity: Integer
      insufficient_data_actions: [Value, Value, ...]
      metric: String
      ok_actions: [Value, Value, ...]
      query: String
      repeat_actions: Boolean
      resource_type: String
      severity: String
      threshold: Number
      time_constraints: [{"duration": Integer, "name": String,
"description": String, "timezone": String, "start": String}, {"duration":
Integer, "name": String, "description": String, "timezone": String,
"start": String}, ...]
```

OS::Aodh::GnocchiResourcesAlarm

> **Note**
>
> Available since 2015.1 (Kilo)

A resource allowing for the watch of some specified resource.

An alarm that evaluates threshold based on some metric for the specified resource.

Required Properties

metric

Metric name watched by the alarm.

String value expected.

Can be updated without replacement.

resource_id

Id of a resource.

String value expected.

Can be updated without replacement.

resource_type

Resource type.

String value expected.

Can be updated without replacement.

threshold

Threshold to evaluate against.

Number value expected.

Can be updated without replacement.

Optional Properties

aggregation_method

The aggregation method to compare to the threshold.

String value expected.

Can be updated without replacement.

Allowed values: mean, sum, last, max, min, std, median, first, count

alarm_actions

A list of URLs (webhooks) to invoke when state transitions to alarm.

List value expected.

Can be updated without replacement.

comparison_operator

Operator used to compare specified statistic with threshold.

String value expected.

Can be updated without replacement.

Allowed values: ge, gt, eq, ne, lt, le

description

Description for the alarm.

String value expected.

Can be updated without replacement.

enabled

True if alarm evaluation/actioning is enabled.

Boolean value expected.

Can be updated without replacement.

Defaults to “true”.

evaluation_periods

Number of periods to evaluate over.

Integer value expected.

Can be updated without replacement.

granularity

The time range in seconds.

Integer value expected.

Can be updated without replacement.

insufficient_data_actions

A list of URLs (webhooks) to invoke when state transitions to insufficient-data.

List value expected.

Can be updated without replacement.

ok_actions

A list of URLs (webhooks) to invoke when state transitions to ok.

List value expected.

Can be updated without replacement.

repeat_actions

False to trigger actions when the threshold is reached AND the alarm's state has changed. By default, actions are called each time the threshold is reached.

Boolean value expected.

Can be updated without replacement.

Defaults to "true".

severity

> **Note**
>
> Available since 5.0.0 (Liberty)

Severity of the alarm.

String value expected.

Can be updated without replacement.

Defaults to "low".

Allowed values: low, moderate, critical

time_constraints

> **Note**
>
> Available since 5.0.0 (Liberty)

Describe time constraints for the alarm. Only evaluate the alarm if the time at evaluation is within this time constraint. Start point(s) of the constraint are specified with a cron expression, whereas its duration is given in seconds.

List value expected.

Updates cause replacement.

Defaults to "[]".

List contents:

*

Map value expected.

Updates cause replacement.

Map properties:

description

Description for the time constraint.

String value expected.

Updates cause replacement.

duration

Duration for the time constraint.

Integer value expected.

Updates cause replacement.

The value must be at least 0.

name

Name for the time constraint.

String value expected.

Updates cause replacement.

start

Start time for the time constraint. A CRON expression property.

String value expected.

Updates cause replacement.

Value must be of type cron_expression

timezone

> Timezone for the time constraint (eg. ‘Taiwan/Taipei’, ‘Europe/Amsterdam’).
>
> String value expected.
>
> Updates cause replacement.
>
> Value must be of type timezone

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Aodh::GnocchiResourcesAlarm
    properties:
      aggregation_method: String
      alarm_actions: [Value, Value, ...]
      comparison_operator: String
      description: String
      enabled: Boolean
      evaluation_periods: Integer
      granularity: Integer
      insufficient_data_actions: [Value, Value, ...]
      metric: String
      ok_actions: [Value, Value, ...]
      repeat_actions: Boolean
      resource_id: String
      resource_type: String
      severity: String
      threshold: Number
      time_constraints: [{"duration": Integer, "name": String,
"description": String, "timezone": String, "start": String}, {"duration":
Integer, "name": String, "description": String, "timezone": String,
"start": String}, ...]
```

OS::Barbican::CertificateContainer

> **Note**
>
> Available since 6.0.0 (Mitaka)

A resource for creating barbican certificate container.

A certificate container is used for storing the secrets that are relevant to certificates.

Optional Properties

certificate_ref

Reference to certificate.

String value expected.

Updates cause replacement.

Value must be of type barbican.secret

intermediates_ref

Reference to intermediates.

String value expected.

Updates cause replacement.

Value must be of type barbican.secret

name

Human-readable name for the container.

String value expected.

Updates cause replacement.

private_key_passphrase_ref

Reference to private key passphrase.

String value expected.

Updates cause replacement.

Value must be of type barbican.secret

private_key_ref

Reference to private key.

String value expected.

Updates cause replacement.

Value must be of type barbican.secret

Attributes

consumers

The URIs to container consumers.

container_ref

The URI to the container.

secret_refs

The URIs to secrets stored in container.

show

Detailed information about resource.

status

The status of the container.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Barbican::CertificateContainer
    properties:
      certificate_ref: String
      intermediates_ref: String
      name: String
      private_key_passphrase_ref: String
      private_key_ref: String
```

OS::Barbican::GenericContainer

> **Note**
>
> Available since 6.0.0 (Mitaka)

A resource for creating Barbican generic container.

A generic container is used for any type of secret that a user may wish to aggregate. There are no restrictions on the amount of secrets that can be held within this container.

Optional Properties

name

Human-readable name for the container.

String value expected.

Updates cause replacement.

secrets

References to secrets that will be stored in container.

List value expected.

Updates cause replacement.

List contents:

*

> Map value expected.
>
> Updates cause replacement.
>
> *Map properties:*
>
> name
>
> > Name of the secret.
> >
> > String value expected.
> >
> > Updates cause replacement.

ref

Reference to the secret.

String value expected.

Updates cause replacement.

Value must be of type barbican.secret

Attributes

consumers

The URIs to container consumers.

container_ref

The URI to the container.

secret_refs

The URIs to secrets stored in container.

show

Detailed information about resource.

status

The status of the container.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Barbican::GenericContainer
    properties:
      name: String
      secrets: [{"name": String, "ref": String}, {"name": String, "ref":
String}, ...]
```

OS::Barbican::Order

> **Note**
>
> Available since 2014.2 (Juno)

A resource allowing for the generation secret material by Barbican.

The resource allows to generate some secret material. It can be, for example, some key or certificate. The order encapsulates the workflow and history for the creation of a secret. The time to generate a secret can vary depending on the type of secret.

Required Properties

type

> **Note**
>
> Available since 5.0.0 (Liberty)

The type of the order.

String value expected.

Updates cause replacement.

Allowed values: key, asymmetric, certificate

Optional Properties

algorithm

The algorithm type used to generate the secret. Required for key and asymmetric types of order.

String value expected.

Updates cause replacement.

bit_length

The bit-length of the secret. Required for key and asymmetric types of order.

Integer value expected.

Updates cause replacement.

ca_id

> **Note**
>
> Available since 5.0.0 (Liberty)

The identifier of the CA to use.

String value expected.

Updates cause replacement.

expiration

The expiration date for the secret in ISO-8601 format.

String value expected.

Updates cause replacement.

Value must be of type expiration

mode

The type/mode of the algorithm associated with the secret information.

String value expected.

Updates cause replacement.

name

Human readable name for the secret.

String value expected.

Updates cause replacement.

pass_phrase

> **Note**
>
> Available since 5.0.0 (Liberty)

The passphrase the created key. Can be set only for asymmetric type of order.

String value expected.

Updates cause replacement.

payload_content_type

The type/format the secret data is provided in.

String value expected.

Updates cause replacement.

profile

> **Note**
>
> Available since 5.0.0 (Liberty)

The profile of certificate to use.

String value expected.

Updates cause replacement.

request_data

> **Note**
>
> Available since 5.0.0 (Liberty)

The content of the CSR. Only for certificate orders.

String value expected.

Updates cause replacement.

request_type

> **Note**
>
> Available since 5.0.0 (Liberty)

The type of the certificate request.

String value expected.

Updates cause replacement.

Allowed values: stored-key, simple-cmc, custom

source_container_ref

> **Note**
>
> Available since 5.0.0 (Liberty)

The source of certificate request.

String value expected.

Updates cause replacement.

Value must be of type barbican.container

subject_dn

> **Note**
>
> Available since 5.0.0 (Liberty)

The subject of the certificate request.

String value expected.

Updates cause replacement.

Attributes

certificate

> **Note**
>
> Available since 5.0.0 (Liberty)

The payload of the created certificate, if available.

container_ref

> **Note**
>
> Available since 5.0.0 (Liberty)

The URI to the created container.

intermediates

> **Note**
>
> Available since 5.0.0 (Liberty)

The payload of the created intermediates, if available.

order_ref

The URI to the order.

private_key

> **Note**
>
> Available since 5.0.0 (Liberty)

The payload of the created private key, if available.

public_key

> **Note**
>
> Available since 5.0.0 (Liberty)

The payload of the created public key, if available.

secret_ref

The URI to the created secret.

show

Detailed information about resource.

status

The status of the order.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Barbican::Order
    properties:
      algorithm: String
      bit_length: Integer
      ca_id: String
      expiration: String
      mode: String
      name: String
      pass_phrase: String
      payload_content_type: String
      profile: String
      request_data: String
      request_type: String
      source_container_ref: String
      subject_dn: String
      type: String
```

OS::Barbican::RSAContainer

> **Note**
>
> Available since 6.0.0 (Mitaka)

A resource for creating barbican RSA container.

An RSA container is used for storing RSA public keys, private keys, and private key pass phrases.

Optional Properties

name

Human-readable name for the container.

String value expected.

Updates cause replacement.

private_key_passphrase_ref

Reference to private key passphrase.

String value expected.

Updates cause replacement.

Value must be of type barbican.secret

private_key_ref

Reference to private key.

String value expected.

Updates cause replacement.

Value must be of type barbican.secret

public_key_ref

Reference to public key.

String value expected.

Updates cause replacement.

Value must be of type barbican.secret

Attributes

consumers

The URIs to container consumers.

container_ref

The URI to the container.

secret_refs

The URIs to secrets stored in container.

show

Detailed information about resource.

status

The status of the container.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Barbican::RSAContainer
    properties:
      name: String
      private_key_passphrase_ref: String
      private_key_ref: String
      public_key_ref: String
```

OS::Barbican::Secret

> **Note**
>
> Available since 2014.2 (Juno)

The resource provides access to the secret/keying stored material.

A secret is a singular item that stored within Barbican. A secret is anything you want it to be; however, the formal use case is a key that you wish to store away from prying eyes. Secret may include private keys, passwords and so on.

Optional Properties

algorithm

The algorithm type used to generate the secret.

String value expected.

Updates cause replacement.

bit_length

The bit-length of the secret.

Integer value expected.

Updates cause replacement.

The value must be at least 0.

expiration

The expiration date for the secret in ISO-8601 format.

String value expected.

Updates cause replacement.

Value must be of type expiration

mode

The type/mode of the algorithm associated with the secret information.

String value expected.

Updates cause replacement.

name

Human readable name for the secret.

String value expected.

Updates cause replacement.

payload

The unencrypted plain text of the secret.

String value expected.

Updates cause replacement.

payload_content_encoding

The encoding format used to provide the payload data.

String value expected.

Updates cause replacement.

Allowed values: base64

payload_content_type

The type/format the secret data is provided in.

String value expected.

Updates cause replacement.

Allowed values: text/plain, application/octet-stream

secret_type

> **Note**
>
> Available since 5.0.0 (Liberty)

The type of the secret.

String value expected.

Updates cause replacement.

Defaults to “opaque”.

Allowed values: symmetric, public, private, certificate, passphrase, opaque

Attributes

decrypted_payload

The decrypted secret payload.

show

Detailed information about resource.

status

The status of the secret.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Barbican::Secret
    properties:
      algorithm: String
      bit_length: Integer
      expiration: String
      mode: String
      name: String
      payload: String
      payload_content_encoding: String
      payload_content_type: String
      secret_type: String
```

OS::Cinder::EncryptedVolumeType

> **Note**
>
> Available since 5.0.0 (Liberty)

A resource for encrypting a cinder volume type.

A Volume Encryption Type is a collection of settings used to conduct encryption for a specific volume type.

Note that default cinder security policy usage of this resource is limited to being used by administrators only.

Required Properties

provider

The class that provides encryption support. For example, nova.volume.encryptors.luks.LuksEncryptor.

String value expected.

Can be updated without replacement.

volume_type

Name or id of volume type (OS::Cinder::VolumeType).

String value expected.

Updates cause replacement.

Value must be of type cinder.vtype

Optional Properties

cipher

The encryption algorithm or mode. For example, aes-xts-plain64.

String value expected.

Can be updated without replacement.

Allowed values: aes-xts-plain64, aes-cbc-essiv

control_location

Notional service where encryption is performed For example, front-end. For Nova.

String value expected.

Can be updated without replacement.

Defaults to “front-end”.

Allowed values: front-end, back-end

key_size

Size of encryption key, in bits. For example, 128 or 256.

Integer value expected.

Can be updated without replacement.

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Cinder::EncryptedVolumeType
    properties:
      cipher: String
      control_location: String
      key_size: Integer
      provider: String
      volume_type: String
```

OS::Cinder::Volume

A resource that implements Cinder volumes.

Cinder volume is a storage in the form of block devices. It can be used, for example, for providing storage to instance. Volume supports creation from snapshot, backup or image. Also volume can be created only by size.

Optional Properties

availability_zone

The availability zone in which the volume will be created.

String value expected.

Updates cause replacement.

backup_id

If specified, the backup to create the volume from.

String value expected.

Can be updated without replacement.

Value must be of type cinder.backup

description

A description of the volume.

String value expected.

Can be updated without replacement.

image

If specified, the name or ID of the image to create the volume from.

String value expected.

Updates cause replacement.

Value must be of type glance.image

metadata

Key/value pairs to associate with the volume.

Map value expected.

Can be updated without replacement.

multiattach

> **Note**
>
> Available since 6.0.0 (Mitaka)

Whether allow the volume to be attached more than once.

Boolean value expected.

Updates cause replacement.

name

A name used to distinguish the volume.

String value expected.

Can be updated without replacement.

read_only

> **Note**
>
> Available since 5.0.0 (Liberty)

Enables or disables read-only access mode of volume.

Boolean value expected.

Can be updated without replacement.

scheduler_hints

> **Note**
>
> Available since 2015.1 (Kilo)

Arbitrary key-value pairs specified by the client to help the Cinder scheduler creating a volume.

Map value expected.

Updates cause replacement.

size

The size of the volume in GB. On update only increase in size is supported. This property is required unless property backup_id or source_volid or snapshot_id is specified.

Integer value expected.

Can be updated without replacement.

The value must be at least 1.

snapshot_id

If specified, the snapshot to create the volume from.

String value expected.

Updates cause replacement.

Value must be of type cinder.snapshot

source_volid

If specified, the volume to use as source.

String value expected.

Updates cause replacement.

Value must be of type cinder.volume

volume_type

If specified, the type of volume to use, mapping to a specific backend.

String value expected.

Can be updated without replacement.

Value must be of type cinder.vtype

Attributes

attachments

The list of attachments of the volume.

availability_zone

The availability zone in which the volume is located.

bootable

Boolean indicating if the volume can be booted or not.

created_at

The timestamp indicating volume creation.

display_description

Description of the volume.

display_name

Name of the volume.

encrypted

Boolean indicating if the volume is encrypted or not.

metadata

Key/value pairs associated with the volume.

metadata_values

Key/value pairs associated with the volume in raw dict form.

multiattach

> **Note**
>
> Available since 6.0.0 (Mitaka)

Boolean indicating whether allow the volume to be attached more than once.

show

Detailed information about resource.

size

The size of the volume in GB.

snapshot_id

The snapshot the volume was created from, if any.

source_volid

The volume used as source, if any.

status

The current status of the volume.

volume_type

The type of the volume mapping to a backend, if any.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Cinder::Volume
    properties:
      availability_zone: String
      backup_id: String
      description: String
      image: String
      metadata: {...}
      multiattach: Boolean
      name: String
      read_only: Boolean
      scheduler_hints: {...}
```

```
size: Integer
snapshot_id: String
source_volid: String
volume_type: String
```

OS::Cinder::VolumeAttachment

Resource for associating volume to instance.

Resource for associating existing volume to instance. Also, the location where the volume is exposed on the instance can be specified.

Required Properties

instance_uuid

The ID of the server to which the volume attaches.

String value expected.

Can be updated without replacement.

volume_id

The ID of the volume to be attached.

String value expected.

Can be updated without replacement.

Value must be of type cinder.volume

Optional Properties

mountpoint

The location where the volume is exposed on the instance. This assignment may not be honored and it is advised that the path /dev/disk/by-id/virtio-<VolumeId> be used instead.

String value expected.

Can be updated without replacement.

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Cinder::VolumeAttachment
    properties:
      instance_uuid: String
      mountpoint: String
      volume_id: String
```

OS::Cinder::VolumeType

> **Note**
>
> Available since 2015.1 (Kilo)

A resource for creating cinder volume types.

Volume type resource allows to define, whether volume, which will be use this type, will public and which projects are allowed to work with it. Also, there can be some user-defined metadata.

Note that default cinder security policy usage of this resource is limited to being used by administrators only.

Required Properties

name

Name of the volume type.

String value expected.

Can be updated without replacement.

Optional Properties

description

> **Note**
>
> Available since 5.0.0 (Liberty)

Description of the volume type.

String value expected.

Can be updated without replacement.

is_public

> **Note**
>
> Available since 5.0.0 (Liberty)

Whether the volume type is accessible to the public.

Boolean value expected.

Can be updated without replacement.

Defaults to “True”.

metadata

The extra specs key and value pairs of the volume type.

Map value expected.

Can be updated without replacement.

projects

> **Note**
>
> Available since 5.0.0 (Liberty)

Projects to add volume type access to. NOTE: This property is only supported since Cinder API V2.

List value expected.

Can be updated without replacement.

Defaults to “[]”.

List contents:

*

> String value expected.
>
> Can be updated without replacement.
>
> Value must be of type keystone.project

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Cinder::VolumeType
    properties:
      description: String
      is_public: Boolean
      metadata: {...}
      name: String
      projects: [String, String, ...]
```

OS::Designate::Domain

> **Note**
>
> Available since 5.0.0 (Liberty)

Heat Template Resource for Designate Domain.

Designate provides DNS-as-a-Service services for OpenStack. So, domain is a realm with an identification string, unique in DNS.

Required Properties

email

Domain email.

String value expected.

Can be updated without replacement.

name

Domain name.

String value expected.

Updates cause replacement.

The length must be no greater than 255.

Optional Properties

description

Description of domain.

String value expected.

Can be updated without replacement.

The length must be no greater than 160.

ttl

Time To Live (Seconds).

Integer value expected.

Can be updated without replacement.

The value must be in the range 1 to 2147483647.

Attributes

serial

DNS domain serial.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Designate::Domain
    properties:
      description: String
      email: String
      name: String
      ttl: Integer
```

OS::Designate::Record

> **Note**
>
> Available since 5.0.0 (Liberty)

Heat Template Resource for Designate Record.

Designate provides DNS-as-a-Service services for OpenStack. Record is storage unit in DNS. So, DNS name server is a server that stores the DNS records for a domain. Each record has a type and type-specific data.

Required Properties

data

DNS record data, varies based on the type of record. For more details, please refer rfc 1035.

String value expected.

Can be updated without replacement.

domain

DNS Domain id or name.

String value expected.

Updates cause replacement.

Value must be of type designate.domain

name

Record name.

String value expected.

Updates cause replacement.

The length must be no greater than 255.

type

DNS Record type.

String value expected.

Can be updated without replacement.

Allowed values: A, AAAA, CNAME, MX, SRV, TXT, SPF, NS, PTR, SSHFP, SOA

Optional Properties

description

Description of record.

String value expected.

Can be updated without replacement.

The length must be no greater than 160.

priority

DNS record priority. It is considered only for MX and SRV types, otherwise, it is ignored.

Integer value expected.

Can be updated without replacement.

The value must be in the range 0 to 65536.

ttl

Time To Live (Seconds).

Integer value expected.

Can be updated without replacement.

The value must be in the range 1 to 2147483647.

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Designate::Record
    properties:
      data: String
      description: String
      domain: String
      name: String
```

```
priority: Integer
ttl: Integer
type: String
```

OS::Glance::Image

> **Note**
>
> Available since 2014.2 (Juno)

A resource managing images in Glance.

A resource provides managing images that are meant to be used with other services.

Required Properties

container_format

Container format of image.

String value expected.

Updates cause replacement.

Allowed values: ami, ari, aki, bare, ova, ovf

disk_format

Disk format of image.

String value expected.

Updates cause replacement.

Allowed values: ami, ari, aki, vhd, vmdk, raw, qcow2, vdi, iso

location

URL where the data for this image already resides. For example, if the image data is stored in swift, you could specify “swift://example.com/container/obj”.

String value expected.

Updates cause replacement.

Optional Properties

id

The image ID. Glance will generate a UUID if not specified.

String value expected.

Updates cause replacement.

is_public

Scope of image accessibility. Public or private. Default value is False means private. Note: The policy setting of glance allows only users with admin roles to create public image by default.

Boolean value expected.

Updates cause replacement.

Defaults to “False”.

min_disk

Amount of disk space (in GB) required to boot image. Default value is 0 if not specified and means no limit on the disk size.

Integer value expected.

Updates cause replacement.

Defaults to “0”.

The value must be at least 0.

min_ram

Amount of ram (in MB) required to boot image. Default value is 0 if not specified and means no limit on the ram size.

Integer value expected.

Updates cause replacement.

Defaults to “0”.

The value must be at least 0.

name

Name for the image. The name of an image is not unique to a Image Service node.

String value expected.

Updates cause replacement.

protected

Whether the image can be deleted. If the value is True, the image is protected and cannot be deleted.

Boolean value expected.

Updates cause replacement.

Defaults to “False”.

tags

> **Note**
>
> Available since 7.0.0 (Newton)

List of image tags.

List value expected.

Can be updated without replacement.

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Glance::Image
    properties:
      container_format: String
      disk_format: String
      id: String
      is_public: Boolean
      location: String
      min_disk: Integer
      min_ram: Integer
      name: String
      protected: Boolean
      tags: [Value, Value, ...]
```

OS::Heat::AccessPolicy

Resource for defining which resources can be accessed by users.

NOTE: Now this resource is actually associated with an AWS user resource, not any OS:: resource though it is registered under the OS namespace below.

Resource for defining resources that users are allowed to access by the DescribeStackResource API.

Required Properties

AllowedResources

Resources that users are allowed to access by the DescribeStackResource API.

List value expected.

Updates cause replacement.

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Heat::AccessPolicy
    properties:
      AllowedResources: [Value, Value, ...]
```

OS::Heat::AutoScalingGroup

> **Note**
>
> Available since 2014.1 (Icehouse)

An autoscaling group that can scale arbitrary resources.

An autoscaling group allows the creation of a desired count of similar resources, which are defined with the resource property in HOT format. If there is a need to

create many of the same resources (e.g. one hundred sets of Server, WaitCondition and WaitConditionHandle or even Neutron Nets), AutoScalingGroup is a convenient and easy way to do that.

Required Properties

max_size

Maximum number of resources in the group.

Integer value expected.

Can be updated without replacement.

The value must be at least 0.

min_size

Minimum number of resources in the group.

Integer value expected.

Can be updated without replacement.

The value must be at least 0.

resource

Resource definition for the resources in the group, in HOT format. The value of this property is the definition of a resource just as if it had been declared in the template itself.

Map value expected.

Can be updated without replacement.

Optional Properties

cooldown

Cooldown period, in seconds.

Integer value expected.

Can be updated without replacement.

desired_capacity

Desired initial number of resources.

Integer value expected.

Can be updated without replacement.

rolling_updates

Policy for rolling updates for this scaling group.

Map value expected.

Can be updated without replacement.

Defaults to “{‘min_in_service’: 0, ‘pause_time’: 0, ‘max_batch_size’: 1}”.

Map properties:

max_batch_size

The maximum number of resources to replace at once.

Integer value expected.

Can be updated without replacement.

Defaults to “1”.

The value must be at least 1.

min_in_service

The minimum number of resources in service while rolling updates are being executed.

Integer value expected.

Can be updated without replacement.

Defaults to “0”.

The value must be at least 0.

pause_time

The number of seconds to wait between batches of updates.

Number value expected.

Can be updated without replacement.

Defaults to “0”.

The value must be at least 0.

Attributes

current_size

> **Note**
>
> Available since 2015.1 (Kilo)

The current size of AutoscalingResourceGroup.

outputs

> **Note**
>
> Available since 2014.2 (Juno)

A map of resource names to the specified attribute of each individual resource that is part of the AutoScalingGroup. This map specifies output parameters that are available once the AutoScalingGroup has been instantiated.

outputs_list

> **Note**
>
> Available since 2014.2 (Juno)

A list of the specified attribute of each individual resource that is part of the AutoScalingGroup. This list of attributes is available as an output once the AutoScalingGroup has been instantiated.

refs

> **Note**
>
> Available since 7.0.0 (Newton)

A list of resource IDs for the resources in the group.

refs_map

> **Note**
>
> Available since 7.0.0 (Newton)

A map of resource names to IDs for the resources in the group.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Heat::AutoScalingGroup
    properties:
      cooldown: Integer
      desired_capacity: Integer
      max_size: Integer
      min_size: Integer
      resource: {...}
      rolling_updates: {"min_in_service": Integer, "pause_time": Number,
"max_batch_size": Integer}
```

OS::Heat::CloudConfig

> **Note**
>
> Available since 2014.1 (Icehouse)

A configuration resource for representing cloud-init cloud-config.

This resource allows cloud-config YAML to be defined and stored by the config API. Any intrinsic functions called in the config will be resolved before storing the result.

This resource will generally be referenced by OS::Nova::Server user_data, or OS::Heat::MultipartMime parts config. Since cloud-config is boot-only configuration, any changes to the definition will result in the replacement of all servers which reference it.

Optional Properties

cloud_config

Map representing the cloud-config data structure which will be formatted as YAML.

Map value expected.

Updates cause replacement.

Attributes

config

The config value of the software config.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Heat::CloudConfig
    properties:
      cloud_config: {...}
```

OS::Heat::InstanceGroup

An instance group that can scale arbitrary instances.

A resource allowing for the creating number of defined with AWS::AutoScaling::LaunchConfiguration instances. Allows to associate scaled resources with loadbalancer resources.

Required Properties

AvailabilityZones

Not Implemented.

List value expected.

Updates cause replacement.

LaunchConfigurationName

The reference to a LaunchConfiguration resource.

String value expected.

Can be updated without replacement.

Size

Desired number of instances.

Integer value expected.

Can be updated without replacement.

Optional Properties

LoadBalancerNames

List of LoadBalancer resources.

List value expected.

Updates cause replacement.

Tags

Tags to attach to this group.

List value expected.

Updates cause replacement.

List contents:

*

Map value expected.

Updates cause replacement.

Map properties:

Key

Tag key.

String value expected.

Updates cause replacement.

Value

Tag value.

String value expected.

Updates cause replacement.

Attributes

InstanceList

A comma-delimited list of server ip addresses. (Heat extension).

show

Detailed information about resource.

update_policy

RollingUpdate

Map value expected.

Updates cause replacement.

Map properties:

MaxBatchSize

Integer value expected.

Updates cause replacement.

Defaults to “1”.

MinInstancesInService

Integer value expected.

Updates cause replacement.

Defaults to “0”.

PauseTime

String value expected.

Updates cause replacement.

Defaults to “PT0S”.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Heat::InstanceGroup
    properties:
      AvailabilityZones: [Value, Value, ...]
      LaunchConfigurationName: String
      LoadBalancerNames: [Value, Value, ...]
      Size: Integer
      Tags: [{"Key": String, "Value": String}, {"Key": String, "Value":
String}, ...]
```

OS::Heat::MultipartMime

> **Note**
>
> Available since 2014.1 (Icehouse)

Assembles a collection of software configurations as a multi-part mime.

Parts in the message can be populated with inline configuration or references to other config resources. If the referenced resource is itself a valid multi-part mime message, that will be broken into parts and those parts appended to this message.

The resulting multi-part mime message will be stored by the configs API and can be referenced in properties such as OS::Nova::Server user_data.

This resource is generally used to build a list of cloud-init configuration elements including scripts and cloud-config. Since cloud-init is boot-only configuration, any changes to the definition will result in the replacement of all servers which reference it.

Optional Properties

parts

Parts belonging to this message.

List value expected.

Updates cause replacement.

Defaults to “[]”.

List contents:

*

Map value expected.

Updates cause replacement.

Map properties:

config

Content of part to attach, either inline or by referencing the ID of another software config resource.

String value expected.

Updates cause replacement.

filename

Optional filename to associate with part.

String value expected.

Updates cause replacement.

subtype

Optional subtype to specify with the type.

String value expected.

Updates cause replacement.

type

Whether the part content is text or multipart.

String value expected.

Updates cause replacement.

Defaults to “text”.

Allowed values: text, multipart

Attributes

config

The config value of the software config.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Heat::MultipartMime
    properties:
      parts: [{"type": String, "subtype": String, "config": String,
"filename": String}, {"type": String, "subtype": String, "config": String,
"filename": String}, ...]
```

OS::Heat::None

> **Note**
>
> Available since 5.0.0 (Liberty)

Enables easily disabling certain resources via the resource_registry.

It does nothing, but can effectively stub out any other resource because it will accept any properties and return any attribute (as None). Note this resource always does nothing on update (e.g it is not replaced even if a change to the stubbed resource properties would cause replacement).

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Heat::None
```

OS::Heat::RandomString

> **Note**
>
> Available since 2014.1 (Icehouse)

A resource which generates a random string.

This is useful for configuring passwords and secrets on services. Random string can be generated from specified character sequences, which means that all characters will be randomly chosen from specified sequences, or with some classes, e.g. letterdigits, which means that all character will be randomly chosen from union of ascii letters and digits. Output string will be randomly generated string with specified length (or with length of 32, if length property doesn't specified).

Optional Properties

character_classes

A list of character class and their constraints to generate the random string from.

List value expected.

Updates cause replacement.

Defaults to "[{'min': 1, 'class': 'lettersdigits'}]".

List contents:

*

> Map value expected.
>
> Updates cause replacement.
>
> *Map properties:*

class

A character class and its corresponding min constraint to generate the random string from.

String value expected.

Updates cause replacement.

Defaults to “lettersdigits”.

Allowed values: lettersdigits, letters, lowercase, uppercase, digits, hexdigits, octdigits

min

The minimum number of characters from this character class that will be in the generated string.

Integer value expected.

Updates cause replacement.

Defaults to “1”.

The value must be in the range 1 to 512.

character_sequences

A list of character sequences and their constraints to generate the random string from.

List value expected.

Updates cause replacement.

List contents:

*

Map value expected.

Updates cause replacement.

Map properties:

min

The minimum number of characters from this sequence that will be in the generated string.

Integer value expected.

Updates cause replacement.

Defaults to “1”.

The value must be in the range 1 to 512.

sequence

A character sequence and its corresponding min constraint to generate the random string from.

String value expected.

Updates cause replacement.

length

Length of the string to generate.

Integer value expected.

Updates cause replacement.

Defaults to “32”.

The value must be in the range 1 to 512.

salt

Value which can be set or changed on stack update to trigger the resource for replacement with a new random string. The salt value itself is ignored by the random generator.

String value expected.

Updates cause replacement.

Attributes

show

Detailed information about resource.

value

The random string generated by this resource. This value is also available by referencing the resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Heat::RandomString
    properties:
      character_classes: [{"min": Integer, "class": String}, {"min":
Integer, "class": String}, ...]
      character_sequences: [{"min": Integer, "sequence": String}, {"min":
Integer, "sequence": String}, ...]
      length: Integer
      salt: String
```

OS::Heat::ResourceChain

> **Note**
>
> Available since 6.0.0 (Mitaka)

Creates one or more resources with the same configuration.

The types of resources to be created are passed into the chain through the `resources` property. One resource will be created for each type listed. Each is passed the configuration specified under `resource_properties`.

The `concurrent` property controls if the resources will be created concurrently. If omitted or set to false, each resource will be treated as having a dependency on the resource before it in the list.

Required Properties

resources

The list of resource types to create. This list may contain type names or aliases defined in the resource registry. Specific template names are not supported.

List value expected.

Can be updated without replacement.

Optional Properties

concurrent

If true, the resources in the chain will be created concurrently. If false or omitted, each resource will be treated as having a dependency on the previous resource in the list.

Boolean value expected.

Updates cause replacement.

Defaults to “False”.

resource_properties

Properties to pass to each resource being created in the chain.

Map value expected.

Updates cause replacement.

Attributes

attributes

A map of resource names to the specified attribute of each individual resource.

refs

A list of resource IDs for the resources in the chain.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Heat::ResourceChain
    properties:
      concurrent: Boolean
      resource_properties: {...}
      resources: [Value, Value, ...]
```

OS::Heat::ResourceGroup

> **Note**
>
> Available since 2014.1 (Icehouse)

Creates one or more identically configured nested resources.

In addition to the refs attribute, this resource implements synthetic attributes that mirror those of the resources in the group. When getting an attribute from this resource, however, a list of attribute values for each resource in the group is returned. To get attribute values for a single resource in the group, synthetic attributes of the form resource.{resource index}.{attribute name} can be used. The resource ID of a particular resource in the group can be obtained via the synthetic attribute resource.{resource index}. Note, that if you get attribute without {resource index}, e.g. [resource, {attribute_name}], you'll get a list of this attribute's value for all resources in group.

While each resource in the group will be identically configured, this resource does allow for some index-based customization of the properties of the resources in the group. For example:

```
resources:
  my_indexed_group:
    type: OS::Heat::ResourceGroup
    properties:
      count: 3
      resource_def:
        type: OS::Nova::Server
        properties:
          # create a unique name for each server
          # using its index in the group
          name: my_server_%index%
          image: CentOS 6.5
          flavor: 4GB Performance
```

would result in a group of three servers having the same image and flavor, but names of my_server_0, my_server_1, and my_server_2. The variable used for substitution can be customized by using the index_var property.

Required Properties

resource_def

Resource definition for the resources in the group. The value of this property is the

definition of a resource just as if it had been declared in the template itself.

Map value expected.

Can be updated without replacement.

Map properties:

metadata

> **Note**
>
> Available since 5.0.0 (Liberty)
>
> Supplied metadata for the resources in the group.
>
> Map value expected.
>
> Can be updated without replacement.

properties

> Property values for the resources in the group.
>
> Map value expected.
>
> Can be updated without replacement.

type

> The type of the resources in the group.
>
> String value expected.
>
> Can be updated without replacement.

Optional Properties

count

The number of resources to create.

Integer value expected.

Can be updated without replacement.

Defaults to “1”.

The value must be at least 0.

index_var

> **Note**
>
> Available since 2014.2 (Juno)

A variable that this resource will use to replace with the current index of a given resource in the group. Can be used, for example, to customize the name property of grouped servers in order to differentiate them when listed with nova client.

String value expected.

Updates cause replacement.

Defaults to “%index%”.

The length must be at least 3.

removal_policies

> **Note**
>
> Available since 2015.1 (Kilo)

Policies for removal of resources on update.

List value expected.

Can be updated without replacement.

Defaults to “[]”.

List contents:

*

> Policy to be processed when doing an update which requires removal of specific resources.
>
> Map value expected.
>
> Can be updated without replacement.

Map properties:

resource_list

List of resources to be removed when doing an update which requires removal of specific resources. The resource may be specified several ways: (1) The resource name, as in the nested stack, (2) The resource reference returned from get_resource in a template, as available via the 'refs' attribute. Note this is destructive on update when specified; even if the count is not being reduced, and once a resource name is removed, it's name is never reused in subsequent updates.

List value expected.

Can be updated without replacement.

Defaults to "[]".

Attributes

attributes

Note

Available since 2014.2 (Juno)

A map of resource names to the specified attribute of each individual resource. Requires heat_template_version: 2014-10-16.

refs

A list of resource IDs for the resources in the group.

refs_map

Note

Available since 7.0.0 (Newton)

A map of resource names to IDs for the resources in the group.

show

Detailed information about resource.

update_policy

batch_create

> **Note**
>
> Available since 5.0.0 (Liberty)

Map value expected.

Updates cause replacement.

Map properties:

max_batch_size

> The maximum number of resources to create at once.
>
> Integer value expected.
>
> Updates cause replacement.
>
> Defaults to “1”.
>
> The value must be at least 1.

pause_time

> The number of seconds to wait between batches.
>
> Number value expected.
>
> Updates cause replacement.
>
> Defaults to “0”.
>
> The value must be at least 0.

rolling_update

> **Note**
>
> Available since 5.0.0 (Liberty)

Map value expected.

Updates cause replacement.

Map properties:

max_batch_size

> The maximum number of resources to replace at once.
>
> Integer value expected.
>
> Updates cause replacement.
>
> Defaults to “1”.
>
> The value must be at least 1.

min_in_service

> The minimum number of resources in service while rolling updates are being executed.
>
> Integer value expected.
>
> Updates cause replacement.
>
> Defaults to “0”.
>
> The value must be at least 0.

pause_time

> The number of seconds to wait between batches of updates.
>
> Number value expected.
>
> Updates cause replacement.
>
> Defaults to “0”.
>
> The value must be at least 0.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Heat::ResourceGroup
    properties:
      count: Integer
      index_var: String
      removal_policies: [{"resource_list": [Value, Value, ...]},
{"resource_list": [Value, Value, ...]}, ...]
      resource_def: {"properties": {...}, "type": String, "metadata":
{...}}
```

OS::Heat::ScalingPolicy

A resource to manage scaling of OS::Heat::AutoScalingGroup.

Note while it may incidentally support AWS::AutoScaling::AutoScalingGroup for now, please don't use it for that purpose and use AWS::AutoScaling::ScalingPolicy instead.

Resource to manage scaling for OS::Heat::AutoScalingGroup, i.e. define which metric should be scaled and scaling adjustment, set cooldown etc.

Required Properties

adjustment_type

Type of adjustment (absolute or percentage).

String value expected.

Can be updated without replacement.

Allowed values: change_in_capacity, exact_capacity, percent_change_in_capacity

auto_scaling_group_id

AutoScaling group ID to apply policy to.

String value expected.

Updates cause replacement.

scaling_adjustment

Size of adjustment.

Number value expected.

Can be updated without replacement.

Optional Properties

cooldown

Cooldown period, in seconds.

Number value expected.

Can be updated without replacement.

min_adjustment_step

Minimum number of resources that are added or removed when the AutoScaling group scales up or down. This can be used only when specifying percent_change_in_capacity for the adjustment_type property.

Integer value expected.

Can be updated without replacement.

The value must be at least 0.

Attributes

alarm_url

A signed url to handle the alarm.

show

Detailed information about resource.

signal_url

> **Note**
>
> Available since 5.0.0 (Liberty)

A url to handle the alarm using native API.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Heat::ScalingPolicy
    properties:
      adjustment_type: String
      auto_scaling_group_id: String
      cooldown: Number
      min_adjustment_step: Integer
      scaling_adjustment: Number
```

OS::Heat::SoftwareComponent

> **Note**
>
> Available since 2014.2 (Juno)

A resource for describing and storing a software component.

This resource is similar to OS::Heat::SoftwareConfig. In contrast to SoftwareConfig which allows for storing only one configuration (e.g. one script), SoftwareComponent allows for storing multiple configurations to address handling of all lifecycle hooks (CREATE, UPDATE, SUSPEND, RESUME, DELETE) for a software component in one place.

This resource is backed by the persistence layer and the API of the SoftwareConfig resource, and only adds handling for the additional ‘configs’ property and attribute.

Required Properties

configs

The list of configurations for the different lifecycle actions of the represented software component.

List value expected.

Updates cause replacement.

The length must be at least 1.

List contents:

*

Map value expected.

Updates cause replacement.

Map properties:

actions

Lifecycle actions to which the configuration applies. The string values provided for this property can include the standard resource actions CREATE, DELETE, UPDATE, SUSPEND and RESUME supported by Heat.

List value expected.

Updates cause replacement.

Defaults to “[‘CREATE’, ‘UPDATE’]”.

The length must be at least 1.

List contents:

*

String value expected.

Updates cause replacement.

config

Configuration script or manifest which specifies what actual configuration is performed.

String value expected.

Updates cause replacement.

tool

The configuration tool used to actually apply the configuration on a server. This string property has to be understood by in-instance tools running inside deployed servers.

String value expected.

Updates cause replacement.

Optional Properties

inputs

Schema representing the inputs that this software config is expecting.

List value expected.

Updates cause replacement.

List contents:

*

Map value expected.

Updates cause replacement.

Map properties:

default

Default value for the input if none is specified.

String value expected.

Updates cause replacement.

description

Description of the input.

String value expected.

Updates cause replacement.

name

Name of the input.

String value expected.

Updates cause replacement.

type

Type of the value of the input.

String value expected.

Updates cause replacement.

Defaults to “String”.

Allowed values: String, Number, CommaDelimitedList, Json, Boolean

options

Map containing options specific to the configuration management tool used by this resource.

Map value expected.

Updates cause replacement.

outputs

Schema representing the outputs that this software config will produce.

List value expected.

Updates cause replacement.

List contents:

*

Map value expected.

Updates cause replacement.

Map properties:

description

Description of the output.

String value expected.

Updates cause replacement.

error_output

Denotes that the deployment is in an error state if this output has a value.

Boolean value expected.

Updates cause replacement.

Defaults to “False”.

name

Name of the output.

String value expected.

Updates cause replacement.

type

Type of the value of the output.

String value expected.

Updates cause replacement.

Defaults to “String”.

Allowed values: String, Number, CommaDelimitedList, Json, Boolean

Attributes

config

The config value of the software config.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Heat::SoftwareComponent
    properties:
      configs: [{"config": String, "actions": [String, String, ...],
"tool": String}, {"config": String, "actions": [String, String, ...],
"tool": String}, ...]
      inputs: [{"name": String, "default": String, "type": String,
"description": String}, {"name": String, "default": String, "type":
String, "description": String}, ...]
      options: {...}
      outputs: [{"name": String, "error_output": Boolean, "type": String,
"description": String}, {"name": String, "error_output": Boolean, "type":
String, "description": String}, ...]
```

OS::Heat::SoftwareConfig

> **Note**
>
> Available since 2014.1 (Icehouse)

A resource for describing and storing software configuration.

The software_configs API which backs this resource creates immutable configs, so any change to the template resource definition will result in a new config being created, and the old one being deleted.

Configs can be defined in the same template which uses them, or they can be created in one stack, and passed to another stack via a parameter.

A config resource can be referenced in other resource properties which are config-aware. This includes the properties OS::Nova::Server user_data, OS::Heat::SoftwareDeployment config and OS::Heat::MultipartMime parts config.

Along with the config script itself, this resource can define schemas for inputs and outputs which the config script is expected to consume and produce. Inputs and outputs are optional and will map to concepts which are specific to the configuration tool being used.

Optional Properties

config

Configuration script or manifest which specifies what actual configuration is performed.

String value expected.

Updates cause replacement.

group

Namespace to group this software config by when delivered to a server. This may imply what configuration tool is going to perform the configuration.

String value expected.

Updates cause replacement.

Defaults to “Heat::Ungrouped”.

inputs

Schema representing the inputs that this software config is expecting.

List value expected.

Updates cause replacement.

List contents:

*

Map value expected.

Updates cause replacement.

Map properties:

default

Default value for the input if none is specified.

String value expected.

Updates cause replacement.

description

Description of the input.

String value expected.

Updates cause replacement.

name

Name of the input.

String value expected.

Updates cause replacement.

type

Type of the value of the input.

String value expected.

Updates cause replacement.

Defaults to “String”.

Allowed values: String, Number, CommaDelimitedList, Json, Boolean

options

Map containing options specific to the configuration management tool used by this resource.

Map value expected.

Updates cause replacement.

outputs

Schema representing the outputs that this software config will produce.

List value expected.

Updates cause replacement.

List contents:

*

Map value expected.

Updates cause replacement.

Map properties:

description

Description of the output.

String value expected.

Updates cause replacement.

error_output

Denotes that the deployment is in an error state if this output has a value.

Boolean value expected.

Updates cause replacement.

Defaults to “False”.

name

Name of the output.

String value expected.

Updates cause replacement.

type

Type of the value of the output.

String value expected.

Updates cause replacement.

Defaults to “String”.

Allowed values: String, Number, CommaDelimitedList, Json, Boolean

Attributes

config

The config value of the software config.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Heat::SoftwareConfig
    properties:
      config: String
      group: String
      inputs: [{"name": String, "default": String, "type": String,
"description": String}, {"name": String, "default": String, "type":
String, "description": String}, ...]
      options: {...}
      outputs: [{"name": String, "error_output": Boolean, "type": String,
"description": String}, {"name": String, "error_output": Boolean, "type":
String, "description": String}, ...]
```

OS::Heat::SoftwareDeployment

> **Note**
>
> Available since 2014.1 (Icehouse)

This resource associates a server with some configuration.

The configuration is to be deployed to that server.

A deployment allows input values to be specified which map to the inputs schema defined in the config resource. These input values are interpreted by the configuration tool in a tool-specific manner.

Whenever this resource goes to an IN_PROGRESS state, it creates an ephemeral config that includes the inputs values plus a number of extra inputs which have names prefixed with deploy_. The extra inputs relate to the current state of the

stack, along with the information and credentials required to signal back the deployment results.

Unless signal_transport=NO_SIGNAL, this resource will remain in an IN_PROGRESS state until the server signals it with the output values for that deployment. Those output values are then available as resource attributes, along with the default attributes deploy_stdout, deploy_stderr and deploy_status_code.

Specifying actions other than the default CREATE and UPDATE will result in the deployment being triggered in those actions. For example this would allow cleanup configuration to be performed during actions SUSPEND and DELETE. A config could be designed to only work with some specific actions, or a config can read the value of the deploy_action input to allow conditional logic to perform different configuration for different actions.

Required Properties

server

ID of resource to apply configuration to. Normally this should be a Nova server ID.

String value expected.

Updates cause replacement.

Optional Properties

actions

Which lifecycle actions of the deployment resource will result in this deployment being triggered.

List value expected.

Can be updated without replacement.

Defaults to “[‘CREATE’, ‘UPDATE’]”.

Allowed values: CREATE, UPDATE, DELETE, SUSPEND, RESUME

config

ID of software configuration resource to execute when applying to the server.

String value expected.

Can be updated without replacement.

input_values

Input values to apply to the software configuration on this server.

Map value expected.

Can be updated without replacement.

name

Name of the derived config associated with this deployment. This is used to apply a sort order to the list of configurations currently deployed to a server.

String value expected.

Can be updated without replacement.

signal_transport

How the server should signal to heat with the deployment output values. CFN_SIGNAL will allow an HTTP POST to a CFN keypair signed URL. TEMP_URL_SIGNAL will create a Swift TempURL to be signaled via HTTP PUT. HEAT_SIGNAL will allow calls to the Heat API resource-signal using the provided keystone credentials. ZAQAR_SIGNAL will create a dedicated zaqar queue to be signaled using the provided keystone credentials. NO_SIGNAL will result in the resource going to the COMPLETE state without waiting for any signal.

String value expected.

Updates cause replacement.

Defaults to “CFN_SIGNAL”.

Allowed values: CFN_SIGNAL, TEMP_URL_SIGNAL, HEAT_SIGNAL, NO_SIGNAL, ZAQAR_SIGNAL

Attributes

deploy_status_code

Returned status code from the configuration execution.

deploy_stderr

Captured stderr from the configuration execution.

deploy_stdout

Captured stdout from the configuration execution.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Heat::SoftwareDeployment
    properties:
      actions: [Value, Value, ...]
      config: String
      input_values: {...}
      name: String
      server: String
      signal_transport: String
```

OS::Heat::SoftwareDeploymentGroup

> **Note**
>
> Available since 5.0.0 (Liberty)

This resource associates a group of servers with some configuration.

The configuration is to be deployed to all servers in the group.

The properties work in a similar way to OS::Heat::SoftwareDeployment, and in addition to the attributes documented, you may pass any attribute supported by OS::Heat::SoftwareDeployment, including those exposing arbitrary outputs, and return a map of deployment names to the specified attribute.

Required Properties

servers

A map of names and server IDs to apply configuration to. The name is arbitrary and is used as the Heat resource name for the corresponding deployment.

Map value expected.

Can be updated without replacement.

Optional Properties

actions

Which lifecycle actions of the deployment resource will result in this deployment being triggered.

List value expected.

Can be updated without replacement.

Defaults to “[‘CREATE’, ‘UPDATE’]”.

Allowed values: CREATE, UPDATE, DELETE, SUSPEND, RESUME

config

ID of software configuration resource to execute when applying to the server.

String value expected.

Can be updated without replacement.

input_values

Input values to apply to the software configuration on this server.

Map value expected.

Can be updated without replacement.

name

Name of the derived config associated with this deployment. This is used to apply a sort order to the list of configurations currently deployed to a server.

String value expected.

Can be updated without replacement.

signal_transport

How the server should signal to heat with the deployment output values. CFN_SIGNAL will allow an HTTP POST to a CFN keypair signed URL. TEMP_URL_SIGNAL will create a Swift TempURL to be signaled via HTTP PUT. HEAT_SIGNAL will allow calls to the Heat API resource-signal using the provided keystone credentials. ZAQAR_SIGNAL will create a dedicated zaqar queue to be signaled using the provided keystone credentials. NO_SIGNAL will result in the resource going to the COMPLETE state without waiting for any signal.

String value expected.

Updates cause replacement.

Defaults to “CFN_SIGNAL”.

Allowed values: CFN_SIGNAL, TEMP_URL_SIGNAL, HEAT_SIGNAL, NO_SIGNAL, ZAQAR_SIGNAL

Attributes

deploy_status_codes

A map of Nova names and returned status code from the configuration execution.

deploy_stderrs

A map of Nova names and captured stderrs from the configuration execution to each server.

deploy_stdouts

A map of Nova names and captured stdouts from the configuration execution to each server.

show

Detailed information about resource.

update_policy

batch_create

> **Note**
>
> Available since 5.0.0 (Liberty)

Map value expected.

Updates cause replacement.

Map properties:

`max_batch_size`

> The maximum number of resources to create at once.
>
> Integer value expected.
>
> Updates cause replacement.
>
> Defaults to “1”.
>
> The value must be at least 1.

pause_time

The number of seconds to wait between batches.

Number value expected.

Updates cause replacement.

Defaults to “0”.

The value must be at least 0.

rolling_update

Note

Available since 5.0.0 (Liberty)

Map value expected.

Updates cause replacement.

Map properties:

max_batch_size

The maximum number of resources to replace at once.

Integer value expected.

Updates cause replacement.

Defaults to “1”.

The value must be at least 1.

min_in_service

The minimum number of resources in service while rolling updates are being executed.

Integer value expected.

Updates cause replacement.

Defaults to “0”.

The value must be at least 0.

pause_time

> The number of seconds to wait between batches of updates.
>
> Number value expected.
>
> Updates cause replacement.
>
> Defaults to “0”.
>
> The value must be at least 0.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Heat::SoftwareDeploymentGroup
    properties:
      actions: [Value, Value, ...]
      config: String
      input_values: {...}
      name: String
      servers: {...}
      signal_transport: String
```

OS::Heat::Stack

A Resource representing a stack.

A resource that allowing for the creating stack, where should be defined stack template in HOT format, parameters (if template has any parameters with no default value), and timeout of creating. After creating current stack will have remote stack.

Required Properties

template

Template that specifies the stack to be created as a resource.

String value expected.

Can be updated without replacement.

Optional Properties

context

Context for this stack.

Map value expected.

Updates cause replacement.

Map properties:

region_name

> Region name in which this stack will be created.
>
> String value expected.
>
> Updates cause replacement.

parameters

Set of parameters passed to this stack.

Map value expected.

Can be updated without replacement.

Defaults to “{}”.

timeout

Number of minutes to wait for this stack creation.

Integer value expected.

Can be updated without replacement.

Attributes

outputs

A dict of key-value pairs output from the stack.

show

Detailed information about resource.

stack_name

Name of the stack.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Heat::Stack
    properties:
      context: {"region_name": String}
      parameters: {...}
      template: String
      timeout: Integer
```

OS::Heat::StructuredConfig

> **Note**
>
> Available since 2014.1 (Icehouse)

A resource which has same logic with OS::Heat::SoftwareConfig.

This resource is like OS::Heat::SoftwareConfig except that the config property is represented by a Map rather than a String.

This is useful for configuration tools which use YAML or JSON as their configuration syntax. The resulting configuration is transferred, stored and returned by the software_configs API as parsed JSON.

Optional Properties

config

Map representing the configuration data structure which will be serialized to JSON format.

Map value expected.

Updates cause replacement.

group

Namespace to group this software config by when delivered to a server. This may imply what configuration tool is going to perform the configuration.

String value expected.

Updates cause replacement.

Defaults to “Heat::Ungrouped”.

inputs

Schema representing the inputs that this software config is expecting.

List value expected.

Updates cause replacement.

List contents:

*

Map value expected.

Updates cause replacement.

Map properties:

default

Default value for the input if none is specified.

String value expected.

Updates cause replacement.

description

Description of the input.

String value expected.

Updates cause replacement.

name

Name of the input.

String value expected.

Updates cause replacement.

type

Type of the value of the input.

String value expected.

Updates cause replacement.

Defaults to “String”.

Allowed values: String, Number, CommaDelimitedList, Json, Boolean

options

Map containing options specific to the configuration management tool used by this resource.

Map value expected.

Updates cause replacement.

outputs

Schema representing the outputs that this software config will produce.

List value expected.

Updates cause replacement.

List contents:

*

Map value expected.

Updates cause replacement.

Map properties:

description

Description of the output.

String value expected.

Updates cause replacement.

error_output

Denotes that the deployment is in an error state if this output has a value.

Boolean value expected.

Updates cause replacement.

Defaults to “False”.

name

Name of the output.

String value expected.

Updates cause replacement.

type

Type of the value of the output.

String value expected.

Updates cause replacement.

Defaults to “String”.

Allowed values: String, Number, CommaDelimitedList, Json, Boolean

Attributes

config

The config value of the software config.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Heat::StructuredConfig
    properties:
      config: {...}
      group: String
      inputs: [{"name": String, "default": String, "type": String,
"description": String}, {"name": String, "default": String, "type":
String, "description": String}, ...]
      options: {...}
      outputs: [{"name": String, "error_output": Boolean, "type": String,
"description": String}, {"name": String, "error_output": Boolean, "type":
String, "description": String}, ...]
```

OS::Heat::StructuredDeployment

> **Note**
>
> Available since 2014.1 (Icehouse)

A resource which has same logic with OS::Heat::SoftwareDeployment.

A deployment resource like OS::Heat::SoftwareDeployment, but which performs input value substitution on the config defined by a OS::Heat::StructuredConfig resource.

Some configuration tools have no concept of inputs, so the input value substitution needs to occur in the deployment resource. An example of this is the JSON metadata consumed by the cfn-init tool.

Where the config contains {get_input: input_name} this will be substituted with the value of input_name in this resource's input_values. If get_input needs to be passed through to the substituted configuration then a different input_key property value can be specified.

Required Properties

server

ID of resource to apply configuration to. Normally this should be a Nova server ID.

String value expected.

Updates cause replacement.

Optional Properties

actions

Which lifecycle actions of the deployment resource will result in this deployment being triggered.

List value expected.

Can be updated without replacement.

Defaults to "['CREATE', 'UPDATE']".

Allowed values: CREATE, UPDATE, DELETE, SUSPEND, RESUME

config

ID of software configuration resource to execute when applying to the server.

String value expected.

Can be updated without replacement.

input_key

Name of key to use for substituting inputs during deployment.

String value expected.

Updates cause replacement.

Defaults to “get_input”.

input_values

Input values to apply to the software configuration on this server.

Map value expected.

Can be updated without replacement.

input_values_validate

Perform a check on the input values passed to verify that each required input has a corresponding value. When the property is set to STRICT and no value is passed, an exception is raised.

String value expected.

Updates cause replacement.

Defaults to “LAX”.

Allowed values: LAX, STRICT

name

Name of the derived config associated with this deployment. This is used to apply a sort order to the list of configurations currently deployed to a server.

String value expected.

Can be updated without replacement.

signal_transport

How the server should signal to heat with the deployment output values. CFN_SIGNAL will allow an HTTP POST to a CFN keypair signed URL. TEMP_URL_SIGNAL will create a Swift TempURL to be signaled via HTTP PUT. HEAT_SIGNAL will allow calls to the Heat API resource-signal using the provided keystone credentials. ZAQAR_SIGNAL will create a dedicated zaqar queue to be signaled using the provided keystone credentials. NO_SIGNAL will result in the resource going to the COMPLETE state without waiting for any signal.

String value expected.

Updates cause replacement.

Defaults to “CFN_SIGNAL”.

Allowed values: CFN_SIGNAL, TEMP_URL_SIGNAL, HEAT_SIGNAL, NO_SIGNAL, ZAQAR_SIGNAL

Attributes

deploy_status_code

Returned status code from the configuration execution.

deploy_stderr

Captured stderr from the configuration execution.

deploy_stdout

Captured stdout from the configuration execution.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Heat::StructuredDeployment
    properties:
      actions: [Value, Value, ...]
      config: String
      input_key: String
      input_values: {...}
      input_values_validate: String
      name: String
      server: String
      signal_transport: String
```

OS::Heat::StructuredDeploymentGroup

> **Note**
>
> Available since 5.0.0 (Liberty)

This resource associates a group of servers with some configuration.

This resource works similar as OS::Heat::SoftwareDeploymentGroup, but for structured resources.

Required Properties

servers

A map of names and server IDs to apply configuration to. The name is arbitrary and is used as the Heat resource name for the corresponding deployment.

Map value expected.

Can be updated without replacement.

Optional Properties

actions

Which lifecycle actions of the deployment resource will result in this deployment being triggered.

List value expected.

Can be updated without replacement.

Defaults to “[‘CREATE’, ‘UPDATE’]”.

Allowed values: CREATE, UPDATE, DELETE, SUSPEND, RESUME

config

ID of software configuration resource to execute when applying to the server.

String value expected.

Can be updated without replacement.

input_key

Name of key to use for substituting inputs during deployment.

String value expected.

Updates cause replacement.

Defaults to “get_input”.

input_values

Input values to apply to the software configuration on this server.

Map value expected.

Can be updated without replacement.

input_values_validate

Perform a check on the input values passed to verify that each required input has a corresponding value. When the property is set to STRICT and no value is passed, an exception is raised.

String value expected.

Updates cause replacement.

Defaults to “LAX”.

Allowed values: LAX, STRICT

name

Name of the derived config associated with this deployment. This is used to apply a sort order to the list of configurations currently deployed to a server.

String value expected.

Can be updated without replacement.

signal_transport

How the server should signal to heat with the deployment output values. CFN_SIGNAL will allow an HTTP POST to a CFN keypair signed URL. TEMP_URL_SIGNAL will create a Swift TempURL to be signaled via HTTP PUT. HEAT_SIGNAL will allow calls to the Heat API resource-signal using the provided keystone credentials. ZAQAR_SIGNAL will create a dedicated zaqar queue to be signaled using the provided keystone credentials. NO_SIGNAL will result in the resource going to the COMPLETE state without waiting for any signal.

String value expected.

Updates cause replacement.

Defaults to “CFN_SIGNAL”.

Allowed values: CFN_SIGNAL, TEMP_URL_SIGNAL, HEAT_SIGNAL, NO_SIGNAL, ZAQAR_SIGNAL

Attributes

deploy_status_codes

A map of Nova names and returned status code from the configuration execution.

deploy_stderrs

A map of Nova names and captured stderrs from the configuration execution to each server.

deploy_stdouts

A map of Nova names and captured stdouts from the configuration execution to each server.

show

Detailed information about resource.

update_policy

batch_create

> **Note**
>
> Available since 5.0.0 (Liberty)

Map value expected.

Updates cause replacement.

Map properties:

max_batch_size

> The maximum number of resources to create at once.
>
> Integer value expected.
>
> Updates cause replacement.
>
> Defaults to “1”.
>
> The value must be at least 1.

pause_time

> The number of seconds to wait between batches.

Number value expected.

Updates cause replacement.

Defaults to “0”.

The value must be at least 0.

rolling_update

Note

Available since 5.0.0 (Liberty)

Map value expected.

Updates cause replacement.

Map properties:

max_batch_size

The maximum number of resources to replace at once.

Integer value expected.

Updates cause replacement.

Defaults to “1”.

The value must be at least 1.

min_in_service

The minimum number of resources in service while rolling updates are being executed.

Integer value expected.

Updates cause replacement.

Defaults to “0”.

The value must be at least 0.

pause_time

The number of seconds to wait between batches of updates.

> Number value expected.
>
> Updates cause replacement.
>
> Defaults to “0”.
>
> The value must be at least 0.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Heat::StructuredDeploymentGroup
    properties:
      actions: [Value, Value, ...]
      config: String
      input_key: String
      input_values: {...}
      input_values_validate: String
      name: String
      servers: {...}
      signal_transport: String
```

OS::Heat::SwiftSignal

> **Note**
>
> Available since 2014.2 (Juno)

Resource for handling signals received by SwiftSignalHandle.

This resource handles signals received by SwiftSignalHandle and is same as WaitCondition resource.

Required Properties

handle

URL of TempURL where resource will signal completion and optionally upload data.

String value expected.

Updates cause replacement.

timeout

The maximum number of seconds to wait for the resource to signal completion. Once the timeout is reached, creation of the signal resource will fail.

Number value expected.

Updates cause replacement.

The value must be in the range 1 to 43200.

Optional Properties

count

The number of success signals that must be received before the stack creation process continues.

Integer value expected.

Updates cause replacement.

Defaults to “1”.

The value must be in the range 1 to 1000.

Attributes

data

JSON data that was uploaded via the SwiftSignalHandle.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Heat::SwiftSignal
    properties:
      count: Integer
      handle: String
      timeout: Number
```

OS::Heat::SwiftSignalHandle

> **Note**
>
> Available since 2014.2 (Juno)

Resource for managing signals from Swift resources.

This resource is same as WaitConditionHandle, but designed for using by Swift resources.

Attributes

curl_cli

Convenience attribute, provides curl CLI command prefix, which can be used for signalling handle completion or failure. You can signal success by adding –data-binary '{"status": "SUCCESS"}' , or signal failure by adding –data-binary '{"status": "FAILURE"}'.

endpoint

Endpoint/url which can be used for signalling handle.

show

Detailed information about resource.

token

Tokens are not needed for Swift TempURLs. This attribute is being kept for compatibility with the OS::Heat::WaitConditionHandle resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Heat::SwiftSignalHandle
```

OS::Heat::TestResource

> **Note**
>
> Available since 5.0.0 (Liberty)

A resource which stores the string value that was provided.

This resource is to be used only for testing. It has control knobs such as ‘update_replace’, ‘fail’, ‘wait_secs’.

Optional Properties

action_wait_secs

Options for simulating waiting.

Map value expected.

Can be updated without replacement.

Map properties:

create

> Seconds to wait after a create. Defaults to the global wait_secs.
>
> Number value expected.
>
> Can be updated without replacement.

delete

> Seconds to wait after a delete. Defaults to the global wait_secs.
>
> Number value expected.
>
> Can be updated without replacement.

update

> Seconds to wait after an update. Defaults to the global wait_secs.
>
> Number value expected.
>
> Can be updated without replacement.

attr_wait_secs

> **Note**
>
> Available since 6.0.0 (Mitaka)

Number value for timeout during resolving output value.

Number value expected.

Can be updated without replacement.

Defaults to “0”.

client_name

Client to poll.

String value expected.

Can be updated without replacement.

Defaults to “”.

constraint_prop_secs

> **Note**
>
> Available since 6.0.0 (Mitaka)

Number value for delay during resolve constraint.

Number value expected.

Can be updated without replacement.

Defaults to “0”.

Value must be of type test_constr

entity_name

Client entity to poll.

String value expected.

Can be updated without replacement.

Defaults to “”.

fail

Value which can be set to fail the resource operation to test failure scenarios.

Boolean value expected.

Can be updated without replacement.

Defaults to “False”.

update_replace

Value which can be set to trigger update replace for the particular resource.

Boolean value expected.

Can be updated without replacement.

Defaults to “False”.

update_replace_value

> **Note**
>
> Available since 7.0.0 (Newton)

Some value that can be stored but can not be updated.

String value expected.

Updates cause replacement.

value

The input string to be stored.

String value expected.

Can be updated without replacement.

Defaults to “test_string”.

wait_secs

Seconds to wait after an action (-1 is infinite).

Number value expected.

Can be updated without replacement.

Defaults to “0”.

Attributes

output

The string that was stored. This value is also available by referencing the resource.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Heat::TestResource
    properties:
      action_wait_secs: {"update": Number, "create": Number, "delete":
Number}
      attr_wait_secs: Number
      client_name: String
      constraint_prop_secs: Number
      entity_name: String
      fail: Boolean
      update_replace: Boolean
      update_replace_value: String
      value: String
      wait_secs: Number
```

OS::Heat::UpdateWaitConditionHandle

> **Note**
>
> Available since 2014.1 (Icehouse)

WaitConditionHandle that clears signals and changes handle on update.

This works identically to a regular WaitConditionHandle, except that on update it clears all signals received and changes the handle. Using this handle means that you must setup the signal senders to send their signals again any time the update handle changes. This allows us to roll out new configurations and be confident that they are rolled out once UPDATE COMPLETE is reached.

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30

...

resources:
  ...
  the_resource:
    type: OS::Heat::UpdateWaitConditionHandle
```

OS::Heat::WaitCondition

> **Note**
>
> Available since 2014.2 (Juno)

Resource for handling signals received by WaitConditionHandle.

Resource takes WaitConditionHandle and starts to create. Resource is in CREATE_IN_PROGRESS status until WaitConditionHandle doesn't receive sufficient number of successful signals (this number can be specified with count property) and successfully creates after that, or fails due to timeout.

Required Properties

handle

A reference to the wait condition handle used to signal this wait condition.

String value expected.

Updates cause replacement.

timeout

The number of seconds to wait for the correct number of signals to arrive.

Number value expected.

Updates cause replacement.

The value must be in the range 1 to 43200.

Optional Properties

count

The number of success signals that must be received before the stack creation process continues.

Integer value expected.

Can be updated without replacement.

Defaults to “1”.

The value must be at least 1.

Attributes

data

JSON string containing data associated with wait condition signals sent to the handle.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Heat::WaitCondition
    properties:
      count: Integer
      handle: String
      timeout: Number
```

OS::Heat::WaitConditionHandle

> **Note**
>
> Available since 2014.2 (Juno)

Resource for managing instance signals.

The main points of this resource are:

- have no dependencies (so the instance can reference it).
- create credentials to allow for signalling from the instance.
- handle signals from the instance, validate and store result.

Optional Properties

signal_transport

> **Note**
>
> Available since 6.0.0 (Mitaka)

How the client will signal the wait condition. CFN_SIGNAL will allow an HTTP POST to a CFN keypair signed URL. TEMP_URL_SIGNAL will create a Swift TempURL to be signalled via HTTP PUT. HEAT_SIGNAL will allow calls to the Heat API resource-signal using the provided keystone credentials. ZAQAR_SIGNAL will create a dedicated zaqar queue to be signalled using the provided keystone credentials. TOKEN_SIGNAL will allow and HTTP POST to a Heat API endpoint with the provided keystone token. NO_SIGNAL will result in the resource going to a signalled state without waiting for any signal.

String value expected.

Updates cause replacement.

Defaults to “TOKEN_SIGNAL”.

Allowed values: CFN_SIGNAL, TEMP_URL_SIGNAL, HEAT_SIGNAL, NO_SIGNAL, ZAQAR_SIGNAL, TOKEN_SIGNAL

Attributes

curl_cli

Convenience attribute, provides curl CLI command prefix, which can be used for signalling handle completion or failure when signal_transport is set to TOKEN_SIGNAL. You can signal success by adding –data-binary ‘{“status”: “SUCCESS”}’ , or signal failure by adding –data-binary ‘{“status”: “FAILURE”}’. This attribute is set to None for all other signal transports.

endpoint

Endpoint/url which can be used for signalling handle when signal_transport is set to TOKEN_SIGNAL. None for all other signal transports.

show

Detailed information about resource.

signal

JSON serialized map that includes the endpoint, token and/or other attributes the client must use for signalling this handle. The contents of this map depend on the type of signal selected in the signal_transport property.

token

Token for stack-user which can be used for signalling handle when signal_transport is set to TOKEN_SIGNAL. None for all other signal transports.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Heat::WaitConditionHandle
    properties:
      signal_transport: String
```

OS::Keystone::Endpoint

> **Note**
>
> Available since 5.0.0 (Liberty) - Supported versions: keystone v3

Heat Template Resource for Keystone Service Endpoint.

Keystone endpoint is just the URL that can be used for accessing a service within OpenStack. Endpoint can be accessed by admin, by services or public, i.e. everyone can use this endpoint.

Required Properties

interface

Interface type of keystone service endpoint.

String value expected.

Can be updated without replacement.

Allowed values: public, internal, admin

service

Name or Id of keystone service.

String value expected.

Can be updated without replacement.

Value must be of type keystone.service

url

URL of keystone service endpoint.

String value expected.

Can be updated without replacement.

Optional Properties

enabled

> **Note**
>
> Available since 6.0.0 (Mitaka)

This endpoint is enabled or disabled.

Boolean value expected.

Can be updated without replacement.

Defaults to “True”.

name

Name of keystone endpoint.

String value expected.

Can be updated without replacement.

region

Name or Id of keystone region.

String value expected.

Can be updated without replacement.

Value must be of type keystone.region

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Keystone::Endpoint
    properties:
      enabled: Boolean
      interface: String
      name: String
      region: String
      service: String
      url: String
```

OS::Keystone::Group

> **Note**
>
> Available since 2015.1 (Kilo) - Supported versions: keystone v3

Heat Template Resource for Keystone Group.

Groups are a container representing a collection of users. A group itself must be owned by a specific domain, and hence all group names are not globally unique, but only unique to their domain.

Optional Properties

description

Description of keystone group.

String value expected.

Can be updated without replacement.

Defaults to “”.

domain

Name or id of keystone domain.

String value expected.

Can be updated without replacement.

Defaults to “default”.

Value must be of type keystone.domain

name

Name of keystone group.

String value expected.

Can be updated without replacement.

roles

List of role assignments.

List value expected.

Can be updated without replacement.

List contents:

*

Map between role with either project or domain.

Map value expected.

Can be updated without replacement.

Map properties:

domain

Keystone domain.

String value expected.

Can be updated without replacement.

Value must be of type keystone.domain

project

Keystone project.

String value expected.

Can be updated without replacement.

Value must be of type keystone.project

role

Keystone role.

String value expected.

Can be updated without replacement.

Value must be of type keystone.role

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Keystone::Group
    properties:
      description: String
      domain: String
      name: String
      roles: [{"role": String, "project": String, "domain": String},
{"role": String, "project": String, "domain": String}, ...]
```

OS::Keystone::GroupRoleAssignment

Note

Available since 5.0.0 (Liberty) - Supported versions: keystone v3

Resource for granting roles to a group.

Resource for specifying groups and their's roles.

Required Properties

group

Name or id of keystone group.

String value expected.

Can be updated without replacement.

Value must be of type keystone.group

Optional Properties

roles

List of role assignments.

List value expected.

Can be updated without replacement.

List contents:

*

> Map between role with either project or domain.
>
> Map value expected.
>
> Can be updated without replacement.
>
> *Map properties:*
>
> domain
>
> > Keystone domain.
> >
> > String value expected.
> >
> > Can be updated without replacement.
> >
> > Value must be of type keystone.domain
>
> project
>
> > Keystone project.
> >
> > String value expected.

Can be updated without replacement.

Value must be of type keystone.project

role

Keystone role.

String value expected.

Can be updated without replacement.

Value must be of type keystone.role

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Keystone::GroupRoleAssignment
    properties:
      group: String
      roles: [{"role": String, "project": String, "domain": String},
{"role": String, "project": String, "domain": String}, ...]
```

OS::Keystone::Project

Note

Available since 2015.1 (Kilo) - Supported versions: keystone v3

Heat Template Resource for Keystone Project.

Projects represent the base unit of ownership in OpenStack, in that all resources in OpenStack should be owned by a specific project. A project itself must be owned by a specific domain, and hence all project names are not globally unique, but unique to their domain. If the domain for a project is not specified, then it is added to the

default domain.

Optional Properties

description

Description of keystone project.

String value expected.

Can be updated without replacement.

Defaults to “”.

domain

Name or id of keystone domain.

String value expected.

Can be updated without replacement.

Defaults to “default”.

Value must be of type keystone.domain

enabled

This project is enabled or disabled.

Boolean value expected.

Can be updated without replacement.

Defaults to “True”.

name

Name of keystone project.

String value expected.

Can be updated without replacement.

parent

> **Note**
>
> Available since 6.0.0 (Mitaka)

The name or ID of parent of this keystone project in hierarchy.

String value expected.

Updates cause replacement.

Value must be of type keystone.project

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Keystone::Project
    properties:
      description: String
      domain: String
      enabled: Boolean
      name: String
      parent: String
```

OS::Keystone::Region

> **Note**
>
> Available since 6.0.0 (Mitaka) - Supported versions: keystone v3

Heat Template Resource for Keystone Region.

This plug-in helps to create, update and delete a keystone region. Also it can be used for enable or disable a given keystone region.

Optional Properties

description

Description of keystone region.

String value expected.

Can be updated without replacement.

enabled

This region is enabled or disabled.

Boolean value expected.

Can be updated without replacement.

Defaults to “True”.

id

The user-defined region ID and should unique to the OpenStack deployment. While creating the region, heat will url encode this ID.

String value expected.

Updates cause replacement.

parent_region

If the region is hierarchically a child of another region, set this parameter to the ID of the parent region.

String value expected.

Can be updated without replacement.

Value must be of type keystone.region

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Keystone::Region
    properties:
      description: String
      enabled: Boolean
      id: String
      parent_region: String
```

OS::Keystone::Role

> **Note**
>
> Available since 2015.1 (Kilo) - Supported versions: keystone v3

Heat Template Resource for Keystone Role.

Roles dictate the level of authorization the end user can obtain. Roles can be granted at either the domain or project level. Role can be assigned to the individual user or at the group level. Role names are globally unique.

Optional Properties

name

Name of keystone role.

String value expected.

Can be updated without replacement.

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Keystone::Role
    properties:
      name: String
```

OS::Keystone::Service

> **Note**
>
> Available since 5.0.0 (Liberty) - Supported versions: keystone v3

Heat Template Resource for Keystone Service.

A resource that allows to create new service and manage it by Keystone.

Required Properties

type

Type of keystone Service.

String value expected.

Can be updated without replacement.

Optional Properties

description

Description of keystone service.

String value expected.

Can be updated without replacement.

enabled

> **Note**
>
> Available since 6.0.0 (Mitaka)

This service is enabled or disabled.

Boolean value expected.

Can be updated without replacement.

Defaults to “True”.

name

Name of keystone service.

String value expected.

Can be updated without replacement.

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Keystone::Service
    properties:
      description: String
      enabled: Boolean
      name: String
      type: String
```

OS::Keystone::User

> **Note**
>
> Available since 2015.1 (Kilo) - Supported versions: keystone v3

Heat Template Resource for Keystone User.

Users represent an individual API consumer. A user itself must be owned by a specific domain, and hence all user names are not globally unique, but only unique to their domain.

Optional Properties

default_project

Default project of keystone user.

String value expected.

Can be updated without replacement.

Value must be of type keystone.project

description

Description of keystone user.

String value expected.

Can be updated without replacement.

Defaults to “”.

domain

Name of keystone domain.

String value expected.

Can be updated without replacement.

Defaults to “default”.

Value must be of type keystone.domain

email

Email address of keystone user.

String value expected.

Can be updated without replacement.

enabled

Keystone user is enabled or disabled.

Boolean value expected.

Can be updated without replacement.

Defaults to “True”.

groups

Keystone user groups.

List value expected.

Can be updated without replacement.

List contents:

*

Keystone user group.

String value expected.

Can be updated without replacement.

Value must be of type keystone.group

name

Name of keystone user.

String value expected.

Can be updated without replacement.

password

Password of keystone user.

String value expected.

Can be updated without replacement.

roles

List of role assignments.

List value expected.

Can be updated without replacement.

List contents:

*

Map between role with either project or domain.

Map value expected.

Can be updated without replacement.

Map properties:

domain

Keystone domain.

String value expected.

Can be updated without replacement.

Value must be of type keystone.domain

project

Keystone project.

String value expected.

Can be updated without replacement.

Value must be of type keystone.project

role

Keystone role.

String value expected.

Can be updated without replacement.

Value must be of type keystone.role

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Keystone::User
    properties:
      default_project: String
      description: String
      domain: String
      email: String
      enabled: Boolean
      groups: [String, String, ...]
      name: String
      password: String
      roles: [{"role": String, "project": String, "domain": String},
{"role": String, "project": String, "domain": String}, ...]
```

OS::Keystone::UserRoleAssignment

> **Note**
>
> Available since 5.0.0 (Liberty) - Supported versions: keystone v3

Resource for granting roles to a user.

Resource for specifying users and their's roles.

Required Properties

user

Name or id of keystone user.

String value expected.

Can be updated without replacement.

Value must be of type keystone.user

Optional Properties

roles

List of role assignments.

List value expected.

Can be updated without replacement.

List contents:

*

> Map between role with either project or domain.
>
> Map value expected.
>
> Can be updated without replacement.
>
> *Map properties:*
>
> domain
>
> > Keystone domain.
> >
> > String value expected.

Can be updated without replacement.

Value must be of type keystone.domain

project

Keystone project.

String value expected.

Can be updated without replacement.

Value must be of type keystone.project

role

Keystone role.

String value expected.

Can be updated without replacement.

Value must be of type keystone.role

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Keystone::UserRoleAssignment
    properties:
      roles: [{"role": String, "project": String, "domain": String},
{"role": String, "project": String, "domain": String}, ...]
      user: String
```

OS::Magnum::Bay

> **Note**
>
> Available since 6.0.0 (Mitaka)

A resource that creates a Magnum Bay.

This resource creates a Magnum bay, which is a collection of node objects where work is scheduled.

Required Properties

baymodel

The name or ID of the bay model.

String value expected.

Updates cause replacement.

Value must be of type magnum.baymodel

Optional Properties

bay_create_timeout

Timeout for creating the bay in minutes. Set to 0 for no timeout.

Integer value expected.

Updates cause replacement.

Defaults to “0”.

The value must be at least 0.

discovery_url

Specifies a custom discovery url for node discovery.

String value expected.

Updates cause replacement.

master_count

The number of master nodes for this bay.

Integer value expected.

Can be updated without replacement.

Defaults to “1”.

The value must be at least 1.

name

The bay name.

String value expected.

Updates cause replacement.

node_count

The node count for this bay.

Integer value expected.

Can be updated without replacement.

Defaults to “1”.

The value must be at least 1.

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Magnum::Bay
    properties:
      bay_create_timeout: Integer
      baymodel: String
      discovery_url: String
      master_count: Integer
      name: String
      node_count: Integer
```

OS::Magnum::BayModel

> **Note**
>
> Available since 5.0.0 (Liberty)

A resource for the BayModel in Magnum.

BayModel is an object that stores template information about the bay which is used to create new bays consistently.

Required Properties

coe

The Container Orchestration Engine for this bay model.

String value expected.

Updates cause replacement.

Allowed values: kubernetes, swarm, mesos

external_network

The external network to attach the Bay.

String value expected.

Updates cause replacement.

Value must be of type neutron.network

image

The image name or UUID to use as a base image for this baymodel.

String value expected.

Updates cause replacement.

Value must be of type glance.image

keypair

The name or id of the nova ssh keypair.

String value expected.

Updates cause replacement.

Value must be of type nova.keypair

Optional Properties

dns_nameserver

The DNS nameserver address.

String value expected.

Updates cause replacement.

Value must be of type ip_addr

docker_volume_size

The size in GB of the docker volume.

Integer value expected.

Updates cause replacement.

The value must be at least 1.

fixed_network

The fixed network to attach the Bay.

String value expected.

Updates cause replacement.

Value must be of type neutron.network

flavor

The flavor of this bay model.

String value expected.

Updates cause replacement.

Value must be of type nova.flavor

http_proxy

> **Note**
>
> Available since 6.0.0 (Mitaka)

The http_proxy address to use for nodes in bay.

String value expected.

Updates cause replacement.

https_proxy

> **Note**
>
> Available since 6.0.0 (Mitaka)

The https_proxy address to use for nodes in bay.

String value expected.

Updates cause replacement.

labels

> **Note**
>
> Available since 6.0.0 (Mitaka)

Arbitrary labels in the form of key=value pairs to associate with a baymodel.

Map value expected.

Updates cause replacement.

master_flavor

The flavor of the master node for this bay model.

String value expected.

Updates cause replacement.

Value must be of type nova.flavor

name

The bay model name.

String value expected.

Updates cause replacement.

network_driver

> **Note**
>
> Available since 6.0.0 (Mitaka)

The name of the driver used for instantiating container networks. By default, Magnum will choose the pre-configured network driver based on COE type.

String value expected.

Updates cause replacement.

no_proxy

> **Note**
>
> Available since 6.0.0 (Mitaka)

A comma separated list of addresses for which proxies should not be used in the bay.

String value expected.

Updates cause replacement.

public

> **Note**
>
> Available since 6.0.0 (Mitaka)

Make the baymodel public.

Boolean value expected.

Updates cause replacement.

Defaults to “False”.

registry_enabled

> **Note**
>
> Available since 6.0.0 (Mitaka)

Enable the docker registry in the bay.

Boolean value expected.

Updates cause replacement.

Defaults to “False”.

tls_disabled

> **Note**
>
> Available since 6.0.0 (Mitaka)

Disable TLS in the bay.

Boolean value expected.

Updates cause replacement.

Defaults to “False”.

volume_driver

> **Note**
>
> Available since 7.0.0 (Newton)

The volume driver name for instantiating container volume.

String value expected.

Updates cause replacement.

Allowed values: cinder, rexray

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Magnum::BayModel
    properties:
      coe: String
      dns_nameserver: String
      docker_volume_size: Integer
      external_network: String
      fixed_network: String
```

```
flavor: String
http_proxy: String
https_proxy: String
image: String
keypair: String
labels: {...}
master_flavor: String
name: String
network_driver: String
no_proxy: String
public: Boolean
registry_enabled: Boolean
tls_disabled: Boolean
volume_driver: String
```

OS::Manila::SecurityService

> **Note**
>
> Available since 5.0.0 (Liberty)

A resource that implements security service of Manila.

A security_service is a set of options that defines a security domain for a particular shared filesystem protocol, such as an Active Directory domain or a Kerberos domain.

Required Properties

type

Security service type.

String value expected.

Updates cause replacement.

Allowed values: ldap, kerberos, active_directory

Optional Properties

description

Security service description.

String value expected.

Can be updated without replacement.

dns_ip

DNS IP address used inside tenant's network.

String value expected.

Can be updated without replacement.

domain

Security service domain.

String value expected.

Can be updated without replacement.

name

Security service name.

String value expected.

Can be updated without replacement.

password

Password used by user.

String value expected.

Can be updated without replacement.

server

Security service IP address or hostname.

String value expected.

Can be updated without replacement.

user

Security service user or group used by tenant.

String value expected.

Can be updated without replacement.

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Manila::SecurityService
    properties:
      description: String
      dns_ip: String
      domain: String
      name: String
      password: String
      server: String
      type: String
      user: String
```

OS::Manila::Share

> **Note**
>
> Available since 5.0.0 (Liberty)

A resource that creates shared mountable file system.

The resource creates a manila share - shared mountable filesystem that can be attached to any client(or clients) that has a network access and permission to mount filesystem. Share is a unit of storage with specific size that supports pre-defined share protocol and advanced security model (access lists, share networks and security services).

Required Properties

share_protocol

Share protocol supported by shared filesystem.

String value expected.

Updates cause replacement.

Allowed values: NFS, CIFS, GlusterFS, HDFS

size

Share storage size in GB.

Integer value expected.

Updates cause replacement.

Optional Properties

access_rules

A list of access rules that define access from IP to Share.

List value expected.

Can be updated without replacement.

Defaults to “[]”.

List contents:

*

Map value expected.

Can be updated without replacement.

Map properties:

access_level

Level of access that need to be provided for guest.

String value expected.

Can be updated without replacement.

Allowed values: ro, rw

access_to

IP or other address information about guest that allowed to access to Share.

String value expected.

Can be updated without replacement.

access_type

Type of access that should be provided to guest.

> String value expected.
>
> Can be updated without replacement.
>
> Allowed values: ip, domain

description

Share description.

String value expected.

Can be updated without replacement.

is_public

Defines if shared filesystem is public or private.

Boolean value expected.

Can be updated without replacement.

Defaults to “False”.

metadata

Metadata key-values defined for share.

Map value expected.

Can be updated without replacement.

name

Share name.

String value expected.

Can be updated without replacement.

share_network

Name or ID of shared network defined for shared filesystem.

String value expected.

Updates cause replacement.

Value must be of type manila.share_network

share_type

Name or ID of shared filesystem type. Types defines some share filesystem profiles that will be used for share creation.

String value expected.

Updates cause replacement.

Value must be of type manila.share_type

snapshot

Name or ID of shared file system snapshot that will be restored and created as a new share.

String value expected.

Updates cause replacement.

Value must be of type manila.share_snapshot

Attributes

availability_zone

The availability zone of shared filesystem.

created_at

Datetime when a share was created.

export_locations

Export locations of share.

host

Share host.

project_id

Share project ID.

share_server_id

ID of server (VM, etc...) on host that is used for exporting network file-system.

show

Detailed information about resource.

status

Current share status.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Manila::Share
    properties:
      access_rules: [{"access_type": String, "access_to": String,
"access_level": String}, {"access_type": String, "access_to": String,
"access_level": String}, ...]
      description: String
      is_public: Boolean
      metadata: {...}
      name: String
      share_network: String
      share_protocol: String
      share_type: String
      size: Integer
      snapshot: String
```

OS::Manila::ShareNetwork

> **Note**
>
> Available since 5.0.0 (Liberty)

A resource that stores network information for share servers.

Stores network information that will be used by share servers, where shares are hosted.

Optional Properties

description

Share network description.

String value expected.

Can be updated without replacement.

name

Name of the share network.

String value expected.

Can be updated without replacement.

neutron_network

Neutron network id.

String value expected.

Can be updated without replacement.

Value must be of type neutron.network

neutron_subnet

Neutron subnet id.

String value expected.

Can be updated without replacement.

Value must be of type neutron.subnet

nova_network

Nova network id.

String value expected.

Can be updated without replacement.

Value must be of type nova.network

security_services

A list of security services IDs or names.

List value expected.

Can be updated without replacement.

Defaults to “[]”.

List contents:

*

String value expected.

Can be updated without replacement.

Attributes

cidr

CIDR of subnet.

ip_version

Version of IP address.

network_type

The physical mechanism by which the virtual network is implemented.

segmentation_id

VLAN ID for VLAN networks or tunnel-id for GRE/VXLAN networks.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Manila::ShareNetwork
    properties:
      description: String
      name: String
      neutron_network: String
      neutron_subnet: String
      nova_network: String
      security_services: [String, String, ...]
```

OS::Manila::ShareType

> **Note**
>
> Available since 5.0.0 (Liberty)

A resource for creating manila share type.

A share_type is an administrator-defined “type of service”, comprised of a tenant visible description, and a list of non-tenant-visible key/value pairs (extra_specs) which the Manila scheduler uses to make scheduling decisions for shared filesystem tasks.

Please note that share type is intended to use mostly by administrators. So it is very likely that Manila will prohibit creation of the resource without administration grants.

Required Properties

driver_handles_share_servers

Required extra specification. Defines if share drivers handles share servers.

Boolean value expected.

Updates cause replacement.

name

Name of the share type.

String value expected.

Updates cause replacement.

Optional Properties

extra_specs

Extra specs key-value pairs defined for share type.

Map value expected.

Can be updated without replacement.

is_public

Defines if share type is accessible to the public.

Boolean value expected.

Updates cause replacement.

Defaults to “True”.

snapshot_support

> **Note**
>
> Available since 6.0.0 (Mitaka)

Boolean extra spec that used for filtering of backends by their capability to create share snapshots.

Boolean value expected.

Updates cause replacement.

Defaults to “True”.

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Manila::ShareType
    properties:
      driver_handles_share_servers: Boolean
      extra_specs: {...}
      is_public: Boolean
      name: String
      snapshot_support: Boolean
```

OS::Mistral::CronTrigger

> **Note**
>
> Available since 5.0.0 (Liberty)

A resource implements Mistral cron trigger.

Cron trigger is an object allowing to run workflow on a schedule. User specifies what workflow with what input needs to be run and also specifies how often it should be run. Pattern property is used to describe the frequency of workflow execution.

Required Properties

workflow

Workflow to execute.

Map value expected.

Updates cause replacement.

Map properties:

input

Input values for the workflow.

Map value expected.

Updates cause replacement.

name

Name of the workflow.

String value expected.

Updates cause replacement.

Value must be of type mistral.workflow

Optional Properties

count

Remaining executions.

Integer value expected.

Updates cause replacement.

first_time

Time of the first execution in format “YYYY-MM-DD HH:MM”.

String value expected.

Updates cause replacement.

name

Name of the cron trigger.

String value expected.

Updates cause replacement.

pattern

Cron expression.

String value expected.

Updates cause replacement.

Value must be of type cron_expression

Attributes

next_execution_time

Time of the next execution in format “YYYY-MM-DD HH:MM:SS”.

remaining_executions

Number of remaining executions.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Mistral::CronTrigger
    properties:
      count: Integer
      first_time: String
      name: String
      pattern: String
      workflow: {"name": String, "input": {...}}
```

OS::Mistral::Workflow

> **Note**
>
> Available since 2015.1 (Kilo)

A resource that implements Mistral workflow.

Workflow represents a process that can be described in a various number of ways and that can do some job interesting to the end user. Each workflow consists of tasks (at least one) describing what exact steps should be made during workflow execution.

For detailed description how to use Workflow, read Mistral documentation.

Required Properties

tasks

Dictionary containing workflow tasks.

List value expected.

Can be updated without replacement.

The length must be at least 1.

List contents:

*

> Map value expected.
>
> Can be updated without replacement.
>
> *Map properties:*
>
> action
>
> > Name of the action associated with the task. Either action or workflow may be defined in the task.
> >
> > String value expected.
> >
> > Can be updated without replacement.

description

Task description.

String value expected.

Can be updated without replacement.

input

Actual input parameter values of the task.

Map value expected.

Can be updated without replacement.

join

Note

Available since 6.0.0 (Mitaka)

Allows to synchronize multiple parallel workflow branches and aggregate their data. Valid inputs: all - the task will run only if all upstream tasks are completed. Any numeric value - then the task will run once at least this number of upstream tasks are completed and corresponding conditions have triggered.

String value expected.

Can be updated without replacement.

keep_result

Note

Available since 5.0.0 (Liberty)

Allowing not to store action results after task completion.

Boolean value expected.

Can be updated without replacement.

name

Task name.

String value expected.

Can be updated without replacement.

on_complete

List of tasks which will run after the task has completed regardless of whether it is successful or not.

List value expected.

Can be updated without replacement.

on_error

List of tasks which will run after the task has completed with an error.

List value expected.

Can be updated without replacement.

on_success

List of tasks which will run after the task has completed successfully.

List value expected.

Can be updated without replacement.

pause_before

Note

Available since 5.0.0 (Liberty)

Defines whether Mistral Engine should put the workflow on hold or not before starting a task.

Boolean value expected.

Can be updated without replacement.

publish

Dictionary of variables to publish to the workflow context.

Map value expected.

Can be updated without replacement.

requires

List of tasks which should be executed before this task. Used only in reverse workflows.

List value expected.

Can be updated without replacement.

retry

Note

Available since 5.0.0 (Liberty)

Defines a pattern how task should be repeated in case of an error.

Map value expected.

Can be updated without replacement.

target

Note

Available since 5.0.0 (Liberty)

It defines an executor to which task action should be sent to.

String value expected.

Can be updated without replacement.

timeout

Note

Available since 5.0.0 (Liberty)

Defines a period of time in seconds after which a task will be failed automatically by engine if hasn't completed.

Integer value expected.

Can be updated without replacement.

wait_after

Note

Available since 5.0.0 (Liberty)

Defines a delay in seconds that Mistral Engine should wait after a task has completed before starting next tasks defined in on-success, on-error or on-complete.

Integer value expected.

Can be updated without replacement.

wait_before

Note

Available since 5.0.0 (Liberty)

Defines a delay in seconds that Mistral Engine should wait before starting a task.

Integer value expected.

Can be updated without replacement.

with_items

Note

Available since 5.0.0 (Liberty)

If configured, it allows to run action or workflow associated with a task multiple times on a provided list of items.

String value expected.

Can be updated without replacement.

workflow

Name of the workflow associated with the task. Can be defined by intrinsic function get_resource or by name of the referenced workflow, i.e. { workflow: wf_name } or { workflow: { get_resource: wf_name }}. Either action or workflow may be defined in the task.

String value expected.

Can be updated without replacement.

Value must be of type mistral.workflow

policies

Note

DEPRECATED since 5.0.0 (Liberty) - Add needed policies directly to the task, Policy keyword is not needed

Note

Available since 2015.1 (Kilo)

Dictionary-like section defining task policies that influence how Mistral Engine runs tasks. Must satisfy Mistral DSL v2.

> Map value expected.
>
> Can be updated without replacement.

type

Workflow type.

String value expected.

Can be updated without replacement.

Allowed values: direct, reverse

Optional Properties

description

Workflow description.

String value expected.

Can be updated without replacement.

input

Dictionary which contains input for workflow.

Map value expected.

Can be updated without replacement.

name

Workflow name.

String value expected.

Updates cause replacement.

output

Any data structure arbitrarily containing YAQL expressions that defines workflow output. May be nested.

Map value expected.

Can be updated without replacement.

params

Workflow additional parameters. If Workflow is reverse typed, params requires ‘task_name’, which defines initial task.

Map value expected.

Can be updated without replacement.

task_defaults

> **Note**
>
> Available since 5.0.0 (Liberty)

Default settings for some of task attributes defined at workflow level.

Map value expected.

Can be updated without replacement.

Map properties:

on_complete

> List of tasks which will run after the task has completed regardless of whether it is successful or not.
>
> List value expected.
>
> Can be updated without replacement.

on_error

> List of tasks which will run after the task has completed with an error.
>
> List value expected.
>
> Can be updated without replacement.

on_success

> List of tasks which will run after the task has completed successfully.
>
> List value expected.
>
> Can be updated without replacement.

pause_before

> Defines whether Mistral Engine should put the workflow on hold or not before starting a task.

Boolean value expected.

Can be updated without replacement.

requires

List of tasks which should be executed before this task. Used only in reverse workflows.

List value expected.

Can be updated without replacement.

retry

Defines a pattern how task should be repeated in case of an error.

Map value expected.

Can be updated without replacement.

timeout

Defines a period of time in seconds after which a task will be failed automatically by engine if hasn’t completed.

Integer value expected.

Can be updated without replacement.

wait_after

Defines a delay in seconds that Mistral Engine should wait after a task has completed before starting next tasks defined in on-success, on-error or on-complete.

Integer value expected.

Can be updated without replacement.

wait_before

Defines a delay in seconds that Mistral Engine should wait before starting a task.

Integer value expected.

Can be updated without replacement.

use_request_body_as_input

Note

Available since 6.0.0 (Mitaka)

Defines the method in which the request body for signaling a workflow would be parsed. In case this property is set to True, the body would be parsed as a simple json where each key is a workflow input, in other cases body would be parsed expecting a specific json format with two keys: “input” and “params”.

Boolean value expected.

Can be updated without replacement.

Attributes

alarm_url

A signed url to create executions for workflows specified in Workflow resource.

data

A dictionary which contains name and input of the workflow.

executions

List of workflows’ executions, each of them is a dictionary with information about execution. Each dictionary returns values for next keys: id, workflow_name, created_at, updated_at, state for current execution state, input, output.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Mistral::Workflow
    properties:
      description: String
      input: {...}
```

```
      name: String
      output: {...}
      params: {...}
      task_defaults: {"timeout": Integer, "pause_before": Boolean,
"on_error": [Value, Value, ...], "wait_after": Integer, "requires":
[Value, Value, ...], "retry": {...}, "on_complete": [Value, Value, ...],
"on_success": [Value, Value, ...], "wait_before": Integer}
      tasks: [{"timeout": Integer, "policies": {...}, "workflow": String,
"requires": [Value, Value, ...], "retry": {...}, "target": String,
"input": {...}, "description": String, "publish": {...}, "keep_result":
Boolean, "on_error": [Value, Value, ...], "action": String, "wait_after":
Integer, "with_items": String, "pause_before": Boolean, "name": String,
"on_complete": [Value, Value, ...], "on_success": [Value, Value, ...],
"wait_before": Integer, "join": String}, {"timeout": Integer, "policies":
{...}, "workflow": String, "requires": [Value, Value, ...], "retry":
{...}, "target": String, "input": {...}, "description": String, "publish":
{...}, "keep_result": Boolean, "on_error": [Value, Value, ...], "action":
String, "wait_after": Integer, "with_items": String, "pause_before":
Boolean, "name": String, "on_complete": [Value, Value, ...], "on_success":
[Value, Value, ...], "wait_before": Integer, "join": String}, ...]
      type: String
      use_request_body_as_input: Boolean
```

OS::Monasca::AlarmDefinition

> **Note**
>
> Available since 7.0.0 (Newton)
>
> **Note**
>
> UNSUPPORTED since 5.0.0 (Liberty)

Heat Template Resource for Monasca Alarm definition.

Monasca Alarm definition helps to define the required expression for a given alarm situation. This plugin helps to create, update and delete the alarm definition.

Alarm definitions is necessary to describe and manage alarms in a one-to-many relationship in order to avoid having to manually declare each alarm even though they may share many common attributes and differ in only one, such as hostname.

Required Properties

expression

Expression of the alarm to evaluate.

String value expected.

Updates cause replacement.

Optional Properties

actions_enabled

Whether to enable the actions or not.

Boolean value expected.

Can be updated without replacement.

Defaults to “True”.

alarm_actions

The notification methods to use when an alarm state is ALARM.

List value expected.

Can be updated without replacement.

Defaults to “[]”.

List contents:

*

Monasca notification.

String value expected.

Can be updated without replacement.

Value must be of type monasca.notification

description

Description of the alarm.

String value expected.

Can be updated without replacement.

match_by

The metric dimensions to match to the alarm dimensions. One or more dimension key names separated by a comma.

List value expected.

Updates cause replacement.

Defaults to “[]”.

name

Name of the alarm. By default, physical resource name is used.

String value expected.

Can be updated without replacement.

ok_actions

The notification methods to use when an alarm state is OK.

List value expected.

Can be updated without replacement.

Defaults to “[]”.

List contents:

*

Monasca notification.

String value expected.

Can be updated without replacement.

Value must be of type monasca.notification

severity

Severity of the alarm.

String value expected.

Can be updated without replacement.

Defaults to “low”.

Allowed values: low, medium, high, critical

undetermined_actions

The notification methods to use when an alarm state is UNDETERMINED.

List value expected.

Can be updated without replacement.

Defaults to “[]”.

List contents:

*

> Monasca notification.
>
> String value expected.
>
> Can be updated without replacement.
>
> Value must be of type monasca.notification

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Monasca::AlarmDefinition
    properties:
      actions_enabled: Boolean
      alarm_actions: [String, String, ...]
      description: String
      expression: String
      match_by: [Value, Value, ...]
      name: String
      ok_actions: [String, String, ...]
      severity: String
      undetermined_actions: [String, String, ...]
```

OS::Monasca::Notification

> **Note**
>
> Available since 7.0.0 (Newton)
>
> **Note**
>
> UNSUPPORTED since 5.0.0 (Liberty)

Heat Template Resource for Monasca Notification.

A resource which is used to notificate if there is some alarm. Monasca Notification helps to declare the hook points, which will be invoked once alarm is generated. This plugin helps to create, update and delete the notification.

Required Properties

address

Address of the notification. It could be a valid email address, url or service key based on notification type.

String value expected.

Can be updated without replacement.

type

Type of the notification.

String value expected.

Can be updated without replacement.

Allowed values: email, webhook, pagerduty

Optional Properties

name

Name of the notification. By default, physical resource name is used.

String value expected.

Can be updated without replacement.

period

> **Note**
>
> Available since 7.0.0 (Newton)

Interval in seconds to invoke webhooks if the alarm state does not transition away from the defined trigger state. The default value is a period interval of 60 seconds. A value of 0 will disable continuous notifications. This property is only applicable for the webhook notification type.

Integer value expected.

Can be updated without replacement.

Defaults to “60”.

Allowed values: 0, 60

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Monasca::Notification
    properties:
      address: String
      name: String
      period: Integer
      type: String
```

OS::Neutron::AddressScope

> **Note**
>
> Available since 6.0.0 (Mitaka)

A resource for Neutron address scope.

This resource can be associated with multiple subnet pools in a one-to-many relationship. The subnet pools under an address scope must not overlap.

Optional Properties

ip_version

Address family of the address scope, which is 4 or 6.

Integer value expected.

Updates cause replacement.

Defaults to “4”.

Allowed values: 4, 6

name

The name for the address scope.

String value expected.

Can be updated without replacement.

shared

Whether the address scope should be shared to other tenants. Note that the default policy setting restricts usage of this attribute to administrative users only, and restricts changing of shared address scope to unshared with update.

Boolean value expected.

Can be updated without replacement.

Defaults to “False”.

tenant_id

The owner tenant ID of the address scope. Only administrative users can specify a tenant ID other than their own.

String value expected.

Updates cause replacement.

Value must be of type keystone.project

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Neutron::AddressScope
    properties:
      ip_version: Integer
      name: String
      shared: Boolean
      tenant_id: String
```

OS::Neutron::Firewall

A resource for the Firewall resource in Neutron FWaaS.

Resource for using the Neutron firewall implementation. Firewall is a network security system that monitors and controls the incoming and outgoing network traffic based on predetermined security rules.

Required Properties

firewall_policy_id

The ID of the firewall policy that this firewall is associated with.

String value expected.

Can be updated without replacement.

Optional Properties

admin_state_up

Administrative state of the firewall. If false (down), firewall does not forward packets and will drop all traffic to/from VMs behind the firewall.

Boolean value expected.

Can be updated without replacement.

Defaults to “True”.

description

Description for the firewall.

String value expected.

Can be updated without replacement.

name

Name for the firewall.

String value expected.

Can be updated without replacement.

value_specs

> **Note**
>
> Available since 5.0.0 (Liberty)

Extra parameters to include in the request. Parameters are often specific to installed hardware or extensions.

Map value expected.

Can be updated without replacement.

Defaults to “{}”.

shared

> **Note**
>
> UNSUPPORTED since 6.0.0 (Mitaka) - There is no such option during 5.0.0, so need to make this property unsupported while it not used.
>
> **Note**
>
> Available since 2015.1 (Kilo)

Whether this firewall should be shared across all tenants. NOTE: The default policy setting in Neutron restricts usage of this property to administrative users only.

Boolean value expected.

Can be updated without replacement.

Attributes

admin_state_up

The administrative state of the firewall.

description

Description of the firewall.

firewall_policy_id

Unique identifier of the firewall policy used to create the firewall.

name

Name for the firewall.

shared

> **Note**
>
> UNSUPPORTED since 6.0.0 (Mitaka) - There is no such option during 5.0.0, so need to make this attribute unsupported, otherwise error will raised.

Shared status of this firewall.

show

Detailed information about resource.

status

The status of the firewall.

tenant_id

Id of the tenant owning the firewall.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Neutron::Firewall
    properties:
      admin_state_up: Boolean
      description: String
      firewall_policy_id: String
      name: String
      value_specs: {...}
```

OS::Neutron::FirewallPolicy

A resource for the FirewallPolicy resource in Neutron FWaaS.

FirewallPolicy resource is an ordered collection of firewall rules. A firewall policy can be shared across tenants.

Required Properties

firewall_rules

An ordered list of firewall rules to apply to the firewall.

List value expected.

Can be updated without replacement.

Optional Properties

audited

Whether this policy should be audited. When set to True, each time the firewall policy or the associated firewall rules are changed, this attribute will be set to False and will have to be explicitly set to True through an update operation.

Boolean value expected.

Can be updated without replacement.

Defaults to “False”.

description

Description for the firewall policy.

String value expected.

Can be updated without replacement.

name

Name for the firewall policy.

String value expected.

Can be updated without replacement.

shared

Whether this policy should be shared across all tenants.

Boolean value expected.

Can be updated without replacement.

Defaults to “False”.

Attributes

audited

Audit status of this firewall policy.

description

Description of the firewall policy.

firewall_rules

List of firewall rules in this firewall policy.

name

Name for the firewall policy.

shared

Shared status of this firewall policy.

show

Detailed information about resource.

tenant_id

Id of the tenant owning the firewall policy.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Neutron::FirewallPolicy
    properties:
      audited: Boolean
      description: String
      firewall_rules: [Value, Value, ...]
      name: String
      shared: Boolean
```

OS::Neutron::FirewallRule

A resource for the FirewallRule resource in Neutron FWaaS.

FirewallRule represents a collection of attributes like ports, ip addresses etc. which define match criteria and action (allow, or deny) that needs to be taken on the matched data traffic.

Optional Properties

action

Action to be performed on the traffic matching the rule.

String value expected.

Can be updated without replacement.

Defaults to “deny”.

Allowed values: allow, deny

description

Description for the firewall rule.

String value expected.

Can be updated without replacement.

destination_ip_address

Destination IP address or CIDR.

String value expected.

Can be updated without replacement.

Value must be of type net_cidr

destination_port

Destination port number or a range.

String value expected.

Can be updated without replacement.

enabled

Whether this rule should be enabled.

Boolean value expected.

Can be updated without replacement.

Defaults to “True”.

ip_version

Internet protocol version.

String value expected.

Can be updated without replacement.

Defaults to “4”.

Allowed values: 4, 6

name

Name for the firewall rule.

String value expected.

Can be updated without replacement.

protocol

Protocol for the firewall rule.

String value expected.

Can be updated without replacement.

Defaults to “any”.

Allowed values: tcp, udp, icmp, any

shared

Whether this rule should be shared across all tenants.

Boolean value expected.

Can be updated without replacement.

Defaults to “False”.

source_ip_address

Source IP address or CIDR.

String value expected.

Can be updated without replacement.

Value must be of type net_cidr

source_port

Source port number or a range.

String value expected.

Can be updated without replacement.

Attributes

action

Allow or deny action for this firewall rule.

description

Description of the firewall rule.

destination_ip_address

Destination ip_address for this firewall rule.

destination_port

Destination port range for this firewall rule.

enabled

Indicates whether this firewall rule is enabled or not.

firewall_policy_id

Unique identifier of the firewall policy to which this firewall rule belongs.

ip_version

Ip_version for this firewall rule.

name

Name for the firewall rule.

position

Position of the rule within the firewall policy.

protocol

Protocol value for this firewall rule.

shared

Shared status of this firewall rule.

show

Detailed information about resource.

source_ip_address

Source ip_address for this firewall rule.

source_port

Source port range for this firewall rule.

tenant_id

Id of the tenant owning the firewall.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Neutron::FirewallRule
    properties:
      action: String
      description: String
      destination_ip_address: String
      destination_port: String
      enabled: Boolean
      ip_version: String
      name: String
      protocol: String
      shared: Boolean
      source_ip_address: String
      source_port: String
```

OS::Neutron::FloatingIP

A resource for managing Neutron floating ips.

Floating IP addresses can change their association between routers by action of the user. One of the most common use cases for floating IPs is to provide public IP addresses to a private cloud, where there are a limited number of IP addresses available. Another is for a public cloud user to have a “static” IP address that can be reassigned when an instance is upgraded or moved.

Required Properties

floating_network

> **Note**
>
> Available since 2014.2 (Juno)

Network to allocate floating IP from.

String value expected.

Updates cause replacement.

Value must be of type neutron.network

Optional Properties

dns_domain

> **Note**
>
> Available since 7.0.0 (Newton)

DNS domain associated with floating ip.

String value expected.

Can be updated without replacement.

Value must be of type dns_domain

dns_name

> **Note**
>
> Available since 7.0.0 (Newton)

DNS name associated with floating ip.

String value expected.

Can be updated without replacement.

Value must be of type rel_dns_name

fixed_ip_address

IP address to use if the port has multiple addresses.

String value expected.

Can be updated without replacement.

Value must be of type ip_addr

floating_ip_address

> **Note**
>
> Available since 5.0.0 (Liberty)

IP address of the floating IP. NOTE: The default policy setting in Neutron restricts usage of this property to administrative users only.

String value expected.

Updates cause replacement.

Value must be of type ip_addr

port_id

ID of an existing port with at least one IP address to associate with this floating IP.

String value expected.

Can be updated without replacement.

Value must be of type neutron.port

value_specs

Extra parameters to include in the “floatingip” object in the creation request.

Parameters are often specific to installed hardware or extensions.

Map value expected.

Updates cause replacement.

Defaults to “{}”.

Attributes

fixed_ip_address

IP address of the associated port, if specified.

floating_ip_address

The allocated address of this IP.

floating_network_id

ID of the network in which this IP is allocated.

port_id

ID of the port associated with this IP.

router_id

ID of the router used as gateway, set when associated with a port.

show

Detailed information about resource.

tenant_id

The tenant owning this floating IP.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Neutron::FloatingIP
    properties:
      dns_domain: String
      dns_name: String
      fixed_ip_address: String
      floating_ip_address: String
      floating_network: String
```

```
port_id: String
value_specs: {...}
```

OS::Neutron::FloatingIPAssociation

A resource for associating floating ips and ports.

This resource allows associating a floating IP to a port with at least one IP address to associate with this floating IP.

Required Properties

floatingip_id

ID of the floating IP to associate.

String value expected.

Can be updated without replacement.

port_id

ID of an existing port with at least one IP address to associate with this floating IP.

String value expected.

Can be updated without replacement.

Value must be of type neutron.port

Optional Properties

fixed_ip_address

IP address to use if the port has multiple addresses.

String value expected.

Can be updated without replacement.

Value must be of type ip_addr

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Neutron::FloatingIPAssociation
    properties:
      fixed_ip_address: String
      floatingip_id: String
      port_id: String
```

OS::Neutron::IKEPolicy

A resource for IKE policy in Neutron.

The Internet Key Exchange policy identifies the authentication and encryption algorithm used during phase one and phase two negotiation of a VPN connection.

Optional Properties

auth_algorithm

Authentication hash algorithm for the ike policy.

String value expected.

Updates cause replacement.

Defaults to “sha1”.

Allowed values: sha1

description

Description for the ike policy.

String value expected.

Can be updated without replacement.

encryption_algorithm

Encryption algorithm for the ike policy.

String value expected.

Updates cause replacement.

Defaults to “aes-128”.

Allowed values: 3des, aes-128, aes-192, aes-256

ike_version

Version for the ike policy.

String value expected.

Updates cause replacement.

Defaults to “v1”.

Allowed values: v1, v2

lifetime

Safety assessment lifetime configuration for the ike policy.

Map value expected.

Updates cause replacement.

Map properties:

units

> Safety assessment lifetime units.
>
> String value expected.
>
> Updates cause replacement.
>
> Defaults to “seconds”.
>
> Allowed values: seconds, kilobytes

value

> Safety assessment lifetime value in specified units.
>
> Integer value expected.
>
> Updates cause replacement.
>
> Defaults to “3600”.

name

Name for the ike policy.

String value expected.

Can be updated without replacement.

pfs

Perfect forward secrecy in lowercase for the ike policy.

String value expected.

Updates cause replacement.

Defaults to “group5”.

Allowed values: group2, group5, group14

phase1_negotiation_mode

Negotiation mode for the ike policy.

String value expected.

Updates cause replacement.

Defaults to “main”.

Allowed values: main

Attributes

auth_algorithm

The authentication hash algorithm used by the ike policy.

description

The description of the ike policy.

encryption_algorithm

The encryption algorithm used by the ike policy.

ike_version

The version of the ike policy.

lifetime

The safety assessment lifetime configuration for the ike policy.

name

The name of the ike policy.

pfs

The perfect forward secrecy of the ike policy.

phase1_negotiation_mode

The negotiation mode of the ike policy.

show

Detailed information about resource.

tenant_id

The unique identifier of the tenant owning the ike policy.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Neutron::IKEPolicy
    properties:
      auth_algorithm: String
      description: String
      encryption_algorithm: String
      ike_version: String
      lifetime: {"value": Integer, "units": String}
      name: String
      pfs: String
      phase1_negotiation_mode: String
```

OS::Neutron::IPsecPolicy

A resource for IPsec policy in Neutron.

The IP security policy specifying the authentication and encryption algorithm, and encapsulation mode used for the established VPN connection.

Optional Properties

auth_algorithm

Authentication hash algorithm for the ipsec policy.

String value expected.

Updates cause replacement.

Defaults to “sha1”.

Allowed values: sha1

description

Description for the ipsec policy.

String value expected.

Can be updated without replacement.

encapsulation_mode

Encapsulation mode for the ipsec policy.

String value expected.

Updates cause replacement.

Defaults to “tunnel”.

Allowed values: tunnel, transport

encryption_algorithm

Encryption algorithm for the ipsec policy.

String value expected.

Updates cause replacement.

Defaults to “aes-128”.

Allowed values: 3des, aes-128, aes-192, aes-256

lifetime

Safety assessment lifetime configuration for the ipsec policy.

Map value expected.

Updates cause replacement.

Map properties:

units

> Safety assessment lifetime units.
>
> String value expected.
>
> Updates cause replacement.
>
> Defaults to “seconds”.
>
> Allowed values: seconds, kilobytes

value

> Safety assessment lifetime value in specified units.
>
> Integer value expected.

> Updates cause replacement.
>
> Defaults to “3600”.

name

Name for the ipsec policy.

String value expected.

Can be updated without replacement.

pfs

Perfect forward secrecy for the ipsec policy.

String value expected.

Updates cause replacement.

Defaults to “group5”.

Allowed values: group2, group5, group14

transform_protocol

Transform protocol for the ipsec policy.

String value expected.

Updates cause replacement.

Defaults to “esp”.

Allowed values: esp, ah, ah-esp

Attributes

auth_algorithm

The authentication hash algorithm of the ipsec policy.

description

The description of the ipsec policy.

encapsulation_mode

The encapsulation mode of the ipsec policy.

encryption_algorithm

The encryption algorithm of the ipsec policy.

lifetime

The safety assessment lifetime configuration of the ipsec policy.

name

The name of the ipsec policy.

pfs

The perfect forward secrecy of the ipsec policy.

show

Detailed information about resource.

tenant_id

The unique identifier of the tenant owning the ipsec policy.

transform_protocol

The transform protocol of the ipsec policy.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Neutron::IPsecPolicy
    properties:
      auth_algorithm: String
      description: String
      encapsulation_mode: String
      encryption_algorithm: String
      lifetime: {"value": Integer, "units": String}
      name: String
      pfs: String
      transform_protocol: String
```

OS::Neutron::IPsecSiteConnection

A resource for IPsec site connection in Neutron.

This resource has details for the site-to-site IPsec connection, including the peer CIDRs, MTU, peer address, DPD settings and status.

Required Properties

ikepolicy_id

Unique identifier for the ike policy associated with the ipsec site connection.

String value expected.

Updates cause replacement.

ipsecpolicy_id

Unique identifier for the ipsec policy associated with the ipsec site connection.

String value expected.

Updates cause replacement.

peer_address

Remote branch router public IPv4 address or IPv6 address or FQDN.

String value expected.

Updates cause replacement.

peer_cidrs

Remote subnet(s) in CIDR format.

List value expected.

Updates cause replacement.

List contents:

*

> String value expected.
>
> Updates cause replacement.
>
> Value must be of type net_cidr

peer_id

Remote branch router identity.

String value expected.

Updates cause replacement.

psk

Pre-shared key string for the ipsec site connection.

String value expected.

Updates cause replacement.

vpnservice_id

Unique identifier for the vpn service associated with the ipsec site connection.

String value expected.

Updates cause replacement.

Optional Properties

admin_state_up

Administrative state for the ipsec site connection.

Boolean value expected.

Can be updated without replacement.

Defaults to “True”.

description

Description for the ipsec site connection.

String value expected.

Can be updated without replacement.

dpd

Dead Peer Detection protocol configuration for the ipsec site connection.

Map value expected.

Updates cause replacement.

Map properties:

actions

> Controls DPD protocol mode.
>
> String value expected.
>
> Updates cause replacement.
>
> Defaults to “hold”.
>
> Allowed values: clear, disabled, hold, restart, restart-by-peer

interval

Number of seconds for the DPD delay.

Integer value expected.

Updates cause replacement.

Defaults to “30”.

timeout

Number of seconds for the DPD timeout.

Integer value expected.

Updates cause replacement.

Defaults to “120”.

initiator

Initiator state in lowercase for the ipsec site connection.

String value expected.

Updates cause replacement.

Defaults to “bi-directional”.

Allowed values: bi-directional, response-only

mtu

Maximum transmission unit size (in bytes) for the ipsec site connection.

Integer value expected.

Updates cause replacement.

Defaults to “1500”.

name

Name for the ipsec site connection.

String value expected.

Can be updated without replacement.

Attributes

admin_state_up

The administrative state of the ipsec site connection.

auth_mode

The authentication mode of the ipsec site connection.

description

The description of the ipsec site connection.

dpd

The dead peer detection protocol configuration of the ipsec site connection.

ikepolicy_id

The unique identifier of ike policy associated with the ipsec site connection.

initiator

The initiator of the ipsec site connection.

ipsecpolicy_id

The unique identifier of ipsec policy associated with the ipsec site connection.

mtu

The maximum transmission unit size (in bytes) of the ipsec site connection.

name

The name of the ipsec site connection.

peer_address

The remote branch router public IPv4 address or IPv6 address or FQDN.

peer_cidrs

The remote subnet(s) in CIDR format of the ipsec site connection.

peer_id

The remote branch router identity of the ipsec site connection.

psk

The pre-shared key string of the ipsec site connection.

route_mode

The route mode of the ipsec site connection.

show

Detailed information about resource.

status

The status of the ipsec site connection.

tenant_id

The unique identifier of the tenant owning the ipsec site connection.

vpnservice_id

The unique identifier of vpn service associated with the ipsec site connection.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Neutron::IPsecSiteConnection
    properties:
      admin_state_up: Boolean
      description: String
      dpd: {"timeout": Integer, "actions": String, "interval": Integer}
      ikepolicy_id: String
      initiator: String
      ipsecpolicy_id: String
      mtu: Integer
      name: String
      peer_address: String
      peer_cidrs: [String, String, ...]
      peer_id: String
      psk: String
      vpnservice_id: String
```

OS::Neutron::LBaaS::HealthMonitor

> **Note**
>
> Available since 6.0.0 (Mitaka)

A resource to handle load balancer health monitors.

This resource creates and manages Neutron LBaaS v2 healthmonitors, which watches status of the load balanced servers.

Required Properties

delay

The minimum time in milliseconds between regular connections of the member.

Integer value expected.

Can be updated without replacement.

The value must be at least 0.

max_retries

Number of permissible connection failures before changing the member status to INACTIVE.

Integer value expected.

Can be updated without replacement.

The value must be in the range 1 to 10.

pool

ID or name of the load balancing pool.

String value expected.

Updates cause replacement.

Value must be of type neutron.lbaas.pool

timeout

Maximum number of milliseconds for a monitor to wait for a connection to be established before it times out.

Integer value expected.

Can be updated without replacement.

The value must be at least 0.

type

One of predefined health monitor types.

String value expected.

Updates cause replacement.

Allowed values: PING, TCP, HTTP, HTTPS

Optional Properties

admin_state_up

The administrative state of the health monitor.

Boolean value expected.

Can be updated without replacement.

Defaults to “True”.

expected_codes

The HTTP status codes expected in response from the member to declare it healthy. Specify one of the following values: a single value, such as 200. a list, such as 200, 202. a range, such as 200-204.

String value expected.

Can be updated without replacement.

Defaults to “200”.

http_method

The HTTP method used for requests by the monitor of type HTTP.

String value expected.

Can be updated without replacement.

Defaults to “GET”.

Allowed values: GET, HEAD, POST, PUT, DELETE, TRACE, OPTIONS, CONNECT, PATCH

tenant_id

ID of the tenant who owns the health monitor.

String value expected.

Updates cause replacement.

url_path

The HTTP path used in the HTTP request used by the monitor to test a member health. A valid value is a string the begins with a forward slash (/).

String value expected.

Can be updated without replacement.

Defaults to “/”.

Attributes

pools

The list of Pools related to this monitor.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Neutron::LBaaS::HealthMonitor
    properties:
      admin_state_up: Boolean
      delay: Integer
      expected_codes: String
      http_method: String
      max_retries: Integer
      pool: String
      tenant_id: String
      timeout: Integer
      type: String
      url_path: String
```

OS::Neutron::LBaaS::L7Policy

> **Note**
>
> Available since 7.0.0 (Newton)

A resource for managing LBaaS v2 L7Policies.

This resource manages Neutron-LBaaS v2 L7Policies, which represent a collection of L7Rules. L7Policy holds the action that should be performed when the rules are matched (Redirect to Pool, Redirect to URL, Reject). L7Policy holds a Listener id, so a Listener can evaluate a collection of L7Policies. L7Policy will return True when all of the L7Rules that belong to this L7Policy are matched. L7Policies under a specific Listener are ordered and the first l7Policy that returns a match will be executed. When none of the policies match the request gets forwarded to listener.default_pool_id.

Required Properties

action

Action type of the policy.

String value expected.

Can be updated without replacement.

Allowed values: REJECT, REDIRECT_TO_POOL, REDIRECT_TO_URL

listener

ID or name of the listener this policy belongs to.

String value expected.

Updates cause replacement.

Value must be of type neutron.lbaas.listener

Optional Properties

admin_state_up

The administrative state of the policy.

Boolean value expected.

Can be updated without replacement.

Defaults to “True”.

description

Description of the policy.

String value expected.

Can be updated without replacement.

name

Name of the policy.

String value expected.

Can be updated without replacement.

position

L7 policy position in ordered policies list. This must be an integer starting from 1. If not specified, policy will be placed at the tail of existing policies list.

Number value expected.

Can be updated without replacement.

The value must be at least 1.

redirect_pool

ID or name of the pool for REDIRECT_TO_POOL action type.

String value expected.

Can be updated without replacement.

Value must be of type neutron.lbaas.pool

redirect_url

URL for REDIRECT_TO_URL action type. This should be a valid URL string.

String value expected.

Can be updated without replacement.

Attributes

rules

L7Rules associated with this policy.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Neutron::LBaaS::L7Policy
    properties:
      action: String
      admin_state_up: Boolean
      description: String
      listener: String
      name: String
      position: Number
      redirect_pool: String
      redirect_url: String
```

OS::Neutron::LBaaS::Listener

> **Note**
>
> Available since 6.0.0 (Mitaka)

A resource for managing LBaaS v2 Listeners.

This resource creates and manages Neutron LBaaS v2 Listeners, which represent a listening endpoint for the vip.

Required Properties

loadbalancer

ID or name of the load balancer with which listener is associated.

String value expected.

Updates cause replacement.

Value must be of type neutron.lbaas.loadbalancer

protocol

Protocol on which to listen for the client traffic.

String value expected.

Updates cause replacement.

Allowed values: TCP, HTTP, HTTPS, TERMINATED_HTTPS

protocol_port

TCP or UDP port on which to listen for client traffic.

Integer value expected.

Updates cause replacement.

The value must be in the range 1 to 65535.

Optional Properties

admin_state_up

The administrative state of this listener.

Boolean value expected.

Can be updated without replacement.

Defaults to “True”.

connection_limit

The maximum number of connections permitted for this load balancer. Defaults to -1, which is infinite.

Integer value expected.

Can be updated without replacement.

Defaults to “-1”.

The value must be at least -1.

default_tls_container_ref

Default TLS container reference to retrieve TLS information.

String value expected.

Can be updated without replacement.

description

Description of this listener.

String value expected.

Can be updated without replacement.

Defaults to “”.

name

Name of this listener.

String value expected.

Can be updated without replacement.

sni_container_refs

List of TLS container references for SNI.

List value expected.

Can be updated without replacement.

tenant_id

The ID of the tenant who owns the listener.

String value expected.

Updates cause replacement.

Attributes

default_pool_id

ID of the default pool this listener is associated to.

loadbalancers

ID of the load balancer this listener is associated to.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Neutron::LBaaS::Listener
    properties:
      admin_state_up: Boolean
      connection_limit: Integer
      default_tls_container_ref: String
      description: String
      loadbalancer: String
      name: String
```

```
protocol: String
protocol_port: Integer
sni_container_refs: [Value, Value, ...]
tenant_id: String
```

OS::Neutron::LBaaS::LoadBalancer

> **Note**
>
> Available since 6.0.0 (Mitaka)

A resource for creating LBaaS v2 Load Balancers.

This resource creates and manages Neutron LBaaS v2 Load Balancers, which allows traffic to be directed between servers.

Required Properties

vip_subnet

The name or ID of the subnet on which to allocate the VIP address.

String value expected.

Updates cause replacement.

Value must be of type neutron.subnet

Optional Properties

admin_state_up

The administrative state of this Load Balancer.

Boolean value expected.

Can be updated without replacement.

Defaults to “True”.

description

Description of this Load Balancer.

String value expected.

Can be updated without replacement.

Defaults to “”.

name

Name of this Load Balancer.

String value expected.

Can be updated without replacement.

provider

Provider for this Load Balancer.

String value expected.

Updates cause replacement.

Value must be of type neutron.lbaas.provider

tenant_id

The ID of the tenant who owns the Load Balancer. Only administrative users can specify a tenant ID other than their own.

String value expected.

Updates cause replacement.

Value must be of type keystone.project

vip_address

IP address for the VIP.

String value expected.

Updates cause replacement.

Value must be of type ip_addr

Attributes

show

Detailed information about resource.

vip_address

The VIP address of the LoadBalancer.

vip_port_id

The VIP port of the LoadBalancer.

vip_subnet_id

The VIP subnet of the LoadBalancer.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Neutron::LBaaS::LoadBalancer
    properties:
      admin_state_up: Boolean
      description: String
      name: String
      provider: String
      tenant_id: String
      vip_address: String
      vip_subnet: String
```

OS::Neutron::LBaaS::Pool

> **Note**
>
> Available since 6.0.0 (Mitaka)

A resource for managing LBaaS v2 Pools.

This resources manages Neutron-LBaaS v2 Pools, which represent a group of nodes. Pools define the subnet where nodes reside, balancing algorithm, and the nodes themselves.

Required Properties

lb_algorithm

The algorithm used to distribute load between the members of the pool.

String value expected.

Can be updated without replacement.

Allowed values: ROUND_ROBIN, LEAST_CONNECTIONS, SOURCE_IP

listener

Listener name or ID to be associated with this pool.

String value expected.

Updates cause replacement.

Value must be of type neutron.lbaas.listener

protocol

Protocol of the pool.

String value expected.

Updates cause replacement.

Allowed values: TCP, HTTP, HTTPS

Optional Properties

admin_state_up

The administrative state of this pool.

Boolean value expected.

Can be updated without replacement.

Defaults to “True”.

description

Description of this pool.

String value expected.

Can be updated without replacement.

Defaults to “”.

name

Name of this pool.

String value expected.

Can be updated without replacement.

session_persistence

Configuration of session persistence.

Map value expected.

Updates cause replacement.

Map properties:

cookie_name

Name of the cookie, required if type is APP_COOKIE.

String value expected.

Updates cause replacement.

type

Method of implementation of session persistence feature.

String value expected.

Updates cause replacement.

Allowed values: SOURCE_IP, HTTP_COOKIE, APP_COOKIE

Attributes

healthmonitor_id

ID of the health monitor associated with this pool.

listeners

Listener associated with this pool.

members

Members associated with this pool.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Neutron::LBaaS::Pool
    properties:
      admin_state_up: Boolean
      description: String
      lb_algorithm: String
      listener: String
      name: String
      protocol: String
      session_persistence: {"type": String, "cookie_name": String}
```

OS::Neutron::LBaaS::PoolMember

> **Note**
>
> Available since 6.0.0 (Mitaka)

A resource for managing LBaaS v2 Pool Members.

A pool member represents a single backend node.

Required Properties

address

IP address of the pool member on the pool network.

String value expected.

Updates cause replacement.

Value must be of type ip_addr

pool

Name or ID of the load balancing pool.

String value expected.

Updates cause replacement.

Value must be of type neutron.lbaas.pool

protocol_port

Port on which the pool member listens for requests or connections.

Integer value expected.

Updates cause replacement.

The value must be in the range 1 to 65535.

Optional Properties

admin_state_up

The administrative state of the pool member.

Boolean value expected.

Can be updated without replacement.

Defaults to “True”.

subnet

Subnet name or ID of this member.

String value expected.

Updates cause replacement.

Value must be of type neutron.subnet

weight

Weight of pool member in the pool (default to 1).

Integer value expected.

Can be updated without replacement.

Defaults to “1”.

The value must be in the range 0 to 256.

Attributes

address

The IP address of the pool member.

pool_id

The ID of the pool to which the pool member belongs.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Neutron::LBaaS::PoolMember
    properties:
      address: String
      admin_state_up: Boolean
      pool: String
      protocol_port: Integer
      subnet: String
      weight: Integer
```

OS::Neutron::MeteringLabel

> **Note**
>
> Available since 2014.1 (Icehouse)

A resource for creating neutron metering label.

The idea is to meter this at the L3 routers levels. The point is to allow operators to configure IP ranges and to assign a label to them. For example we will be able to set two labels; one for the internal traffic, and the other one for the external traffic. Each label will measure the traffic for a specific set of IP range. Then, bandwidth measurement will be sent for each label to the Oslo notification system and could be collected by Ceilometer.

Optional Properties

description

Description of the metering label.

String value expected.

Updates cause replacement.

name

Name of the metering label.

String value expected.

Updates cause replacement.

shared

> **Note**
>
> Available since 2015.1 (Kilo)

Whether the metering label should be shared across all tenants.

Boolean value expected.

Updates cause replacement.

Defaults to “False”.

Attributes

description

Description of the metering label.

name

Name of the metering label.

shared

Shared status of the metering label.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Neutron::MeteringLabel
    properties:
      description: String
      name: String
      shared: Boolean
```

OS::Neutron::MeteringRule

> **Note**
>
> Available since 2014.1 (Icehouse)

A resource to create rule for some label.

Resource for allowing specified label to measure the traffic for a specific set of ip range.

Required Properties

metering_label_id

The metering label ID to associate with this metering rule.

String value expected.

Updates cause replacement.

remote_ip_prefix

Indicates remote IP prefix to be associated with this metering rule.

String value expected.

Updates cause replacement.

Optional Properties

direction

The direction in which metering rule is applied, either ingress or egress.

String value expected.

Updates cause replacement.

Defaults to “ingress”.

Allowed values: ingress, egress

excluded

Specify whether the remote_ip_prefix will be excluded or not from traffic counters of the metering label. For example to not count the traffic of a specific IP address of a range.

Boolean value expected.

Updates cause replacement.

Defaults to “False”.

Attributes

direction

The direction in which metering rule is applied.

excluded

Exclude state for cidr.

metering_label_id

The metering label ID to associate with this metering rule.

remote_ip_prefix

CIDR to be associated with this metering rule.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Neutron::MeteringRule
    properties:
      direction: String
      excluded: Boolean
      metering_label_id: String
      remote_ip_prefix: String
```

OS::Neutron::Net

A resource for managing Neutron net.

A network is a virtual isolated layer-2 broadcast domain which is typically reserved to the tenant who created it, unless the network has been explicitly configured to be shared.

Optional Properties

admin_state_up

A boolean value specifying the administrative status of the network.

Boolean value expected.

Can be updated without replacement.

Defaults to “True”.

dhcp_agent_ids

The IDs of the DHCP agent to schedule the network. Note that the default policy setting in Neutron restricts usage of this property to administrative users only.

List value expected.

Can be updated without replacement.

dns_domain

> **Note**
>
> Available since 7.0.0 (Newton)

DNS domain associated with this network.

String value expected.

Can be updated without replacement.

Value must be of type dns_domain

name

A string specifying a symbolic name for the network, which is not required to be unique.

String value expected.

Can be updated without replacement.

port_security_enabled

> **Note**
>
> Available since 5.0.0 (Liberty)

Flag to enable/disable port security on the network. It provides the default value for the attribute of the ports created on this network.

Boolean value expected.

Can be updated without replacement.

qos_policy

> **Note**
>
> Available since 6.0.0 (Mitaka)

The name or ID of QoS policy to attach to this network.

String value expected.

Can be updated without replacement.

Value must be of type neutron.qos_policy

shared

Whether this network should be shared across all tenants. Note that the default policy setting restricts usage of this attribute to administrative users only.

Boolean value expected.

Can be updated without replacement.

Defaults to “False”.

tenant_id

The ID of the tenant which will own the network. Only administrative users can set the tenant identifier; this cannot be changed using authorization policies.

String value expected.

Updates cause replacement.

value_specs

Extra parameters to include in the request. Parameters are often specific to installed hardware or extensions.

Map value expected.

Can be updated without replacement.

Defaults to “{}”.

Attributes

admin_state_up

The administrative status of the network.

mtu

> **Note**
>
> Available since 5.0.0 (Liberty)

The maximum transmission unit size(in bytes) for the network.

name

The name of the network.

port_security_enabled

> **Note**
>
> Available since 5.0.0 (Liberty)

Port security enabled of the network.

qos_policy_id

> **Note**
>
> Available since 6.0.0 (Mitaka)

The QoS policy ID attached to this network.

show

Detailed information about resource.

status

The status of the network.

subnets

Subnets of this network.

tenant_id

The tenant owning this network.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Neutron::Net
    properties:
      admin_state_up: Boolean
      dhcp_agent_ids: [Value, Value, ...]
      dns_domain: String
      name: String
      port_security_enabled: Boolean
      qos_policy: String
      shared: Boolean
      tenant_id: String
      value_specs: {...}
```

OS::Neutron::NetworkGateway

> **Note**
>
> Available since 2014.1 (Icehouse)

Network Gateway resource in Neutron Network Gateway.

Resource for connecting internal networks with specified devices.

Required Properties

devices

Device info for this network gateway.

List value expected.

Can be updated without replacement.

The length must be at least 1.

List contents:

*

> Map value expected.
>
> Can be updated without replacement.
>
> *Map properties:*
>
> id
>
> > The device id for the network gateway.
> >
> > String value expected.
> >
> > Can be updated without replacement.
>
> interface_name
>
> > The interface name for the network gateway.
> >
> > String value expected.
> >
> > Can be updated without replacement.

Optional Properties

connections

Connection info for this network gateway.

List value expected.

Can be updated without replacement.

Defaults to “{}”.

List contents:

*

Map value expected.

Can be updated without replacement.

Map properties:

network

Note

Available since 2014.2 (Juno)

The internal network to connect on the network gateway.

String value expected.

Can be updated without replacement.

Value must be of type neutron.network

segmentation_type

L2 segmentation strategy on the external side of the network gateway.

String value expected.

Can be updated without replacement.

Defaults to “flat”.

Allowed values: flat, vlan

segmentation_id

> The id for L2 segment on the external side of the network gateway. Must be specified when using vlan.
>
> Integer value expected.
>
> Can be updated without replacement.
>
> The value must be in the range 0 to 4094.

name

The name of the network gateway.

String value expected.

Can be updated without replacement.

Attributes

default

A boolean value of default flag.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Neutron::NetworkGateway
    properties:
      connections: [{"segmentation_id": Integer, "network_id": String,
"segmentation_type": String, "network": String}, {"segmentation_id":
Integer, "network_id": String, "segmentation_type": String, "network":
String}, ...]
      devices: [{"interface_name": String, "id": String},
{"interface_name": String, "id": String}, ...]
      name: String
```

OS::Neutron::Port

A resource for managing Neutron ports.

A port represents a virtual switch port on a logical network switch. Virtual instances attach their interfaces into ports. The logical port also defines the MAC address and the IP address(es) to be assigned to the interfaces plugged into them. When IP addresses are associated to a port, this also implies the port is associated with a subnet, as the IP address was taken from the allocation pool for a specific subnet.

Required Properties

network

> **Note**
>
> Available since 2014.2 (Juno)

Network this port belongs to. If you plan to use current port to assign Floating IP, you should specify fixed_ips with subnet. Note if this changes to a different network update, the port will be replaced.

String value expected.

Updates cause replacement.

Value must be of type neutron.network

Optional Properties

admin_state_up

The administrative state of this port.

Boolean value expected.

Can be updated without replacement.

Defaults to “True”.

allowed_address_pairs

Additional MAC/IP address pairs allowed to pass through the port.

List value expected.

Updates cause replacement.

List contents:

*

Map value expected.

Updates cause replacement.

Map properties:

ip_address

IP address to allow through this port.

String value expected.

Updates cause replacement.

Value must be of type net_cidr

mac_address

MAC address to allow through this port.

String value expected.

Updates cause replacement.

Value must be of type mac_addr

binding:vnic_type

Note

Available since 2015.1 (Kilo)

The vnic type to be bound on the neutron port. To support SR-IOV PCI passthrough networking, you can request that the neutron port to be realized as normal (virtual nic), direct (pci passthrough), or macvtap (virtual interface with a tap-like software interface). Note that this only works for Neutron deployments that support the bindings extension.

String value expected.

Can be updated without replacement.

Allowed values: normal, direct, macvtap

device_id

Device ID of this port.

String value expected.

Can be updated without replacement.

device_owner

Name of the network owning the port. The value is typically network:floatingip or network:router_interface or network:dhcp.

String value expected.

Can be updated without replacement.

dns_name

> **Note**
>
> Available since 7.0.0 (Newton)

DNS name associated with the port.

String value expected.

Can be updated without replacement.

Value must be of type dns_name

fixed_ips

Desired IPs for this port.

List value expected.

Can be updated without replacement.

List contents:

*

> Map value expected.
>
> Can be updated without replacement.
>
> *Map properties:*
>
> ip_address
>
> IP address desired in the subnet for this port.

String value expected.

Can be updated without replacement.

Value must be of type ip_addr

subnet

Note

Available since 2014.2 (Juno)

Subnet in which to allocate the IP address for this port.

String value expected.

Can be updated without replacement.

Value must be of type neutron.subnet

mac_address

MAC address to give to this port.

String value expected.

Updates cause replacement.

Value must be of type mac_addr

name

A symbolic name for this port.

String value expected.

Can be updated without replacement.

port_security_enabled

Note

Available since 5.0.0 (Liberty)

Flag to enable/disable port security on the port. When disable this feature(set it to False), there will be no packages filtering, like security-group and address-pairs.

Boolean value expected.

Can be updated without replacement.

qos_policy

> **Note**
>
> Available since 6.0.0 (Mitaka)

The name or ID of QoS policy to attach to this port.

String value expected.

Can be updated without replacement.

Value must be of type neutron.qos_policy

security_groups

Security group IDs to associate with this port.

List value expected.

Can be updated without replacement.

value_specs

Extra parameters to include in the request.

Map value expected.

Can be updated without replacement.

Defaults to “{}”.

replacement_policy

> **Note**
>
> DEPRECATED since 6.0.0 (Mitaka) - Replacement policy used to work around flawed nova/neutron port interaction which has been fixed since Liberty.
>
> **Note**
>
> Available since 2014.2 (Juno)

Policy on how to respond to a stack-update for this resource. REPLACE_ALWAYS will replace the port regardless of any property changes. AUTO will update the existing port for any changed update-allowed property.

String value expected.

Can be updated without replacement.

Defaults to “AUTO”.

Allowed values: REPLACE_ALWAYS, AUTO

Attributes

admin_state_up

The administrative state of this port.

allowed_address_pairs

Additional MAC/IP address pairs allowed to pass through a port.

device_id

Unique identifier for the device.

device_owner

Name of the network owning the port.

dns_assignment

> **Note**
>
> Available since 7.0.0 (Newton)

The DNS assigned to this port.

fixed_ips

Fixed IP addresses.

mac_address

MAC address of the port.

name

Friendly name of the port.

network_id

Unique identifier for the network owning the port.

port_security_enabled

> **Note**
>
> Available since 5.0.0 (Liberty)

Port security enabled of the port.

qos_policy_id

> **Note**
>
> Available since 6.0.0 (Mitaka)

The QoS policy ID attached to this port.

security_groups

A list of security groups for the port.

show

Detailed information about resource.

status

The status of the port.

subnets

A list of all subnet attributes for the port.

tenant_id

Tenant owning the port.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Neutron::Port
    properties:
      admin_state_up: Boolean
      allowed_address_pairs: [{"ip_address": String, "mac_address":
String}, {"ip_address": String, "mac_address": String}, ...]
      binding:vnic_type: String
```

```
      device_id: String
      device_owner: String
      dns_name: String
      fixed_ips: [{"ip_address": String, "subnet": String, "subnet_id":
String}, {"ip_address": String, "subnet": String, "subnet_id": String},
...]
      mac_address: String
      name: String
      network: String
      port_security_enabled: Boolean
      qos_policy: String
      security_groups: [Value, Value, ...]
      value_specs: {...}
```

OS::Neutron::ProviderNet

> **Note**
>
> Available since 2014.1 (Icehouse)

A resource for managing Neutron provider networks.

Provider networks specify details of physical realisation of the existing network.

Required Properties

network_type

A string specifying the provider network type for the network.

String value expected.

Can be updated without replacement.

Allowed values: vlan, flat

Optional Properties

admin_state_up

A boolean value specifying the administrative status of the network.

Boolean value expected.

Can be updated without replacement.

Defaults to “True”.

name

A string specifying a symbolic name for the network, which is not required to be unique.

String value expected.

Can be updated without replacement.

physical_network

A string specifying physical network mapping for the network.

String value expected.

Can be updated without replacement.

router_external

> **Note**
>
> Available since 6.0.0 (Mitaka)

Whether the network contains an external router.

Boolean value expected.

Can be updated without replacement.

Defaults to “False”.

segmentation_id

A string specifying the segmentation id for the network.

String value expected.

Can be updated without replacement.

shared

Whether this network should be shared across all tenants.

Boolean value expected.

Can be updated without replacement.

Defaults to “True”.

Attributes

show

Detailed information about resource.

status

The status of the network.

subnets

Subnets of this network.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Neutron::ProviderNet
    properties:
      admin_state_up: Boolean
      name: String
      network_type: String
      physical_network: String
      router_external: Boolean
      segmentation_id: String
      shared: Boolean
```

OS::Neutron::QoSBandwidthLimitRule

> **Note**
>
> Available since 6.0.0 (Mitaka)

A resource for Neutron QoS bandwidth limit rule.

This rule can be associated with QoS policy, and then the policy can be used by neutron port and network, to provide bandwidth limit QoS capabilities.

The default policy usage of this resource is limited to administrators only.

Required Properties

max_kbps

Max bandwidth in kbps.

Integer value expected.

Can be updated without replacement.

The value must be at least 0.

policy

ID or name of the QoS policy.

String value expected.

Updates cause replacement.

Value must be of type neutron.qos_policy

Optional Properties

max_burst_kbps

Max burst bandwidth in kbps.

Integer value expected.

Can be updated without replacement.

Defaults to “0”.

The value must be at least 0.

tenant_id

The owner tenant ID of this rule.

String value expected.

Updates cause replacement.

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Neutron::QoSBandwidthLimitRule
    properties:
      max_burst_kbps: Integer
      max_kbps: Integer
      policy: String
      tenant_id: String
```

OS::Neutron::QoSDscpMarkingRule

> **Note**
>
> Available since 7.0.0 (Newton)

A resource for Neutron QoS DSCP marking rule.

This rule can be associated with QoS policy, and then the policy can be used by neutron port and network, to provide DSCP marking QoS capabilities.

The default policy usage of this resource is limited to administrators only.

Required Properties

dscp_mark

DSCP mark between 0 and 56, except 2-6, 42, 44, and 50-54.

Integer value expected.

Can be updated without replacement.

Allowed values: 0, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 46, 48, 56

policy

ID or name of the QoS policy.

String value expected.

Updates cause replacement.

Value must be of type neutron.qos_policy

Optional Properties

tenant_id

The owner tenant ID of this rule.

String value expected.

Updates cause replacement.

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Neutron::QoSDscpMarkingRule
    properties:
      dscp_mark: Integer
      policy: String
      tenant_id: String
```

OS::Neutron::QoSPolicy

> **Note**
>
> Available since 6.0.0 (Mitaka)

A resource for Neutron QoS Policy.

This QoS policy can be associated with neutron resources, such as port and network, to provide QoS capabilities.

The default policy usage of this resource is limited to administrators only.

Optional Properties

description

The description for the QoS policy.

String value expected.

Can be updated without replacement.

name

The name for the QoS policy.

String value expected.

Can be updated without replacement.

shared

Whether this QoS policy should be shared to other tenants.

Boolean value expected.

Can be updated without replacement.

Defaults to “False”.

tenant_id

The owner tenant ID of this QoS policy.

String value expected.

Updates cause replacement.

Attributes

rules

A list of all rules for the QoS policy.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Neutron::QoSPolicy
    properties:
      description: String
      name: String
      shared: Boolean
      tenant_id: String
```

OS::Neutron::RBACPolicy

> **Note**
>
> Available since 6.0.0 (Mitaka)

A Resource for managing RBAC policy in Neutron.

This resource creates and manages Neutron RBAC policy, which allows to share Neutron networks to subsets of tenants.

Required Properties

action

Action for the RBAC policy.

String value expected.

Updates cause replacement.

object_id

ID or name of the RBAC object.

String value expected.

Updates cause replacement.

object_type

Type of the object that RBAC policy affects.

String value expected.

Updates cause replacement.

target_tenant

ID of the tenant to which the RBAC policy will be enforced.

String value expected.

Can be updated without replacement.

Optional Properties

tenant_id

The owner tenant ID. Only required if the caller has an administrative role and wants to create a RBAC for another tenant.

String value expected.

Updates cause replacement.

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Neutron::RBACPolicy
    properties:
      action: String
      object_id: String
      object_type: String
      target_tenant: String
      tenant_id: String
```

OS::Neutron::Router

A resource that implements Neutron router.

Router is a physical or virtual network device that passes network traffic between different networks.

Optional Properties

admin_state_up

The administrative state of the router.

Boolean value expected.

Can be updated without replacement.

Defaults to “True”.

distributed

> **Note**
>
> Available since 2015.1 (Kilo)

Indicates whether or not to create a distributed router. NOTE: The default policy setting in Neutron restricts usage of this property to administrative users only. This property can not be used in conjunction with the L3 agent ID.

Boolean value expected.

Updates cause replacement.

external_gateway_info

External network gateway configuration for a router.

Map value expected.

Can be updated without replacement.

Map properties:

enable_snat

Enables Source NAT on the router gateway. NOTE: The default policy setting in Neutron restricts usage of this property to administrative users only.

Boolean value expected.

Can be updated without replacement.

external_fixed_ips

Note

Available since 6.0.0 (Mitaka)

External fixed IP addresses for the gateway.

List value expected.

Can be updated without replacement.

List contents:

*

Map value expected.

Can be updated without replacement.

Map properties:

ip_address

External fixed IP address.

String value expected.

Can be updated without replacement.

Value must be of type ip_addr

subnet

Subnet of external fixed IP address.

String value expected.

Can be updated without replacement.

Value must be of type neutron.subnet

network

ID or name of the external network for the gateway.

String value expected.

Can be updated without replacement.

ha

Note

Available since 2015.1 (Kilo)

Indicates whether or not to create a highly available router. NOTE: The default policy setting in Neutron restricts usage of this property to administrative users only. And now neutron do not support distributed and ha at the same time.

Boolean value expected.

Updates cause replacement.

l3_agent_ids

Note

Available since 2015.1 (Kilo)

ID list of the L3 agent. User can specify multi-agents for highly available router. NOTE: The default policy setting in Neutron restricts usage of this property to administrative users only.

List value expected.

Can be updated without replacement.

List contents:

*

String value expected.

Can be updated without replacement.

name

The name of the router.

String value expected.

Can be updated without replacement.

value_specs

Extra parameters to include in the creation request.

Map value expected.

Can be updated without replacement.

Defaults to “{}”.

Attributes

admin_state_up

Administrative state of the router.

external_gateway_info

Gateway network for the router.

name

Friendly name of the router.

show

Detailed information about resource.

status

The status of the router.

tenant_id

Tenant owning the router.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Neutron::Router
    properties:
      admin_state_up: Boolean
      distributed: Boolean
      external_gateway_info: {"enable_snat": Boolean,
"external_fixed_ips": [{"ip_address": String, "subnet": String},
{"ip_address": String, "subnet": String}, ...], "network": String}
      ha: Boolean
      l3_agent_ids: [String, String, ...]
      name: String
      value_specs: {...}
```

OS::Neutron::RouterInterface

A resource for managing Neutron router interfaces.

Router interfaces associate routers with existing subnets or ports.

Required Properties

router

The router.

String value expected.

Updates cause replacement.

Value must be of type neutron.router

Optional Properties

port

> **Note**
>
> Available since 2015.1 (Kilo)

The port, either subnet or port should be specified.

String value expected.

Updates cause replacement.

Value must be of type neutron.port

subnet

The subnet, either subnet or port should be specified.

String value expected.

Updates cause replacement.

Value must be of type neutron.subnet

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Neutron::RouterInterface
    properties:
      port: String
      router: String
      subnet: String
```

OS::Neutron::SecurityGroup

> **Note**
>
> Available since 2014.1 (Icehouse)

A resource for managing Neutron security groups.

Security groups are sets of IP filter rules that are applied to an instance's networking. They are project specific, and project members can edit the default rules for their group and add new rules sets. All projects have a "default" security group, which is applied to instances that have no other security group defined.

Optional Properties

description

Description of the security group.

String value expected.

Can be updated without replacement.

name

A string specifying a symbolic name for the security group, which is not required to be unique.

String value expected.

Can be updated without replacement.

rules

List of security group rules.

List value expected.

Can be updated without replacement.

Defaults to “[]”.

List contents:

*

Map value expected.

Can be updated without replacement.

Map properties:

direction

The direction in which the security group rule is applied. For a compute instance, an ingress security group rule matches traffic that is incoming (ingress) for that instance. An egress rule is applied to traffic leaving the instance.

String value expected.

Can be updated without replacement.

Defaults to “ingress”.

Allowed values: ingress, egress

ethertype

Ethertype of the traffic.

String value expected.

Can be updated without replacement.

Defaults to “IPv4”.

Allowed values: IPv4, IPv6

port_range_max

The maximum port number in the range that is matched by the security group rule. The port_range_min attribute constrains the port_range_max attribute. If the protocol is ICMP, this value must be an ICMP type.

Integer value expected.

Can be updated without replacement.

The value must be in the range 0 to 65535.

port_range_min

The minimum port number in the range that is matched by the security group rule. If the protocol is TCP or UDP, this value must be less than or equal to the value of the port_range_max attribute. If the protocol is ICMP, this value must be an ICMP type.

Integer value expected.

Can be updated without replacement.

The value must be in the range 0 to 65535.

protocol

The protocol that is matched by the security group rule. Valid values include tcp, udp, and icmp.

String value expected.

Can be updated without replacement.

remote_group_id

The remote group ID to be associated with this security group rule. If no value is specified then this rule will use this security group for the remote_group_id. The remote mode parameter must be set to “remote_group_id”.

String value expected.

Can be updated without replacement.

Value must be of type neutron.security_group

remote_ip_prefix

The remote IP prefix (CIDR) to be associated with this security group rule.

String value expected.

Can be updated without replacement.

Value must be of type net_cidr

remote_mode

Whether to specify a remote group or a remote IP prefix.

String value expected.

Can be updated without replacement.

Defaults to “remote_ip_prefix”.

Allowed values: remote_ip_prefix, remote_group_id

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Neutron::SecurityGroup
    properties:
      description: String
      name: String
      rules: [{"protocol": String, "port_range_max": Integer,
"remote_mode": String, "direction": String, "remote_group_id": String,
"port_range_min": Integer, "remote_ip_prefix": String, "ethertype":
String}, {"protocol": String, "port_range_max": Integer, "remote_mode":
String, "direction": String, "remote_group_id": String, "port_range_min":
Integer, "remote_ip_prefix": String, "ethertype": String}, ...]
```

OS::Neutron::SecurityGroupRule

> **Note**
>
> Available since 7.0.0 (Newton)

A resource for managing Neutron security group rules.

Rules to use in security group resource.

Required Properties

security_group

Security group name or ID to add rule.

String value expected.

Updates cause replacement.

Value must be of type neutron.security_group

Optional Properties

description

Description of the security group rule.

String value expected.

Updates cause replacement.

direction

The direction in which the security group rule is applied. For a compute instance, an ingress security group rule matches traffic that is incoming (ingress) for that instance. An egress rule is applied to traffic leaving the instance.

String value expected.

Updates cause replacement.

Defaults to “ingress”.

Allowed values: ingress, egress

ethertype

Ethertype of the traffic.

String value expected.

Updates cause replacement.

Defaults to “IPv4”.

Allowed values: IPv4, IPv6

port_range_max

The maximum port number in the range that is matched by the security group rule. The port_range_min attribute constrains the port_range_max attribute. If the protocol is ICMP, this value must be an ICMP code.

Integer value expected.

Updates cause replacement.

The value must be in the range 0 to 65535.

port_range_min

The minimum port number in the range that is matched by the security group rule. If the protocol is TCP or UDP, this value must be less than or equal to the value of the port_range_max attribute. If the protocol is ICMP, this value must be an ICMP type.

Integer value expected.

Updates cause replacement.

The value must be in the range 0 to 65535.

protocol

The protocol that is matched by the security group rule. Allowed values are ah, dccp, egp, esp, gre, icmp, icmpv6, igmp, ipv6-encap, ipv6-frag, ipv6-icmp, ipv6-nonxt, ipv6-opts, ipv6-route, ospf, pgm, rsvp, sctp, tcp, udp, udplite, vrrp and integer representations [0-255].

String value expected.

Updates cause replacement.

Defaults to “tcp”.

Allowed values: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, ah, dccp, egp, esp, gre, icmp, icmpv6, igmp, ipv6-encap, ipv6-frag, ipv6-icmp, ipv6-nonxt, ipv6-opts, ipv6-route, ospf, pgm, rsvp, sctp, tcp, udp, udplite, vrrp

remote_group

The remote group name or ID to be associated with this security group rule.

String value expected.

Updates cause replacement.

Value must be of type neutron.security_group

remote_ip_prefix

The remote IP prefix (CIDR) to be associated with this security group rule.

String value expected.

Updates cause replacement.

Value must be of type net_cidr

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Neutron::SecurityGroupRule
    properties:
      description: String
      direction: String
      ethertype: String
      port_range_max: Integer
      port_range_min: Integer
      protocol: String
      remote_group: String
      remote_ip_prefix: String
      security_group: String
```

OS::Neutron::Subnet

A resource for managing Neutron subnets.

A subnet represents an IP address block that can be used for assigning IP addresses to virtual instances. Each subnet must have a CIDR and must be associated with a network. IPs can be either selected from the whole subnet CIDR, or from “allocation pools” that can be specified by the user.

Required Properties

network

> **Note**
>
> Available since 2014.2 (Juno)

The ID of the attached network.

String value expected.

Updates cause replacement.

Value must be of type neutron.network

Optional Properties

allocation_pools

The start and end addresses for the allocation pools.

List value expected.

Can be updated without replacement.

List contents:

*

Map value expected.

Can be updated without replacement.

Map properties:

end

End address for the allocation pool.

String value expected.

Can be updated without replacement.

Value must be of type ip_addr

start

Start address for the allocation pool.

String value expected.

Can be updated without replacement.

Value must be of type ip_addr

cidr

The CIDR.

String value expected.

Updates cause replacement.

Value must be of type net_cidr

dns_nameservers

A specified set of DNS name servers to be used.

List value expected.

Can be updated without replacement.

Defaults to “[]”.

enable_dhcp

Set to true if DHCP is enabled and false if DHCP is disabled.

Boolean value expected.

Can be updated without replacement.

Defaults to “True”.

gateway_ip

The gateway IP address. Set to any of [null | ~ | “”] to create/update a subnet without a gateway. If omitted when creation, neutron will assign the first free IP address within the subnet to the gateway automatically. If remove this from template when update, the old gateway IP address will be detached.

String value expected.

Can be updated without replacement.

host_routes

A list of host route dictionaries for the subnet.

List value expected.

Can be updated without replacement.

List contents:

*

Map value expected.

Can be updated without replacement.

Map properties:

destination

The destination for static route.

String value expected.

Can be updated without replacement.

Value must be of type net_cidr

nexthop

The next hop for the destination.

String value expected.

Can be updated without replacement.

Value must be of type ip_addr

ip_version

The IP version, which is 4 or 6.

Integer value expected.

Updates cause replacement.

Defaults to “4”.

Allowed values: 4, 6

ipv6_address_mode

Note

Available since 2015.1 (Kilo)

IPv6 address mode.

String value expected.

Updates cause replacement.

Allowed values: dhcpv6-stateful, dhcpv6-stateless, slaac

ipv6_ra_mode

Note

Available since 2015.1 (Kilo)

IPv6 RA (Router Advertisement) mode.

String value expected.

Updates cause replacement.

Allowed values: dhcpv6-stateful, dhcpv6-stateless, slaac

name

The name of the subnet.

String value expected.

Can be updated without replacement.

prefixlen

> **Note**
>
> Available since 6.0.0 (Mitaka)

Prefix length for subnet allocation from subnet pool.

Integer value expected.

Updates cause replacement.

The value must be at least 0.

subnetpool

> **Note**
>
> Available since 6.0.0 (Mitaka)

The name or ID of the subnet pool.

String value expected.

Updates cause replacement.

Value must be of type neutron.subnetpool

tenant_id

The ID of the tenant who owns the network. Only administrative users can specify a tenant ID other than their own.

String value expected.

Updates cause replacement.

value_specs

Extra parameters to include in the request.

Map value expected.

Can be updated without replacement.

Defaults to “{}”.

Attributes

allocation_pools

Ip allocation pools and their ranges.

cidr

CIDR block notation for this subnet.

dns_nameservers

List of dns nameservers.

enable_dhcp

‘true’ if DHCP is enabled for this subnet; ‘false’ otherwise.

gateway_ip

Ip of the subnet’s gateway.

host_routes

Additional routes for this subnet.

ip_version

Ip version for the subnet.

name

Friendly name of the subnet.

network_id

Parent network of the subnet.

show

Detailed information about resource.

tenant_id

Tenant owning the subnet.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Neutron::Subnet
    properties:
      allocation_pools: [{"end": String, "start": String}, {"end": String,
"start": String}, ...]
      cidr: String
      dns_nameservers: [Value, Value, ...]
      enable_dhcp: Boolean
      gateway_ip: String
      host_routes: [{"destination": String, "nexthop": String},
{"destination": String, "nexthop": String}, ...]
      ip_version: Integer
      ipv6_address_mode: String
      ipv6_ra_mode: String
      name: String
      network: String
      prefixlen: Integer
      subnetpool: String
      tenant_id: String
      value_specs: {...}
```

OS::Neutron::SubnetPool

> **Note**
>
> Available since 6.0.0 (Mitaka)

A resource that implements neutron subnet pool.

This resource can be used to create a subnet pool with a large block of addresses and create subnets from it.

Required Properties

prefixes

List of subnet prefixes to assign.

List value expected.

Can be updated without replacement.

The length must be at least 1.

List contents:

*

String value expected.

Can be updated without replacement.

Value must be of type net_cidr

Optional Properties

address_scope

An address scope ID to assign to the subnet pool.

String value expected.

Can be updated without replacement.

Value must be of type neutron.address_scope

default_prefixlen

The size of the prefix to allocate when the cidr or prefixlen attributes are not specified while creating a subnet.

Integer value expected.

Can be updated without replacement.

The value must be at least 0.

default_quota

A per-tenant quota on the prefix space that can be allocated from the subnet pool for tenant subnets.

Integer value expected.

Can be updated without replacement.

The value must be at least 0.

is_default

Whether this is default IPv4/IPv6 subnet pool. There can only be one default subnet pool for each IP family. Note that the default policy setting restricts administrative users to set this to True.

Boolean value expected.

Can be updated without replacement.

Defaults to “False”.

max_prefixlen

Maximum prefix size that can be allocated from the subnet pool.

Integer value expected.

Can be updated without replacement.

The value must be at least 0.

min_prefixlen

Smallest prefix size that can be allocated from the subnet pool.

Integer value expected.

Can be updated without replacement.

The value must be at least 0.

name

Name of the subnet pool.

String value expected.

Can be updated without replacement.

shared

Whether the subnet pool will be shared across all tenants. Note that the default policy setting restricts usage of this attribute to administrative users only.

Boolean value expected.

Updates cause replacement.

Defaults to “False”.

tenant_id

The ID of the tenant who owns the subnet pool. Only administrative users can specify a tenant ID other than their own.

String value expected.

Updates cause replacement.

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Neutron::SubnetPool
    properties:
      address_scope: String
      default_prefixlen: Integer
      default_quota: Integer
      is_default: Boolean
      max_prefixlen: Integer
      min_prefixlen: Integer
      name: String
      prefixes: [String, String, ...]
      shared: Boolean
      tenant_id: String
```

OS::Neutron::VPNService

A resource for VPN service in Neutron.

VPN service is a high level object that associates VPN with a specific subnet and router.

Required Properties

router

> **Note**
>
> Available since 2015.1 (Kilo)

The router to which the vpn service will be inserted.

String value expected.

Updates cause replacement.

Value must be of type neutron.router

subnet

> **Note**
>
> Available since 2014.2 (Juno)

Subnet in which the vpn service will be created.

String value expected.

Updates cause replacement.

Value must be of type neutron.subnet

Optional Properties

admin_state_up

Administrative state for the vpn service.

Boolean value expected.

Can be updated without replacement.

Defaults to “True”.

description

Description for the vpn service.

String value expected.

Can be updated without replacement.

name

Name for the vpn service.

String value expected.

Can be updated without replacement.

Attributes

admin_state_up

The administrative state of the vpn service.

description

The description of the vpn service.

name

The name of the vpn service.

router_id

The unique identifier of the router to which the vpn service was inserted.

show

Detailed information about resource.

status

The status of the vpn service.

subnet_id

The unique identifier of the subnet in which the vpn service was created.

tenant_id

The unique identifier of the tenant owning the vpn service.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Neutron::VPNService
    properties:
      admin_state_up: Boolean
      description: String
      name: String
      router: String
      subnet: String
```

OS::Nova::Flavor

> **Note**
>
> Available since 2014.2 (Juno)

A resource for creating OpenStack virtual hardware templates.

Due to default nova security policy usage of this resource is limited to being used by administrators only. The rights may also be delegated to other users by redefining

the access controls on the nova-api server.

Note that the current implementation of the Nova Flavor resource does not allow specifying the name and flavorid properties for the resource. This is done to avoid potential naming collision upon flavor creation as all flavor have a global scope.

Required Properties

ram

Memory in MB for the flavor.

Integer value expected.

Updates cause replacement.

vcpus

Number of VCPUs for the flavor.

Integer value expected.

Updates cause replacement.

Optional Properties

disk

Size of local disk in GB. The “0” size is a special case that uses the native base image size as the size of the ephemeral root volume.

Integer value expected.

Updates cause replacement.

Defaults to “0”.

ephemeral

Size of a secondary ephemeral data disk in GB.

Integer value expected.

Updates cause replacement.

Defaults to “0”.

extra_specs

Key/Value pairs to extend the capabilities of the flavor.

Map value expected.

Can be updated without replacement.

flavorid

> **Note**
>
> Available since 7.0.0 (Newton)

Unique ID of the flavor. If not specified, an UUID will be auto generated and used.

String value expected.

Updates cause replacement.

is_public

> **Note**
>
> Available since 6.0.0 (Mitaka)

Scope of flavor accessibility. Public or private. Default value is True, means public, shared across all projects.

Boolean value expected.

Updates cause replacement.

Defaults to “True”.

name

> **Note**
>
> Available since 7.0.0 (Newton)

Name of the flavor.

String value expected.

Updates cause replacement.

rxtx_factor

RX/TX factor.

Number value expected.

Updates cause replacement.

Defaults to “1.0”.

swap

Swap space in MB.

Integer value expected.

Updates cause replacement.

Defaults to “0”.

Attributes

extra_specs

> **Note**
>
> Available since 7.0.0 (Newton)

Extra specs of the flavor in key-value pairs.

is_public

> **Note**
>
> Available since 6.0.0 (Mitaka)

Whether the flavor is shared across all projects.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Nova::Flavor
    properties:
      disk: Integer
      ephemeral: Integer
      extra_specs: {...}
      flavorid: String
      is_public: Boolean
      name: String
      ram: Integer
```

```
rxtx_factor: Number
swap: Integer
vcpus: Integer
```

OS::Nova::FloatingIP

> **Note**
>
> Available since 2014.1 (Icehouse)

A resource for managing Nova floating IPs.

Floating IP addresses can change their association between instances by action of the user. One of the most common use cases for floating IPs is to provide public IP addresses to a private cloud, where there are a limited number of IP addresses available. Another is for a public cloud user to have a “static” IP address that can be reassigned when an instance is upgraded or moved.

Optional Properties

pool

Allocate a floating IP from a given floating IP pool.

String value expected.

Updates cause replacement.

Attributes

ip

Allocated floating IP address.

pool

Pool from which floating IP is allocated.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Nova::FloatingIP
    properties:
      pool: String
```

OS::Nova::FloatingIPAssociation

> **Note**
>
> Available since 2014.1 (Icehouse)

A resource associates Nova floating IP with Nova server resource.

Resource for associating existing Nova floating IP and Nova server.

Required Properties

floating_ip

ID of the floating IP to assign to the server.

String value expected.

Can be updated without replacement.

server_id

Server to assign floating IP to.

String value expected.

Can be updated without replacement.

Value must be of type nova.server

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Nova::FloatingIPAssociation
    properties:
      floating_ip: String
      server_id: String
```

OS::Nova::HostAggregate

> **Note**
>
> Available since 6.0.0 (Mitaka)

A resource for further partition an availability zone with hosts.

While availability zones are visible to users, host aggregates are only visible to administrators. Host aggregates started out as a way to use Xen hypervisor resource pools, but has been generalized to provide a mechanism to allow administrators to assign key-value pairs to groups of machines. Each node can have multiple aggregates, each aggregate can have multiple key-value pairs, and the same key-value pair can be assigned to multiple aggregate. This information can be used in the scheduler to enable advanced scheduling, to set up xen hypervisor resources pools or to define logical groups for migration.

Required Properties

availability_zone

Name for the availability zone.

String value expected.

Can be updated without replacement.

name

Name for the aggregate.

String value expected.

Can be updated without replacement.

Optional Properties

hosts

List of hosts to join aggregate.

List value expected.

Can be updated without replacement.

List contents:

*

> String value expected.
>
> Can be updated without replacement.
>
> Value must be of type nova.host

metadata

Arbitrary key/value metadata to store information for aggregate.

Map value expected.

Can be updated without replacement.

Defaults to “{}”.

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Nova::HostAggregate
    properties:
      availability_zone: String
      hosts: [String, String, ...]
      metadata: {...}
      name: String
```

OS::Nova::KeyPair

> **Note**
>
> Available since 2014.1 (Icehouse)

A resource for creating Nova key pairs.

A keypair is a ssh key that can be injected into a server on launch.

Note that if a new key is generated setting save_private_key to True results in the system saving the private key which can then be retrieved via the private_key attribute of this resource.

Setting the public_key property means that the private_key attribute of this resource will always return an empty string regardless of the save_private_key setting since there will be no private key data to save.

Required Properties

name

The name of the key pair.

String value expected.

Updates cause replacement.

The length must be in the range 1 to 255.

Optional Properties

public_key

The optional public key. This allows users to supply the public key from a pre-existing key pair. If not supplied, a new key pair will be generated.

String value expected.

Updates cause replacement.

save_private_key

True if the system should remember a generated private key; False otherwise.

Boolean value expected.

Updates cause replacement.

Defaults to “False”.

Attributes

private_key

The private key if it has been saved.

public_key

The public key.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Nova::KeyPair
    properties:
      name: String
      public_key: String
      save_private_key: Boolean
```

OS::Nova::Server

A resource for managing Nova instances.

A Server resource manages the running virtual machine instance within an OpenStack cloud.

Required Properties

flavor

The ID or name of the flavor to boot onto.

String value expected.

Can be updated without replacement.

Value must be of type nova.flavor

Optional Properties

admin_pass

The administrator password for the server.

String value expected.

Can be updated without replacement.

availability_zone

Name of the availability zone for server placement.

String value expected.

Updates cause replacement.

block_device_mapping

Block device mappings for this server.

List value expected.

Updates cause replacement.

List contents:

*

Map value expected.

Updates cause replacement.

Map properties:

delete_on_termination

Indicate whether the volume should be deleted when the server is terminated.

Boolean value expected.

Updates cause replacement.

device_name

A device name where the volume will be attached in the system at /dev/device_name. This value is typically vda.

String value expected.

Updates cause replacement.

snapshot_id

The ID of the snapshot to create a volume from.

String value expected.

Updates cause replacement.

Value must be of type cinder.snapshot

volume_id

The ID of the volume to boot from. Only one of volume_id or snapshot_id should be provided.

String value expected.

Updates cause replacement.

Value must be of type cinder.volume

volume_size

The size of the volume, in GB. It is safe to leave this blank and have the Compute service infer the size.

Integer value expected.

Updates cause replacement.

block_device_mapping_v2

Note

Available since 2015.1 (Kilo)

Block device mappings v2 for this server.

List value expected.

Updates cause replacement.

List contents:

*

Map value expected.

Updates cause replacement.

Map properties:

boot_index

Integer used for ordering the boot disks.

Integer value expected.

Updates cause replacement.

delete_on_termination

Indicate whether the volume should be deleted when the server is terminated.

Boolean value expected.

Updates cause replacement.

device_name

A device name where the volume will be attached in the system at /dev/device_name. This value is typically vda.

String value expected.

Updates cause replacement.

device_type

Device type: at the moment we can make distinction only between disk and cdrom.

String value expected.

Updates cause replacement.

Allowed values: cdrom, disk

disk_bus

Bus of the device: hypervisor driver chooses a suitable default if omitted.

String value expected.

Updates cause replacement.

Allowed values: ide, lame_bus, scsi, usb, virtio

image

Note

Available since 7.0.0 (Newton)

The ID or name of the image to create a volume from.

String value expected.

Updates cause replacement.

Value must be of type glance.image

snapshot_id

The ID of the snapshot to create a volume from.

String value expected.

Updates cause replacement.

Value must be of type cinder.snapshot

swap_size

The size of the swap, in MB.

Integer value expected.

Updates cause replacement.

volume_id

The volume_id can be boot or non-boot device to the server.

String value expected.

Updates cause replacement.

Value must be of type cinder.volume

volume_size

Size of the block device in GB. If it is omitted, hypervisor driver calculates size.

Integer value expected.

Updates cause replacement.

image_id

Note

DEPRECATED since 7.0.0 (Newton) - Use property image.

Note

Available since 5.0.0 (Liberty)

The ID of the image to create a volume from.

String value expected.

Updates cause replacement.

Value must be of type glance.image

config_drive

If True, enable config drive on the server.

Boolean value expected.

Updates cause replacement.

diskConfig

Control how the disk is partitioned when the server is created.

String value expected.

Updates cause replacement.

Allowed values: AUTO, MANUAL

flavor_update_policy

Policy on how to apply a flavor update; either by requesting a server resize or by replacing the entire server.

String value expected.

Can be updated without replacement.

Defaults to “RESIZE”.

Allowed values: RESIZE, REPLACE

image

The ID or name of the image to boot with.

String value expected.

Can be updated without replacement.

Value must be of type glance.image

image_update_policy

Policy on how to apply an image-id update; either by requesting a server rebuild or by replacing the entire server.

String value expected.

Can be updated without replacement.

Defaults to “REBUILD”.

Allowed values: REBUILD, REPLACE, REBUILD_PRESERVE_EPHEMERAL

key_name

Name of keypair to inject into the server.

String value expected.

Updates cause replacement.

Value must be of type nova.keypair

metadata

Arbitrary key/value metadata to store for this server. Both keys and values must be 255 characters or less. Non-string values will be serialized to JSON (and the serialized string must be 255 characters or less).

Map value expected.

Can be updated without replacement.

name

Server name.

String value expected.

Can be updated without replacement.

networks

An ordered list of nics to be added to this server, with information about connected networks, fixed ips, port etc.

List value expected.

Can be updated without replacement.

List contents:

*

Map value expected.

Can be updated without replacement.

Map properties:

network

Name or ID of network to create a port on.

String value expected.

Can be updated without replacement.

Value must be of type neutron.network

port_extra_properties

Note

Available since 6.0.0 (Mitaka)

Dict, which has expand properties for port. Used only if port property is not specified for creating port.

Map value expected.

Can be updated without replacement.

Map properties:

admin_state_up

The administrative state of this port.

Boolean value expected.

Can be updated without replacement.

Defaults to “True”.

allowed_address_pairs

Additional MAC/IP address pairs allowed to pass through the port.

List value expected.

Can be updated without replacement.

List contents:

*

Map value expected.

Can be updated without replacement.

Map properties:

ip_address

IP address to allow through this port.

String value expected.

Can be updated without replacement.

Value must be of type net_cidr

mac_address

MAC address to allow through this port.

String value expected.

Can be updated without replacement.

Value must be of type mac_addr

binding:vnic_type

Note

Available since 2015.1 (Kilo)

The vnic type to be bound on the neutron port. To support SR-IOV PCI passthrough networking, you can request that the neutron port to be realized as normal (virtual nic), direct (pci passthrough), or macvtap (virtual interface with a tap-like software interface). Note that this only works for Neutron deployments that support the bindings extension.

String value expected.

Can be updated without replacement.

Allowed values: normal, direct, macvtap

mac_address

MAC address to give to this port.

String value expected.

Can be updated without replacement.

Value must be of type mac_addr

port_security_enabled

Note

Available since 5.0.0 (Liberty)

Flag to enable/disable port security on the port. When disable this feature(set it to False), there will be no packages filtering, like security-group and address-pairs.

Boolean value expected.

Can be updated without replacement.

qos_policy

Note

Available since 6.0.0 (Mitaka)

The name or ID of QoS policy to attach to this port.

String value expected.

Can be updated without replacement.

Value must be of type neutron.qos_policy

value_specs

Extra parameters to include in the request.

Map value expected.

Can be updated without replacement.

Defaults to “{}”.

subnet

Note

Available since 5.0.0 (Liberty)

Subnet in which to allocate the IP address for port. Used for creating port, based on derived properties. If subnet is specified, network property becomes optional.

String value expected.

Can be updated without replacement.

fixed_ip

Fixed IP address to specify for the port created on the requested network.

String value expected.

Can be updated without replacement.

Value must be of type ip_addr

floating_ip

Note

Available since 6.0.0 (Mitaka)

ID of the floating IP to associate.

String value expected.

Can be updated without replacement.

port

ID of an existing port to associate with this server.

String value expected.

Can be updated without replacement.

Value must be of type neutron.port

personality

A map of files to create/overwrite on the server upon boot. Keys are file names and values are the file contents.

Map value expected.

Updates cause replacement.

Defaults to “{}”.

reservation_id

A UUID for the set of servers being requested.

String value expected.

Updates cause replacement.

scheduler_hints

Arbitrary key-value pairs specified by the client to help boot a server.

Map value expected.

Updates cause replacement.

security_groups

List of security group names or IDs. Cannot be used if neutron ports are associated with this server; assign security groups to the ports instead.

List value expected.

Updates cause replacement.

Defaults to “[]”.

software_config_transport

How the server should receive the metadata required for software configuration. POLL_SERVER_CFN will allow calls to the cfn API action DescribeStackResource authenticated with the provided keypair. POLL_SERVER_HEAT will allow calls to the Heat API resource-show using the provided keystone credentials. POLL_TEMP_URL will create and populate a Swift TempURL with metadata for polling. ZAQAR_MESSAGE will create a dedicated zaqar queue and post the metadata for polling.

String value expected.

Can be updated without replacement.

Defaults to “POLL_SERVER_CFN”.

Allowed values: POLL_SERVER_CFN, POLL_SERVER_HEAT, POLL_TEMP_URL, ZAQAR_MESSAGE

user_data

User data script to be executed by cloud-init.

String value expected.

Can be updated without replacement.

Defaults to “”.

user_data_format

How the user_data should be formatted for the server. For HEAT_CFNTOOLS, the user_data is bundled as part of the heat-cfntools cloud-init boot configuration data. For RAW the user_data is passed to Nova unmodified. For SOFTWARE_CONFIG user_data is bundled as part of the software config data, and metadata is derived from any associated SoftwareDeployment resources.

String value expected.

Updates cause replacement.

Defaults to “HEAT_CFNTOOLS”.

Allowed values: HEAT_CFNTOOLS, RAW, SOFTWARE_CONFIG

user_data_update_policy

> **Note**
>
> Available since 6.0.0 (Mitaka)

Policy on how to apply a user_data update; either by ignoring it or by replacing the entire server.

String value expected.

Can be updated without replacement.

Defaults to “REPLACE”.

Allowed values: REPLACE, IGNORE

Attributes

accessIPv4

The manually assigned alternative public IPv4 address of the server.

accessIPv6

The manually assigned alternative public IPv6 address of the server.

addresses

A dict of all network addresses with corresponding port_id. Each network will have two keys in dict, they are network name and network id. The port ID may be obtained through the following expression: “{get_attr: [<server>, addresses, <network name_or_id>, 0, port]}”.

console_urls

> **Note**
>
> Available since 2015.1 (Kilo)

URLs of server’s consoles. To get a specific console type, the requested type can be specified as parameter to the get_attr function, e.g. get_attr: [<server>, console_urls, novnc]. Currently supported types are novnc, xvpvnc, spice-html5, rdp-html5, serial.

instance_name

AWS compatible instance name.

name

Name of the server.

networks

A dict of assigned network addresses of the form: {“public”: [ip1, ip2...], “private”: [ip3, ip4], “public_uuid”: [ip1, ip2...], “private_uuid”: [ip3, ip4]}. Each network will have two keys in dict, they are network name and network id.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Nova::Server
    properties:
      admin_pass: String
      availability_zone: String
      block_device_mapping: [{"delete_on_termination": Boolean,
"device_name": String, "volume_size": Integer, "snapshot_id": String,
"volume_id": String}, {"delete_on_termination": Boolean, "device_name":
String, "volume_size": Integer, "snapshot_id": String, "volume_id":
String}, ...]
      block_device_mapping_v2: [{"disk_bus": String, "device_type":
String, "image": String, "snapshot_id": String, "boot_index": Integer,
"delete_on_termination": Boolean, "volume_id": String, "image_id": String,
"swap_size": Integer, "device_name": String, "volume_size": Integer},
{"disk_bus": String, "device_type": String, "image": String,
"snapshot_id": String, "boot_index": Integer, "delete_on_termination":
Boolean, "volume_id": String, "image_id": String, "swap_size": Integer,
"device_name": String, "volume_size": Integer}, ...]
      config_drive: Boolean
      diskConfig: String
      flavor: String
      flavor_update_policy: String
      image: String
      image_update_policy: String
      key_name: String
      metadata: {...}
      name: String
      networks: [{"port": String, "floating_ip": String, "fixed_ip":
String, "uuid": String, "subnet": String, "port_extra_properties":
{"qos_policy": String, "binding:vnic_type": String, "mac_address": String,
"admin_state_up": Boolean, "allowed_address_pairs": [{"ip_address":
String, "mac_address": String}, {"ip_address": String, "mac_address":
String}, ...], "value_specs": {...}, "port_security_enabled": Boolean},
"network": String}, {"port": String, "floating_ip": String, "fixed_ip":
String, "uuid": String, "subnet": String, "port_extra_properties":
{"qos_policy": String, "binding:vnic_type": String, "mac_address": String,
"admin_state_up": Boolean, "allowed_address_pairs": [{"ip_address":
String, "mac_address": String}, {"ip_address": String, "mac_address":
```

```
String}, ...], "value_specs": {...}, "port_security_enabled": Boolean},
"network": String}, ...]
      personality: {...}
      reservation_id: String
      scheduler_hints: {...}
      security_groups: [Value, Value, ...]
      software_config_transport: String
      user_data: String
      user_data_format: String
      user_data_update_policy: String
```

OS::Nova::ServerGroup

> **Note**
>
> Available since 2014.2 (Juno)

A resource for managing a Nova server group.

Server groups allow you to make sure instances (VM/VPS) are on the same hypervisor host or on a different one.

Optional Properties

name

Server Group name.

String value expected.

Updates cause replacement.

policies

A list of string policies to apply. Defaults to anti-affinity.

List value expected.

Updates cause replacement.

Defaults to “[‘anti-affinity’]”.

Allowed values: anti-affinity, affinity

List contents:

*

String value expected.

Updates cause replacement.

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Nova::ServerGroup
    properties:
      name: String
      policies: [String, String, ...]
```

OS::Sahara::Cluster

A resource for managing Sahara clusters.

The Cluster entity represents a collection of VM instances that all have the same data processing framework installed. It is mainly characterized by a VM image with a pre-installed framework which will be used for cluster deployment. Users may choose one of the pre-configured Cluster Templates to start a Cluster. To get access to VMs after a Cluster has started, the user should specify a keypair.

Required Properties

cluster_template_id

ID of the Cluster Template used for Node Groups and configurations.

String value expected.

Updates cause replacement.

hadoop_version

Version of Hadoop running on instances.

String value expected.

Updates cause replacement.

plugin_name

Plugin name.

String value expected.

Updates cause replacement.

Value must be of type sahara.plugin

Optional Properties

default_image_id

> **Note**
>
> Available since 2015.1 (Kilo)

Default name or UUID of the image used to boot Hadoop nodes.

String value expected.

Updates cause replacement.

Value must be of type sahara.image

key_name

Keypair added to instances to make them accessible for user.

String value expected.

Updates cause replacement.

Value must be of type nova.keypair

name

Hadoop cluster name.

String value expected.

Updates cause replacement.

neutron_management_network

Name or UUID of network.

String value expected.

Updates cause replacement.

Value must be of type neutron.network

shares

> **Note**
>
> Available since 6.0.0 (Mitaka)

List of manila shares to be mounted.

List value expected.

Updates cause replacement.

List contents:

*

> Map value expected.
>
> Updates cause replacement.
>
> *Map properties:*
>
> access_level
>
> > Governs permissions set in manila for the cluster ips.
> >
> > String value expected.
> >
> > Updates cause replacement.
> >
> > Defaults to “rw”.
> >
> > Allowed values: rw, ro
>
> id
>
> > Id of the manila share.
> >
> > String value expected.
> >
> > Updates cause replacement.
>
> path
>
> > Local path on each cluster node on which to mount the share. Defaults to ‘/mnt/{share_id}’.
> >
> > String value expected.

Updates cause replacement.

use_autoconfig

Note

Available since 5.0.0 (Liberty)

Configure most important configs automatically.

Boolean value expected.

Updates cause replacement.

Attributes

info

Cluster information.

show

Detailed information about resource.

status

Cluster status.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Sahara::Cluster
    properties:
      cluster_template_id: String
      default_image_id: String
      hadoop_version: String
      key_name: String
      name: String
      neutron_management_network: String
      plugin_name: String
      shares: [{"path": String, "id": String, "access_level": String},
{"path": String, "id": String, "access_level": String}, ...]
      use_autoconfig: Boolean
```

OS::Sahara::ClusterTemplate

> **Note**
>
> Available since 2014.2 (Juno)

A resource for managing Sahara cluster templates.

A Cluster Template is designed to bring Node Group Templates together to form a Cluster. A Cluster Template defines what Node Groups will be included and how many instances will be created in each. Some data processing framework configurations can not be applied to a single node, but to a whole Cluster. A user can specify these kinds of configurations in a Cluster Template. Sahara enables users to specify which processes should be added to an anti-affinity group within a Cluster Template. If a process is included into an anti-affinity group, it means that VMs where this process is going to be launched should be scheduled to different hardware hosts.

Required Properties

hadoop_version

Version of Hadoop running on instances.

String value expected.

Can be updated without replacement.

plugin_name

Plugin name.

String value expected.

Can be updated without replacement.

Value must be of type sahara.plugin

Optional Properties

anti_affinity

List of processes to enable anti-affinity for.

List value expected.

Can be updated without replacement.

List contents:

*

String value expected.

Can be updated without replacement.

cluster_configs

Cluster configs dictionary.

Map value expected.

Can be updated without replacement.

default_image_id

ID of the default image to use for the template.

String value expected.

Can be updated without replacement.

Value must be of type sahara.image

description

Description of the Sahara Group Template.

String value expected.

Can be updated without replacement.

Defaults to “”.

name

Name for the Sahara Cluster Template.

String value expected.

Can be updated without replacement.

The length must be in the range 1 to 50.

Value must match pattern: ^(([a-zA-Z]|[a-zA-Z][a-zA-Z0-9\-]*[a-zA-Z0-9])\.)*([A-Za-z]|[A-Za-z][A-Za-z0-9\-]*[A-Za-z0-9])$

neutron_management_network

Name or UUID of network.

String value expected.

Can be updated without replacement.

Value must be of type neutron.network

node_groups

Node groups.

List value expected.

Can be updated without replacement.

List contents:

*

Map value expected.

Can be updated without replacement.

Map properties:

count

Number of instances in the Node group.

Integer value expected.

Can be updated without replacement.

The value must be at least 1.

name

Name of the Node group.

String value expected.

Can be updated without replacement.

node_group_template_id

ID of the Node Group Template.

String value expected.

Can be updated without replacement.

shares

Note

Available since 6.0.0 (Mitaka)

List of manila shares to be mounted.

List value expected.

Can be updated without replacement.

List contents:

*

Map value expected.

Can be updated without replacement.

Map properties:

access_level

Governs permissions set in manila for the cluster ips.

String value expected.

Can be updated without replacement.

Defaults to “rw”.

Allowed values: rw, ro

id

Id of the manila share.

String value expected.

Can be updated without replacement.

path

Local path on each cluster node on which to mount the share. Defaults to ‘/mnt/{share_id}’.

String value expected.

Can be updated without replacement.

use_autoconfig

> **Note**
>
> Available since 5.0.0 (Liberty)

Configure most important configs automatically.

Boolean value expected.

Updates cause replacement.

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Sahara::ClusterTemplate
    properties:
      anti_affinity: [String, String, ...]
      cluster_configs: {...}
      default_image_id: String
      description: String
      hadoop_version: String
      name: String
      neutron_management_network: String
      node_groups: [{"name": String, "node_group_template_id": String,
"count": Integer}, {"name": String, "node_group_template_id": String,
"count": Integer}, ...]
      plugin_name: String
      shares: [{"path": String, "id": String, "access_level": String},
{"path": String, "id": String, "access_level": String}, ...]
      use_autoconfig: Boolean
```

OS::Sahara::DataSource

> **Note**
>
> Available since 5.0.0 (Liberty)

A resource for creating sahara data source.

A data source stores an URL which designates the location of input or output data and any credentials needed to access the location.

Required Properties

type

Type of the data source.

String value expected.

Can be updated without replacement.

Allowed values: swift, hdfs, maprfs, manila

url

URL for the data source.

String value expected.

Can be updated without replacement.

Optional Properties

credentials

Credentials used for swift. Not required if sahara is configured to use proxy users and delegated trusts for access.

Map value expected.

Can be updated without replacement.

Map properties:

password

> Password for accessing the data source URL.
>
> String value expected.

Can be updated without replacement.

user

Username for accessing the data source URL.

String value expected.

Can be updated without replacement.

description

Description of the data source.

String value expected.

Can be updated without replacement.

Defaults to “”.

name

Name of the data source.

String value expected.

Can be updated without replacement.

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Sahara::DataSource
    properties:
      credentials: {"user": String, "password": String}
      description: String
      name: String
      type: String
      url: String
```

OS::Sahara::ImageRegistry

> **Note**
>
> Available since 6.0.0 (Mitaka)

A resource for registering an image in sahara.

Allows to register an image in the sahara image registry and add tags.

Required Properties

image

ID or name of the image to register.

String value expected.

Updates cause replacement.

Value must be of type glance.image

username

Username of privileged user in the image.

String value expected.

Can be updated without replacement.

Optional Properties

description

Description of the image.

String value expected.

Can be updated without replacement.

Defaults to “”.

tags

Tags to add to the image.

List value expected.

Can be updated without replacement.

Defaults to “[]”.

List contents:

*

String value expected.

Can be updated without replacement.

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Sahara::ImageRegistry
    properties:
      description: String
      image: String
      tags: [String, String, ...]
      username: String
```

OS::Sahara::JobBinary

Note

Available since 5.0.0 (Liberty)

A resource for creating sahara job binary.

A job binary stores an URL to a single script or Jar file and any credentials needed to retrieve the file.

Required Properties

url

URL for the job binary. Must be in the format swift://<container>/<path> or internal-db://<uuid>.

String value expected.

Can be updated without replacement.

Optional Properties

credentials

Credentials used for swift. Not required if sahara is configured to use proxy users and delegated trusts for access.

Map value expected.

Can be updated without replacement.

Map properties:

password

> Password for accessing the job binary URL.
>
> String value expected.
>
> Can be updated without replacement.

user

> Username for accessing the job binary URL.
>
> String value expected.
>
> Can be updated without replacement.

description

Description of the job binary.

String value expected.

Can be updated without replacement.

Defaults to “”.

name

Name of the job binary.

String value expected.

Can be updated without replacement.

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Sahara::JobBinary
    properties:
      credentials: {"user": String, "password": String}
      description: String
      name: String
      url: String
```

OS::Sahara::NodeGroupTemplate

> **Note**
>
> Available since 2014.2 (Juno)

A resource for managing Sahara node group templates.

A Node Group Template describes a group of nodes within cluster. It contains a list of hadoop processes that will be launched on each instance in a group. Also a Node Group Template may provide node scoped configurations for those processes.

Required Properties

flavor

Name or ID Nova flavor for the nodes.

String value expected.

Can be updated without replacement.

Value must be of type nova.flavor

hadoop_version

Version of Hadoop running on instances.

String value expected.

Can be updated without replacement.

node_processes

List of processes to run on every node.

List value expected.

Can be updated without replacement.

The length must be at least 1.

List contents:

*

String value expected.

Can be updated without replacement.

plugin_name

Plugin name.

String value expected.

Can be updated without replacement.

Value must be of type sahara.plugin

Optional Properties

auto_security_group

Defines whether auto-assign security group to this Node Group template.

Boolean value expected.

Can be updated without replacement.

availability_zone

Availability zone to create servers in.

String value expected.

Can be updated without replacement.

description

Description of the Node Group Template.

String value expected.

Can be updated without replacement.

Defaults to “”.

floating_ip_pool

Name or UUID of the Neutron floating IP network or name of the Nova floating ip pool to use. Should not be provided when used with Nova-network that auto-assign floating IPs.

String value expected.

Can be updated without replacement.

image_id

ID of the image to use for the template.

String value expected.

Can be updated without replacement.

Value must be of type sahara.image

is_proxy_gateway

> **Note**
>
> Available since 5.0.0 (Liberty)

Provide access to nodes using other nodes of the cluster as proxy gateways.

Boolean value expected.

Can be updated without replacement.

name

Name for the Sahara Node Group Template.

String value expected.

Can be updated without replacement.

The length must be in the range 1 to 50.

Value must match pattern: ^(([a-zA-Z]|[a-zA-Z][a-zA-Z0-9\-]*[a-zA-Z0-9])\.)*([A-Za-z]|[A-Za-z][A-Za-z0-9\-]*[A-Za-z0-9])$

node_configs

Dictionary of node configurations.

Map value expected.

Can be updated without replacement.

security_groups

List of security group names or IDs to assign to this Node Group template.

List value expected.

Can be updated without replacement.

List contents:

*

String value expected.

Can be updated without replacement.

shares

Note

Available since 6.0.0 (Mitaka)

List of manila shares to be mounted.

List value expected.

Can be updated without replacement.

List contents:

*

Map value expected.

Can be updated without replacement.

Map properties:

access_level

Governs permissions set in manila for the cluster ips.

String value expected.

Can be updated without replacement.

Defaults to “rw”.

Allowed values: rw, ro

id

Id of the manila share.

String value expected.

Can be updated without replacement.

path

Local path on each cluster node on which to mount the share. Defaults to '/mnt/{share_id}'.

String value expected.

Can be updated without replacement.

use_autoconfig

Note

Available since 5.0.0 (Liberty)

Configure most important configs automatically.

Boolean value expected.

Can be updated without replacement.

volume_local_to_instance

Note

Available since 5.0.0 (Liberty)

Create volumes on the same physical port as an instance.

Boolean value expected.

Can be updated without replacement.

volume_type

Type of the volume to create on Cinder backend.

String value expected.

Can be updated without replacement.

Value must be of type cinder.vtype

volumes_availability_zone

Availability zone to create volumes in.

String value expected.

Can be updated without replacement.

volumes_per_node

Volumes per node.

Integer value expected.

Can be updated without replacement.

The value must be at least 0.

volumes_size

Size of the volumes, in GB.

Integer value expected.

Can be updated without replacement.

The value must be at least 1.

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Sahara::NodeGroupTemplate
    properties:
      auto_security_group: Boolean
      availability_zone: String
      description: String
      flavor: String
      floating_ip_pool: String
      hadoop_version: String
      image_id: String
      is_proxy_gateway: Boolean
      name: String
```

```
      node_configs: {...}
      node_processes: [String, String, ...]
      plugin_name: String
      security_groups: [String, String, ...]
      shares: [{"path": String, "id": String, "access_level": String},
{"path": String, "id": String, "access_level": String}, ...]
      use_autoconfig: Boolean
      volume_local_to_instance: Boolean
      volume_type: String
      volumes_availability_zone: String
      volumes_per_node: Integer
      volumes_size: Integer
```

OS::Senlin::Cluster

> **Note**
>
> Available since 6.0.0 (Mitaka)

A resource that creates a Senlin Cluster.

Cluster resource in senlin can create and manage objects of the same nature, e.g. Nova servers, Heat stacks, Cinder volumes, etc. The collection of these objects is referred to as a cluster.

Required Properties

profile

The name or id of the Senlin profile.

String value expected.

Can be updated without replacement.

Value must be of type senlin.profile

Optional Properties

desired_capacity

Desired initial number of resources in cluster.

Integer value expected.

Can be updated without replacement.

Defaults to “0”.

max_size

Maximum number of resources in the cluster. -1 means unlimited.

Integer value expected.

Can be updated without replacement.

Defaults to “-1”.

The value must be at least -1.

metadata

Metadata key-values defined for cluster.

Map value expected.

Can be updated without replacement.

min_size

Minimum number of resources in the cluster.

Integer value expected.

Can be updated without replacement.

Defaults to “0”.

The value must be at least 0.

name

Name of the cluster. By default, physical resource name is used.

String value expected.

Can be updated without replacement.

timeout

The number of seconds to wait for the cluster actions.

Integer value expected.

Can be updated without replacement.

The value must be at least 0.

Attributes

desired_capacity

Desired capacity of the cluster.

max_size

Max size of the cluster.

metadata

Cluster metadata.

min_size

Min size of the cluster.

name

Cluster name.

nodes

Nodes list in the cluster.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Senlin::Cluster
    properties:
      desired_capacity: Integer
      max_size: Integer
      metadata: {...}
      min_size: Integer
      name: String
      profile: String
      timeout: Integer
```

OS::Senlin::Node

> **Note**
>
> Available since 6.0.0 (Mitaka)

A resource that creates a Senlin Node.

Node is an object that belongs to at most one Cluster, it can be created based on a profile.

Required Properties

profile

Name or ID of senlin profile to create this node.

String value expected.

Can be updated without replacement.

Value must be of type senlin.profile

Optional Properties

metadata

Metadata key-values defined for node.

Map value expected.

Can be updated without replacement.

name

Name of the senlin node. By default, physical resource name is used.

String value expected.

Can be updated without replacement.

Attributes

cluster_id

The cluster ID this node belongs to.

details

The details of physical object.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Senlin::Node
    properties:
      metadata: {...}
      name: String
      profile: String
```

OS::Senlin::Policy

> **Note**
>
> Available since 6.0.0 (Mitaka)

A resource that creates a Senlin Policy.

A policy is a set of rules that can be checked and/or enforced when an action is performed on a Cluster.

Required Properties

type

The type of senlin policy.

String value expected.

Updates cause replacement.

Value must be of type senlin.policy_type

Optional Properties

bindings

A list of clusters to which this policy is attached.

List value expected.

Can be updated without replacement.

List contents:

*

Map value expected.

Can be updated without replacement.

Map properties:

cluster

The name or ID of target cluster.

String value expected.

Can be updated without replacement.

Value must be of type senlin.cluster

enabled

Whether enable this policy on that cluster.

Boolean value expected.

Can be updated without replacement.

Defaults to “True”.

name

Name of the senlin policy. By default, physical resource name is used.

String value expected.

Can be updated without replacement.

properties

Properties of this policy.

Map value expected.

Updates cause replacement.

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Senlin::Policy
    properties:
      bindings: [{"enabled": Boolean, "cluster": String}, {"enabled":
Boolean, "cluster": String}, ...]
      name: String
      properties: {...}
      type: String
```

OS::Senlin::Profile

> **Note**
>
> Available since 6.0.0 (Mitaka)

A resource that creates a Senlin Profile.

Profile resource in senlin is a template describing how to create nodes in cluster.

Required Properties

type

The type of profile.

String value expected.

Updates cause replacement.

Value must be of type senlin.profile_type

Optional Properties

metadata

Metadata key-values defined for profile.

Map value expected.

Can be updated without replacement.

name

Name of the senlin profile. By default, physical resource name is used.

String value expected.

Can be updated without replacement.

properties

Properties for profile.

Map value expected.

Updates cause replacement.

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Senlin::Profile
    properties:
      metadata: {...}
      name: String
      properties: {...}
      type: String
```

OS::Senlin::Receiver

> **Note**
>
> Available since 6.0.0 (Mitaka)

A resource that creates Senlin Receiver.

Receiver is an abstract resource created at the senlin engine that can be used to hook the engine to some external event/alarm sources.

Required Properties

action

The action to be executed when the receiver is signaled.

String value expected.

Updates cause replacement.

Allowed values: CLUSTER_SCALE_OUT, CLUSTER_SCALE_IN

cluster

Name or ID of target cluster.

String value expected.

Updates cause replacement.

Value must be of type senlin.cluster

Optional Properties

name

Name of the senlin receiver. By default, physical resource name is used.

String value expected.

Updates cause replacement.

params

The parameters passed to action when the receiver is signaled.

Map value expected.

Updates cause replacement.

type

Type of receiver.

String value expected.

Updates cause replacement.

Defaults to “webhook”.

Allowed values: webhook

Attributes

channel

The channel for receiving signals.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Senlin::Receiver
    properties:
      action: String
      cluster: String
      name: String
      params: {...}
      type: String
```

OS::Swift::Container

A resource for managing Swift containers.

A container defines a namespace for objects. An object with the same name in two different containers represents two different objects.

Optional Properties

PurgeOnDelete

> **Note**
>
> Available since 2015.1 (Kilo)

If True, delete any objects in the container when the container is deleted. Otherwise, deleting a non-empty container will result in an error.

Boolean value expected.

Updates cause replacement.

Defaults to “False”.

X-Account-Meta

A map of user-defined meta data to associate with the account. Each key in the map will set the header X-Account-Meta-{key} with the corresponding value.

Map value expected.

Updates cause replacement.

Defaults to “{}”.

X-Container-Meta

A map of user-defined meta data to associate with the container. Each key in the map will set the header X-Container-Meta-{key} with the corresponding value.

Map value expected.

Updates cause replacement.

Defaults to “{}”.

X-Container-Read

Specify the ACL permissions on who can read objects in the container.

String value expected.

Updates cause replacement.

X-Container-Write

Specify the ACL permissions on who can write objects to the container.

String value expected.

Updates cause replacement.

name

Name for the container. If not specified, a unique name will be generated.

String value expected.

Updates cause replacement.

Attributes

BytesUsed

The number of bytes stored in the container.

DomainName

The host from the container URL.

HeadContainer

A map containing all headers for the container.

ObjectCount

The number of objects stored in the container.

RootURL

The parent URL of the container.

WebsiteURL

The URL of the container.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Swift::Container
    properties:
      PurgeOnDelete: Boolean
      X-Account-Meta: {...}
      X-Container-Meta: {...}
      X-Container-Read: String
      X-Container-Write: String
      name: String
```

OS::Trove::Cluster

> **Note**
>
> Available since 2015.1 (Kilo)

A resource for managing Trove clusters.

A Cluster is an opaque cluster used to store Database clusters.

Required Properties

datastore_type

Name of registered datastore type.

String value expected.

Updates cause replacement.

The length must be no greater than 255.

datastore_version

Name of the registered datastore version. It must exist for provided datastore type. Defaults to using single active version. If several active versions exist for provided datastore type, explicit value for this parameter must be specified.

String value expected.

Updates cause replacement.

The length must be no greater than 255.

instances

List of database instances.

List value expected.

Updates cause replacement.

List contents:

*

> Map value expected.
>
> Updates cause replacement.
>
> *Map properties:*
>
> flavor
>
> > Flavor of the instance.
> >
> > String value expected.
> >
> > Updates cause replacement.
> >
> > Value must be of type trove.flavor

volume_size

Size of the instance disk volume in GB.

Integer value expected.

Updates cause replacement.

The value must be in the range 1 to 150.

Optional Properties

name

Name of the cluster to create.

String value expected.

Updates cause replacement.

The length must be no greater than 255.

Attributes

instances

A list of instances ids.

ip

A list of cluster instance IPs.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Trove::Cluster
    properties:
      datastore_type: String
      datastore_version: String
      instances: [{"flavor": String, "volume_size": Integer}, {"flavor":
String, "volume_size": Integer}, ...]
      name: String
```

OS::Trove::Instance

> **Note**
>
> Available since 2014.1 (Icehouse)

OpenStack cloud database instance resource.

Trove is Database as a Service for OpenStack. It's designed to run entirely on OpenStack, with the goal of allowing users to quickly and easily utilize the features of a relational or non-relational database without the burden of handling complex administrative tasks.

Required Properties

flavor

Reference to a flavor for creating DB instance.

String value expected.

Can be updated without replacement.

Value must be of type trove.flavor

size

Database volume size in GB.

Integer value expected.

Can be updated without replacement.

The value must be in the range 1 to 150.

Optional Properties

availability_zone

Name of the availability zone for DB instance.

String value expected.

Updates cause replacement.

databases

List of databases to be created on DB instance creation.

List value expected.

Can be updated without replacement.

Defaults to “[]”.

List contents:

*

Map value expected.

Can be updated without replacement.

Map properties:

character_set

Set of symbols and encodings.

String value expected.

Can be updated without replacement.

Defaults to “utf8”.

collate

Set of rules for comparing characters in a character set.

String value expected.

Can be updated without replacement.

Defaults to “utf8_general_ci”.

name

Specifies database names for creating databases on instance creation.

String value expected.

Can be updated without replacement.

The length must be no greater than 64.

Value must match pattern: [a-zA-Z0-9_]+[a-zA-Z0-9_@?#\s]*[a-zA-Z0-9_]+

datastore_type

Name of registered datastore type.

String value expected.

Updates cause replacement.

The length must be no greater than 255.

datastore_version

Name of the registered datastore version. It must exist for provided datastore type. Defaults to using single active version. If several active versions exist for provided datastore type, explicit value for this parameter must be specified.

String value expected.

Updates cause replacement.

The length must be no greater than 255.

name

Name of the DB instance to create.

String value expected.

Can be updated without replacement.

The length must be no greater than 255.

networks

List of network interfaces to create on instance.

List value expected.

Updates cause replacement.

Defaults to “[]”.

List contents:

*

Map value expected.

Updates cause replacement.

Map properties:

fixed_ip

Fixed IPv4 address for this NIC.

String value expected.

Updates cause replacement.

Value must be of type ip_addr

network

Name or UUID of the network to attach this NIC to. Either port or network must be specified.

String value expected.

Updates cause replacement.

Value must be of type neutron.network

port

Name or UUID of Neutron port to attach this NIC to. Either port or network must be specified.

String value expected.

Updates cause replacement.

Value must be of type neutron.port

replica_count

Note

Available since 5.0.0 (Liberty)

The number of replicas to be created.

Integer value expected.

Updates cause replacement.

replica_of

Note

Available since 5.0.0 (Liberty)

Identifier of the source instance to replicate.

String value expected.

Updates cause replacement.

restore_point

DB instance restore point.

String value expected.

Updates cause replacement.

users

List of users to be created on DB instance creation.

List value expected.

Can be updated without replacement.

Defaults to “[]”.

List contents:

*

Map value expected.

Can be updated without replacement.

Map properties:

databases

Names of databases that those users can access on instance creation.

List value expected.

Can be updated without replacement.

The length must be at least 1.

List contents:

*

String value expected.

Can be updated without replacement.

host

The host from which a user is allowed to connect to the

database.

String value expected.

Can be updated without replacement.

Defaults to “%”.

name

User name to create a user on instance creation.

String value expected.

Can be updated without replacement.

The length must be no greater than 16.

Value must match pattern: [a-zA-Z0-9_]+[a-zA-Z0-9_@?#\s]*[a-zA-Z0-9_]+

password

Password for those users on instance creation.

String value expected.

Can be updated without replacement.

Value must match pattern: [a-zA-Z0-9_]+[a-zA-Z0-9_@?#\s]*[a-zA-Z0-9_]+

Attributes

hostname

Hostname of the instance.

href

Api endpoint reference of the instance.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Trove::Instance
    properties:
      availability_zone: String
      databases: [{"name": String, "character_set": String, "collate":
String}, {"name": String, "character_set": String, "collate": String},
...]
      datastore_type: String
      datastore_version: String
      flavor: String
      name: String
      networks: [{"port": String, "fixed_ip": String, "network": String},
{"port": String, "fixed_ip": String, "network": String}, ...]
      replica_count: Integer
      replica_of: String
      restore_point: String
      size: Integer
      users: [{"name": String, "databases": [String, String, ...], "host":
String, "password": String}, {"name": String, "databases": [String,
String, ...], "host": String, "password": String}, ...]
```

OS::Zaqar::Queue

> **Note**
>
> Available since 2014.2 (Juno)

A resource for managing Zaqar queues.

Queue is a logical entity that groups messages. Ideally a queue is created per work type. For example, if you want to compress files, you would create a queue dedicated for this job. Any application that reads from this queue would only compress files.

Required Properties

name

Name of the queue instance to create.

String value expected.

Updates cause replacement.

Optional Properties

metadata

Arbitrary key/value metadata to store contextual information about this queue.

Map value expected.

Can be updated without replacement.

Attributes

href

The resource href of the queue.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Zaqar::Queue
    properties:
      metadata: {...}
      name: String
```

CloudFormation Compatible Resource Types

AWS::AutoScaling::AutoScalingGroup

> **Note**
>
> Available since 2014.1 (Icehouse)

Required Properties

AvailabilityZones

Not Implemented.

List value expected.

Updates cause replacement.

MaxSize

Maximum number of instances in the group.

Integer value expected.

Can be updated without replacement.

MinSize

Minimum number of instances in the group.

Integer value expected.

Can be updated without replacement.

Optional Properties

Cooldown

Cooldown period, in seconds.

Integer value expected.

Can be updated without replacement.

DesiredCapacity

Desired initial number of instances.

Integer value expected.

Can be updated without replacement.

HealthCheckGracePeriod

> **Note**
>
> Not implemented.

HealthCheckType

> **Note**
>
> Not implemented.

InstanceId

The ID of an existing instance to use to create the Auto Scaling group. If specify this property, will create the group use an existing instance instead of a launch configuration.

String value expected.

Updates cause replacement.

Value must be of type nova.server

LaunchConfigurationName

The reference to a LaunchConfiguration resource.

String value expected.

Can be updated without replacement.

LoadBalancerNames

List of LoadBalancer resources.

List value expected.

Updates cause replacement.

Tags

Tags to attach to this group.

List value expected.

Updates cause replacement.

List contents:

*

Map value expected.

Updates cause replacement.

Map properties:

Key

String value expected.

Updates cause replacement.

Value

String value expected.

Updates cause replacement.

VPCZoneIdentifier

Use only with Neutron, to list the internal subnet to which the instance will be attached; needed only if multiple exist; list length must be exactly 1.

List value expected.

Updates cause replacement.

List contents:

*

UUID of the internal subnet to which the instance will be attached.

String value expected.

Updates cause replacement.

Attributes

InstanceList

A comma-delimited list of server ip addresses. (Heat extension).

show

Detailed information about resource.

update_policy

AutoScalingRollingUpdate

Map value expected.

Updates cause replacement.

Map properties:

MaxBatchSize

> Integer value expected.
>
> Updates cause replacement.
>
> Defaults to “1”.

MinInstancesInService

> Integer value expected.
>
> Updates cause replacement.
>
> Defaults to “0”.

PauseTime

> String value expected.
>
> Updates cause replacement.
>
> Defaults to “PT0S”.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: AWS::AutoScaling::AutoScalingGroup
    properties:
      AvailabilityZones: [Value, Value, ...]
      Cooldown: Integer
```

```
      DesiredCapacity: Integer
      InstanceId: String
      LaunchConfigurationName: String
      LoadBalancerNames: [Value, Value, ...]
      MaxSize: Integer
      MinSize: Integer
      Tags: [{"Key": String, "Value": String}, {"Key": String, "Value":
String}, ...]
      VPCZoneIdentifier: [String, String, ...]
```

AWS::AutoScaling::LaunchConfiguration

Optional Properties

BlockDeviceMappings

Block device mappings to attach to instance.

List value expected.

Updates cause replacement.

List contents:

*

> Map value expected.
>
> Updates cause replacement.
>
> *Map properties:*
>
> DeviceName
>
> > A device name where the volume will be attached in the system at /dev/device_name.e.g. vdb
> >
> > String value expected.
> >
> > Updates cause replacement.
>
> Ebs
>
> > The ebs volume to attach to the instance.
> >
> > Map value expected.
> >
> > Updates cause replacement.

Map properties:

DeleteOnTermination

Indicate whether the volume should be deleted when the instance is terminated.

Boolean value expected.

Updates cause replacement.

Defaults to “True”.

Iops

Note

Not implemented.

SnapshotId

The ID of the snapshot to create a volume from.

String value expected.

Updates cause replacement.

Value must be of type cinder.snapshot

VolumeSize

The size of the volume, in GB. Must be equal or greater than the size of the snapshot. It is safe to leave this blank and have the Compute service infer the size.

String value expected.

Updates cause replacement.

VolumeType

Note

Not implemented.

NoDevice

Note

Not implemented.

VirtualName

Note

Not implemented.

ImageId

Glance image ID or name.

String value expected.

Updates cause replacement.

Value must be of type glance.image

InstanceId

The ID of an existing instance you want to use to create the launch configuration. All properties are derived from the instance with the exception of BlockDeviceMapping.

String value expected.

Updates cause replacement.

Value must be of type nova.server

InstanceType

Nova instance type (flavor).

String value expected.

Updates cause replacement.

Value must be of type nova.flavor

KernelId

Note

Not implemented.

KeyName

Optional Nova keypair name.

String value expected.

Updates cause replacement.

Value must be of type nova.keypair

NovaSchedulerHints

Scheduler hints to pass to Nova (Heat extension).

List value expected.

Updates cause replacement.

List contents:

*

Map value expected.

Updates cause replacement.

Map properties:

Key

String value expected.

Updates cause replacement.

Value

String value expected.

Updates cause replacement.

RamDiskId

> **Note**
>
> Not implemented.

SecurityGroups

Security group names to assign.

List value expected.

Updates cause replacement.

UserData

User data to pass to instance.

String value expected.

Updates cause replacement.

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: AWS::AutoScaling::LaunchConfiguration
    properties:
      BlockDeviceMappings: [{"Ebs": {"VolumeSize": String,
"DeleteOnTermination": Boolean, "SnapshotId": String}, "DeviceName":
String}, {"Ebs": {"VolumeSize": String, "DeleteOnTermination": Boolean,
"SnapshotId": String}, "DeviceName": String}, ...]
      ImageId: String
      InstanceId: String
      InstanceType: String
      KeyName: String
      NovaSchedulerHints: [{"Key": String, "Value": String}, {"Key":
String, "Value": String}, ...]
      SecurityGroups: [Value, Value, ...]
      UserData: String
```

AWS::AutoScaling::ScalingPolicy

Required Properties

AdjustmentType

Type of adjustment (absolute or percentage).

String value expected.

Can be updated without replacement.

Allowed values: ChangeInCapacity, ExactCapacity, PercentChangeInCapacity

AutoScalingGroupName

AutoScaling group name to apply policy to.

String value expected.

Updates cause replacement.

ScalingAdjustment

Size of adjustment.

Integer value expected.

Can be updated without replacement.

Optional Properties

Cooldown

Cooldown period, in seconds.

Integer value expected.

Can be updated without replacement.

MinAdjustmentStep

Minimum number of resources that are added or removed when the AutoScaling group scales up or down. This can be used only when specifying PercentChangeInCapacity for the AdjustmentType property.

Integer value expected.

Can be updated without replacement.

The value must be at least 0.

Attributes

AlarmUrl

A signed url to handle the alarm. (Heat extension).

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: AWS::AutoScaling::ScalingPolicy
    properties:
      AdjustmentType: String
      AutoScalingGroupName: String
      Cooldown: Integer
      MinAdjustmentStep: Integer
      ScalingAdjustment: Integer
```

AWS::CloudFormation::Stack

Represents a child stack to allow composition of templates.

Required Properties

TemplateURL

The URL of a template that specifies the stack to be created as a resource.

String value expected.

Can be updated without replacement.

Optional Properties

Parameters

The set of parameters passed to this nested stack.

Map value expected.

Can be updated without replacement.

TimeoutInMinutes

The length of time, in minutes, to wait for the nested stack creation.

Integer value expected.

Can be updated without replacement.

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: AWS::CloudFormation::Stack
    properties:
      Parameters: {...}
      TemplateURL: String
      TimeoutInMinutes: Integer
```

AWS::CloudFormation::WaitCondition

> **Note**
>
> Available since 2014.1 (Icehouse)

Required Properties

Handle

A reference to the wait condition handle used to signal this wait condition.

String value expected.

Updates cause replacement.

Timeout

The number of seconds to wait for the correct number of signals to arrive.

Integer value expected.

Updates cause replacement.

The value must be in the range 1 to 43200.

Optional Properties

Count

The number of success signals that must be received before the stack creation process continues.

Integer value expected.

Can be updated without replacement.

Defaults to “1”.

The value must be at least 1.

Attributes

Data

JSON string containing data associated with wait condition signals sent to the handle.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: AWS::CloudFormation::WaitCondition
    properties:
      Count: Integer
      Handle: String
      Timeout: Integer
```

AWS::CloudFormation::WaitConditionHandle

> **Note**
>
> Available since 2014.1 (Icehouse)

AWS WaitConditionHandle resource.

the main point of this class is to : have no dependencies (so the instance can reference it) generate a unique url (to be returned in the reference) then the cfn-signal will use this url to post to and WaitCondition will poll it to see if has been written to.

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: AWS::CloudFormation::WaitConditionHandle
```

AWS::EC2::EIP

Optional Properties

Domain

Set to “vpc” to have IP address allocation associated to your VPC.

String value expected.

Updates cause replacement.

Allowed values: vpc

InstanceId

Instance ID to associate with EIP.

String value expected.

Can be updated without replacement.

Value must be of type nova.server

Attributes

AllocationId

ID that AWS assigns to represent the allocation of the address for use with Amazon VPC. Returned only for VPC elastic IP addresses.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: AWS::EC2::EIP
    properties:
      Domain: String
      InstanceId: String
```

AWS::EC2::EIPAssociation

Optional Properties

AllocationId

Allocation ID for VPC EIP address.

String value expected.

Can be updated without replacement.

EIP

EIP address to associate with instance.

String value expected.

Can be updated without replacement.

Value must be of type ip_addr

InstanceId

Instance ID to associate with EIP specified by EIP property.

String value expected.

Can be updated without replacement.

Value must be of type nova.server

NetworkInterfaceId

Network interface ID to associate with EIP.

String value expected.

Can be updated without replacement.

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: AWS::EC2::EIPAssociation
    properties:
      AllocationId: String
      EIP: String
      InstanceId: String
      NetworkInterfaceId: String
```

AWS::EC2::Instance

Required Properties

ImageId

Glance image ID or name.

String value expected.

Updates cause replacement.

Value must be of type glance.image

InstanceType

Nova instance type (flavor).

String value expected.

Can be updated without replacement.

Value must be of type nova.flavor

Optional Properties

AvailabilityZone

Availability zone to launch the instance in.

String value expected.

Updates cause replacement.

BlockDeviceMappings

Block device mappings to attach to instance.

List value expected.

Updates cause replacement.

List contents:

*

Map value expected.

Updates cause replacement.

Map properties:

DeviceName

A device name where the volume will be attached in the system at /dev/device_name.e.g. vdb

String value expected.

Updates cause replacement.

Ebs

The ebs volume to attach to the instance.

Map value expected.

Updates cause replacement.

Map properties:

DeleteOnTermination

Indicate whether the volume should be deleted when the instance is terminated.

Boolean value expected.

Updates cause replacement.

Defaults to “True”.

Iops

Note

Not implemented.

SnapshotId

The ID of the snapshot to create a volume from.

String value expected.

Updates cause replacement.

Value must be of type cinder.snapshot

VolumeSize

The size of the volume, in GB. Must be equal or greater than the size of the snapshot. It is safe to leave this blank and have the Compute service infer the size.

String value expected.

Updates cause replacement.

VolumeType

Note

Not implemented.

NoDevice

Note

Not implemented.

VirtualName

Note

Not implemented.

DisableApiTermination

Note

Not implemented.

KernelId

Note

Not implemented.

KeyName

Optional Nova keypair name.

String value expected.

Updates cause replacement.

Value must be of type nova.keypair

Monitoring

> **Note**
>
> Not implemented.

NetworkInterfaces

Network interfaces to associate with instance.

List value expected.

Can be updated without replacement.

NovaSchedulerHints

Scheduler hints to pass to Nova (Heat extension).

List value expected.

Updates cause replacement.

List contents:

*

> Map value expected.
>
> Updates cause replacement.
>
> *Map properties:*
>
> Key
>
> > String value expected.
> >
> > Updates cause replacement.
>
> Value
>
> > String value expected.
> >
> > Updates cause replacement.

PlacementGroupName

> **Note**
>
> Not implemented.

PrivateIpAddress

> **Note**
>
> Not implemented.

RamDiskId

> **Note**
>
> Not implemented.

SecurityGroupIds

Security group IDs to assign.

List value expected.

Updates cause replacement.

SecurityGroups

Security group names to assign.

List value expected.

Updates cause replacement.

SourceDestCheck

> **Note**
>
> Not implemented.

SubnetId

Subnet ID to launch instance in.

String value expected.

Can be updated without replacement.

Tags

Tags to attach to instance.

List value expected.

Can be updated without replacement.

List contents:

*

Map value expected.

Can be updated without replacement.

Map properties:

Key

String value expected.

Can be updated without replacement.

Value

String value expected.

Can be updated without replacement.

Tenancy

Note

Not implemented.

UserData

User data to pass to instance.

String value expected.

Updates cause replacement.

Volumes

Volumes to attach to instance.

List value expected.

Updates cause replacement.

Defaults to “[]”.

List contents:

*

Map value expected.

Updates cause replacement.

Map properties:

Device

The device where the volume is exposed on the instance. This assignment may not be honored and it is advised that the path /dev/disk/by-id/virtio-<VolumeId> be used instead.

String value expected.

Updates cause replacement.

VolumeId

The ID of the volume to be attached.

String value expected.

Updates cause replacement.

Value must be of type cinder.volume

Attributes

AvailabilityZone

The Availability Zone where the specified instance is launched.

PrivateDnsName

Private DNS name of the specified instance.

PrivateIp

Private IP address of the specified instance.

PublicDnsName

Public DNS name of the specified instance.

PublicIp

Public IP address of the specified instance.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: AWS::EC2::Instance
    properties:
      AvailabilityZone: String
      BlockDeviceMappings: [{"Ebs": {"VolumeSize": String,
"DeleteOnTermination": Boolean, "SnapshotId": String}, "DeviceName":
String}, {"Ebs": {"VolumeSize": String, "DeleteOnTermination": Boolean,
"SnapshotId": String}, "DeviceName": String}, ...]
      ImageId: String
      InstanceType: String
      KeyName: String
      NetworkInterfaces: [Value, Value, ...]
      NovaSchedulerHints: [{"Key": String, "Value": String}, {"Key":
String, "Value": String}, ...]
      SecurityGroupIds: [Value, Value, ...]
      SecurityGroups: [Value, Value, ...]
      SubnetId: String
      Tags: [{"Key": String, "Value": String}, {"Key": String, "Value":
String}, ...]
      UserData: String
      Volumes: [{"Device": String, "VolumeId": String}, {"Device": String,
"VolumeId": String}, ...]
```

AWS::EC2::InternetGateway

Optional Properties

Tags

List value expected.

Updates cause replacement.

List contents:

*

> **Note**
>
> Not implemented.

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: AWS::EC2::InternetGateway
    properties:
      Tags: [{"Key": String, "Value": String}, {"Key": String, "Value":
String}, ...]
```

AWS::EC2::NetworkInterface

Required Properties

SubnetId

Subnet ID to associate with this interface.

String value expected.

Updates cause replacement.

Value must be of type neutron.subnet

Optional Properties

Description

Description for this interface.

String value expected.

Updates cause replacement.

GroupSet

List of security group IDs associated with this interface.

List value expected.

Can be updated without replacement.

PrivateIpAddress

String value expected.

Updates cause replacement.

SourceDestCheck

> **Note**
>
> Not implemented.

Tags

List value expected.

Updates cause replacement.

List contents:

*

> **Note**
>
> Not implemented.

Attributes

PrivateIpAddress

Private IP address of the network interface.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: AWS::EC2::NetworkInterface
    properties:
      Description: String
      GroupSet: [Value, Value, ...]
      PrivateIpAddress: String
      SubnetId: String
      Tags: [{"Key": String, "Value": String}, {"Key": String, "Value":
String}, ...]
```

AWS::EC2::RouteTable

> **Note**
>
> Available since 2014.1 (Icehouse)

Required Properties

VpcId

VPC ID for where the route table is created.

String value expected.

Updates cause replacement.

Optional Properties

Tags

List value expected.

Updates cause replacement.

List contents:

*

> **Note**
>
> Not implemented.

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: AWS::EC2::RouteTable
    properties:
      Tags: [{"Key": String, "Value": String}, {"Key": String, "Value":
String}, ...]
      VpcId: String
```

AWS::EC2::SecurityGroup

Required Properties

GroupDescription

Description of the security group.

String value expected.

Updates cause replacement.

Optional Properties

SecurityGroupEgress

List value expected.

Can be updated without replacement.

List contents:

*

List of security group egress rules.

Map value expected.

Can be updated without replacement.

Map properties:

CidrIp

String value expected.

Can be updated without replacement.

FromPort

String value expected.

Can be updated without replacement.

IpProtocol

String value expected.

Can be updated without replacement.

SourceSecurityGroupId

String value expected.

Can be updated without replacement.

SourceSecurityGroupName

String value expected.

Can be updated without replacement.

SourceSecurityGroupOwnerId

Note

Not implemented.

ToPort

String value expected.

Can be updated without replacement.

SecurityGroupIngress

List value expected.

Can be updated without replacement.

List contents:

*

List of security group ingress rules.

Map value expected.

Can be updated without replacement.

Map properties:

CidrIp

String value expected.

Can be updated without replacement.

FromPort

String value expected.

Can be updated without replacement.

IpProtocol

String value expected.

Can be updated without replacement.

SourceSecurityGroupId

String value expected.

Can be updated without replacement.

SourceSecurityGroupName

String value expected.

Can be updated without replacement.

SourceSecurityGroupOwnerId

Note

Not implemented.

ToPort

String value expected.

Can be updated without replacement.

VpcId

Physical ID of the VPC. Not implemented.

String value expected.

Updates cause replacement.

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: AWS::EC2::SecurityGroup
    properties:
      GroupDescription: String
      SecurityGroupEgress: [{"ToPort": String, "CidrIp": String,
"FromPort": String, "IpProtocol": String, "SourceSecurityGroupName":
String, "SourceSecurityGroupId": String}, {"ToPort": String, "CidrIp":
String, "FromPort": String, "IpProtocol": String,
"SourceSecurityGroupName": String, "SourceSecurityGroupId": String}, ...]
      SecurityGroupIngress: [{"ToPort": String, "CidrIp": String,
"FromPort": String, "IpProtocol": String, "SourceSecurityGroupName":
String, "SourceSecurityGroupId": String}, {"ToPort": String, "CidrIp":
String, "FromPort": String, "IpProtocol": String,
```

```
"SourceSecurityGroupName": String, "SourceSecurityGroupId": String}, ...]
      VpcId: String
```

AWS::EC2::Subnet

Required Properties

CidrBlock

CIDR block to apply to subnet.

String value expected.

Updates cause replacement.

VpcId

Ref structure that contains the ID of the VPC on which you want to create the subnet.

String value expected.

Updates cause replacement.

Optional Properties

AvailabilityZone

Availability zone in which you want the subnet.

String value expected.

Updates cause replacement.

Tags

List value expected.

Updates cause replacement.

List contents:

*

> **Note**
>
> Not implemented.

Attributes

AvailabilityZone

Availability Zone of the subnet.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: AWS::EC2::Subnet
    properties:
      AvailabilityZone: String
      CidrBlock: String
      Tags: [{"Key": String, "Value": String}, {"Key": String, "Value":
String}, ...]
      VpcId: String
```

AWS::EC2::SubnetRouteTableAssociation

Required Properties

RouteTableId

Route table ID.

String value expected.

Updates cause replacement.

SubnetId

Subnet ID.

String value expected.

Updates cause replacement.

Value must be of type neutron.subnet

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: AWS::EC2::SubnetRouteTableAssociation
    properties:
      RouteTableId: String
      SubnetId: String
```

AWS::EC2::VPC

Optional Properties

CidrBlock

CIDR block to apply to the VPC.

String value expected.

Updates cause replacement.

InstanceTenancy

> **Note**
>
> Not implemented.

Tags

List value expected.

Updates cause replacement.

List contents:

*

> **Note**
>
> Not implemented.

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: AWS::EC2::VPC
    properties:
      CidrBlock: String
      Tags: [{"Key": String, "Value": String}, {"Key": String, "Value":
String}, ...]
```

AWS::EC2::VPCGatewayAttachment

Required Properties

VpcId

VPC ID for this gateway association.

String value expected.

Updates cause replacement.

Optional Properties

InternetGatewayId

ID of the InternetGateway.

String value expected.

Updates cause replacement.

VpnGatewayId

> **Note**
>
> Not implemented.

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: AWS::EC2::VPCGatewayAttachment
    properties:
      InternetGatewayId: String
      VpcId: String
```

AWS::EC2::Volume

Required Properties

AvailabilityZone

The availability zone in which the volume will be created.

String value expected.

Updates are not supported. Resource update will fail on any attempt to update this property.

Optional Properties

Size

The size of the volume in GB.

Integer value expected.

Updates are not supported. Resource update will fail on any attempt to update this property.

The value must be at least 1.

SnapshotId

If specified, the backup used as the source to create the volume.

String value expected.

Updates are not supported. Resource update will fail on any attempt to update this property.

Value must be of type cinder.backup

Tags

The list of tags to associate with the volume.

List value expected.

Updates are not supported. Resource update will fail on any attempt to update this property.

List contents:

*

Map value expected.

Updates are not supported. Resource update will fail on any attempt to update this property.

Map properties:

Key

String value expected.

Updates are not supported. Resource update will fail on any attempt to update this property.

Value

String value expected.

Updates are not supported. Resource update will fail on any attempt to update this property.

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: AWS::EC2::Volume
    properties:
      AvailabilityZone: String
      Size: Integer
      SnapshotId: String
      Tags: [{"Key": String, "Value": String}, {"Key": String, "Value":
String}, ...]
```

AWS::EC2::VolumeAttachment

Required Properties

Device

The device where the volume is exposed on the instance. This assignment may not be honored and it is advised that the path /dev/disk/by-id/virtio-<VolumeId> be used instead.

String value expected.

Updates are not supported. Resource update will fail on any attempt to update this property.

Value must match pattern: /dev/vd[b-z]

InstanceId

The ID of the instance to which the volume attaches.

String value expected.

Updates are not supported. Resource update will fail on any attempt to update this property.

Value must be of type nova.server

VolumeId

The ID of the volume to be attached.

String value expected.

Updates are not supported. Resource update will fail on any attempt to update this property.

Value must be of type cinder.volume

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: AWS::EC2::VolumeAttachment
    properties:
      Device: String
      InstanceId: String
      VolumeId: String
```

AWS::ElasticLoadBalancing::LoadBalancer

Implements a HAProxy-bearing instance as a nested stack.

The template for the nested stack can be redefined with `loadbalancer_template` option in `heat.conf`.

Generally the image used for the instance must have the following packages installed or available for installation at runtime:

```
- heat-cfntools and its dependencies like python-psutil
- cronie
- socat
- haproxy
```

Current default builtin template uses Fedora 21 x86_64 base cloud image

(https://getfedora.org/cloud/download/) and apart from installing packages goes through some hoops around SELinux due to pecularities of heat-cfntools.

Required Properties

AvailabilityZones

The Availability Zones in which to create the load balancer.

List value expected.

Updates cause replacement.

Listeners

One or more listeners for this load balancer.

List value expected.

Updates cause replacement.

List contents:

*

Map value expected.

Updates cause replacement.

Map properties:

InstancePort

TCP port on which the instance server is listening.

Integer value expected.

Updates cause replacement.

LoadBalancerPort

The external load balancer port number.

Integer value expected.

Updates cause replacement.

PolicyNames

Note

Not implemented.

Protocol

The load balancer transport protocol to use.

String value expected.

Updates cause replacement.

Allowed values: TCP, HTTP

SSLCertificateId

Note

Not implemented.

Optional Properties

AppCookieStickinessPolicy

Note

Not implemented.

HealthCheck

An application health check for the instances.

Map value expected.

Updates cause replacement.

Map properties:

HealthyThreshold

The number of consecutive health probe successes required before moving the instance to the healthy state.

Integer value expected.

Updates cause replacement.

Interval

The approximate interval, in seconds, between health checks of an individual instance.

Integer value expected.

Updates cause replacement.

Target

The port being checked.

String value expected.

Updates cause replacement.

Timeout

Health probe timeout, in seconds.

Integer value expected.

Updates cause replacement.

UnhealthyThreshold

The number of consecutive health probe failures required before moving the instance to the unhealthy state

Integer value expected.

Updates cause replacement.

Instances

The list of instance IDs load balanced.

List value expected.

Can be updated without replacement.

LBCookieStickinessPolicy

> **Note**
>
> Not implemented.

SecurityGroups

List of Security Groups assigned on current LB.

List value expected.

Can be updated without replacement.

Subnets

> **Note**
>
> Not implemented.

Attributes

CanonicalHostedZoneName

The name of the hosted zone that is associated with the LoadBalancer.

CanonicalHostedZoneNameID

The ID of the hosted zone name that is associated with the LoadBalancer.

DNSName

The DNS name for the LoadBalancer.

SourceSecurityGroup.GroupName

The security group that you can use as part of your inbound rules for your LoadBalancer’s back-end instances.

SourceSecurityGroup.OwnerAlias

Owner of the source security group.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: AWS::ElasticLoadBalancing::LoadBalancer
    properties:
      AvailabilityZones: [Value, Value, ...]
      HealthCheck: {"HealthyThreshold": Integer, "Target": String,
"UnhealthyThreshold": Integer, "Timeout": Integer, "Interval": Integer}
      Instances: [Value, Value, ...]
      Listeners: [{"Protocol": String, "InstancePort": Integer,
"LoadBalancerPort": Integer}, {"Protocol": String, "InstancePort":
Integer, "LoadBalancerPort": Integer}, ...]
      SecurityGroups: [Value, Value, ...]
```

AWS::IAM::AccessKey

Required Properties

UserName

The name of the user that the new key will belong to.

String value expected.

Updates cause replacement.

Optional Properties

Serial

> **Note**
>
> Not implemented.

Status

> **Note**
>
> Not implemented.

Attributes

SecretAccessKey

Keypair secret key.

UserName

Username associated with the AccessKey.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: AWS::IAM::AccessKey
    properties:
      UserName: String
```

AWS::IAM::User

Optional Properties

Groups

Not Implemented.

List value expected.

Updates cause replacement.

LoginProfile

A login profile for the user.

Map value expected.

Updates cause replacement.

Map properties:

Password

String value expected.

Updates cause replacement.

Path

Not Implemented.

String value expected.

Updates cause replacement.

Policies

Access policies to apply to the user.

List value expected.

Updates cause replacement.

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: AWS::IAM::User
    properties:
      Groups: [Value, Value, ...]
      LoginProfile: {"Password": String}
      Path: String
      Policies: [Value, Value, ...]
```

AWS::S3::Bucket

Optional Properties

AccessControl

A predefined access control list (ACL) that grants permissions on the bucket.

String value expected.

Updates cause replacement.

Allowed values: Private, PublicRead, PublicReadWrite, AuthenticatedRead, BucketOwnerRead, BucketOwnerFullControl

Tags

Tags to attach to the bucket.

List value expected.

Updates cause replacement.

List contents:

*

Map value expected.

Updates cause replacement.

Map properties:

Key

The tag key name.

String value expected.

Updates cause replacement.

Value

The tag value.

String value expected.

Updates cause replacement.

WebsiteConfiguration

Information used to configure the bucket as a static website.

Map value expected.

Updates cause replacement.

Map properties:

ErrorDocument

The name of the error document.

String value expected.

Updates cause replacement.

IndexDocument

The name of the index document.

String value expected.

Updates cause replacement.

Attributes

DomainName

The DNS name of the specified bucket.

WebsiteURL

The website endpoint for the specified bucket.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: AWS::S3::Bucket
    properties:
      AccessControl: String
      Tags: [{"Key": String, "Value": String}, {"Key": String, "Value":
String}, ...]
      WebsiteConfiguration: {"IndexDocument": String, "ErrorDocument":
String}
```

Unsupported Heat Resource Types

These resources are enabled, but are not officially supported.

OS::Aodh::CombinationAlarm

> **Note**
>
> DEPRECATED since 7.0.0 (Newton) - The combination alarm is deprecated and disabled by default in Aodh.
>
> **Note**
>
> Available since 2014.1 (Icehouse)

A resource that implements combination of Aodh alarms.

Allows to use alarm as a combination of other alarms with some operator: activate this alarm if any alarm in combination has been activated or if all alarms in combination have been activated.

Required Properties

alarm_ids

List of alarm identifiers to combine.

List value expected.

Can be updated without replacement.

The length must be at least 1.

Optional Properties

alarm_actions

A list of URLs (webhooks) to invoke when state transitions to alarm.

List value expected.

Can be updated without replacement.

description

Description for the alarm.

String value expected.

Can be updated without replacement.

enabled

True if alarm evaluation/actioning is enabled.

Boolean value expected.

Can be updated without replacement.

Defaults to “true”.

insufficient_data_actions

A list of URLs (webhooks) to invoke when state transitions to insufficient-data.

List value expected.

Can be updated without replacement.

ok_actions

A list of URLs (webhooks) to invoke when state transitions to ok.

List value expected.

Can be updated without replacement.

operator

Operator used to combine the alarms.

String value expected.

Can be updated without replacement.

Allowed values: and, or

repeat_actions

False to trigger actions when the threshold is reached AND the alarm’s state has changed. By default, actions are called each time the threshold is reached.

Boolean value expected.

Can be updated without replacement.

Defaults to “true”.

severity

> **Note**
>
> Available since 5.0.0 (Liberty)

Severity of the alarm.

String value expected.

Can be updated without replacement.

Defaults to “low”.

Allowed values: low, moderate, critical

time_constraints

> **Note**
>
> Available since 5.0.0 (Liberty)

Describe time constraints for the alarm. Only evaluate the alarm if the time at evaluation is within this time constraint. Start point(s) of the constraint are specified with a cron expression, whereas its duration is given in seconds.

List value expected.

Updates cause replacement.

Defaults to “[]”.

List contents:

*

> Map value expected.
>
> Updates cause replacement.
>
> *Map properties:*
>
> description
>
> > Description for the time constraint.
> >
> > String value expected.
> >
> > Updates cause replacement.

duration

Duration for the time constraint.

Integer value expected.

Updates cause replacement.

The value must be at least 0.

name

Name for the time constraint.

String value expected.

Updates cause replacement.

start

Start time for the time constraint. A CRON expression property.

String value expected.

Updates cause replacement.

Value must be of type cron_expression

timezone

Timezone for the time constraint (eg. ‘Taiwan/Taipei’, ‘Europe/Amsterdam’).

String value expected.

Updates cause replacement.

Value must be of type timezone

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Aodh::CombinationAlarm
    properties:
      alarm_actions: [Value, Value, ...]
      alarm_ids: [Value, Value, ...]
      description: String
      enabled: Boolean
      insufficient_data_actions: [Value, Value, ...]
      ok_actions: [Value, Value, ...]
      operator: String
      repeat_actions: Boolean
      severity: String
      time_constraints: [{"duration": Integer, "name": String,
"description": String, "timezone": String, "start": String}, {"duration":
Integer, "name": String, "description": String, "timezone": String,
"start": String}, ...]
```

OS::Heat::HARestarter

> **Note**
>
> DEPRECATED since 2015.1 (Kilo) - The HARestarter resource type is deprecated and will be removed in a future release of Heat, once it has support for auto-healing any type of resource. Note that HARestarter does *not* actually restart servers - it deletes and then recreates them. It also does the same to all dependent resources, and may therefore exhibit unexpected and undesirable behaviour. Instead, use the mark-unhealthy API to mark a resource as needing replacement, and then a stack update to perform the replacement while respecting the dependencies and not deleting them unnecessarily.

Required Properties

InstanceId

Instance ID to be restarted.

String value expected.

Updates cause replacement.

Value must be of type nova.server

Attributes

AlarmUrl

A signed url to handle the alarm (Heat extension).

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Heat::HARestarter
    properties:
      InstanceId: String
```

OS::Neutron::ExtraRoute

> **Note**
>
> UNSUPPORTED - Use this resource at your own risk.

Resource for specifying extra routes for Neutron router.

Resource allows to specify nexthop IP and destination network for router.

Required Properties

destination

Network in CIDR notation.

String value expected.

Updates cause replacement.

nexthop

Nexthop IP address.

String value expected.

Updates cause replacement.

router_id

The router id.

String value expected.

Updates cause replacement.

Value must be of type neutron.router

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Neutron::ExtraRoute
    properties:
      destination: String
      nexthop: String
      router_id: String
```

OS::Neutron::HealthMonitor

> **Note**
>
> DEPRECATED since 7.0.0 (Newton) - Neutron LBaaS v1 is deprecated in the Liberty release and is planned to be removed in a future release. Going forward, the LBaaS V2 should be used.

A resource for managing health monitors for loadbalancers in Neutron.

A health monitor is used to determine whether or not back-end members of the VIP's pool are usable for processing a request. A pool can have several health monitors associated with it. There are different types of health monitors supported by the OpenStack LBaaS service:

- PING: used to ping the members using ICMP.
- TCP: used to connect to the members using TCP.
- HTTP: used to send an HTTP request to the member.
- HTTPS: used to send a secure HTTP request to the member.

Required Properties

delay

The minimum time in milliseconds between regular connections of the member.

Integer value expected.

Can be updated without replacement.

max_retries

Number of permissible connection failures before changing the member status to INACTIVE.

Integer value expected.

Can be updated without replacement.

timeout

Maximum number of milliseconds for a monitor to wait for a connection to be established before it times out.

Integer value expected.

Can be updated without replacement.

type

One of predefined health monitor types.

String value expected.

Updates cause replacement.

Allowed values: PING, TCP, HTTP, HTTPS

Optional Properties

admin_state_up

The administrative state of the health monitor.

Boolean value expected.

Can be updated without replacement.

Defaults to “True”.

expected_codes

The list of HTTP status codes expected in response from the member to declare it healthy.

String value expected.

Can be updated without replacement.

http_method

The HTTP method used for requests by the monitor of type HTTP.

String value expected.

Can be updated without replacement.

url_path

The HTTP path used in the HTTP request used by the monitor to test a member health.

String value expected.

Can be updated without replacement.

Attributes

admin_state_up

The administrative state of this health monitor.

delay

The minimum time in milliseconds between regular connections of the member.

expected_codes

The list of HTTP status codes expected in response from the member to declare it healthy.

http_method

The HTTP method used for requests by the monitor of type HTTP.

max_retries

Number of permissible connection failures before changing the member status to INACTIVE.

show

Detailed information about resource.

tenant_id

Tenant owning the health monitor.

timeout

Maximum number of milliseconds for a monitor to wait for a connection to be established before it times out.

type

One of predefined health monitor types.

url_path

The HTTP path used in the HTTP request used by the monitor to test a member health.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Neutron::HealthMonitor
    properties:
      admin_state_up: Boolean
      delay: Integer
      expected_codes: String
      http_method: String
      max_retries: Integer
      timeout: Integer
      type: String
      url_path: String
```

OS::Neutron::LoadBalancer

> **Note**
>
> DEPRECATED since 7.0.0 (Newton) - Neutron LBaaS v1 is deprecated in the Liberty release and is planned to be removed in a future release. Going forward, the LBaaS V2 should be used.

> **Note**
>
> Available since 2014.1 (Icehouse)

A resource to link a neutron pool with servers.

A loadbalancer allows linking a neutron pool with specified servers to some port.

Required Properties

pool_id

The ID of the load balancing pool.

String value expected.

Can be updated without replacement.

protocol_port

Port number on which the servers are running on the members.

Integer value expected.

Updates cause replacement.

The value must be in the range 0 to 65535.

Optional Properties

members

The list of Nova server IDs load balanced.

List value expected.

Can be updated without replacement.

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Neutron::LoadBalancer
    properties:
      members: [Value, Value, ...]
      pool_id: String
      protocol_port: Integer
```

OS::Neutron::Pool

> **Note**
>
> DEPRECATED since 7.0.0 (Newton) - Neutron LBaaS v1 is deprecated in the Liberty release and is planned to be removed in a future release. Going forward, the LBaaS V2 should be used.

A resource for managing load balancer pools in Neutron.

A load balancing pool is a logical set of devices, such as web servers, that you group together to receive and process traffic. The loadbalancing function chooses a member of the pool according to the configured load balancing method to handle the new requests or connections received on the VIP address. There is only one pool for a VIP.

Required Properties

lb_method

The algorithm used to distribute load between the members of the pool.

String value expected.

Can be updated without replacement.

Allowed values: ROUND_ROBIN, LEAST_CONNECTIONS, SOURCE_IP

protocol

Protocol for balancing.

String value expected.

Updates cause replacement.

Allowed values: TCP, HTTP, HTTPS

subnet

> **Note**
>
> Available since 2014.2 (Juno)

The subnet for the port on which the members of the pool will be connected.

String value expected.

Updates cause replacement.

Value must be of type neutron.subnet

vip

IP address and port of the pool.

Map value expected.

Updates cause replacement.

Map properties:

address

> IP address of the vip.
>
> String value expected.
>
> Updates cause replacement.
>
> Value must be of type ip_addr

admin_state_up

> The administrative state of this vip.
>
> Boolean value expected.
>
> Updates cause replacement.

Defaults to “True”.

connection_limit

The maximum number of connections per second allowed for the vip.

Integer value expected.

Updates cause replacement.

description

Description of the vip.

String value expected.

Updates cause replacement.

name

Name of the vip.

String value expected.

Updates cause replacement.

protocol_port

TCP port on which to listen for client traffic that is associated with the vip address.

Integer value expected.

Updates cause replacement.

session_persistence

Configuration of session persistence.

Map value expected.

Updates cause replacement.

Map properties:

cookie_name

Name of the cookie, required if type is APP_COOKIE.

String value expected.

Updates cause replacement.

type

Method of implementation of session persistence feature.

String value expected.

Updates cause replacement.

Allowed values: SOURCE_IP, HTTP_COOKIE, APP_COOKIE

subnet

Subnet of the vip.

String value expected.

Updates cause replacement.

Value must be of type neutron.subnet

Optional Properties

admin_state_up

The administrative state of this pool.

Boolean value expected.

Can be updated without replacement.

Defaults to “True”.

description

Description of the pool.

String value expected.

Can be updated without replacement.

monitors

List of health monitors associated with the pool.

List value expected.

Can be updated without replacement.

Defaults to “[]”.

name

Name of the pool.

String value expected.

Updates cause replacement.

provider

> **Note**
>
> Available since 5.0.0 (Liberty)

LBaaS provider to implement this load balancer instance.

String value expected.

Updates cause replacement.

Value must be of type neutron.lb.provider

Attributes

admin_state_up

The administrative state of this pool.

description

Description of the pool.

lb_method

The algorithm used to distribute load between the members of the pool.

name

Name of the pool.

protocol

Protocol to balance.

provider

> **Note**
>
> Available since 5.0.0 (Liberty)

Provider implementing this load balancer instance.

show

Detailed information about resource.

subnet_id

The subnet for the port on which the members of the pool will be connected.

tenant_id

Tenant owning the pool.

vip

Vip associated with the pool.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Neutron::Pool
    properties:
      admin_state_up: Boolean
      description: String
      lb_method: String
      monitors: [Value, Value, ...]
      name: String
      protocol: String
      provider: String
      subnet: String
      vip: {"protocol_port": Integer, "address": String,
"connection_limit": Integer, "admin_state_up": Boolean, "subnet": String,
"name": String, "session_persistence": {"type": String, "cookie_name":
String}, "description": String}
```

OS::Neutron::PoolMember

> **Note**
>
> DEPRECATED since 7.0.0 (Newton) - Neutron LBaaS v1 is deprecated in the Liberty release and is planned to be removed in a future release. Going forward, the LBaaS V2 should be used.

> **Note**
>
> Available since 2014.1 (Icehouse)

A resource to handle loadbalancer members.

A pool member represents the application running on backend server.

Required Properties

address

IP address of the pool member on the pool network.

String value expected.

Updates cause replacement.

Value must be of type ip_addr

pool_id

The ID of the load balancing pool.

String value expected.

Can be updated without replacement.

protocol_port

TCP port on which the pool member listens for requests or connections.

Integer value expected.

Updates cause replacement.

The value must be in the range 0 to 65535.

Optional Properties

admin_state_up

The administrative state of the pool member.

Boolean value expected.

Updates cause replacement.

Defaults to “True”.

weight

Weight of pool member in the pool (default to 1).

Integer value expected.

Can be updated without replacement.

The value must be in the range 0 to 256.

Attributes

address

IP address of the pool member.

admin_state_up

The administrative state of this pool member.

pool_id

The ID of the load balancing pool.

protocol_port

TCP port on which the pool member listens for requests or connections.

show

Detailed information about resource.

tenant_id

Tenant owning the pool member.

weight

Weight of the pool member in the pool.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Neutron::PoolMember
    properties:
      address: String
      admin_state_up: Boolean
      pool_id: String
      protocol_port: Integer
      weight: Integer
```

OS::Neutron::PortPair

> **Note**
>
> UNSUPPORTED since 7.0.0 (Newton)

A resource for neutron networking-sfc port-pair.

This plug-in requires networking-sfc>=1.0.0. So to enable this plug-in, install this library and restart the heat-engine.

A Port Pair represents a service function instance. The ingress port and the egress port of the service function may be specified. If a service function has one bidirectional port, the ingress port has the same value as the egress port.

Required Properties

egress

ID or name of the egress neutron port.

String value expected.

Updates cause replacement.

Value must be of type neutron.port

ingress

ID or name of the ingress neutron port.

String value expected.

Updates cause replacement.

Value must be of type neutron.port

Optional Properties

description

Description for the Port Pair.

String value expected.

Can be updated without replacement.

name

Name for the Port Pair.

String value expected.

Can be updated without replacement.

service_function_parameters

Dictionary of service function parameter. Currently only correlation=None is supported.

Map value expected.

Updates cause replacement.

Defaults to “{‘correlation’: None}”.

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Neutron::PortPair
    properties:
      description: String
      egress: String
      ingress: String
      name: String
      service_function_parameters: {...}
```

Contributed Heat Resource Types

These resources are not enabled by default.

OS::Nova::Server

> **Note**
>
> UNSUPPORTED - This resource is not supported, use at your own risk.

Resource for Rackspace Cloud Servers.

This resource overloads existent integrated OS::Nova::Server resource and is used for Rackspace Cloud Servers.

Required Properties

flavor

The ID or name of the flavor to boot onto.

String value expected.

Can be updated without replacement.

Value must be of type nova.flavor

Optional Properties

admin_pass

The administrator password for the server.

String value expected.

Can be updated without replacement.

availability_zone

Name of the availability zone for server placement.

String value expected.

Updates cause replacement.

block_device_mapping

Block device mappings for this server.

List value expected.

Updates cause replacement.

List contents:

*

Map value expected.

Updates cause replacement.

Map properties:

delete_on_termination

Indicate whether the volume should be deleted when the server is terminated.

Boolean value expected.

Updates cause replacement.

device_name

A device name where the volume will be attached in the system at /dev/device_name. This value is typically vda.

String value expected.

Updates cause replacement.

snapshot_id

The ID of the snapshot to create a volume from.

String value expected.

Updates cause replacement.

Value must be of type cinder.snapshot

volume_id

The ID of the volume to boot from. Only one of volume_id or snapshot_id should be provided.

String value expected.

Updates cause replacement.

Value must be of type cinder.volume

volume_size

The size of the volume, in GB. It is safe to leave this blank and have the Compute service infer the size.

Integer value expected.

Updates cause replacement.

block_device_mapping_v2

Note

Available since 2015.1 (Kilo)

Block device mappings v2 for this server.

List value expected.

Updates cause replacement.

List contents:

*

Map value expected.

Updates cause replacement.

Map properties:

boot_index

Integer used for ordering the boot disks.

Integer value expected.

Updates cause replacement.

delete_on_termination

Indicate whether the volume should be deleted when the server is terminated.

Boolean value expected.

Updates cause replacement.

device_name

A device name where the volume will be attached in the system at /dev/device_name. This value is typically vda.

String value expected.

Updates cause replacement.

device_type

Device type: at the moment we can make distinction only between disk and cdrom.

String value expected.

Updates cause replacement.

Allowed values: cdrom, disk

disk_bus

Bus of the device: hypervisor driver chooses a suitable default if omitted.

String value expected.

Updates cause replacement.

Allowed values: ide, lame_bus, scsi, usb, virtio

image

Note

Available since 7.0.0 (Newton)

The ID or name of the image to create a volume from.

String value expected.

Updates cause replacement.

Value must be of type glance.image

snapshot_id

The ID of the snapshot to create a volume from.

String value expected.

Updates cause replacement.

Value must be of type cinder.snapshot

swap_size

The size of the swap, in MB.

Integer value expected.

Updates cause replacement.

volume_id

The volume_id can be boot or non-boot device to the server.

String value expected.

Updates cause replacement.

Value must be of type cinder.volume

volume_size

Size of the block device in GB. If it is omitted, hypervisor driver calculates size.

Integer value expected.

Updates cause replacement.

image_id

Note

DEPRECATED since 7.0.0 (Newton) - Use property image.

Note

Available since 5.0.0 (Liberty)

> The ID of the image to create a volume from.
>
> String value expected.
>
> Updates cause replacement.
>
> Value must be of type glance.image

config_drive

If True, enable config drive on the server.

Boolean value expected.

Updates cause replacement.

diskConfig

Control how the disk is partitioned when the server is created.

String value expected.

Updates cause replacement.

Allowed values: AUTO, MANUAL

flavor_update_policy

Policy on how to apply a flavor update; either by requesting a server resize or by replacing the entire server.

String value expected.

Can be updated without replacement.

Defaults to “RESIZE”.

Allowed values: RESIZE, REPLACE

image

The ID or name of the image to boot with.

String value expected.

Can be updated without replacement.

Value must be of type glance.image

image_update_policy

Policy on how to apply an image-id update; either by requesting a server rebuild or by replacing the entire server.

String value expected.

Can be updated without replacement.

Defaults to “REBUILD”.

Allowed values: REBUILD, REPLACE, REBUILD_PRESERVE_EPHEMERAL

key_name

Name of keypair to inject into the server.

String value expected.

Updates cause replacement.

Value must be of type nova.keypair

metadata

Arbitrary key/value metadata to store for this server. Both keys and values must be 255 characters or less. Non-string values will be serialized to JSON (and the serialized string must be 255 characters or less).

Map value expected.

Can be updated without replacement.

name

Server name.

String value expected.

Can be updated without replacement.

networks

An ordered list of nics to be added to this server, with information about connected networks, fixed ips, port etc.

List value expected.

Can be updated without replacement.

List contents:

*

Map value expected.

Can be updated without replacement.

Map properties:

network

Name or ID of network to create a port on.

String value expected.

Can be updated without replacement.

Value must be of type neutron.network

port_extra_properties

Note

Available since 6.0.0 (Mitaka)

Dict, which has expand properties for port. Used only if port property is not specified for creating port.

Map value expected.

Can be updated without replacement.

Map properties:

admin_state_up

The administrative state of this port.

Boolean value expected.

Can be updated without replacement.

Defaults to “True”.

allowed_address_pairs

Additional MAC/IP address pairs allowed to pass through the port.

List value expected.

Can be updated without replacement.

List contents:

- *

  Map value expected.

  Can be updated without replacement.

  Map properties:

  ip_address

  IP address to allow through this port.

  String value expected.

  Can be updated without replacement.

  Value must be of type net_cidr

  mac_address

  MAC address to allow through this port.

  String value expected.

  Can be updated without replacement.

  Value must be of type mac_addr

  binding:vnic_type

Note

Available since 2015.1 (Kilo)

The vnic type to be bound on the neutron port. To support SR-IOV PCI passthrough networking, you can request that the neutron port to be realized as normal (virtual nic), direct (pci passthrough), or macvtap (virtual interface with a tap-like software interface). Note that this only works for Neutron deployments that support the bindings extension.

String value expected.

Can be updated without replacement.

Allowed values: normal, direct, macvtap

mac_address

MAC address to give to this port.

String value expected.

Can be updated without replacement.

Value must be of type mac_addr

port_security_enabled

Note

Available since 5.0.0 (Liberty)

Flag to enable/disable port security on the port. When disable this feature(set it to False), there will be no packages filtering, like security-group and address-pairs.

Boolean value expected.

Can be updated without replacement.

qos_policy

Note

Available since 6.0.0 (Mitaka)

The name or ID of QoS policy to attach to this port.

String value expected.

Can be updated without replacement.

Value must be of type neutron.qos_policy

value_specs

Extra parameters to include in the request.

Map value expected.

Can be updated without replacement.

Defaults to “{}”.

subnet

Note

Available since 5.0.0 (Liberty)

Subnet in which to allocate the IP address for port. Used for creating port, based on derived properties. If subnet is specified, network property becomes optional.

String value expected.

Can be updated without replacement.

fixed_ip

Fixed IP address to specify for the port created on the requested network.

String value expected.

Can be updated without replacement.

Value must be of type ip_addr

floating_ip

Note

Available since 6.0.0 (Mitaka)

ID of the floating IP to associate.

String value expected.

Can be updated without replacement.

port

ID of an existing port to associate with this server.

String value expected.

Can be updated without replacement.

Value must be of type neutron.port

personality

A map of files to create/overwrite on the server upon boot. Keys are file names and values are the file contents.

Map value expected.

Updates cause replacement.

Defaults to “{}”.

reservation_id

A UUID for the set of servers being requested.

String value expected.

Updates cause replacement.

scheduler_hints

Arbitrary key-value pairs specified by the client to help boot a server.

Map value expected.

Updates cause replacement.

security_groups

List of security group names or IDs. Cannot be used if neutron ports are associated with this server; assign security groups to the ports instead.

List value expected.

Updates cause replacement.

Defaults to “[]”.

software_config_transport

How the server should receive the metadata required for software configuration. POLL_TEMP_URL is the only supported transport on Rackspace Cloud. This property is retained for compatibility.

String value expected.

Can be updated without replacement.

Defaults to “POLL_TEMP_URL”.

Allowed values: POLL_TEMP_URL

user_data

User data script to be executed by cloud-init.

String value expected.

Can be updated without replacement.

Defaults to “”.

user_data_format

How the user_data should be formatted for the server. For RAW the user_data is passed to Nova unmodified. For SOFTWARE_CONFIG user_data is bundled as part of the software config data, and metadata is derived from any associated SoftwareDeployment resources.

String value expected.

Updates cause replacement.

Defaults to “RAW”.

Allowed values: RAW, SOFTWARE_CONFIG

user_data_update_policy

> **Note**
>
> Available since 6.0.0 (Mitaka)

Policy on how to apply a user_data update; either by ignoring it or by replacing the entire server.

String value expected.

Can be updated without replacement.

Defaults to “REPLACE”.

Allowed values: REPLACE, IGNORE

Attributes

accessIPv4

The manually assigned alternative public IPv4 address of the server.

accessIPv6

The manually assigned alternative public IPv6 address of the server.

addresses

A dict of all network addresses with corresponding port_id. Each network will have two keys in dict, they are network name and network id. The port ID may be obtained through the following expression: “{get_attr: [<server>, addresses, <network name_or_id>, 0, port]}”.

console_urls

> **Note**
>
> Available since 2015.1 (Kilo)

URLs of server’s consoles. To get a specific console type, the requested type can be specified as parameter to the get_attr function, e.g. get_attr: [<server>, console_urls, novnc]. Currently supported types are novnc, xvpvnc, spice-html5, rdp-html5, serial.

instance_name

AWS compatible instance name.

name

Name of the server.

networks

A dict of assigned network addresses of the form: {“public”: [ip1, ip2...], “private”: [ip3, ip4], “public_uuid”: [ip1, ip2...], “private_uuid”: [ip3, ip4]}. Each network will have two keys in dict, they are network name and network id.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: OS::Nova::Server
    properties:
      admin_pass: String
      availability_zone: String
      block_device_mapping: [{"device_name": String, "volume_size":
Integer, "snapshot_id": String, "volume_id": String,
"delete_on_termination": Boolean}, {"device_name": String, "volume_size":
Integer, "snapshot_id": String, "volume_id": String,
"delete_on_termination": Boolean}, ...]
      block_device_mapping_v2: [{"volume_id": String, "disk_bus": String,
"image_id": String, "device_type": String, "swap_size": Integer, "image":
String, "snapshot_id": String, "device_name": String, "volume_size":
Integer, "boot_index": Integer, "delete_on_termination": Boolean},
{"volume_id": String, "disk_bus": String, "image_id": String,
"device_type": String, "swap_size": Integer, "image": String,
"snapshot_id": String, "device_name": String, "volume_size": Integer,
"boot_index": Integer, "delete_on_termination": Boolean}, ...]
      config_drive: Boolean
      diskConfig: String
      flavor: String
      flavor_update_policy: String
      image: String
      image_update_policy: String
      key_name: String
      metadata: {...}
      name: String
```

```
      networks: [{"port": String, "floating_ip": String, "fixed_ip":
String, "uuid": String, "subnet": String, "port_extra_properties":
{"qos_policy": String, "port_security_enabled": Boolean,
"binding:vnic_type": String, "mac_address": String, "admin_state_up":
Boolean, "allowed_address_pairs": [{"ip_address": String, "mac_address":
String}, {"ip_address": String, "mac_address": String}, ...],
"value_specs": {...}}, "network": String}, {"port": String, "floating_ip":
String, "fixed_ip": String, "uuid": String, "subnet": String,
"port_extra_properties": {"qos_policy": String, "port_security_enabled":
Boolean, "binding:vnic_type": String, "mac_address": String,
"admin_state_up": Boolean, "allowed_address_pairs": [{"ip_address":
String, "mac_address": String}, {"ip_address": String, "mac_address":
String}, ...], "value_specs": {...}}, "network": String}, ...]
      personality: {...}
      reservation_id: String
      scheduler_hints: {...}
      security_groups: [Value, Value, ...]
      software_config_transport: String
      user_data: String
      user_data_format: String
      user_data_update_policy: String
```

Rackspace Cloud Resource Types

These resources are not enabled by default.

The resources in this module are for using Heat with the Rackspace Cloud. These resources either allow using Rackspace services that don't have equivalent services in OpenStack or account for differences between a generic OpenStack deployment and the Rackspace Cloud.

Rackspace resources depend on the dev branch of pyrax to work properly. More information about them can be found in the RACKSPACE_README.

Rackspace::AutoScale::Group

> **Note**
>
> UNSUPPORTED - This resource is not supported, use at your own risk.

Represents a scaling group.

Required Properties

groupConfiguration

Group configuration.

Map value expected.

Can be updated without replacement.

Map properties:

cooldown

Number of seconds after capacity changes during which further capacity changes are disabled.

Number value expected.

Can be updated without replacement.

maxEntities

Maximum number of entities in this scaling group.

Integer value expected.

Can be updated without replacement.

metadata

Arbitrary key/value metadata to associate with this group.

Map value expected.

Can be updated without replacement.

minEntities

Minimum number of entities in this scaling group.

Integer value expected.

Can be updated without replacement.

name

Name of the scaling group.

String value expected.

Can be updated without replacement.

launchConfiguration

Launch configuration.

Map value expected.

Can be updated without replacement.

Map properties:

args

Type-specific launch arguments.

Map value expected.

Can be updated without replacement.

Map properties:

loadBalancers

List of load balancers to hook the server up to. If not specified, no load balancing will be configured.

List value expected.

Can be updated without replacement.

Defaults to “[]”.

List contents:

*

Map value expected.

Can be updated without replacement.

Map properties:

loadBalancerId

ID of the load balancer.

String value expected.

Can be updated without replacement.

port

Server port to connect the load balancer to.

Integer value expected.

Can be updated without replacement.

server

Server creation arguments, as accepted by the Cloud Servers server creation API.

Map value expected.

Can be updated without replacement.

Map properties:

config_drive

Enable config drive on the instance.

Boolean value expected.

Can be updated without replacement.

diskConfig

Configuration specifying the partition layout. AUTO to create a partition utilizing the entire disk, and MANUAL to create a partition matching the source image.

String value expected.

Can be updated without replacement.

Allowed values: AUTO, MANUAL

flavorRef

The ID or name of the flavor to boot onto.

String value expected.

Can be updated without replacement.

Value must be of type nova.flavor

imageRef

The ID or name of the image to boot with.

String value expected.

Can be updated without replacement.

Value must be of type glance.image

key_name

Name of a previously created SSH keypair to allow key-based authentication to the server.

String value expected.

Can be updated without replacement.

metadata

Metadata key and value pairs.

Map value expected.

Can be updated without replacement.

name

Server name.

String value expected.

Can be updated without replacement.

networks

Networks to attach to. If unspecified, the instance will be attached to the public Internet and private ServiceNet networks.

List value expected.

Can be updated without replacement.

List contents:

*

Map value expected.

Can be updated without replacement.

Map properties:

uuid

UUID of network to attach to.

String value expected.

Can be updated without replacement.

personality

File path and contents.

Map value expected.

Can be updated without replacement.

user_data

User data for bootstrapping the instance.

String value expected.

Can be updated without replacement.

stack

The attributes that Auto Scale uses to create a new stack. The attributes that you specify for the stack entity apply to all new stacks in the scaling group. Note the stack arguments are directly passed to Heat when creating a stack.

Map value expected.

Can be updated without replacement.

Map properties:

disable_rollback

Keep the resources that have been created if the stack fails to create. Defaults to True.

Boolean value expected.

Can be updated without replacement.

Defaults to “True”.

environment

The environment for the stack.

Map value expected.

Can be updated without replacement.

files

The contents of files that the template references.

Map value expected.

Can be updated without replacement.

parameters

Key/value pairs of the parameters and their values to pass to the parameters in the template.

Map value expected.

Can be updated without replacement.

template

The template that describes the stack. Either the template or template_url property must be specified.

String value expected.

Can be updated without replacement.

template_url

A URI to a template. Either the template or template_url property must be specified.

String value expected.

Can be updated without replacement.

timeout_mins

The stack creation timeout in minutes.

Integer value expected.

Can be updated without replacement.

type

Launch configuration method. Only launch_server and launch_stack are currently supported.

String value expected.

Can be updated without replacement.

Allowed values: launch_server, launch_stack

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: Rackspace::AutoScale::Group
    properties:
      groupConfiguration: {"minEntities": Integer, "name": String,
"metadata": {...}, "maxEntities": Integer, "cooldown": Number}
      launchConfiguration: {"type": String, "args": {"stack":
{"template_url": String, "timeout_mins": Integer, "files": {...},
"template": String, "environment": {...}, "parameters": {...},
"disable_rollback": Boolean}, "loadBalancers": [{"port": Integer,
"loadBalancerId": String}, {"port": Integer, "loadBalancerId": String},
...], "server": {"networks": [{"uuid": String}, {"uuid": String}, ...],
"user_data": String, "diskConfig": String, "key_name": String, "imageRef":
String, "metadata": {...}, "name": String, "config_drive": Boolean,
"flavorRef": String, "personality": {...}}}}
```

Rackspace::AutoScale::ScalingPolicy

> **Note**
>
> UNSUPPORTED - This resource is not supported, use at your own risk.

Represents a Rackspace Auto Scale scaling policy.

Required Properties

group

Scaling group ID that this policy belongs to.

String value expected.

Updates cause replacement.

name

Name of this scaling policy.

String value expected.

Can be updated without replacement.

type

Type of this scaling policy. Specifies how the policy is executed.

String value expected.

Can be updated without replacement.

Allowed values: webhook, schedule, cloud_monitoring

Optional Properties

args

Type-specific arguments for the policy.

Map value expected.

Can be updated without replacement.

change

Amount to add to or remove from current number of instances. Incompatible with changePercent and desiredCapacity.

Integer value expected.

Can be updated without replacement.

changePercent

Percentage-based change to add or remove from current number of instances. Incompatible with change and desiredCapacity.

Number value expected.

Can be updated without replacement.

cooldown

Number of seconds after a policy execution during which further executions are disabled.

Number value expected.

Can be updated without replacement.

desiredCapacity

Absolute number to set the number of instances to. Incompatible with change and changePercent.

Integer value expected.

Can be updated without replacement.

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: Rackspace::AutoScale::ScalingPolicy
    properties:
      args: {...}
      change: Integer
      changePercent: Number
      cooldown: Number
      desiredCapacity: Integer
      group: String
      name: String
      type: String
```

Rackspace::AutoScale::WebHook

> **Note**
>
> UNSUPPORTED - This resource is not supported, use at your own risk.

Represents a Rackspace AutoScale webhook.

Exposes the URLs of the webhook as attributes.

Required Properties

name

The name of this webhook.

String value expected.

Can be updated without replacement.

policy

The policy that this webhook should apply to, in {group_id}:{policy_id} format. Generally a Ref to a Policy resource.

String value expected.

Updates cause replacement.

Optional Properties

metadata

Arbitrary key/value metadata for this webhook.

Map value expected.

Can be updated without replacement.

Attributes

capabilityUrl

The url for executing the webhook (doesn't require auth).

executeUrl

The url for executing the webhook (requires auth).

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: Rackspace::AutoScale::WebHook
    properties:
      metadata: {...}
      name: String
      policy: String
```

Rackspace::Cloud::DNS

> **Note**
>
> UNSUPPORTED - This resource is not supported, use at your own risk.

Represents a DNS resource.

Required Properties

emailAddress

Email address to use for contacting the domain administrator.

String value expected.

Can be updated without replacement.

name

Specifies the name for the domain or subdomain. Must be a valid domain name.

String value expected.

Updates cause replacement.

The length must be at least 3.

Optional Properties

comment

Optional free form text comment

String value expected.

Can be updated without replacement.

The length must be no greater than 160.

records

Domain records

List value expected.

Can be updated without replacement.

List contents:

*

> Map value expected.
>
> Can be updated without replacement.
>
> *Map properties:*
>
> comment
>
> > Optional free form text comment
> >
> > String value expected.

Can be updated without replacement.

The length must be no greater than 160.

data

Type specific record data

String value expected.

Can be updated without replacement.

name

Specifies the name for the domain or subdomain. Must be a valid domain name.

String value expected.

Can be updated without replacement.

The length must be at least 3.

priority

Required for MX and SRV records, but forbidden for other record types. If specified, must be an integer from 0 to 65535.

Integer value expected.

Can be updated without replacement.

The value must be in the range 0 to 65535.

ttl

How long other servers should cache recorddata.

Integer value expected.

Can be updated without replacement.

Defaults to “3600”.

The value must be at least 300.

type

Specifies the record type.

String value expected.

Can be updated without replacement.

Allowed values: A, AAAA, NS, MX, CNAME, TXT, SRV

ttl

How long other servers should cache recorddata.

Integer value expected.

Can be updated without replacement.

Defaults to “3600”.

The value must be at least 300.

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: Rackspace::Cloud::DNS
    properties:
      comment: String
      emailAddress: String
      name: String
      records: [{"type": String, "data": String, "ttl": Integer,
"comment": String, "name": String, "priority": Integer}, {"type": String,
"data": String, "ttl": Integer, "comment": String, "name": String,
"priority": Integer}, ...]
      ttl: Integer
```

Rackspace::Cloud::LBNode

Represents a single node of a Rackspace Cloud Load Balancer

Required Properties

address

IP address for the node.

String value expected.

Updates cause replacement.

load_balancer

The ID of the load balancer to associate the node with.

String value expected.

Updates cause replacement.

port

Integer value expected.

Updates cause replacement.

Optional Properties

condition

String value expected.

Can be updated without replacement.

Defaults to “ENABLED”.

Allowed values: ENABLED, DISABLED, DRAINING

draining_timeout

The time to wait, in seconds, for the node to drain before it is deleted.

Integer value expected.

Can be updated without replacement.

Defaults to “0”.

The value must be at least 0.

type

String value expected.

Can be updated without replacement.

Allowed values: PRIMARY, SECONDARY

weight

Number value expected.

Can be updated without replacement.

The value must be in the range 1 to 100.

Attributes

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: Rackspace::Cloud::LBNode
    properties:
      address: String
      condition: String
      draining_timeout: Integer
      load_balancer: String
      port: Integer
      type: String
      weight: Number
```

Rackspace::Cloud::LoadBalancer

> **Note**
>
> UNSUPPORTED - This resource is not supported, use at your own risk.

Represents a Rackspace Cloud Loadbalancer.

Required Properties

nodes

List value expected.

Can be updated without replacement.

List contents:

*

Map value expected.

Can be updated without replacement.

Map properties:

addresses

IP addresses for the load balancer node. Must have at least one address.

List value expected.

Can be updated without replacement.

List contents:

*

String value expected.

Can be updated without replacement.

condition

String value expected.

Can be updated without replacement.

Defaults to “ENABLED”.

Allowed values: ENABLED, DISABLED, DRAINING

port

Integer value expected.

Can be updated without replacement.

type

String value expected.

Can be updated without replacement.

Defaults to “PRIMARY”.

Allowed values: PRIMARY, SECONDARY

weight

Number value expected.

Can be updated without replacement.

Defaults to “1”.

The value must be in the range 1 to 100.

port

Integer value expected.

Can be updated without replacement.

protocol

String value expected.

Can be updated without replacement.

Allowed values: DNS_TCP, DNS_UDP, FTP, HTTP, HTTPS, IMAPS, IMAPv4, LDAP, LDAPS, MYSQL, POP3, POP3S, SMTP, TCP, TCP_CLIENT_FIRST, UDP, UDP_STREAM, SFTP

virtualIps

List value expected.

Updates cause replacement.

The length must be at least 1.

List contents:

*

Map value expected.

Updates cause replacement.

Map properties:

id

ID of a shared VIP to use instead of creating a new one. This property cannot be specified if type or version is specified.

Number value expected.

Updates cause replacement.

ipVersion

IP version of the VIP. This property cannot be specified if ‘id’ is specified. This property must be specified if id is not specified.

String value expected.

Updates cause replacement.

Allowed values: IPV6, IPV4

type

The type of VIP (public or internal). This property cannot be specified if ‘id’ is specified. This property must be specified if id is not specified.

String value expected.

Updates cause replacement.

Allowed values: SERVICENET, PUBLIC

Optional Properties

accessList

List value expected.

Updates cause replacement.

List contents:

*

Map value expected.

Updates cause replacement.

Map properties:

address

String value expected.

Updates cause replacement.

type

String value expected.

Updates cause replacement.

Allowed values: ALLOW, DENY

algorithm

String value expected.

Can be updated without replacement.

Allowed values: LEAST_CONNECTIONS, RANDOM, ROUND_ROBIN, WEIGHTED_LEAST_CONNECTIONS, WEIGHTED_ROUND_ROBIN

connectionLogging

Boolean value expected.

Can be updated without replacement.

connectionThrottle

Map value expected.

Can be updated without replacement.

Map properties:

maxConnectionRate

Number value expected.

Can be updated without replacement.

The value must be in the range 0 to 100000.

maxConnections

Integer value expected.

Can be updated without replacement.

The value must be in the range 1 to 100000.

minConnections

Integer value expected.

Can be updated without replacement.

The value must be in the range 1 to 1000.

rateInterval

Number value expected.

Can be updated without replacement.

The value must be in the range 1 to 3600.

contentCaching

String value expected.

Can be updated without replacement.

Allowed values: ENABLED, DISABLED

errorPage

String value expected.

Can be updated without replacement.

halfClosed

Boolean value expected.

Can be updated without replacement.

healthMonitor

Map value expected.

Can be updated without replacement.

Map properties:

attemptsBeforeDeactivation

Number value expected.

Can be updated without replacement.

The value must be in the range 1 to 10.

bodyRegex

String value expected.

Can be updated without replacement.

delay

Number value expected.

Can be updated without replacement.

The value must be in the range 1 to 3600.

hostHeader

String value expected.

Can be updated without replacement.

path

String value expected.

Can be updated without replacement.

statusRegex

String value expected.

Can be updated without replacement.

timeout

Number value expected.

Can be updated without replacement.

The value must be in the range 1 to 300.

type

String value expected.

Can be updated without replacement.

Allowed values: CONNECT, HTTP, HTTPS

httpsRedirect

> **Note**
>
> Available since 2015.1 (Kilo)

Enables or disables HTTP to HTTPS redirection for the load balancer. When enabled, any HTTP request returns status code 301 (Moved Permanently), and the requester is redirected to the requested URL via the HTTPS protocol on port 443. Only available for HTTPS protocol (port=443), or HTTP protocol with a properly configured SSL termination (secureTrafficOnly=true, securePort=443).

Boolean value expected.

Can be updated without replacement.

Defaults to “False”.

metadata

Map value expected.

Can be updated without replacement.

name

String value expected.

Can be updated without replacement.

sessionPersistence

String value expected.

Can be updated without replacement.

Allowed values: HTTP_COOKIE, SOURCE_IP

sslTermination

Map value expected.

Can be updated without replacement.

Map properties:

certificate

> String value expected.
>
> Can be updated without replacement.

intermediateCertificate

String value expected.

Can be updated without replacement.

privatekey

String value expected.

Can be updated without replacement.

securePort

Integer value expected.

Can be updated without replacement.

Defaults to “443”.

secureTrafficOnly

Boolean value expected.

Can be updated without replacement.

Defaults to “False”.

timeout

Number value expected.

Can be updated without replacement.

The value must be in the range 1 to 120.

Attributes

PublicIp

Public IP address of the specified instance.

show

Detailed information about resource.

virtualIps

A list of assigned virtual ip addresses

HOT Syntax

```
heat_template_version: 2015-04-30
```

```
...
resources:
  ...
  the_resource:
    type: Rackspace::Cloud::LoadBalancer
    properties:
      accessList: [{"address": String, "type": String}, {"address":
String, "type": String}, ...]
      algorithm: String
      connectionLogging: Boolean
      connectionThrottle: {"rateInterval": Number, "maxConnectionRate":
Number, "minConnections": Integer, "maxConnections": Integer}
      contentCaching: String
      errorPage: String
      halfClosed: Boolean
      healthMonitor: {"timeout": Number, "type": String, "bodyRegex":
String, "hostHeader": String, "path": String, "delay": Number,
"attemptsBeforeDeactivation": Number, "statusRegex": String}
      httpsRedirect: Boolean
      metadata: {...}
      name: String
      nodes: [{"port": Integer, "weight": Number, "condition": String,
"type": String, "addresses": [String, String, ...]}, {"port": Integer,
"weight": Number, "condition": String, "type": String, "addresses":
[String, String, ...]}, ...]
      port: Integer
      protocol: String
      sessionPersistence: String
      sslTermination: {"privatekey": String, "secureTrafficOnly": Boolean,
"securePort": Integer, "certificate": String, "intermediateCertificate":
String}
      timeout: Number
      virtualIps: [{"type": String, "id": Number, "ipVersion": String},
{"type": String, "id": Number, "ipVersion": String}, ...]
```

DockerInc Resource

This resource is not enabled by default.

This plugin enables the use of Docker containers in a Heat template and requires the docker-py package. You can find more information in the DOCKER_README.

DockerInc::Docker::Container

> **Note**
>
> UNSUPPORTED - This resource is not supported, use at your own risk.

Optional Properties

cap_add

> **Note**
>
> Available since 2015.1 (Kilo)

Be used to add kernel capabilities (only supported for API version >= 1.2.0).

List value expected.

Updates cause replacement.

Defaults to “[]”.

List contents:

*

> The security features provided by Linux kernels.
>
> String value expected.
>
> Updates cause replacement.
>
> Allowed values: SETPCAP, SYS_MODULE, SYS_RAWIO, SYS_PACCT, SYS_ADMIN, SYS_NICE, SYS_RESOURCE, SYS_TIME, SYS_TTY_CONFIG, MKNOD, AUDIT_WRITE, AUDIT_CONTROL, MAC_OVERRIDE, MAC_ADMIN, NET_ADMIN, SYSLOG, CHOWN, NET_RAW, DAC_OVERRIDE, FOWNER, DAC_READ_SEARCH, FSETID, KILL, SETGID, SETUID, LINUX_IMMUTABLE, NET_BIND_SERVICE, NET_BROADCAST, IPC_LOCK, IPC_OWNER, SYS_CHROOT, SYS_PTRACE, SYS_BOOT, LEASE, SETFCAP, WAKE_ALARM, BLOCK_SUSPEND, ALL

cap_drop

> **Note**

Available since 2015.1 (Kilo)

Be used to drop kernel capabilities (only supported for API version >= 1.2.0).

List value expected.

Updates cause replacement.

Defaults to “[]”.

List contents:

*

The security features provided by Linux kernels.

String value expected.

Updates cause replacement.

Allowed values: SETPCAP, SYS_MODULE, SYS_RAWIO, SYS_PACCT, SYS_ADMIN, SYS_NICE, SYS_RESOURCE, SYS_TIME, SYS_TTY_CONFIG, MKNOD, AUDIT_WRITE, AUDIT_CONTROL, MAC_OVERRIDE, MAC_ADMIN, NET_ADMIN, SYSLOG, CHOWN, NET_RAW, DAC_OVERRIDE, FOWNER, DAC_READ_SEARCH, FSETID, KILL, SETGID, SETUID, LINUX_IMMUTABLE, NET_BIND_SERVICE, NET_BROADCAST, IPC_LOCK, IPC_OWNER, SYS_CHROOT, SYS_PTRACE, SYS_BOOT, LEASE, SETFCAP, WAKE_ALARM, BLOCK_SUSPEND, ALL

cmd

Command to run after spawning the container.

List value expected.

Updates cause replacement.

Defaults to “[]”.

cpu_set

Note

Available since 5.0.0 (Liberty)

The CPUs in which to allow execution (only supported for API version >= 1.12).

String value expected.

Updates cause replacement.

cpu_shares

> **Note**
>
> Available since 5.0.0 (Liberty)

Relative weight which determines the allocation of the CPU processing power(only supported for API version >= 1.8).

Integer value expected.

Updates cause replacement.

Defaults to “0”.

devices

> **Note**
>
> Available since 5.0.0 (Liberty)

Device mappings (only supported for API version >= 1.14).

List value expected.

Updates cause replacement.

Defaults to “[]”.

List contents:

*

> Map value expected.
>
> Updates cause replacement.
>
> *Map properties:*
>
> path_in_container
>
> > The device path of the container mappings to the host.
> >
> > String value expected.
> >
> > Updates cause replacement.

The length must be no greater than 255.

Value must match pattern: ^/dev/[/_\-a-zA-Z0-9]+$

path_on_host

The device path on the host.

String value expected.

Updates cause replacement.

The length must be no greater than 255.

Value must match pattern: ^/dev/[/_\-a-zA-Z0-9]+$

permissions

The permissions of the container to read/write/create the devices.

String value expected.

Updates cause replacement.

Defaults to “rwm”.

Allowed values: r, w, m, rw, rm, wm, rwm

dns

Set custom dns servers.

List value expected.

Updates cause replacement.

docker_endpoint

Docker daemon endpoint (by default the local docker daemon will be used).

String value expected.

Updates cause replacement.

env

Set environment variables.

List value expected.

Updates cause replacement.

hostname

Hostname of the container.

String value expected.

Updates cause replacement.

Defaults to “”.

image

Image name.

String value expected.

Updates cause replacement.

links

Links to other containers.

Map value expected.

Updates cause replacement.

memory

Memory limit (Bytes).

Integer value expected.

Updates cause replacement.

name

Name of the container.

String value expected.

Updates cause replacement.

open_stdin

Open stdin.

Boolean value expected.

Updates cause replacement.

Defaults to “False”.

port_bindings

TCP/UDP ports bindings.

Map value expected.

Updates cause replacement.

port_specs

TCP/UDP ports mapping.

List value expected.

Updates cause replacement.

privileged

Enable extended privileges.

Boolean value expected.

Updates cause replacement.

Defaults to “False”.

read_only

> **Note**
>
> Available since 2015.1 (Kilo)

If true, mount the container’s root filesystem as read only (only supported for API version >= 1.17).

Boolean value expected.

Updates cause replacement.

Defaults to “False”.

restart_policy

> **Note**
>
> Available since 2015.1 (Kilo)

Restart policies (only supported for API version >= 1.2.0).

Map value expected.

Updates cause replacement.

Defaults to “{}”.

Map properties:

MaximumRetryCount

> A maximum restart count for the on-failure policy.
>
> Integer value expected.
>
> Updates cause replacement.
>
> Defaults to “0”.

Name

> The behavior to apply when the container exits.
>
> String value expected.
>
> Updates cause replacement.
>
> Defaults to “no”.
>
> Allowed values: no, on-failure, always

stdin_once

If true, close stdin after the 1 attached client disconnects.

Boolean value expected.

Updates cause replacement.

Defaults to “False”.

tty

Allocate a pseudo-tty.

Boolean value expected.

Updates cause replacement.

Defaults to “False”.

user

Username or UID.

String value expected.

Updates cause replacement.

Defaults to “”.

volumes

Create a bind mount.

Map value expected.

Updates cause replacement.

Defaults to “{}”.

volumes_from

Mount all specified volumes.

List value expected.

Updates cause replacement.

Defaults to “”.

Attributes

info

Container info.

logs

Container logs.

logs_head

Container first logs line.

logs_tail

Container last logs line.

network_gateway

Container ip gateway.

network_info

Container network info.

network_ip

Container ip address.

network_tcp_ports

Container TCP ports.

network_udp_ports

Container UDP ports.

show

Detailed information about resource.

HOT Syntax

```
heat_template_version: 2015-04-30
...
resources:
  ...
  the_resource:
    type: DockerInc::Docker::Container
    properties:
      cap_add: [String, String, ...]
      cap_drop: [String, String, ...]
      cmd: [Value, Value, ...]
      cpu_set: String
      cpu_shares: Integer
      devices: [{"path_on_host": String, "permissions": String,
"path_in_container": String}, {"path_on_host": String, "permissions":
String, "path_in_container": String}, ...]
      dns: [Value, Value, ...]
      docker_endpoint: String
      env: [Value, Value, ...]
      hostname: String
      image: String
      links: {...}
      memory: Integer
      name: String
      open_stdin: Boolean
      port_bindings: {...}
      port_specs: [Value, Value, ...]
      privileged: Boolean
      read_only: Boolean
      restart_policy: {"Name": String, "MaximumRetryCount": Integer}
      stdin_once: Boolean
      tty: Boolean
      user: String
      volumes: {...}
      volumes_from: [Value, Value, ...]
```

CloudFormation Compatible Functions

There are a number of functions that you can use to help you write CloudFormation

compatible templates. While most CloudFormation functions are supported in HOT version '2013-05-23', *Fn::Select* is the only CloudFormation function supported in HOT templates since version '2014-10-16' which is introduced in Juno.

All of these functions (except *Ref*) start with *Fn::*.

Ref

Returns the value of the named parameter or resource.

Parameters

name : String
The name of the resource or parameter.

Usage

```
{Ref: my_server}
```

Returns the nova instance ID. For example, `d8093de0-850f-4513-b202-7979de6c0d55`.

Fn::Base64

This is a placeholder for a function to convert an input string to Base64. This function in Heat actually performs no conversion. It is included for the benefit of CFN templates that convert UserData to Base64. Heat only accepts UserData in plain text.

Parameters

value : String
The string to convert.

Usage

```
{"Fn::Base64": "convert this string please."}
```

Returns the original input string.

Fn::FindInMap

Returns the value corresponding to keys into a two-level map declared in the Mappings section.

Parameters

map_name : String
The logical name of a mapping declared in the Mappings section that contains the keys and values.

top_level_key : String
The top-level key name. It's value is a list of key-value pairs.

second_level_key : String
The second-level key name, which is set to one of the keys from the list assigned to top_level_key.

Usage

```
Mapping:
  MyContacts:
    jone: {phone: 337, email: a@b.com}
    jim: {phone: 908, email: g@b.com}

{"Fn::FindInMap": ["MyContacts", "jim", "phone" ] }
```

Returns `908`.

Fn::GetAtt

Returns an attribute of a resource within the template.

Parameters

resource : String
The name of the resource.

attribute : String
The name of the attribute.

Usage

```
{Fn::GetAtt: [my_server, PublicIp]}
```

Returns an IP address such as `10.0.0.2`.

Fn::GetAZs

Returns the Availability Zones within the given region.

Note: AZ's and regions are not fully implemented in Heat.

Parameters

region : String
The name of the region.

Usage

```
{Fn::GetAZs: ""}
```

Returns the list provided by `nova availability-zone-list`.

Fn::Join

Like python join, it joins a list of strings with the given delimiter.

Parameters

delimiter : String
The string to join the list with.
list : list
The list to join.

Usage

```
{Fn::Join: [",", ["beer", "wine", "more beer"]]}
```

Returns `beer, wine, more beer`.

Fn::Select

Select an item from a list.

Heat extension: Select an item from a map

Parameters

selector : string or integer
The number of item in the list or the name of the item in the map.
collection : map or list
The collection to select the item from.

Usage

For a list lookup:

```
{ "Fn::Select" : [ "2", [ "apples", "grapes", "mangoes" ] ] }
```

Returns `mangoes`.

For a map lookup:

```
{ "Fn::Select" : [ "red", {"red": "a", "flu": "b"} ] }
```

Returns a.

Fn::Split

This is the reverse of Join. Convert a string into a list based on the delimiter.

Parameters

delimiter : string
Matching string to split on.

string : String
The string to split.

Usage

```
{ "Fn::Split" : [ ",", "str1,str2,str3,str4"]}
```

Returns `{["str1", "str2", "str3", "str4"]}`.

Fn::Replace

Find and replace one string with another.

Parameters

substitutions : map
A map of substitutions.

string: String
The string to do the substitutions in.

Usage

```
{"Fn::Replace": [
 {'$var1': 'foo', '%var2%': 'bar'},
  '$var1 is %var2%'
]}
```

Returns `"foo is bar"`.

Fn::ResourceFacade

When writing a Template Resource:

- user writes a template that will fill in for a resource (the resource is the facade).
- when they are writing their template they need to access the metadata from the facade.

Parameters

attribute_name : String
One of `Metadata`, `DeletionPolicy` or `UpdatePolicy`.

Usage

```
{'Fn::ResourceFacade': 'Metadata'}
{'Fn::ResourceFacade': 'DeletionPolicy'}
{'Fn::ResourceFacade': 'UpdatePolicy'}
```

Example

Here is a top level template `top.yaml`

```
resources:
  my_server:
    type: OS::Nova::Server
    metadata:
      key: value
      some: more stuff
```

Here is a resource template `my_actual_server.yaml`

```
resources:
  _actual_server_:
    type: OS::Nova::Server
    metadata: {'Fn::ResourceFacade': Metadata}
```

The environment file `env.yaml`

```
resource_registry:
  resources:
    my_server:
      "OS::Nova::Server": my_actual_server.yaml
```

To use it

```
$ heat stack-create -f top.yaml -e env.yaml
```

What happened is the metadata in `top.yaml` (key: value, some: more stuff) gets passed into the resource template via the Fn::ResourceFacade function.

Fn::MemberListToMap

Convert an AWS style member list into a map.

Parameters

key name: string
The name of the key (normally “Name” or “Key”).

value name: string
The name of the value (normally “Value”).

list: A list of strings
The string to convert.

Usage

```
{'Fn::MemberListToMap': ['Name', 'Value', ['.member.0.Name=key',
                                           '.member.0.Value=door',
                                           '.member.1.Name=colour',
                                           '.member.1.Value=green']]}
```

Returns `{'key': 'door', 'colour': 'green'}`.

Fn::Equals

Compares whether two values are equal. And returns true if the two values are equal or false if they aren’t.

Parameters

value1:
A value of any type that you want to compare.

value2:
A value of any type that you want to compare.

Usage

```
{'Fn::Equals': [{'Ref': 'env_type'}, 'prod']}
```

Returns true if the param ‘env_type’ equals to ‘prod’, otherwise returns false.

This page documents the templates at https://git.openstack.org/cgit/openstack/heat-templates/

Example HOT Templates

- Hello World HOT Template
- Description
- Parameters

Example CFN Templates

- AWS Wordpress Single Instance Template
- Description
- Parameters

Hello World HOT Template

https://git.openstack.org/cgit/openstack/heat-templates/tree/hot/hello_world.yaml

Description

Hello world HOT template that just defines a single compute instance. Contains just base features to verify base HOT support.

Parameters

key_name **(required)**

type

string

description

Name of an existing key pair to use for the instance

flavor **(optional)**

type

string

description

Flavor for the instance to be created

image **(required)**

type

string

description

Image *ID* or image name to use for the instance

admin_pass **(required)**

type

string

description

The admin password for the instance

db_port **(optional)**

type

number

description

The database port number

AWS Wordpress Single Instance Template

https://git.openstack.org/cgit/openstack/heat-templates/tree/cfn/F18/WordPress_Single_Instance.template

Description

AWS CloudFormation Sample Template WordPress_Single_Instance: WordPress is web software you can use to create a beautiful website or blog. This template installs a single-instance WordPress deployment using a local MySQL database to store the data.

Parameters

KeyName **(required)**

type

string

description

Name of an existing EC2 KeyPair to enable SSH access to the instance

InstanceType **(optional)**

type

string

description

The EC2 instance type

DBName **(optional)**

type

string

description

The WordPress database name

DBUsernameName **(optional)**

type

string

description

The WordPress database admin account username

DBPassword **(optional)**

type

string

description

The WordPress database admin account password

DBRootPassword **(optional)**

type

string

description

Root password for MySQL

LinuxDistribution **(optional)**

type

string

description

Linux distribution of choice

Glossary

API server

HTTP REST API service for heat.

CFN

An abbreviated form of “AWS CloudFormation”.

Constraint

Defines valid input *template*.

Dependency

When a *resource* must wait for another resource to finish creation before being created itself. Heat adds an implicit dependency when a resource references another resource or one of its *attributes*. An explicit dependency can also be created by the user in the template definition.

Environment

Used to affect the run-time behavior of the template. Provides a way to override the default resource implementation and parameters passed to Heat. See *Environments*.

Heat Orchestration Template

A particular *template* format that is native to Heat. Heat Orchestration Templates are expressed in YAML and are not backwards-compatible with CloudFormation templates.

HOT

An acronym for “Input parameters

See *Parameters*.

Metadata

May refer to *Nova Instance metadata*, or the *Metadata service*.

Metadata service
A Compute service that enables virtual machine instances to retrieve instance-specific data. See Metadata service (OpenStack Administrator Guide).

Multi-region
A feature of Heat that supports deployment to multiple regions.

Nested resource
A *nested stack*.

Nested stack
A *template* referenced by URL inside of another template. Used to reduce redundant resource definitions and group complex architectures into logical groups.

Nova Instance metadata
User-provided *key:value* pairs associated with a Compute Instance. See Instance specific data (OpenStack Operations Guide).

OpenStack
Open source software for building private and public clouds.

Orchestrate
Arrange or direct the elements of a situation to produce a desired effect.

Outputs
A top-level block in a *template* that defines what data will be returned by a stack after instantiation.

Parameters
A top-level block in a *template* that defines what data can be passed to customise a template when it is used to create or update a *stack*.

Provider resource
A *provider template*. The parent resource's properties become the What are "Providers"? (OpenStack Wiki).

Provider template
Allows user-definable *outputs* become the parent stack's *attributes*.

Resource
An element of OpenStack infrastructure instantiated from a particular *Nested resource*.

Resource attribute
Data that can be obtained from a *resource*, e.g. a server's public IP or name. Usually passed to another resource's *outputs*.

Resource group
A *resource provider* that creates one or more identically configured *nested resources*.

Resource Metadata
A *resource property* that contains CFN-style template metadata. See AWS::CloudFormation::Init (AWS CloudFormation User Guide)

Resource plugin
Python code that understands how to instantiate and manage a Heat Resource Plugins (OpenStack wiki).

Resource property
Data utilized for the instantiation of a *resource*. Can be defined statically in a *template* or passed in as *input parameters*.

Resource provider
The implementation of a particular resource type. May be a *Provider template*.

Stack
A collection of instantiated *resources* that are defined in a single *template*.

Stack resource
A *resource provider* that allows the management of a *resource* in a parent stack.

Template
An orchestration document that details everything needed to carry out an *orchestration*.

Template resource
See *Provider resource*.

User data
A *resource property* that contains a user-provided data blob. User data gets passed to cloud-init to automatically configure instances at boot time. See also User data (OpenStack End User Guide).

Wait condition
A *resource provider* that provides a way to communicate data or events from servers back to the orchestration engine. Most commonly used to pause the creation of the *stack* while the server is being configured.

Installing OpenStack and Heat on RHEL/Fedora/CentOS

Go to the OpenStack Documentation for the latest version of the Installation Guide for Red Hat Enterprise Linux, CentOS and Fedora which includes a chapter on installing the Orchestration module (Heat).

There are instructions for installing the RDO OpenStack distribution on Fedora and CentOS.

If installing with packstack, you can install heat by specifying `--os-heat-install=y` in your packstack invocation, or setting `CONFIG_HEAT_INSTALL=y` in your answers file.

If installing with TripleO Heat will be installed by default.

Installing Heat on Ubuntu

Heat is packaged for Debian, and Ubuntu (from 13.10)

Go to the OpenStack Documentation for the latest version of the Installation Guide for Ubuntu which includes a chapter on installing the Orchestration module (Heat).

There is a Juju Charm for Heat <https://jujucharms.com/heat/> available.

Scaling a Deployment

When deploying in an environment where a large number of incoming requests need to be handled, the API and engine services can be overloaded. In those scenarios, in order to increase the system performance, it can be helpful to run multiple load-balanced APIs and engines.

This guide details how to scale out the REST API, the CFN API, and the engine, also known as the *heat-api*, *heat-api-cfn*, and *heat-engine* services, respectively.

Assumptions

This guide, using a devstack installation of OpenStack, assumes that:

> - You have configured devstack from Single Machine Installation Guide;
> - You have set up heat on devstack, as defined at heat and DevStack;
> - You have installed HAProxy on the devstack server.

Architecture

This section shows the basic heat architecture, the load balancing mechanism used and the target scaled out architecture.

Basic Architecture

The heat architecture is as defined at heat architecture and shown in the diagram below, where we have a CLI that sends HTTP requests to the REST and CFN APIs,

which in turn make calls using AMQP to the heat-engine.

```
                  |- [REST API] -|
[CLI] -- <HTTP> --                -- <AMQP> -- [ENGINE]
                  |- [CFN API]  -|
```

Load Balancing

As there is a need to use a load balancer mechanism between the multiple APIs and the CLI, a proxy has to be deployed.

Because the heat CLI and APIs communicate by exchanging HTTP requests and responses, a HAProxy HTTP load balancer server will be deployed between them.

This way, the proxy will take the CLIs requests to the APIs and act on their behalf. Once the proxy receives a response, it will be redirected to the caller CLI.

A round-robin distribution of messages from the AMQP queue will act as the load balancer for multiple engines. Check that your AMQP service is configured to distribute messages round-robin (RabbitMQ does this by default).

Target Architecture

A scaled out heat architecture is represented in the diagram below:

```
                             |- [REST-API] -|
                             |-    ...     -|
                             |- [REST-API] -|             |- [ENGINE] -|
[CLI] -- <HTTP> -- [PROXY] --                -- <AMQP> -- |-    ...   -|
                             |- [API-CFN]  -|             |- [ENGINE] -|
                             |-    ...     -|
                             |- [API-CFN]  -|
```

Thus, a request sent from the CLI looks like:

> - CLI contacts the proxy;
> - The HAProxy server, acting as a load balancer, redirects the call to an API instance;
> - The API server sends messages to the AMQP queue, and the engines pick up messages in round-robin fashion.

Deploying Multiple APIs

In order to run a heat component separately, you have to execute one of the python scripts located at the *bin* directory of your heat repository.

These scripts take as argument a configuration file. When using devstack, the configuration file is located at */etc/heat/heat.conf*. For instance, to start new REST and CFN API services, you must run:

```
python bin/heat-api --config-file=/etc/heat/heat.conf
python bin/heat-api-cfn --config-file=/etc/heat/heat.conf
```

Each API service must have a unique address to listen. This address have to be defined in the configuration file. For REST and CFN APIs, modify the *[heat_api]* and *[heat_api_cfn]* blocks, respectively.

```
[heat_api]
bind_port = {API_PORT}
bind_host = {API_HOST}

...

[heat_api_cfn]
bind_port = {API_CFN_PORT}
bind_host = {API_CFN_HOST}
```

If you wish to run multiple API processes on the same machine, you must create multiple copies of the heat.conf file, each containing a unique port number.

In addition, if you want to run some API services in different machines than the devstack server, you have to update the loopback addresses found at the *sql_connection* and *rabbit_host* properties to the devstack server's IP, which must be reachable from the remote machine.

Deploying Multiple Engines

All engines must be configured to use the same AMQP service. Ensure that all of the *rabbit_** and *kombu_** configuration options in the *[DEFAULT]* section of */etc/heat/heat.conf* match across each machine that will be running an engine. By using the same AMQP configuration, each engine will subscribe to the same AMQP *engine* queue and pick up work in round-robin fashion with the other engines.

One or more engines can be deployed per host. Depending on the host's CPU

architecture, it may be beneficial to deploy several engines on a single machine.

To start multiple engines on the same machine, simply start multiple *heat-engine* processes:

```
python bin/heat-engine --config-file=/etc/heat/heat.conf &
python bin/heat-engine --config-file=/etc/heat/heat.conf &
```

Deploying the Proxy

In order to simplify the deployment of the HAProxy server, we will replace the REST and CFN APIs deployed when installing devstack by the HAProxy server. This way, there is no need to update, on the CLI, the addresses where it should look for the APIs. In this case, when it makes a call to any API, it will find the proxy, acting on their behalf.

Note that the addresses that the HAProxy will be listening to are the pairs *API_HOST:API-PORT* and *API_CFN_HOST:API_CFN_PORT*, found at the *[heat_api]* and *[heat_api_cfn]* blocks on the devstack server's configuration file. In addition, the original *heat-api* and *heat-api-cfn* processes running in these ports have to be killed, because these addresses must be free to be used by the proxy.

To deploy the HAProxy server on the devstack server, run *haproxy -f apis-proxy.conf*, where this configuration file looks like:

```
global
    daemon
    maxconn 4000

defaults
    log  global
    maxconn  8000
    option  redispatch
    retries  3
    timeout  http-request 10s
    timeout  queue 1m
    timeout  connect 10s
    timeout  client 1m
    timeout  server 1m
    timeout  check 10s

listen rest_api_proxy
    # The values required below are the original ones that were in
    # /etc/heat/heat.conf on the devstack server.
```

```
    bind {API_HOST}:{API_PORT}
    balance  source
    option  tcpka
    option  httpchk
    # The values required below are the different addresses supplied when
    # running the REST API instances.
    server SERVER_1 {HOST_1}:{PORT_1}
    ...
    server SERVER_N {HOST_N}:{PORT_N}

listen cfn_api_proxy
    # The values required below are the original ones that were in
    # /etc/heat/heat.conf on the devstack server.
    bind {API_CFN_HOST}:{API_CFN_PORT}
    balance  source
    option  tcpka
    option  httpchk
    # The values required below are the different addresses supplied when
    # running the CFN API instances.
    server SERVER_1 {HOST_1}:{PORT_1}
    ...
    server SERVER_N {HOST_N}:{PORT_N}
```

Sample

This section aims to clarify some aspects of the scaling out solution, as well as to show more details of the configuration by describing a concrete sample.

Architecture

This section shows a basic OpenStack architecture and the target one that will be used for testing of the scaled-out heat services.

Basic Architecture

For this sample, consider that:

> - We have an architecture composed by 3 machines configured in a LAN, with the addresses A: 10.0.0.1; B: 10.0.0.2; and C: 10.0.0.3;
> - The OpenStack devstack installation, including the heat module, has been done in the machine A, as shown in the

Assumptions section.

Target Architecture

At this moment, everything is running in a single devstack server. The next subsections show how to deploy a scaling out heat architecture by:

- Running one REST and one CFN API on the machines B and C;
- Setting up the HAProxy server on the machine A.

Running the API and Engine Services

For each machine, B and C, you must do the following steps:

- Clone the heat repository https://git.openstack.org/cgit/openstack/heat, run:

::

```
git clone https://git.openstack.org/openstack/heat
```

- Create a local copy of the configuration file */etc/heat/heat.conf* from the machine A;
- Make required changes on the configuration file;
- Enter the heat local repository and run:

```
python bin/heat-api --config-file=/etc/heat/heat.conf
python bin/heat-api-cfn --config-file=/etc/heat/heat.conf
```

- Start as many *heat-engine* processes as you want running on that machine:

```
python bin/heat-engine --config-file=/etc/heat/heat.conf &
python bin/heat-engine --config-file=/etc/heat/heat.conf &
...
```

Changes On Configuration

The original file from A looks like:

```
[DEFAULT]
...
sql_connection = mysql+pymysql://root:admin@127.0.0.1/heat?charset=utf8
rabbit_host = localhost
```

```
...
[heat_api]
bind_port = 8004
bind_host = 10.0.0.1
...
[heat_api_cfn]
bind_port = 8000
bind_host = 10.0.0.1
```

After the changes for B, it looks like:

```
[DEFAULT]
...
sql_connection = mysql+pymysql://root:admin@10.0.0.1/heat?charset=utf8
rabbit_host = 10.0.0.1
...
[heat_api]
bind_port = 8004
bind_host = 10.0.0.2
...
[heat_api_cfn]
bind_port = 8000
bind_host = 10.0.0.2
```

Setting Up HAProxy

On the machine A, kill the *heat-api* and *heat-api-cfn* processes by running *pkill heat-api* and *pkill heat-api-cfn*. After, run *haproxy -f apis-proxy.conf* with the following configuration:

```
 global
    daemon
    maxconn 4000

defaults
    log  global
    maxconn  8000
    option  redispatch
    retries  3
    timeout  http-request 10s
    timeout  queue 1m
    timeout  connect 10s
    timeout  client 1m
    timeout  server 1m
```

```
        timeout  check 10s

    listen rest_api_proxy
        bind 10.0.0.1:8004
        balance  source
        option  tcpka
        option  httpchk
        server rest-server-1 10.0.0.2:8004
        server rest-server-2 10.0.0.3:8004

    listen cfn_api_proxy
        bind 10.0.0.1:8000
        balance  source
        option  tcpka
        option  httpchk
        server cfn-server-1 10.0.0.2:8000
        server cfn-server-2 10.0.0.3:8000
```

Man pages for services and utilities

Heat services

- heat-engine
  - SYNOPSIS
  - DESCRIPTION
  - INVENTORY
  - OPTIONS
  - FILES
- heat-api
  - SYNOPSIS
  - DESCRIPTION
  - INVENTORY
  - OPTIONS
  - FILES
- heat-api-cfn
  - SYNOPSIS
  - DESCRIPTION
  - INVENTORY
  - OPTIONS
  - FILES
- heat-api-cloudwatch
  - SYNOPSIS
  - DESCRIPTION
  - OPTIONS
  - FILES

Heat utilities

- heat-manage
  - SYNOPSIS
  - DESCRIPTION

        - OPTIONS
        - FILES
        - BUGS
    - heat-db-setup
        - SYNOPSIS
        - DESCRIPTION
        - COMMANDS
        - OPTIONS
        - EXAMPLES
        - BUGS
    - heat-keystone-setup
        - SYNOPSIS
        - DESCRIPTION
        - EXAMPLES
        - BUGS
    - heat-keystone-setup-domain
        - SYNOPSIS
        - DESCRIPTION
        - OPTIONS
        - EXAMPLES
        - BUGS

heat-engine

SYNOPSIS

```
heat-engine [options]
```

DESCRIPTION

heat-engine is the heat project server with an internal RPC api called by the heat-api server.

INVENTORY

The heat-engine does all the orchestration work and is the layer in which the resource integration is implemented.

OPTIONS

--config-file

Path to a config file to use. Multiple config files can be specified, with values in later files taking precedence.

--config-dir

Path to a config directory to pull .conf files from. This file set is sorted, so as to provide a predictable parse order if individual options are over-ridden. The set is parsed after the file(s), if any, specified via –config-file, hence over-ridden options in the directory take precedence.

--version

Show program’s version number and exit. The output could be empty if the distribution didn’t specify any version information.

FILES

- /etc/heat/heat.conf

heat-api

SYNOPSIS

```
heat-api [options]
```

DESCRIPTION

heat-api provides an external REST API to the heat project.

INVENTORY

heat-api is a service that exposes an external REST based api to the heat-engine service. The communication between the heat-api and heat-engine uses message queue based RPC.

OPTIONS

--config-file

Path to a config file to use. Multiple config files can be specified, with values in later files taking precedence.

--config-dir

Path to a config directory to pull .conf files from. This file set is sorted, so as to provide a predictable parse order if individual options are over-ridden. The set is parsed after the file(s), if any, specified via –config-file, hence over-ridden options in the directory take precedence.

--version

Show program’s version number and exit. The output could be empty if the distribution didn’t specify any version information.

FILES

- /etc/heat/heat.conf

heat-api-cfn

SYNOPSIS

```
heat-api-cfn [options]
```

DESCRIPTION

heat-api-cfn is a CloudFormation compatible API service to the heat project.

INVENTORY

heat-api-cfn is a service that exposes an external REST based api to the heat-engine service. The communication between the heat-api-cfn and heat-engine uses message queue based RPC.

OPTIONS

--config-file

Path to a config file to use. Multiple config files can be specified, with values in later files taking precedence.

--config-dir

Path to a config directory to pull .conf files from. This file set is sorted, so as to provide a predictable parse order if individual options are over-ridden. The set is parsed after the file(s), if any, specified via –config-file, hence over-ridden options in the directory take precedence.

--version

Show program’s version number and exit. The output could be empty if the distribution didn’t specify any version information.

FILES

- /etc/heat/heat.conf

heat-api-cloudwatch

SYNOPSIS

```
heat-api-cloudwatch [options]
```

DESCRIPTION

heat-api-cloudwatch is a CloudWatch-like API service to the heat project.

OPTIONS

--config-file

Path to a config file to use. Multiple config files can be specified, with values in later files taking precedence.

--config-dir

Path to a config directory to pull .conf files from. This file set is sorted, so as to provide a predictable parse order if individual options are over-ridden. The set is parsed after the file(s), if any, specified via –config-file, hence over-ridden options in the directory take precedence.

--version

Show program's version number and exit. The output could be empty if the distribution didn't specify any version information.

FILES

- /etc/heat/heat.conf

heat-manage

SYNOPSIS

```
heat-manage <action> [options]
```

DESCRIPTION

heat-manage helps manage heat specific database operations.

OPTIONS

The standard pattern for executing a heat-manage command is: `heat-manage <command> [<args>]`

Run with -h to see a list of available commands: `heat-manage -h`

Commands are `db_version`, `db_sync`, `purge_deleted` and `service`. Detailed descriptions are below.

Heat Db version

```
heat-manage db_version
```

> Print out the db schema version.

`heat-manage db_sync`

> Sync the database up to the most recent version.

`heat-manage purge_deleted [-g {days,hours,minutes,seconds}] [-p project_id] [age]`

> Purge db entries marked as deleted and older than [age]. When project_id argument is provided, only entries belonging to this project will be purged.

`heat-manage service list`

> Shows details for all currently running heat-engines.

`heat-manage service clean`

> Clean dead engine records.

`heat-manage --version`

> Shows program's version number and exit. The output could be empty if the distribution didn't specify any version information.

FILES

The /etc/heat/heat.conf file contains global options which can be used to configure some aspects of heat-manage, for example the DB connection and logging.

BUGS

- Heat issues are tracked in Launchpad so you can view or report bugs here OpenStack Heat Bugs

heat-db-setup

SYNOPSIS

`heat-db-setup [COMMANDS] [OPTIONS]`

DESCRIPTION

heat-db-setup is a tool which configures the local MySQL database for heat. Typically distro-specific tools would provide this functionality so please read the

distro-specific documentation for configuring heat.

COMMANDS

`rpm`

Indicate the distribution is a RPM packaging based distribution.

`deb`

Indicate the distribution is a DEB packaging based distribution.

OPTIONS

`-h, --help`

Print usage information.

`-p, --password`

Specify the password for the ‘heat’ MySQL user that the script will use to connect to the ‘heat’ MySQL database. By default, the password ‘heat’ will be used.

`-r, --rootpw`

Specify the root MySQL password. If the script installs the MySQL server, it will set the root password to this value instead of prompting for a password. If the MySQL server is already installed, this password will be used to connect to the database instead of having to prompt for it.

`-y, --yes`

In cases where the script would normally ask for confirmation before doing something, such as installing mysql-server, just assume yes. This is useful if you want to run the script non-interactively.

EXAMPLES

```
heat-db-setup rpm -p heat_password -r mysql_pwd -y

heat-db-setup deb -p heat_password -r mysql_pwd -y

heat-db-setup rpm
```

BUGS

Heat bugs are managed through Launchpad OpenStack Heat Bugs

heat-keystone-setup

SYNOPSIS

```
heat-keystone-setup
```

DESCRIPTION

Warning: This script is deprecated, please use other tool to setup keystone for heat.

The `heat-keystone-setup` tool configures keystone for use with heat. This script requires admin keystone credentials to be available in the shell environment and write access to `/etc/keystone`.

Distributions may provide other tools to setup keystone for use with Heat, so check the distro documentation first.

EXAMPLES

```
heat-keystone-setup
```

BUGS

Heat bugs are managed through Launchpad OpenStack Heat Bugs

heat-keystone-setup-domain

SYNOPSIS

```
heat-keystone-setup-domain [OPTIONS]
```

DESCRIPTION

The heat-keystone-setup-domain tool configures keystone by creating a ‘stack user domain’ and the user credential used to manage this domain. A ‘stack user domain’

can be treated as a namespace for projects, groups and users created by heat. The domain will have an admin user that manages other users, groups and projects in the domain.

This script requires admin keystone credentials to be available in the shell environment by setting OS_USERNAME and OS_PASSWORD.

After running this script, a user needs to take actions to check or modify the heat configuration file (e.g. /etc/heat/heat.conf). The tool is NOT performing these updates on behalf of the user.

Distributions may provide other tools to setup 'stack user domain' for use with heat, so check the distro documentation first. Other tools are available to set up the 'stack user domain', for example python-openstackclient, which is preferred to this tool where it is available.

OPTIONS

-h, --help

Print usage information.

--config-dir <DIR>

Path to a config directory from which to read the `heat.conf` file(s). This file set is sorted, so as to provide a predictable parse order if individual options are over-ridden. The set is parsed after the file(s) specified via previous –config-file, arguments hence over-ridden options in the directory take precedence.

--config-file <PATH>

Path to a config file to use. Multiple config files can be specified, with values in later files taking precedence. The default files used is /etc/heat/heat.conf.

--stack-domain-admin <USERNAME>

Name of a user for Keystone to create, which has roles sufficient to manage users (i.e. stack domain users) and projects (i.e. stack domain projects) in the 'stack user domain'.

Another way to specify the admin user name is by setting an environment variable STACK_DOMAIN_ADMIN before running this tool. If both command line arguments and environment variable are specified, the command line arguments take precedence.

--stack-domain-admin-password <PASSWORD>

Password for the ‘stack-domain-admin’ user.

The password can be instead specified using an environment variable STACK_DOMAIN_ADMIN_PASSWORD before invoking this tool. If both command line arguments and environment variable are specified, the command line arguments take precedence.

--stack-user-domain-name <DOMAIN>

Name of domain to create for stack users.

The domain name can be instead specified using an environment variable STACK_USER_DOMAIN_NAME before invoking this tool. If both command line arguments and environment variable are specified, the command line argument take precedence.

--version

Show program’s version number and exit. The output could be empty if the distribution didn’t specify any version information.

EXAMPLES

> heat-keystone-setup-domain
>
> **heat-keystone-setup-domain –stack-user-domain-name heat_user_domain**
>
> > –stack-domain-admin heat_domain_admin –stack-domain-admin-password verysecrete

BUGS

Heat bugs are managed through Launchpad OpenStack Heat Bugs

Heat Contribution Guidelines

In the Contributions Guide, you will find documented policies for developing with heat. This includes the processes we use for blueprints and specs, bugs, contributor onboarding, core reviewer memberships, and other procedural items.

Policies

- Blueprints and Specs
  - Spec Notes
  - Heat Spec Lite
  - Lite spec Submission Guidelines
  - Lite spec from existing bugs

Blueprints and Specs

The Heat team uses the heat-specs repository for its specification reviews. Detailed information can be found here.

Please note that we use a template for spec submissions. It is not required to fill out all sections in the template.

Spec Notes

There are occasions when a spec is approved and the code does not land in the cycle it was targeted for. For these cases, the workflow to get the spec into the next release is as below:

- Anyone can propose a patch to heat-specs which moves a spec from the previous release backlog into the new release directory.

The specs which are moved in this way can be fast-tracked into the next release. Please note that it is required to re-propose the spec for the new release and it'll be evaluated based on the resources available and cycle priorities.

Heat Spec Lite

Lite specs are small feature requests tracked as Launchpad bugs, with status 'Wishlist' and tagged with 'spec-lite' tag. These allow for submission and review of

these feature requests before code is submitted.

These can be used for small features that don't warrant a detailed spec to be proposed, evaluated, and worked on. The team evaluates these requests as it evaluates specs.

Once a spec-lite bug has been approved/triaged as a Request for Enhancement(RFE), it'll be targeted for a release.

The workflow for the life of a spec-lite in Launchpad is as follows:

- File a bug with a small summary of what the requested change is and tag it as spec-lite.
- The bug is triaged and importance changed to Wishlist.
- The bug is evaluated and marked as Triaged to announce approval or to Won't fix to announce rejection or Invalid to request a full spec.
- The bug is moved to In Progress once the code is up and ready to review.
- The bug is moved to Fix Committed once the patch lands.

In summary:

State	Meaning
New	This is where spec-lite starts, as filed by the community.
Triaged	Drivers - Move to this state to mean, "you can start working on it"
Won't Fix	Drivers - Move to this state to reject a lite-spec.
Invalid	Drivers - Move to this state to request a full spec for this request

The drivers team will discuss the following bug reports in IRC meetings:

- heat RFE's
- python-heatclient RFE's

Lite spec Submission Guidelines

When a bug is submitted, there are two fields that must be filled: 'summary' and 'further information'. The 'summary' must be brief enough to fit in one line.

The 'further information' section must be a description of what you would like to see implemented in heat. The description should provide enough details for a knowledgeable developer to understand what is the existing problem and what's the proposed solution.

Add spec-lite tag to the bug.

Lite spec from existing bugs

If there's an already existing bug that describes a small feature suitable for a spec-lite, add a `spec-lite' tag to the bug. There is no need to create a new bug. The comments and history of the existing bug are important for it's review.

Heat and DevStack

Heat is fully integrated into DevStack. This is a convenient way to try out or develop heat alongside the current development state of all the other OpenStack projects. Heat on DevStack works on both Ubuntu and Fedora.

These instructions assume you already have a working DevStack installation which can launch basic instances.

Configure DevStack to enable heat

Heat is configured by default on devstack for Icehouse and Juno releases. Newer versions of OpenStack require enabling heat services in devstack local.conf.

Add the following to [[local|localrc]] section of local.conf:

```
[[local|localrc]]

#Enable heat services
enable_service h-eng h-api h-api-cfn h-api-cw
```

It would also be useful to automatically download and register a VM image that heat can launch. To do that add the following to [[local|localrc]] section of local.conf:

```
IMAGE_URL_SITE="http://download.fedoraproject.org"
IMAGE_URL_PATH="/pub/fedora/linux/releases/21/Cloud/Images/x86_64/"
IMAGE_URL_FILE="Fedora-Cloud-Base-20141203-21.x86_64.qcow2"
IMAGE_URLS+=","$IMAGE_URL_SITE$IMAGE_URL_PATH$IMAGE_URL_FILE
```

URLs for any cloud image may be specified, but fedora images from F20 contain the heat-cfntools package which is required for some heat functionality.

That is all the configuration that is required. When you run ./stack.sh the heat processes will be launched in screen with the labels prefixed with h-.

Configure DevStack to enable ceilometer and aodh (if using alarms)

To use aodh alarms you need to enable ceilometer and aodh in devstack. Adding the following lines to [[local|localrc]] section of local.conf will enable the services:

```
CEILOMETER_BACKEND=mongodb
enable_plugin ceilometer https://git.openstack.org/openstack/ceilometer
enable_plugin aodh https://git.openstack.org/openstack/aodh
```

Configure DevStack to enable OSprofiler

Adding the following line to [[local|localrc]] section of local.conf will add the profiler notifier to your ceilometer:

```
CEILOMETER_NOTIFICATION_TOPICS=notifications,profiler
```

Enable the profiler in /etc/heat/heat.conf:

```
$ echo -e "[profiler]\nprofiler_enabled = True\n"\
      "trace_sqlalchemy = True\n"\
      >> /etc/heat/heat.conf
```

Change the default hmac_key in /etc/heat/api-paste.ini:

```
$ sed -i "s/hmac_keys =.*/hmac_keys = SECRET_KEY/" /etc/heat/api-paste.ini
```

Run any command with –profile SECRET_KEY:

```
$ heat --profile SECRET_KEY stack-list
# it will print <Trace ID>
```

Get pretty HTML with traces:

```
$ osprofiler trace show --html <Profile ID>
```

Note that osprofiler should be run with the admin user name & tenant.

Create a stack

Now that you have a working heat environment you can go to *Creating your first stack*.

Heat architecture

Heat is a service to orchestrate multiple composite cloud applications using the AWS CloudFormation template format, through both an OpenStack-native REST API and a CloudFormation-compatible Query API.

Detailed description

What is the purpose of the project and vision for it?

Heat provides an AWS CloudFormation implementation for OpenStack that orchestrates an AWS CloudFormation template describing a cloud application by executing appropriate OpenStack API calls to generate running cloud applications.

Describe the relevance of the project to other OpenStack projects and the OpenStack mission to provide a ubiquitous cloud computing platform:

The software integrates other core components of OpenStack into a one-file template system. The templates allow creation of most OpenStack resource types (such as instances, floating IPs, volumes, security groups and users), as well as some more advanced functionality such as instance high availability, instance autoscaling, and nested stacks. By providing very tight integration with other OpenStack core projects, all OpenStack core projects could receive a larger user base.

Currently no other CloudFormation implementation exists for OpenStack. The developers believe cloud developers have a strong desire to move workloads from AWS to OpenStack deployments. Given the missing gap of a well-implemented and integrated CloudFormation API in OpenStack, we provide a high quality implementation of this gap improving the ubiquity of OpenStack.

Heat services

The developers are focused on creating an OpenStack style project using OpenStack design tenets, implemented in Python. We have started with full integration with keystone. We have a number of components.

As the developers have only started development in March 2012, the architecture is evolving rapidly.

heat

The heat tool is a CLI which communicates with the heat-api to execute AWS CloudFormation APIs. End developers could also use the heat REST API directly.

heat-api

The heat-api component provides an OpenStack-native REST API that processes API requests by sending them to the heat-engine over RPC.

heat-api-cfn

The heat-api-cfn component provides an AWS Query API that is compatible with AWS CloudFormation and processes API requests by sending them to the heat-engine over RPC.

heat-engine

The heat-engine's main responsibility is to orchestrate the launching of templates and provide events back to the API consumer.

The templates integrate well with Puppet and Chef.

Heat Resource Plug-in Development Guide

Heat allows service providers to extend the capabilities of the orchestration service by writing their own resource plug-ins. These plug-ins are written in Python and included in a directory configured by the service provider. This guide describes a resource plug-in structure and life cycle in order to assist developers in writing their own resource plug-ins.

Resource Plug-in Life Cycle

A resource plug-in is relatively simple in that it needs to extend a base `Resource` class and implement some relevant life cycle handler methods. The basic life cycle methods of a resource are:

create
: The plug-in should create a new physical resource.

update
: The plug-in should update an existing resource with new configuration or tell the engine that the resource must be destroyed and re-created. This method is optional; the default behavior is to create a replacement resource and then delete the old resource.

suspend
: The plug-in should suspend operation of the physical resource; this is an optional operation.

resume
: The plug-in should resume operation of the physical resource; this is an optional operation.

delete
: The plug-in should delete the physical resource.

The base class `Resource` implements each of these life cycle methods and defines one or more handler methods that plug-ins can implement in order to manifest and manage the actual physical resource abstracted by the plug-in. These handler methods will be described in detail in the following sections.

Heat Resource Base Class

Plug-ins must extend the class `heat.engine.resource.Resource`.

This class is responsible for managing the overall life cycle of the plug-in. It defines methods corresponding to the life cycle as well as the basic hooks for plug-ins to handle the work of communicating with specific down-stream services. For example, when the engine determines it is time to create a resource, it calls the `create` method of the applicable plug-in. This method is implemented in the `Resource` base class and handles most of the bookkeeping and interaction with the engine. This method then calls a `handle_create` method defined in the plug-in class (if implemented) which is responsible for using specific service calls or other methods needed to instantiate the desired physical resource (server, network, volume, etc).

Resource Status and Action

The base class handles reporting state of the resource back to the engine. A resource's state is the combination of the life cycle action and the status of that action. For example, if a resource is created successfully, the status of that resource will be `CREATE COMPLETE`. Alternatively, if the plug-in encounters an error when attempting to create the physical resource, the status would be `CREATE FAILED`. The base class handles the reporting and persisting of resource state, so a plug-in's handler methods only need to return data or raise exceptions as appropriate.

Resource Support Status

New resource should be marked from which OpenStack release it will be available with *support_status* option. For more details, see *Heat Support Status usage Guide*.

Properties and Attributes

A resource's *properties* define the settings the template author can manipulate when including that resource in a template. Some examples would be:

- Which flavor and image to use for a Nova server
- The port to listen to on Neutron LBaaS nodes
- The size of a Cinder volume

Attributes describe runtime state data of the physical resource that the plug-in can expose to other resources in a Stack. Generally, these aren't available until the physical resource has been created and is in a usable state. Some examples would be:

- The host id of a Nova server
- The status of a Neutron network

- The creation time of a Cinder volume

Defining Resource Properties

Each property that a resource supports must be defined in a schema that informs the engine and validation logic what the properties are, what type each is, and validation constraints. The schema is a dictionary whose keys define property names and whose values describe the constraints on that property. This dictionary must be assigned to the `properties_schema` attribute of the plug-in.

```
from heat.common.i18n import _
from heat.engine import constraints
from heat.engine import properties

    nested_schema = {
        "foo": properties.Schema(
            properties.Schema.STRING,
            _('description of foo field'),
            constraints=[
                constraints.AllowedPattern('(Ba[rc]?)+'),
                constraints.Length(max=10,
                                   description="don't go crazy")
            ]
        )
    }
    properties_schema = {
        "property_name": properties.Schema(
            properties.Schema.MAP,
            _('Internationalized description of property'),
            required=True,
            default={"Foo": "Bar"},
            schema=nested_schema
        )
    }
```

As shown above, some properties may themselves be complex and reference nested schema definitions. Following are the parameters to the `Schema` constructor; all but the first have defaults.

data_type:

> Defines the type of the property's value. The valid types are the members of the list `properties.Schema.TYPES`, currently `INTEGER`, `STRING`, `NUMBER`, `BOOLEAN`, `MAP`, and `LIST`; please use those symbolic names rather than the literals to which they are

> equated. For `LIST` and `MAP` type properties, the `schema` referenced constrains the format of complex items in the list or map.

***description*:**
A description of the property and its function; also used in documentation generation. Default is `None` — but you should always provide a description.

***default*:**
The default value to assign to this property if none was supplied in the template. Default is `None`.

***schema*:**
This property's value is complex and its members must conform to this referenced schema in order to be valid. The referenced schema dictionary has the same format as the `properties_schema`. Default is `None`.

***required*:**
`True` if the property must have a value for the template to be valid; `False` otherwise. The default is `False`

***constraints*:**
A list of constraints that apply to the property's value. See Property Constraints.

***update_allowed*:**
`True` if an existing resource can be updated, `False` means update is accomplished by delete and re-create. Default is `False`.

***immutable*:**
`True` means updates are not supported, resource update will fail on every change of this property. `False` otherwise. Default is `False`.

Accessing property values of the plug-in at runtime is then a simple call to:

```
self.properties['PropertyName']
```

Based on the property type, properties without a set value will return the default "empty" value for that type:

Type	Empty Value
String	''
Number	0
Integer	0
List	[]
Map	{}
Boolean	False

Property Constraints

Following are the available kinds of constraints. The description is optional and, if given, states the constraint in plain language for the end user.

***AllowedPattern(regex, description)*:**
Constrains the value to match the given regular expression; applicable to STRING.

***AllowedValues(allowed, description)*:**
Lists the allowed values. `allowed` must be a `collections.Sequence` or `basestring`. Applicable to all types of value except MAP.

***Length(min, max, description)*:**
Constrains the length of the value. Applicable to STRING, LIST, MAP. Both `min` and `max` default to `None`.

***Range(min, max, description)*:**
Constrains a numerical value. Applicable to INTEGER and NUMBER. Both `min` and `max` default to `None`.

***CustomConstraint(name, description, environment)*:**
This constructor brings in a named constraint class from an environment. If the given environment is `None` (its default) then the environment used is the global one.

Defining Resource Attributes

Attributes communicate runtime state of the physical resource. Note that some plug-ins do not define any attributes and doing so is optional. If the plug-in needs to expose attributes, it will define an `attributes_schema` similar to the properties schema described above. This schema, however, is much simpler to define as each item in the dictionary only defines the attribute name and a description of the attribute.

```
attributes_schema = {
    "foo": _("The foo attribute"),
    "bar": _("The bar attribute"),
    "baz": _("The baz attribute")
}
```

If attributes are defined, their values must also be resolved by the plug-in. The simplest way to do this is to override the `_resolve_attribute` method from the `Resource` class:

```
def _resolve_attribute(self, name):
    # _example_get_physical_resource is just an example and is not defined
    # in the Resource class
    phys_resource = self._example_get_physical_resource()
```

```
    if phys_resource:
        if not hasattr(phys_resource, name):
                # this is usually not needed, but this is a simple example
                raise exception.InvalidTemplateAttribute(name)
        return getattr(phys_resource, name)
    return None
```

If the plug-in needs to be more sophisticated in its attribute resolution, the plug-in may instead choose to override `FnGetAtt`. However, if this method is chosen, validation and accessibility of the attribute would be the plug-in's responsibility.

Property and Attribute Example

Assume the following simple property and attribute definition:

```
properties_schema = {
    'foo': properties.Schema(
        properties.Schema.STRING,
        _('foo prop description'),
        default='foo',
        required=True
    ),
    'bar': properties.Schema(
        properties.Schema.INTEGER,
        _('bar prop description'),
        required=True,
        constraints=[
            constraints.Range(5, 10)
        ]
    )
}

attributes_schema = {
    'Attr_1': 'The first attribute',
    'Attr_2': 'The second attribute'
}
```

Also assume the plug-in defining the above has been registered under the template reference name 'Resource::Foo' (see Registering Resource Plug-ins). A template author could then use this plug-in in a stack by simply making following declarations in a template:

```
# ... other sections omitted for brevity ...
```

```
resources:
  resource-1:
    type: Resource::Foo
    properties:
      foo: Value of the foo property
      bar: 7

outputs:
  foo-attrib-1:
    value: { get_attr: [resource-1, Attr_1] }
    description: The first attribute of the foo resource
  foo-attrib-2:
    value: { get_attr: [resource-1, Attr_2] }
    description: The second attribute of the foo resource
```

Life Cycle Handler Methods

To do the work of managing the physical resource the plug-in supports, the following life cycle handler methods should be implemented. Note that the plug-in need not implement *all* of these methods; optional handlers will be documented as such.

Generally, the handler methods follow a basic pattern. The basic handler method for any life cycle step follows the format `handle_<life cycle step>`. So for the create step, the handler method would be `handle_create`. Once a handler is called, an optional `check_<life cycle step>_complete` may also be implemented so that the plug-in may return immediately from the basic handler and then take advantage of cooperative multi-threading built in to the base class and periodically poll a down-stream service for completion; the check method is polled until it returns `True`. Again, for the create step, this method would be `check_create_complete`.

Create

`handle_create`(*self*)

Create a new physical resource. This function should make the required calls to create the physical resource and return as soon as there is enough information to identify the resource. The function should return this identifying information and implement `check_create_complete` which will take this information in as a parameter and then periodically be polled. This allows for cooperative multi-threading between multiple resources that have had their dependencies satisfied.

Note once the native identifier of the physical resource is known, this function

should call `self.resource_id_set` passing the native identifier of the physical resource. This will persist the identifier and make it available to the plug-in by accessing `self.resource_id`.

Returns: A representation of the created physical resource

Raise: any `Exception` if the create failed

`check_create_complete`(***self*, *token***)**

If defined, will be called with the return value of `handle_create`

Parameters: **token** – the return value of `handle_create`; used to poll the physical resource's status.

Returns: `True` if the physical resource is active and ready for use; `False` otherwise.

Raise: any `Exception` if the create failed.

Update (Optional)

Note that there is a default implementation of `handle_update` in `heat.engine.resource.Resource` that simply raises an exception indicating that updates require the engine to delete and re-create the resource (this is the default behavior) so implementing this is optional.

`handle_update`(***self*, *json_snippet*, *templ_diff*, *prop_diff***)**

Update the physical resources using updated information.

Parameters:

- **json_snippet** (*collections.Mapping*) – the resource definition from the updated template
- **templ_diff** (*collections.Mapping*) – changed values from the original template definition
- **prop_diff** (*collections.Mapping*) – property values that are different between the original definition and the updated definition; keys are property names and values are the new values. Deleted or properties that were originally present but now absent have values of `None`

check_update_complete(*self*, *token*)

If defined, will be called with the return value of `handle_update`

Parameters: **token** – the return value of `handle_update`; used to poll the physical resource's status.

Returns: `True` if the update has finished; `False` otherwise.

Raise: any `Exception` if the update failed.

Suspend (Optional)

These handler functions are optional and only need to be implemented if the physical resource supports suspending

handle_suspend(*self*)

If the physical resource supports it, this function should call the native API and suspend the resource's operation. This function should return information sufficient for `check_suspend_complete` to poll the native API to verify the operation's status.

Returns: a token containing enough information for `check_suspend_complete` to verify operation status.

Raise: any `Exception` if the suspend operation fails.

check_suspend_complete(*self*, *token*)

Verify the suspend operation completed successfully.

Parameters: **token** – the return value of `handle_suspend`

Returns: `True` if the suspend operation completed and the physical resource is now suspended; `False` otherwise.

Raise: any `Exception` if the suspend operation failed.

Resume (Optional)

These handler functions are optional and only need to be implemented if the physical resource supports resuming from a suspended state

handle_resume(*self*)

If the physical resource supports it, this function should call the native API and resume a suspended resource's operation. This function should return information sufficient for `check_resume_complete` to poll the native API to verify the operation's status.

Returns: a token containing enough information for `check_resume_complete` to verify operation status.

Raise: any `Exception` if the resume operation fails.

check_resume_complete(*self*, *token*)

Verify the resume operation completed successfully.

Parameters: **token** – the return value of `handle_resume`

Returns: `True` if the resume operation completed and the physical resource is now active; `False` otherwise.

Raise: any Exception if the resume operation failed.

Delete

handle_delete(*self*)

Delete the physical resource.

Returns: a token containing sufficient data to verify the operations status

Raise: any `Exception` if the delete operation failed

> **Note**
>
> As of the Liberty release, implementing handle_delete is optional. The parent resource class can handle the most common pattern for deleting resources:
>
> ```
> def handle_delete(self): if self.resource_id is not None: try: self.client().<entity>.delete(self.resource_id) except Exception as ex: self.client_plugin().ignore_not_found(ex) return None return self.resource_id
> ```

For this to work for a particular resource, the entity and default_client_name attributes must be overridden in the resource implementation. For example, entity of Aodh Alarm should equals to "alarm" and default_client_name to "aodh".

`handle_delete_snapshot`(*self*, *snapshot*)

Delete resource snapshot.

Parameters: **snapshot** – dictionary describing current snapshot.

Returns: a token containing sufficient data to verify the operations status

Raise: any `Exception` if the delete operation failed

`handle_snapshot_delete`(*self*, *state*)

Called instead of `handle_delete` when the deletion policy is SNAPSHOT. Create backup of resource and then delete resource.

Parameters: **state** – the (action, status) tuple of the resource to make sure that backup may be created for the current resource

Returns: a token containing sufficient data to verify the operations status

Raise: any `Exception` if the delete operation failed

check_delete_complete(*self*, *token*)

Verify the delete operation completed successfully.

Parameters: **token** – the return value of `handle_delete` or `handle_snapshot_delete` (for deletion policy - Snapshot) used to verify the status of the operation

Returns: `True` if the delete operation completed and the physical resource is deleted; `False` otherwise.

Raise: any `Exception` if the delete operation failed.

check_delete_snapshot_complete(*self*, *token*)

Verify the delete snapshot operation completed successfully.

Parameters: **token** – the return value of `handle_delete_snapshot` used to verify the status of the operation

Returns: `True` if the delete operation completed and the snapshot is deleted; `False` otherwise.

Raise: any `Exception` if the delete operation failed.

Resource Dependencies

Ideally, your resource should not have any 'hidden' dependencies, i.e. Heat should be able to infer any inbound or outbound dependencies of your resource instances from resource properties and the other resources/resource attributes they reference. This is handled by `heat.engine.resource.Resource.add_dependencies()`.

If this is not possible, please do not simply override add_dependencies() in your resource plugin! This has previously caused problems for multiple operations, usually due to uncaught exceptions, If you feel you need to override add_dependencies(), please reach out to Heat developers on the #heat IRC channel on FreeNode or on the openstack-dev mailing list to discuss the possibility of a better solution.

Registering Resource Plug-ins

To make your plug-in available for use in stack templates, the plug-in must register a reference name with the engine. This is done by defining a `resource_mapping` function in your plug-in module that returns a map of template resource type names and their corresponding implementation classes:

```
def resource_mapping():
    return { 'My::Custom::Plugin': MyResourceClass }
```

This would allow a template author to define a resource as:

```
resources:
  my_resource:
    type: My::Custom::Plugin
    properties:
    # ... your plug-in's properties ...
```

Note that you can define multiple plug-ins per module by simply returning a map containing a unique template type name for each. You may also use this to register a single resource plug-in under multiple template type names (which you would only want to do when constrained by backwards compatibility).

Configuring the Engine

In order to use your plug-in, Heat must be configured to read your resources from a particular directory. The `plugin_dirs` configuration option lists the directories on the local file system where the engine will search for plug-ins. Simply place the file containing your resource in one of these directories and the engine will make them available next time the service starts.

See one of the Installation Guides at http://docs.OpenStack.org/ for more information on configuring the orchestration service.

Testing

Tests can live inside the plug-in under the `tests` namespace/directory. The Heat plug-in loader will implicitly not load anything under that directory. This is useful when your plug-in tests have dependencies you don't want installed in production.

Putting It All Together

You can find the plugin classes in `heat/engine/resources`. An exceptionally simple one to start with is `random_string.py`; it is unusual in that it does not manipulate anything in the cloud!

Resource Contributions

The Heat team is interested in adding new resources that give Heat access to additional OpenStack or StackForge projects. The following checklist defines the requirements for a candidate resource to be considered for inclusion:

- Must wrap an OpenStack or StackForge project, or a third party project that is relevant to OpenStack users.
- Must have its dependencies listed in OpenStack's `global-requirements.txt` file, or else it should be able to conditionally disable itself when there are missing dependencies, without crashing or otherwise affecting the normal operation of the heat-engine service.
- The resource's support status flag must be set to `UNSUPPORTED`, to indicate that the Heat team is not responsible for supporting this resource.
- The code must be of comparable quality to official resources. The Heat team can help with this during the review phase.

If you have a resource that is a good fit, you are welcome to contact the Heat team. If for any reason your resource does not meet the above requirements, but you still think it can be useful to other users, you are encouraged to host it on your own repository and share it as a regular Python installable package. You can find example resource plug-ins that have all the required packaging files in the `contrib` directory of the official Heat git repository.

Heat Stack Lifecycle Scheduler Hints

This is a mechanism whereby when heat processes a stack with Server or Volume resources, the stack id, root stack id, stack resource uuid, stack resource name and the path in the stack can be passed by heat to nova and cinder as scheduler hints.

Enabling the scheduler hints

By default, passing the lifecycle scheduler hints is disabled. To enable it, set stack_scheduler_hints to True in heat.conf.

The hints

When heat processes a stack, and the feature is enabled, the stack id, root stack id, stack resource uuid, stack resource name, and the path in the stack (as a list of tuple, (stackresourcename, stackname)) will be passed by heat to nova and cinder as scheduler hints.

Purpose

A heat provider may have a need for custom code to examine stack requests prior to performing the operations to create or update a stack. After the custom code completes, the provider may want to provide hints to the nova or cinder schedulers with stack related identifiers, for processing by any custom scheduler plug-ins configured for nova or cinder.

Guru Meditation Reports

Heat contains a mechanism whereby developers and system administrators can generate a report about the state of a running Heat executable. This report is called a *Guru Meditation Report* (*GMR* for short).

Generating a GMR

A *GMR* can be generated by sending the *USR2* signal to any Heat process with support (see below). The *GMR* will then be outputted standard error for that particular process.

For example, suppose that `heat-api` has process id `10172`, and was run with `2>/var/log/heat/heat-api-err.log`. Then, `kill -USR2 10172` will trigger the Guru Meditation report to be printed to `/var/log/heat/heat-api-err.log`.

Structure of a GMR

The *GMR* is designed to be extensible; any particular executable may add its own sections. However, the base *GMR* consists of several sections:

Package
: Shows information about the package to which this process belongs, including version information

Threads
: Shows stack traces and thread ids for each of the threads within this process

Green Threads
: Shows stack traces for each of the green threads within this process (green threads don't have thread ids)

Configuration
: Lists all the configuration options currently accessible via the CONF object for the current process

Adding support for GMRs to new executable

Adding support for a *GMR* to a given executable is fairly easy.

First import the module (currently residing in oslo-incubator), as well as the Heat version module:

```
from oslo_reports import guru_meditation_report as gmr
from heat import version
```

Then, register any additional sections (optional):

```
TextGuruMeditation.register_section('Some Special Section',
                                    some_section_generator)
```

Finally (under main), before running the "main loop" of the executable (usually `server.start()` or something similar), register the *GMR* hook:

```
TextGuruMeditation.setup_autorun(version)
```

Extending the GMR

As mentioned above, additional sections can be added to the GMR for a particular executable. For more information, see the documentation about `oslo.reports`: oslo.reports

Heat Support Status usage Guide

Heat allows to use for each resource, property, attribute special option named *support_status*, which describes current state of object: current status, since what time this status is actual, any additional information about object's state. This guide describes a detailed state life cycle of resources, properties and attributes.

Support Status option and its parameters

Support status of object may be specified by using class `SupportStatus`, which has follow options:

***status*:**

Current status of object. Allowed values:

- SUPPORTED. Default value of status parameter. All objects with this status are available and can be used.
- DEPRECATED. Object with this status is available, but using it in code or templates is undesirable. As usual, can be reference in message to new object, which can be used instead of deprecated resource.
- HIDDEN. The last step in the deprecation process. Old stacks containing resources in this status will continue functioning. Certain functionality is disabled for resources in this status (resource-type-list, resource-type-show, and resource-type-template). Resources in HIDDEN status are not included in the documentation. A known limitation is that new stacks can be created with HIDDEN resources. See below for more details about the removal and deprecation process.
- UNSUPPORTED. Resources with UNSUPPORTED status are not supported by Heat team, i.e. user can use it, but it may be broken.

***version*:**

Release name, since which current status is active. Parameter is optional, but should be defined or changed any time SupportStatus is specified or status changed. It used for better understanding from which release object in current status. .. note:

`Since Liberty release mark looks like 5.0.0 instead of 2015.2.`

***message*:**

Any additional information about object's state, e.g. `'Use property new_property instead.'`.

***previous_status*:**

Option, which allows to display object's previous status, if any. This is helpful for displaying full life cycle of object. Type of *previous_status* is SupportStatus.

Life cycle of resource, property, attribute

This section describes life cycle of such objects as resource, property and attribute. All these objects have same life cycle:

```
UNSUPPORTED -> SUPPORTED -> DEPRECATED -> HIDDEN
                                       \
                                        -> UNSUPPORTED
```

where UNSUPPORTED is optional.

Creating process of object

During creating object there is a reason to add support status. So new object should contains *support_status* parameter equals to `SupportStatus` class with defined version of object and, maybe, some message. This parameter allows user to understand, from which this object OpenStack release this object is available and can be used.

Deprecating process of object

When some object becomes obsolete, user should know about that, so there is need to add information about deprecation in *support_status* of object. Status of `SupportStatus` must equals to DEPRECATED. If there is no *version* parameter, need to add one with current release otherwise move current status to *previous_status*

and add to *version* current release as value. If some new object replaces old object, it will be good decision to add some information about new object to *support_status* message of old object, e.g. 'Use property new_property instead.'.

Removing process of object

After at least one full release cycle deprecated object should be hidden and *support_status* status should equals to HIDDEN. HIDDEN status means hiding object from documentation and from result of `resource-type-list` CLI command, if object is resource. Also, `resource-type-show` command with such resource will raise NotSupported exception.

Using Support Status during code writing

When adding new objects or adding objects instead of some old (e.g. property subnet instead of subnet_id in OS::Neutron::RouterInterface), there is some information about time of adding objects (since which release it will be available or unavailable). This section described `SupportStatus` during creating/deprecating/removing resources and properties and attributes. Note, that `SupportStatus` locates in support.py, so you need to import *support*. For specifying status, use *support* constant names, e.g. support.SUPPORTED. All constant names described in section above.

Using Support Status during creation

Option *support_status* may be used for whole resource:

```
class ResourceWithType(resource.Resource):

    support_status=support.SupportStatus(
        version='5.0.0',
        message=_('Optional message')
    )
```

To define *support_status* for property or attribute, follow next steps:

```
PROPERTY: properties.Schema(
    ...
    support_status=support.SupportStatus(
        version='5.0.0',
```

```
            message=_('Optional message')
        )
    )
```

Same support_status definition for attribute schema.

Note, that in this situation status parameter of `SupportStatus` uses default value, equals to SUPPORTED.

Using Support Status during deprecation and hiding

When time of deprecation or hiding resource/property/attribute comes, follow next steps:

- If there is some support_status in object, add previous_status parameter with current `SupportStatus` value and change all other parameters for current status, version and, maybe, message.
- If there is no support_status option, add new one with parameters status equals to current status, version equals to current release note and, optionally, some message.

Using Support Status during resource deprecating looks like:

```
class ResourceWithType(resource.Resource):

    support_status=support.SupportStatus(
        status=support.DEPRECATED,
        version='5.0.0',
        message=_('Optional message'),
        previous_status=support.SupportStatus(version='2014.2')
    )
```

Using Support Status during attribute (or property) deprecating looks like:

```
ATTRIBUTE: attributes.Schema(
    ...
    support_status=support.SupportStatus(
        status=support.DEPRECATED,
        version='5.0.0',
        message=_('Optional message like: Use attribute new_attr'),
        previous_status=support.SupportStatus(
            version='2014.2',
            message=_('Feature available since 2014.2'))
    )
```

```
)
```

Same *support_status* defining for property schema.

Note, that during hiding object status should be equal support.HIDDEN instead of support.DEPRECATED. Besides that, SupportStatus with DEPRECATED status should be moved to *previous_status*, e.g.:

```
support.SupportStatus(
    status=support.HIDDEN,
    version='5.0.0',
    message=_('Some message'),
    previous_status=support.SupportStatus(
        status=support.DEPRECATED,
        version='2015.1',
        previous_status=support.SupportStatus(version='2014.2')
    )
)
```

During hiding properties, if some hidden property has alternative, use translation mechanism for translating properties from old to new one. See below, how to use this mechanism.

Translating mechanism for hidden properties

Sometimes properties become deprecated and replaced by another. There is translation mechanism for that. Mechanism used for such cases:

- If there are two properties in properties_schema, which have STRING, INTEGER, NUMBER or BOOLEAN type.
- If there are two properties: one in LIST or MAP property sub-schema and another on the top schema.
- If there are two properties in LIST property.
- If there are non-LIST property and LIST property, which is designed to replace non-LIST property.
- If there is STRING property, which contains name or ID of some entity, e.g. subnet, and should be resolved to entity's ID.

Mechanism has rules and executes them. To define rule, `TranslationRule` class called and specifies *translation_path* - list with path in properties_schema for property which will be affected; *value* - value, which will be added to property,

specified by previous parameter; *value_name* - name of old property, used for case 4; *value_path* - list with path in properties_schema for property which will be used for getting value. `TranslationRule` supports next rules:

- *ADD*. This rule allows to add some value to LIST-type properties. Only LIST-type values can be added to such properties. Using for other cases is prohibited and will be returned with error.
- *REPLACE*. This rule allows to replace some property value to another. Used for all types of properties. Note, that if property has list type, then value will be replaced for all elements of list, where it needed. If element in such property must be replaced by value of another element of this property, *value_name* must be defined.
- *DELETE*. This rule allows to delete some property. If property has list type, then deleting affects value in all list elements.
- *RESOLVE* - This rule allows to resolve some property using client and the *finder* function. Finders may require an additional *entity* key.

Each resource, which has some hidden properties, which can be replaced by new, must overload translation_rules method, which should return a list of `TranslationRules`, for example:

```
def translation_rules(self):
     return [properties.TranslationRule(
         self.properties,
         properties.TranslationRule.REPLACE,
         translation_path=[self.NETWORKS, self.NETWORK_ID],
         value_name=self.NETWORK_UUID)]
```

Using Rally on Heat gates

Heat gate allows to use Rally for performance testing for each particular patch. This functionality can be used for checking patch on performance regressions and also for detecting any floating bugs for common scenarios.

How to run Rally for particular patch

As was mentioned above Heat allows to execute Rally scenarios as a gate job for particular patch. It can be done by posting comment with text `check experimental` for patch on review. It will run bunch of jobs, one of which has name `gate-rally-dsvm-fakevirt-heat`.

List of scenarios, which will be executed, is presented in file `heat-fakevirt.yaml`. Default version of this file is available here: https://github.com/openstack/heat/blob/master/rally-scenarios/heat-fakevirt.yaml

Obviously performance analysis make sense, when it can be compared with some another performance data. So two different approaches can be used for it:

> - Comparison of one part of code with some custom changes (see *Check performance or how to detect regression*)
> - Comparison of two different code parts (see *Compare output API performance*)

Examples of using Rally

Previously two main approaches of using Rally job for Heat were highlighted. Corresponding examples will be described in this part of documentation.

However need to note, that there are a lot of other ways how to use Rally job for Heat performance. For example, this job can be launched periodically (twice in week) for random patches and these results will be compared between each other. It allows to see, that Heat has not any performance regressions.

Check performance or how to detect regression

The easiest way of using Rally is to execute already existing scenarios. One of the examples is presented in patch https://review.openstack.org/#/c/279450/ . In this patch was executed scenario already existing in Rally `HeatStacks.create_and_delete_stack`. During executing this scenario Rally creates and then, when stack is created, delete Heat stack. All existing scenarios can be found here: https://github.com/openstack/rally/blob/master/rally/plugins/openstack/scenarios/heat/stacks.py

Mentioned scenario uses Heat template as a parameter for task. The template path should be mentioned for argument `template_path`. It can be one of Heat templates presented in Rally repository (https://github.com/openstack/rally/tree/master/samples/tasks/scenarios/heat/templates) or new one, like it was done for mentioned patch. New added template should be placed in `rally-scenarios/extra/` directory.

Also it's possible to specify other fields for each Rally task, like `sla` or `context`. More

information about other configuration setting is available by link https://rally.readthedocs.org/en/latest/tutorial.html

Mentioned patch was proposed for confirmation caching mechanism of Heat template validation process (see https://specs.openstack.org/openstack/heat-specs/specs/liberty/constraint-validation-cache.html). So it contains some changes in OS::Heat::TestResource resource, which allows to demonstrate mentioned caching feature improvements.

Initially test was run against current devstack installation, where caching is disabled (e.g. Patch Set 7). The follow results were gotten:

Action	Min (sec)	Max (sec)	Avg (sec)	Success	Count
heat.create_stack	38.223	48.085	42.971	100.0%	10
heat.delete_stack	11.755	18.155	14.085	100.0%	10
total	50.188	65.361	57.057	100.0%	10

In the next patch set (Patch Set 8) was updated by adding Depends-On reference to commit message. It let to execute the same test with patch for devstack, which turns on caching (https://review.openstack.org/#/c/279400/). The results for this case were:

Action	Min (sec)	Max (sec)	Avg (sec)	Success	Count
heat.create_stack	11.863	16.074	14.174	100.0%	10
heat.delete_stack	9.144	11.663	10.595	100.0%	10
total	21.557	27.18	24.77	100.0%	10

Comparison average values for create_stack action in the first and the second executions shows, that with enabled caching create_stack works faster in 3 times. It is a tangible improvement for create_stack operation. Need to note, that in described test delay for each constraint validation request takes 0.3 sec. as specified in `constraint_prop_secs` property of TestResource. It may be more, than real time delay, but it allows to confirm, that caching works correct.

Also this approach may be used for detecting regressions. In this case workflow may be presented as follow list of steps:

- add to task list (`heat-fakevirt.yaml`) existing or new tasks.

> - wait a result of this execution.
> - upload patchset with changes (new feature) and launch the same test again.
> - compare performance results.

Compare output API performance

Another example of using Rally job is writing custom Rally scenarios in Heat repository. There is an example of this is presented on review: https://review.openstack.org/#/c/270225/

It's similar on the first example, but requires more Rally specific coding. New tasks in `heat-fakevirt.yaml` use undefined in Rally repository scenarios:

- CustomHeatBenchmark.create_stack_and_show_output_new
- CustomHeatBenchmark.create_stack_and_show_output_old
- CustomHeatBenchmark.create_stack_and_list_output_new
- CustomHeatBenchmark.create_stack_and_list_output_old

All these scenarios are defined in the same patch and placed in `rally-scenarios/plugins/` directory.

The aim of these scenarios and tasks is to demonstrate differences between new and old API calls. Heat client has a two commands for operating stack outputs: `heat output-list` and `heat output-show <output-id>`. Previously there are no special API calls for getting this information from server and this data was obtained from whole Heat Stack object. This was changed after implementation new API for outputs: https://specs.openstack.org/openstack/heat-specs/specs/mitaka/api-calls-for-output.html

As described in the mentioned specification outputs can be obtained via special requests to Heat API. According to this changes code in Heat client was updated to use new API, if it's available.

The initial problem for this change was performance issue, which can be formulated as: execution command `heat output-show <output-id>` with old approach required resolving all outputs in Heat Stack, before getting only one output specified by user.

The same issue was and with `heat output-list`, which required to resolve all outputs only for providing list of output keys without resolved values.

Two scenarios with suffix `*_new` use new output API. These scenarios are not presented in Rally yet, because it's new API. Another two scenarios with suffix `*_old` are based on the old approach of getting outputs. This code was partially replaced

by new API, so it's not possible to use it on fresh devstack. As result this custom code was written as two custom scenarios.

All these scenarios were added to task list and executed in the same time. Results of execution are shown below:

create_stack_and_show_output_old

Action	Min (sec)	Max (sec)	Avg (sec)	Success	Count
heat.create_stack	13.559	14.298	13.899	100.0%	5
heat.show_output_old	5.214	5.297	5.252	100.0%	5
heat.delete_stack	5.445	6.962	6.008	100.0%	5
total	24.243	26.146	25.159	100.0%	5

create_stack_and_show_output_new

Action	Min (sec)	Max (sec)	Avg (sec)	Success	Count
heat.create_stack	13.719	14.286	13.935	100.0%	5
heat.show_output_new	0.699	0.835	0.762	100.0%	5
heat.delete_stack	5.398	6.457	5.636	100.0%	5
total	19.873	21.21	20.334	100.0%	5

Average value for execution `output-show` for old approach obviously more, then for new API. It happens, because new API resolve only one specified output.

Same results are for `output-list`:

create_stack_and_list_output_old

Action	Min (sec)	Max (sec)	Avg (sec)	Success	Count
heat.create_stack	13.861	14.573	14.141	100.0%	5
heat.list_output_old	5.247	5.339	5.281	100.0%	5
heat.delete_stack	6.727	6.845	6.776	100.0%	5
total	25.886	26.696	26.199	100.0%	5

create_stack_and_list_output_new

Action	Min (sec)	Max (sec)	Avg (sec)	Success	Count
heat.create_stack	13.902	21.117	16.729	100.0%	5
heat.list_output_new	0.147	0.363	0.213	100.0%	5
heat.delete_stack	6.616	8.202	7.022	100.0%	5
total	20.838	27.908	23.964	100.0%	5

It's also expected, because for getting list of output names is not necessary resolved values, how it is done in new API.

All mentioned results clearly show performance changes and allow to confirm, that new approach works correctly.

The `heat.api.aws.ec2token` Module

class `heat.api.aws.ec2token.EC2Token`(***app***, ***conf***)

Bases: heat.common.wsgi.Middleware

Authenticate an EC2 request with keystone and convert to token.

ssl_options

heat.api.aws.ec2token.EC2Token_filter_factory(*global_conf*, ***local_conf*)

Factory method for paste.deploy.

heat.api.aws.ec2token.list_opts()

The heat.api.aws.exception Module

Heat API exception subclasses - maps API response errors to AWS Errors.

exception **heat.api.aws.exception.AlreadyExistsError**(*detail=None*)

Bases: heat.api.aws.exception.HeatAPIException

Resource with the name requested already exists.

code = *400*
explanation = *u'Resource with the name requested already exists'*
title = *'AlreadyExists'*

exception **heat.api.aws.exception.HeatAPIException**(*detail=None*)

Bases: webob.exc.HTTPError

webob HTTPError subclass that creates a serialized body.

Subclass webob HTTPError so we can correctly serialize the wsgi response into the http response body, using the format specified by the request. Note this should not be used directly, instead use the subclasses defined below which map to AWS API errors.

code = *400*
err_type = *'Sender'*
explanation = *u'Generic HeatAPIException, please use specific subclasses!'*

get_unserialized_body()

Return a dict suitable for serialization in the wsgi controller.

This wraps the exception details in a format which maps to the expected format for the AWS API.

title = '*HeatAPIException*'

exception

heat.api.aws.exception.HeatAPINotImplementedError(*detail=None*)

Bases: heat.api.aws.exception.HeatAPIException

API action is not yet implemented.

code = *500*
err_type = '*Server*'
explanation = *u*'*The requested action is not yet implemented*'
title = '*APINotImplemented*'

exception **heat.api.aws.exception.HeatAccessDeniedError(*detail=None*)**

Bases: heat.api.aws.exception.HeatAPIException

Authentication fails due to user IAM group memberships.

This is the response given when authentication fails due to user IAM group memberships meaning we deny access.

code = *403*
explanation = *u*'*User is not authorized to perform action*'
title = '*AccessDenied*'

exception

heat.api.aws.exception.HeatActionInProgressError(*detail=None*)

Bases: heat.api.aws.exception.HeatAPIException

Cannot perform action on stack in its current state.

code = *400*
explanation = '*Cannot perform action on stack while other actions are in progress*'
title = '*InvalidAction*'

exception

heat.api.aws.exception.HeatIncompleteSignatureError(*detail=None*)

Bases: heat.api.aws.exception.HeatAPIException

The request signature does not conform to AWS standards.

code = *400*
explanation = *u'The request signature does not conform to AWS standards'*
title = '*IncompleteSignature*'

***exception* heat.api.aws.exception.HeatInternalFailureError(*detail=None*)**

Bases: heat.api.aws.exception.HeatAPIException

The request processing has failed due to some unknown error.

code = *500*
err_type = '*Server*'
explanation = *u'The request processing has failed due to an internal error'*
title = '*InternalFailure*'

***exception* heat.api.aws.exception.HeatInvalidActionError(*detail=None*)**

Bases: heat.api.aws.exception.HeatAPIException

The action or operation requested is invalid.

code = *400*
explanation = *u'The action or operation requested is invalid'*
title = '*InvalidAction*'

exception

heat.api.aws.exception.HeatInvalidClientTokenIdError(*detail=None*)

Bases: heat.api.aws.exception.HeatAPIException

The X.509 certificate or AWS Access Key ID provided does not exist.

code = *403*
explanation = *u'The certificate or AWS Key ID provided does not exist'*
title = '*InvalidClientTokenId*'

exception

heat.api.aws.exception.HeatInvalidParameterCombinationError(*detail=None*)

Bases: heat.api.aws.exception.HeatAPIException

Parameters that must not be used together were used together.

code = *400*
explanation = *u'Incompatible parameters were used together'*
title = *'InvalidParameterCombination'*

exception

heat.api.aws.exception.HeatInvalidParameterValueError(*detail=None*)

Bases: heat.api.aws.exception.HeatAPIException

A bad or out-of-range value was supplied for the input parameter.

code = *400*
explanation = *u'A bad or out-of-range value was supplied'*
title = *'InvalidParameterValue'*

exception

heat.api.aws.exception.HeatInvalidQueryParameterError(*detail=None*)

Bases: heat.api.aws.exception.HeatAPIException

AWS query string is malformed, does not adhere to AWS standards.

code = *400*
explanation = *u'AWS query string is malformed, does not adhere to AWS spec'*
title = *'InvalidQueryParameter'*

exception

heat.api.aws.exception.HeatMalformedQueryStringError(*detail=None*)

Bases: heat.api.aws.exception.HeatAPIException

The query string is malformed.

code = *404*
explanation = *u'The query string is malformed'*
title = *'MalformedQueryString'*

exception **heat.api.aws.exception.HeatMissingActionError(*detail=None*)**

Bases: heat.api.aws.exception.HeatAPIException

The request is missing an action or operation parameter.

code = *400*
explanation = *u'The request is missing an action or operation parameter'*
title = '*MissingAction*'

exception
heat.api.aws.exception.HeatMissingAuthenticationTokenError(*detail=None*)

Bases: heat.api.aws.exception.HeatAPIException

Does not contain a valid AWS Access Key or certificate.

Request must contain either a valid (registered) AWS Access Key ID or X.509 certificate.

code = *403*
explanation = *u'Does not contain a valid AWS Access Key or certificate'*
title = '*MissingAuthenticationToken*'

exception
heat.api.aws.exception.HeatMissingParameterError(*detail=None*)

Bases: heat.api.aws.exception.HeatAPIException

A mandatory input parameter is missing.

An input parameter that is mandatory for processing the request is missing.

code = *400*
explanation = *u'A mandatory input parameter is missing'*
title = '*MissingParameter*'

exception **heat.api.aws.exception.HeatOptInRequiredError(*detail=None*)**

Bases: heat.api.aws.exception.HeatAPIException

The AWS Access Key ID needs a subscription for the service.

code = *403*
explanation = *u'The AWS Access Key ID needs a subscription for the service'*
title = '*OptInRequired*'

***exception* heat.api.aws.exception.HeatRequestExpiredError(*detail=None*)**

Bases: heat.api.aws.exception.HeatAPIException

Request expired or more than 15 minutes in the future.

Request is past expires date or the request date (either with 15 minute padding), or the request date occurs more than 15 minutes in the future.

code = *400*
explanation = *u'Request expired or more than 15mins in the future'*
title = '*RequestExpired*'

***exception* heat.api.aws.exception.HeatRequestLimitExceeded(*detail=None*)**

Bases: heat.api.aws.exception.HeatAPIException

Payload size of the request exceeds maximum allowed size.

code = *400*
explanation = *u'Payload exceeds maximum allowed size'*
title = '*RequestLimitExceeded*'

***exception* heat.api.aws.exception.HeatServiceUnavailableError(*detail=None*)**

Bases: heat.api.aws.exception.HeatAPIException

The request has failed due to a temporary failure of the server.

code = *503*
err_type = '*Server*'
explanation = *u'Service temporarily unavailable'*
title = '*ServiceUnavailable*'

***exception* heat.api.aws.exception.HeatSignatureError(*detail=None*)**

Bases: heat.api.aws.exception.HeatAPIException

Authentication fails due to a bad signature.

code = *403*

explanation = *u'The request signature we calculated does not match the signature you provided'*

title = '*SignatureDoesNotMatch*'

***exception* heat.api.aws.exception.HeatThrottlingError(*detail=None*)**

Bases: heat.api.aws.exception.HeatAPIException

Request was denied due to request throttling.

code = *400*

explanation = *u'Request was denied due to request throttling'*

title = '*Throttling*'

heat.api.aws.exception.map_remote_error(*ex*)

Map rpc_common.RemoteError exceptions to HeatAPIException subclasses.

Map rpc_common.RemoteError exceptions returned by the engine to HeatAPIException subclasses which can be used to return properly formatted AWS error responses.

The heat.api.aws.utils Module

Helper utilities related to the AWS API implementations.

heat.api.aws.utils.extract_param_list(*params*, *prefix=''*)

Extract a list-of-dicts based on parameters containing AWS style list.

MetricData.member.1.MetricName=buffers MetricData.member.1.Unit=Bytes MetricData.member.1.Value=231434333 MetricData.member.2.MetricName=buffers2 MetricData.member.2.Unit=Bytes MetricData.member.2.Value=12345

This can be extracted by passing prefix=MetricData, resulting in a list containing two dicts.

heat.api.aws.utils.extract_param_pairs(*params*, *prefix=''*, *keyname=''*, *valuename=''*)

Extract user input params from AWS style parameter-pair encoded list.

In the AWS API list items appear as two key-value pairs (passed as query parameters) with keys of the form below:

Prefix.member.1.keyname=somekey Prefix.member.1.keyvalue=somevalue
Prefix.member.2.keyname=anotherkey Prefix.member.2.keyvalue=somevalue

We reformat this into a dict here to match the heat engine API expected format.

heat.api.aws.utils.format_response(*action*, *response*)

Format response from engine into API format.

heat.api.aws.utils.get_param_value(*params*, *key*)

Looks up an expected parameter in a parsed params dict.

Helper function, looks up an expected parameter in a parsed params dict and returns the result. If params does not contain the requested key we raise an exception of the appropriate type.

heat.api.aws.utils.reformat_dict_keys(*keymap=None*, *inputdict=None*)

Utility function for mapping one dict format to another.

The heat.api.cfn.v1.signal Module

class **heat.api.cfn.v1.signal.SignalController**(*options*)

Bases: object

signal(*req*, *arn*, *body=None*)

update_waitcondition(*req*, *body*, *arn*)

heat.api.cfn.v1.signal.create_resource(*options*)

Signal resource factory method.

The heat.api.cfn.v1.stacks Module

Stack endpoint for Heat CloudFormation v1 API.

class **heat.api.cfn.v1.stacks.StackController(*options*)**

Bases: object

WSGI controller for stacks resource in Heat CloudFormation v1 API.

Implements the API actions.

CREATE_OR_UPDATE_ACTION = (*'CreateStack', 'UpdateStack'*)
CREATE_STACK = '*CreateStack*'
UPDATE_STACK = '*UpdateStack*'

cancel_update(*req*)

create(*req*)

create_or_update(*req*, *action=None*)

Implements CreateStack and UpdateStack API actions.

Create or update stack as defined in template file.

default(*req*, ***args*)

delete(*req*)

Implements the DeleteStack API action.

Deletes the specified stack.

describe(*req*)

Implements DescribeStacks API action.

Gets detailed information for a stack (or all stacks).

describe_stack_resource(*req*)

Implements the DescribeStackResource API action.

Return the details of the given resource belonging to the given stack.

describe_stack_resources(*req*)

Implements the DescribeStackResources API action.

Return details of resources specified by the parameters.

StackName: returns all resources belonging to the stack.

PhysicalResourceId: returns all resources belonging to the stack this resource is associated with.

Only one of the parameters may be specified.

Optional parameter:

LogicalResourceId: filter the resources list by the logical resource id.

estimate_template_cost(*req*)

Implements the EstimateTemplateCost API action.

Get the estimated monthly cost of a template.

events_list(*req*)

Implements the DescribeStackEvents API action.

Returns events related to a specified stack (or all stacks).

get_template(*req*)

Implements the GetTemplate API action.

Get the template body for an existing stack.

list(*req*)

Implements ListStacks API action.

Lists summary information for all stacks.

list_stack_resources(*req*)

Implements the ListStackResources API action.

Return summary of the resources belonging to the specified stack.

update(*req*)

validate_template(*req*)

Implements the ValidateTemplate API action.

Validates the specified template.

heat.api.cfn.v1.stacks.create_resource(*options*)

Stacks resource factory method.

The heat.api.cfn.versions Module

Controller that returns information on the heat API versions.

Now it's a subclass of module versions, because of identity with OpenStack module versions.

The heat.api.cloudwatch.watch Module

Endpoint for heat AWS-compatible CloudWatch API.

class **heat.api.cloudwatch.watch.WatchController**(*options*)

Bases: object

WSGI controller for CloudWatch resource in heat API.

Implements the API actions.

delete_alarms(*req*)

Implements DeleteAlarms API action.

describe_alarm_history(*req*)

Implements DescribeAlarmHistory API action.

describe_alarms(*req*)

Implements DescribeAlarms API action.

describe_alarms_for_metric(*req*)

Implements DescribeAlarmsForMetric API action.

disable_alarm_actions(*req*)

Implements DisableAlarmActions API action.

enable_alarm_actions(*req*)

Implements EnableAlarmActions API action.

get_metric_statistics(*req*)

Implements GetMetricStatistics API action.

list_metrics(*req*)

Implements ListMetrics API action.

Lists metric datapoints associated with a particular alarm, or all alarms if none specified.

put_metric_alarm(*req*)

Implements PutMetricAlarm API action.

put_metric_data(*req*)

Implements PutMetricData API action.

set_alarm_state(*req*)

Implements SetAlarmState API action.

heat.api.cloudwatch.watch.create_resource(*options*)

Watch resource factory method.

The heat.api.middleware.fault Module

A middleware that turns exceptions into parsable string.

Inspired by Cinder's faultwrapper.

class **heat.api.middleware.fault.Fault**(*error*)

Bases: object

class **heat.api.middleware.fault.FaultWrapper**(*application*)

Bases: heat.common.wsgi.Middleware

Replace error body with something the client can parse.

error_map = *{'InvalidTenant': <class 'webob.exc.HTTPForbidden'>, 'ResourceTypeUnavailable': <class 'webob.exc.HTTPBadRequest'>, 'InvalidSchemaError': <class 'webob.exc.HTTPBadRequest'>, 'ObjectFieldInvalid': <class 'webob.exc.HTTPBadRequest'>, 'NotSupported': <class 'webob.exc.HTTPBadRequest'>, 'InvalidTemplateReference': <class 'webob.exc.HTTPBadRequest'>, 'ResourcePropertyConflict': <class 'webob.exc.HTTPBadRequest'>, 'Forbidden': <class 'webob.exc.HTTPForbidden'>, 'ActionInProgress': <class 'webob.exc.HTTPConflict'>, 'InvalidTemplateVersion': <class 'webob.exc.HTTPBadRequest'>, 'UnsupportedObjectError': <class 'webob.exc.HTTPBadRequest'>, 'ServerBuildFailed': <class 'webob.exc.HTTPInternalServerError'>, 'StopActionFailed': <class 'webob.exc.HTTPInternalServerError'>, 'ReadOnlyFieldError': <class*

'webob.exc.HTTPBadRequest'>, 'EventSendFailed': <class
'webob.exc.HTTPInternalServerError'>, 'NotFound': <class
'webob.exc.HTTPNotFound'>, 'RevertFailed': <class
'webob.exc.HTTPInternalServerError'>, 'PropertyUnspecifiedError':
<class 'webob.exc.HTTPBadRequest'>, 'RequestLimitExceeded': <class
'webob.exc.HTTPBadRequest'>, 'ObjectActionError': <class
'webob.exc.HTTPBadRequest'>, 'ResourceActionNotSupported': <class
'webob.exc.HTTPBadRequest'>, 'UnknownUserParameter': <class
'webob.exc.HTTPBadRequest'>, 'EntityNotFound': <class
'webob.exc.HTTPNotFound'>, 'OrphanedObjectError': <class
'webob.exc.HTTPBadRequest'>, 'AttributeError': <class
'webob.exc.HTTPBadRequest'>, 'PhysicalResourceNameAmbiguity':
<class 'webob.exc.HTTPBadRequest'>, 'UserParameterMissing': <class
'webob.exc.HTTPBadRequest'>, 'ValueError': <class
'webob.exc.HTTPBadRequest'>, 'IncompatibleObjectVersion': <class
'webob.exc.HTTPBadRequest'>, 'MissingCredentialError': <class
'webob.exc.HTTPBadRequest'>, 'StackExists': <class
'webob.exc.HTTPConflict'>, 'ImmutableParameterModified': <class
'webob.exc.HTTPBadRequest'>, 'ResourceNotAvailable': <class
'webob.exc.HTTPNotFound'>, 'Invalid': <class
'webob.exc.HTTPBadRequest'>, 'InvalidBreakPointHook': <class
'webob.exc.HTTPBadRequest'>, 'InvalidGlobalResource': <class
'webob.exc.HTTPInternalServerError'>, 'StackValidationFailed': <class
'webob.exc.HTTPBadRequest'>, 'InvalidTemplateSection': <class
'webob.exc.HTTPBadRequest'>}

`process_request`(*req*)

The `heat.api.middleware.ssl` Module

class `heat.api.middleware.ssl.SSLMiddleware`(*application*, **args*, ***kwargs*)

Bases: `oslo_middleware.ssl.SSLMiddleware`

`heat.api.middleware.ssl.list_opts`()

The heat.api.middleware.version_negotiation Module

Inspects the requested URI for a version string and/or Accept headers.

Also attempts to negotiate an API controller to return.

***class* heat.api.middleware.version_negotiation.VersionNegotiationFilter(*version_controller*, *app*, *conf*, ***local_conf*)**

Bases: heat.common.wsgi.Middleware

process_request(*req*)

Process Accept header or simply return correct API controller.

If there is a version identifier in the URI, return the correct API controller, otherwise, if we find an Accept: header, process it

The heat.api.openstack.v1.actions Module

***class* heat.api.openstack.v1.actions.ActionController(*options*)**

Bases: object

WSGI controller for Actions in Heat v1 API.

Implements the API for stack actions

ACTIONS = *('suspend', 'resume', 'check', 'cancel_update')*

CANCEL_UPDATE = '*cancel_update*'

CHECK = '*check*'

REQUEST_SCOPE = '*actions*'

RESUME = '*resume*'

SUSPEND = '*suspend*'

action(*controller*, *req*, *tenant_id*, ***kwargs*)

Performs a specified action on a stack.

The body is expecting to contain exactly one item whose key specifies the action.

heat.api.openstack.v1.actions.create_resource(*options*)

Actions action factory method.

The heat.api.openstack.v1.build_info Module

***class* heat.api.openstack.v1.build_info.BuildInfoController(*options*)**

Bases: object

WSGI controller for BuildInfo in Heat v1 API.

Returns build information for current app.

REQUEST_SCOPE = '*build_info*'

build_info(*controller*, *req*, *tenant_id*, ***kwargs*)

heat.api.openstack.v1.build_info.create_resource(*options*)

BuildInfo factory method.

The heat.api.openstack.v1.events Module

class **heat.api.openstack.v1.events.EventController(*options*)**

Bases: object

WSGI controller for Events in Heat v1 API.

Implements the API actions.

REQUEST_SCOPE = '*events*'

index(*controller*, *req*, *tenant_id*, ***kwargs*)

Lists summary information for all events.

show(*controller*, *req*, *tenant_id*, ***kwargs*)

Gets detailed information for an event.

heat.api.openstack.v1.events.create_resource(*options*)

Events resource factory method.

heat.api.openstack.v1.events.format_event(*req*, *event*, *keys=None*)

The heat.api.openstack.v1.resources Module

class **heat.api.openstack.v1.resources.ResourceController(*options*)**

Bases: object

WSGI controller for Resources in Heat v1 API.

Implements the API actions.

REQUEST_SCOPE = '*resource*'

index(*controller*, *req*, *tenant_id*, ***kwargs*)

Lists information for all resources.

mark_unhealthy(*controller*, *req*, *tenant_id*, ***kwargs*)

Mark a resource as healthy or unhealthy.

metadata(*controller*, *req*, *tenant_id*, ***kwargs*)

Gets metadata information for a resource.

show(*controller*, *req*, *tenant_id*, ***kwargs*)

Gets detailed information for a resource.

signal(*controller*, *req*, *tenant_id*, ***kwargs*)

heat.api.openstack.v1.resources.create_resource(*options*)

Resources resource factory method.

heat.api.openstack.v1.resources.format_resource(*req*, *res*, *keys=None*)

The heat.api.openstack.v1.services Module

***class* heat.api.openstack.v1.services.ServiceController(*options*)**

Bases: object

WSGI controller for reporting the heat engine status in Heat v1 API.

REQUEST_SCOPE = '*service*'

index(*controller*, *req*, *tenant_id*, ***kwargs*)

heat.api.openstack.v1.services.create_resource(*options*)

The heat.api.openstack.v1.software_configs Module

***class* heat.api.openstack.v1.software_configs.SoftwareConfigController(*options*)**

Bases: object

WSGI controller for Software config in Heat v1 API.

Implements the API actions.

REQUEST_SCOPE = '*software_configs*'

create(*controller*, *req*, *tenant_id*, ***kwargs*)

Create a new software config.

default(*req*, ***args*)

delete(*controller*, *req*, *tenant_id*, ***kwargs*)

Delete an existing software config.

global_index(*controller*, *req*, *tenant_id*, ***kwargs*)

index(*controller*, *req*, *tenant_id*, ***kwargs*)

Lists summary information for all software configs.

show(*controller*, *req*, *tenant_id*, ***kwargs*)

Gets detailed information for a software config.

heat.api.openstack.v1.software_configs.create_resource(*options*)

Software configs resource factory method.

The heat.api.openstack.v1.software_deployments Module

***class* heat.api.openstack.v1.software_deployments.SoftwareDeploymentController(*options*)**

Bases: object

WSGI controller for Software deployments in Heat v1 API.

Implements the API actions.

REQUEST_SCOPE = '*software_deployments*'

create(*controller*, *req*, *tenant_id*, ***kwargs*)

Create a new software deployment.

default(*req*, ***args*)

delete(*controller*, *req*, *tenant_id*, ***kwargs*)

Delete an existing software deployment.

index(*controller*, *req*, *tenant_id*, ***kwargs*)

List software deployments.

metadata(*controller*, *req*, *tenant_id*, ***kwargs*)

List software deployments grouped by the group name.

This is done for the requested server.

show(*controller*, *req*, *tenant_id*, ***kwargs*)

Gets detailed information for a software deployment.

update(*controller*, *req*, *tenant_id*, ***kwargs*)

Update an existing software deployment.

heat.api.openstack.v1.software_deployments.create_resource(*options*)

Software deployments resource factory method.

The heat.api.openstack.v1.stacks Module

Stack endpoint for Heat v1 REST API.

class **heat.api.openstack.v1.stacks.InstantiationData**(*data*, *patch=False*)

Bases: object

The data to create or update a stack.

The data accompanying a PUT or POST request.

PARAMS = (*'stack_name'*, *'template'*, *'template_url'*, *'parameters'*, *'environment'*, *'files'*, *'environment_files'*)
PARAM_ENVIRONMENT = *'environment'*
PARAM_ENVIRONMENT_FILES = *'environment_files'*
PARAM_FILES = *'files'*
PARAM_STACK_NAME = *'stack_name'*
PARAM_TEMPLATE = *'template'*
PARAM_TEMPLATE_URL = *'template_url'*
PARAM_USER_PARAMS = *'parameters'*

args()

Get any additional arguments supplied by the user.

environment()

Get the user-supplied environment for the stack in YAML format.

If the user supplied Parameters then merge these into the environment global options.

environment_files()

files()

static **parse_error_check**(**args*, ***kwds*)

stack_name()

Return the stack name.

template()

Get template file contents.

Get template file contents, either inline, from stack adopt data or from a URL, in JSON or YAML format.

class **heat.api.openstack.v1.stacks.StackController**(***options***)

Bases: object

WSGI controller for stacks resource in Heat v1 API.

Implements the API actions.

REQUEST_SCOPE = '*stacks*'

abandon(***controller***, ***req***, ***tenant_id***, ******kwargs***)

Abandons specified stack.

Abandons specified stack by deleting the stack and it's resources from the database, but underlying resources will not be deleted.

create(***controller***, ***req***, ***tenant_id***, ******kwargs***)

Create a new stack.

default(*req*, ***args*)

delete(*controller*, *req*, *tenant_id*, ***kwargs*)

Delete the specified stack.

delete_snapshot(*controller*, *req*, *tenant_id*, ***kwargs*)

detail(*controller*, *req*, *tenant_id*, ***kwargs*)

Lists detailed information for all stacks.

environment(*controller*, *req*, *tenant_id*, ***kwargs*)

Get the environment for an existing stack.

export(*controller*, *req*, *tenant_id*, ***kwargs*)

Export specified stack.

Return stack data in JSON format.

files(*controller*, *req*, *tenant_id*, ***kwargs*)

Get the files for an existing stack.

generate_template(*controller*, *req*, *tenant_id*, ***kwargs*)

Generates a template based on the specified type.

global_index(*controller*, *req*, *tenant_id*, ***kwargs*)

index(*controller*, *req*, *tenant_id*, ***kwargs*)

Lists summary information for all stacks.

list_outputs(*controller*, *req*, *tenant_id*, ***kwargs*)

list_resource_types(*controller*, *req*, *tenant_id*, ***kwargs*)

Returns a resource types list which may be used in template.

list_snapshots(*controller*, *req*, *tenant_id*, ***kwargs*)

list_template_functions(*controller*, *req*, *tenant_id*, ***kwargs*)

Returns a list of available functions in a given template.

list_template_versions(*controller*, *req*, *tenant_id*, ***kwargs*)

Returns a list of available template versions.

lookup(*controller*, *req*, *tenant_id*, ***kwargs*)

Redirect to the canonical URL for a stack.

prepare_args(*data*)

preview(*controller*, *req*, *tenant_id*, ***kwargs*)

Preview the outcome of a template and its params.

preview_update(*controller*, *req*, *tenant_id*, ***kwargs*)

Preview update for existing stack with a new template/parameters.

preview_update_patch(*controller*, *req*, *tenant_id*, ***kwargs*)

Preview PATCH update for existing stack.

resource_schema(*controller*, *req*, *tenant_id*, ***kwargs*)

Returns the schema of the given resource type.

restore_snapshot(*controller*, *req*, *tenant_id*, ***kwargs*)

show(*controller*, *req*, *tenant_id*, ***kwargs*)

Gets detailed information for a stack.

show_output(*controller*, *req*, *tenant_id*, ***kwargs*)

show_snapshot(*controller*, *req*, *tenant_id*, ***kwargs*)

snapshot(*controller*, *req*, *tenant_id*, ***kwargs*)

template(*controller*, *req*, *tenant_id*, ***kwargs*)

Get the template body for an existing stack.

update(*controller*, *req*, *tenant_id*, ***kwargs*)

Update an existing stack with a new template and/or parameters.

update_patch(*controller*, *req*, *tenant_id*, ***kwargs*)

Update an existing stack with a new template.

Update an existing stack with a new template by patching the parameters Add the flag patch to the args so the engine code can distinguish

validate_template(*controller*, *req*, *tenant_id*, ***kwargs*)

Implements the ValidateTemplate API action.

Validates the specified template.

class **heat.api.openstack.v1.stacks.StackSerializer**

Bases: `heat.common.serializers.JSONResponseSerializer`

Handles serialization of specific controller method responses.

create(*response*, *result*)

heat.api.openstack.v1.stacks.create_resource(*options*)

Stacks resource factory method.

The heat.api.openstack.v1.util Module

heat.api.openstack.v1.util.get_allowed_params(*params*, *whitelist*)

Extract from `params` all entries listed in `whitelist`.

The returning dict will contain an entry for a key if, and only if, there's an entry in

`whitelist` for that key and at least one entry in `params`. If `params` contains multiple entries for the same key, it will yield an array of values: `{key: [v1, v2,...]}`

Parameters:
- **params** – a NestedMultiDict from webob.Request.params
- **whitelist** – an array of strings to whitelist

Returns: a dict with {key: value} pairs

heat.api.openstack.v1.util.identified_stack(*handler*)

Decorator that passes a stack identifier instead of path components.

This is a handler method decorator.

heat.api.openstack.v1.util.make_link(*req*, *identity*, *relationship='self'*)

Return a link structure for the supplied identity dictionary.

heat.api.openstack.v1.util.make_url(*req*, *identity*)

Return the URL for the supplied identity dictionary.

heat.api.openstack.v1.util.policy_enforce(*handler*)

Decorator that enforces policies.

Checks the path matches the request context and enforce policy defined in policy.json.

This is a handler method decorator.

The heat.api.openstack.v1.views.stacks_view Module

heat.api.openstack.v1.views.stacks_view.collection(*req*, *stacks*, *count=None*, *tenant_safe=True*)

heat.api.openstack.v1.views.stacks_view.format_stack(*req*, *stack*, *keys=None*, *tenant_safe=True*)

The heat.api.openstack.v1.views.views_common Module

heat.api.openstack.v1.views.views_common.get_collection_links(*request*, *items*)

Retrieve 'next' link, if applicable.

The heat.api.openstack.versions Module

Controller that returns information on the heat API versions.

Now it's a subclass of module versions, because of identity with cfn module versions. It can be changed, if there will be another API version.

The heat.api.versions Module

Controller that returns information on the heat API versions.

class **heat.api.versions.Controller**(*conf*)

Bases: `object`

A controller that produces information on the heat API versions.

get_href(*req*)

The heat.engine.api Module

heat.engine.api.build_resource_description(*docstring*)

heat.engine.api.extract_args(*params*)

Extract arguments passed as parameters and return them as a dictionary.

Extract any arguments passed as parameters through the API and return them as a dictionary. This allows us to filter the passed args and do type conversion where appropriate

heat.engine.api.format_event(*event*, *stack_identifier*, *root_stack_identifier=None*)

heat.engine.api.format_notification_body(*stack*)

heat.engine.api.format_resource_attributes(*resource*, *with_attr=None*)

heat.engine.api.format_resource_properties(*resource*)

heat.engine.api.format_snapshot(*snapshot*)

heat.engine.api.format_software_config(*sc*, *detail=True*)

heat.engine.api.format_software_deployment(*sd*)

heat.engine.api.format_stack(*stack*, *preview=False*, *resolve_outputs=True*)

Return a representation of the given stack.

Return a representation of the given stack that matches the API output expectations.

heat.engine.api.format_stack_db_object(*stack*)

Return a summary representation of the given stack.

Given a stack versioned db object, return a representation of the given stack for a stack listing.

heat.engine.api.format_stack_output(*stack*, *outputs*, *k*, *resolve_value=True*)

heat.engine.api.format_stack_outputs(*stack*, *outputs*,

***resolve_value=False*)**

Return a representation of the given output template.

Return a representation of the given output template for the given stack that matches the API output expectations.

heat.engine.api.format_stack_preview(*stack*)

heat.engine.api.format_stack_resource(*resource*, *detail=True*, *with_props=False*, *with_attr=None*)

Return a representation of the given resource.

Return a representation of the given resource that matches the API output expectations.

heat.engine.api.format_validate_parameter(*param*)

Format a template parameter for validate template API call.

Formats a template parameter and its schema information from the engine's internal representation (i.e. a Parameter object and its associated Schema object) to a representation expected by the current API (for example to be compatible to CFN syntax).

heat.engine.api.format_watch(*watch*)

heat.engine.api.format_watch_data(*wd*, *rule_names*)

heat.engine.api.translate_filters(*params*)

Translate filter names to their corresponding DB field names.

Parameters: **params** – A dictionary containing keys from engine.api.STACK_KEYS and other keys previously leaked to users.

Returns: A dict containing only valid DB filed names.

The heat.engine.attributes Module

class **heat.engine.attributes.Attribute(*attr_name*, *schema*)**

Bases: object

An Attribute schema.

as_output(*resource_name*, *template_type*='*cfn*')

Output entry for a provider template with the given resource name.

Parameters:
- **resource_name** – the logical name of the provider resource
- **template_type** – the template type to generate

Returns: This attribute as a template 'Output' entry for cfn template and 'output' entry for hot template

support_status()

class **heat.engine.attributes.Attributes(*res_name*, *schema*, *resolver*)**

Bases: _abcoll.Mapping

Models a collection of Resource Attributes.

static **as_outputs(*resource_name*, *resource_class*, *template_type*='*cfn*')**

Dict of Output entries for a provider template with resource name.

Parameters:
- **resource_name** – logical name of the resource
- **resource_class** – resource implementation class

Returns: The attributes of the specified resource_class as a template Output map

reset_resolved_values()

static **schema_from_outputs(*json_snippet*)**

class **heat.engine.attributes.DynamicSchemeAttributes(*res_name*, *schema*, *resolver*)**

Bases: heat.engine.attributes.Attributes

The collection of attributes for resources without static attr scheme.

The class defines collection of attributes for such entities as Resource Group, Software Deployment and so on that doesn't have static attribute scheme. The attribute scheme for such kind of resources can contain attribute from attribute scheme (like other resources) and dynamic attributes (nested stack attrs or API response attrs).

***class* heat.engine.attributes.Schema(*description=None*, *support_status=<heat.engine.support.SupportStatus object at 0x7f507f82c710>*, *cache_mode='cache_local'*, *type=None*)**

Bases: heat.engine.constraints.Schema

Simple schema class for attributes.

Schema objects are serializable to dictionaries following a superset of the HOT input Parameter schema using dict().

BOOLEAN = '*Boolean*'
CACHE_LOCAL = '*cache_local*'
CACHE_MODES = (*'cache_local'*, *'cache_none'*)
CACHE_NONE = '*cache_none*'
DESCRIPTION = '*description*'
INTEGER = '*Integer*'
KEYS = (*'description'*, *'type'*)
LIST = '*List*'
MAP = '*Map*'
STRING = '*String*'
TYPE = '*type*'
TYPES = (*'String'*, *'Map'*, *'List'*, *'Integer'*, *'Boolean'*)

***classmethod* from_attribute(*schema_dict*)**

Return a Property Schema corresponding to a Attribute Schema.

heat.engine.attributes.schemata(*schema*)

Return dictionary of Schema objects for given dictionary of schemata.

heat.engine.attributes.select_from_attribute(*attribute_value*, *path*)

Select an element from an attribute value.

Parameters:
- **attribute_value** – the attribute value.
- **path** – a list of path components to select from the attribute.

Returns: the selected attribute component value.

The heat.engine.cfn.functions Module

***class* heat.engine.cfn.functions.Base64(*stack*, *fn_name*, *args*)**

Bases: heat.engine.function.Function

A placeholder function for converting to base64.

Takes the form:

```
{ "Fn::Base64" : "<string>" }
```

This function actually performs no conversion. It is included for the benefit of templates that convert UserData to Base64. Heat accepts UserData in plain text.

result()

***class* heat.engine.cfn.functions.FindInMap(*stack*, *fn_name*, *args*)**

Bases: heat.engine.function.Function

A function for resolving keys in the template mappings.

Takes the form:

```
{ "Fn::FindInMap" : [ "mapping",
                      "key",
                      "value" ] }
```

result()

***class* heat.engine.cfn.functions.GetAZs(*stack*, *fn_name*, *args*)**

Bases: heat.engine.function.Function

A function for retrieving the availability zones.

Takes the form:

```
{ "Fn::GetAZs" : "<region>" }
```

result()

class **heat.engine.cfn.functions.GetAtt**(***stack***, ***fn_name***, ***args***)

Bases: heat.engine.function.Function

A function for resolving resource attributes.

Takes the form:

```
{ "Fn::GetAtt" : [ "<resource_name>",
                   "<attribute_name" ] }
```

dep_attrs(***resource_name***)

dependencies(***path***)

result()

validate()

class **heat.engine.cfn.functions.Join**(***stack***, ***fn_name***, ***args***)

Bases: heat.engine.function.Function

A function for joining strings.

Takes the form:

```
{ "Fn::Join" : [ "<delim>", [ "<string_1>", "<string_2>", ... ] ] }
```

And resolves to:

```
"<string_1><delim><string_2><delim>..."
```

result()

class **heat.engine.cfn.functions.MemberListToMap**(***stack***, ***fn_name***, ***args***)

Bases: heat.engine.function.Function

A function to convert lists with enumerated keys and values to mapping.

Takes the form:

```
{ 'Fn::MemberListToMap' : [ 'Name',
                            'Value',
                            [ '.member.0.Name=<key_0>',
                              '.member.0.Value=<value_0>',
                              ... ] ] }
```

And resolves to:

```
{ "<key_0>" : "<value_0>", ... }
```

The first two arguments are the names of the key and value.

result()

class **heat.engine.cfn.functions.ParamRef**(***stack***, ***fn_name***, ***args***)

Bases: heat.engine.function.Function

A function for resolving parameter references.

Takes the form:

```
{ "Ref" : "<param_name>" }
```

result()

heat.engine.cfn.functions.Ref(***stack***, ***fn_name***, ***args***)

A function for resolving parameters or resource references.

Takes the form:

```
{ "Ref" : "<param_name>" }
```

or:

```
{ "Ref" : "<resource_name>" }
```

class **heat.engine.cfn.functions.Replace**(***stack***, ***fn_name***, ***args***)

Bases: heat.engine.function.Function

A function for performing string substitutions.

Takes the form:

```
{ "Fn::Replace" : [
    { "<key_1>": "<value_1>", "<key_2>": "<value_2>", ... },
    "<key_1> <key_2>"
  ] }
```

And resolves to:

```
"<value_1> <value_2>"
```

This is implemented using python str.replace on each key. The order in which replacements are performed is undefined.

result()

class **heat.engine.cfn.functions.ResourceFacade**(***stack***, ***fn_name***, ***args***)

Bases: heat.engine.function.Function

A function for retrieving data in a parent provider template.

A function for obtaining data from the facade resource from within the corresponding provider template.

Takes the form:

```
{ "Fn::ResourceFacade": "<attribute_type>" }
```

where the valid attribute types are “Metadata”, “DeletionPolicy” and “UpdatePolicy”.

DELETION_POLICY = '***DeletionPolicy***'

METADATA = '***Metadata***'

UPDATE_POLICY = '***UpdatePolicy***'

result()

class **heat.engine.cfn.functions.ResourceRef**(***stack***, ***fn_name***, ***args***)

Bases: heat.engine.function.Function

A function for resolving resource references.

Takes the form:

```
{ "Ref" : "<resource_name>" }
```

dependencies(*path*)

result()

class **heat.engine.cfn.functions.Select**(*stack*, *fn_name*, *args*)

Bases: heat.engine.function.Function

A function for selecting an item from a list or map.

Takes the form (for a list lookup):

```
{ "Fn::Select" : [ "<index>", [ "<value_1>", "<value_2>", ... ] ] }
```

Takes the form (for a map lookup):

```
{ "Fn::Select" : [ "<index>", { "<key_1>": "<value_1>", ... } ] }
```

If the selected index is not found, this function resolves to an empty string.

result()

class **heat.engine.cfn.functions.Split**(*stack*, *fn_name*, *args*)

Bases: heat.engine.function.Function

A function for splitting strings.

Takes the form:

```
{ "Fn::Split" : [ "<delim>", "<string_1><delim><string_2>..." ] }
```

And resolves to:

```
[ "<string_1>", "<string_2>", ... ]
```

result()

The heat.engine.cfn.template Module

class **heat.engine.cfn.template.CfnTemplate**(*tmpl*, *template_id=None*,

files=None, env=None)

Bases: heat.engine.cfn.template.CfnTemplateBase

CONDITIONS = '*Conditions*'
SECTIONS = (*'AWSTemplateFormatVersion'*, *'HeatTemplateFormatVersion'*, *'Description'*, *'Mappings'*, *'Parameters'*, *'Resources'*, *'Outputs'*, *'Conditions'*)
condition_functions = {*'Fn::Equals': <class 'heat.engine.hot.functions.Equals'>*, *'Ref': <class 'heat.engine.cfn.functions.ParamRef'>*, *'Fn::FindInMap': <class 'heat.engine.cfn.functions.FindInMap'>*}

class heat.engine.cfn.template.CfnTemplateBase(*template*, *template_id=None*, *files=None*, *env=None*)

Bases: heat.engine.template.Template

The base implementation of cfn template.

ALTERNATE_VERSION = '*HeatTemplateFormatVersion*'
DESCRIPTION = '*Description*'
MAPPINGS = '*Mappings*'
OUTPUTS = '*Outputs*'
OUTPUT_DESCRIPTION = '*Description*'
OUTPUT_KEYS = (*'Description'*, *'Value'*)
OUTPUT_VALUE = '*Value*'
PARAMETERS = '*Parameters*'
RESOURCES = '*Resources*'
RES_DELETION_POLICY = '*DeletionPolicy*'
RES_DEPENDS_ON = '*DependsOn*'
RES_DESCRIPTION = '*Description*'
RES_METADATA = '*Metadata*'
RES_PROPERTIES = '*Properties*'
RES_TYPE = '*Type*'
RES_UPDATE_POLICY = '*UpdatePolicy*'
SECTIONS = (*'AWSTemplateFormatVersion'*, *'HeatTemplateFormatVersion'*, *'Description'*, *'Mappings'*, *'Parameters'*, *'Resources'*, *'Outputs'*)
SECTIONS_NO_DIRECT_ACCESS = *set(['AWSTemplateFormatVersion'*, *'HeatTemplateFormatVersion'*, *'Parameters'])*
VERSION = '*AWSTemplateFormatVersion*'

add_resource(*definition*, *name=None*)

deletion_policies = {'Snapshot': 'Snapshot', 'Retain': 'Retain', 'Delete': 'Delete'}

functions = {'Fn::GetAtt': <class 'heat.engine.cfn.functions.GetAtt'>, 'Fn::FindInMap': <class 'heat.engine.cfn.functions.FindInMap'>, 'Ref': <function Ref at 0x7f507f538b90>, 'Fn::Base64': <class 'heat.engine.cfn.functions.Base64'>, 'Fn::Select': <class 'heat.engine.cfn.functions.Select'>, 'Fn::Join': <class 'heat.engine.cfn.functions.Join'>, 'Fn::GetAZs': <class 'heat.engine.cfn.functions.GetAZs'>}

get_section_name(*section*)

param_schemata(*param_defaults=None*)

parameters(*stack_identifier*, *user_params*, *param_defaults=None*)

resource_definitions(*stack*)

validate_resource_definitions(*stack*)

class heat.engine.cfn.template.HeatTemplate(*template*, *template_id=None*, *files=None*, *env=None*)

Bases: heat.engine.cfn.template.CfnTemplateBase

functions = {'Fn::GetAtt': <class 'heat.engine.cfn.functions.GetAtt'>, 'Ref': <function Ref at 0x7f507f538b90>, 'Fn::Replace': <class 'heat.engine.cfn.functions.Replace'>, 'Fn::MemberListToMap': <class 'heat.engine.cfn.functions.MemberListToMap'>, 'Fn::FindInMap': <class 'heat.engine.cfn.functions.FindInMap'>, 'Fn::Base64': <class 'heat.engine.cfn.functions.Base64'>, 'Fn::ResourceFacade': <class 'heat.engine.cfn.functions.ResourceFacade'>, 'Fn::Split': <class 'heat.engine.cfn.functions.Split'>, 'Fn::Select': <class 'heat.engine.cfn.functions.Select'>, 'Fn::Join': <class 'heat.engine.cfn.functions.Join'>, 'Fn::GetAZs': <class 'heat.engine.cfn.functions.GetAZs'>}

The heat.engine.check_resource Module

***class* heat.engine.check_resource.CheckResource(*engine_id*, *rpc_client*, *thread_group_mgr*)**

Bases: object

check(*cnxt*, *resource_id*, *current_traversal*, *resource_data*, *is_update*, *adopt_stack_data*, *rsrc*, *stack*)

Process a node in the dependency graph.

The node may be associated with either an update or a cleanup of its associated resource.

heat.engine.check_resource.check_resource_cleanup(*rsrc*, *template_id*, *resource_data*, *engine_id*, *timeout*)

Delete the Resource if appropriate.

heat.engine.check_resource.check_resource_update(*rsrc*, *template_id*, *resource_data*, *engine_id*, *stack*)

Create or update the Resource if appropriate.

heat.engine.check_resource.check_stack_complete(*cnxt*, *stack*, *current_traversal*, *sender_id*, *deps*, *is_update*)

Mark the stack complete if the update is complete.

Complete is currently in the sense that all desired resources are in service, not that superfluous ones have been cleaned up.

heat.engine.check_resource.construct_input_data(*rsrc*, *curr_stack*)

heat.engine.check_resource.load_resource(*cnxt*, *resource_id*, *resource_data*, *is_update*)

heat.engine.check_resource.propagate_check_resource(*cnxt*, *rpc_client*, *next_res_id*, *current_traversal*, *predecessors*, *sender_key*, *sender_data*, *is_update*, *adopt_stack_data*)

Trigger processing of node if all of its dependencies are satisfied.

The heat.engine.clients.client_plugin Module

class **heat.engine.clients.client_plugin.ClientPlugin**(*context*)

Bases: object

client(*version=None*)

clients

context

default_version = *None*

does_endpoint_exist(*service_type*, *service_name*)

exceptions_module = *None*

ignore_conflict_and_not_found

Raises the exception unless it is a conflict or not-found.

ignore_not_found

Raises the exception unless it is a not-found.

invalidate()

Invalidate/clear any cached client.

is_client_exception(*ex*)

Returns True if the current exception comes from the client.

is_conflict(*ex*)

Returns True if the exception is a conflict.

is_not_found(*ex*)

Returns True if the exception is a not-found.

is_over_limit(*ex*)

Returns True if the exception is an over-limit.

service_types = *[]*

supported_versions = *[]*

url_for(***kwargs*)

heat.engine.clients.client_plugin.retry_if_connection_err(*exception*)

heat.engine.clients.client_plugin.retry_if_result_is_false(*result*)

The heat.engine.clients.os.aodh Module

class **heat.engine.clients.os.aodh.AodhClientPlugin**(*context*)

Bases: heat.engine.clients.client_plugin.ClientPlugin

ALARMING = '*alarming*'

V2 = '*2*'

default_version = '*2*'

exceptions_module = *<module* '*aodhclient.exceptions*' *from* '*/home/jenkins/workspace/heat-docs-ubuntu-xenial/.tox/venv/local/lib/python2.7/site-packages/aodhclient/exceptions.pyc*'*>*

is_conflict(*ex*)

is_not_found(*ex*)

is_over_limit(*ex*)

service_types = ['*alarming*']

supported_versions = ['*2*']

The heat.engine.clients.os.barbican Module

class heat.engine.clients.os.barbican.BarbicanClientPlugin(*context*)

Bases: heat.engine.clients.client_plugin.ClientPlugin

KEY_MANAGER = '*key-manager*'

create_certificate(***props*)

create_generic_container(***props*)

create_rsa(***props*)

get_container_by_ref(*container_ref*)

get_secret_by_ref(*secret_ref*)

is_not_found(*ex*)

service_types = ['*key-manager*']

class heat.engine.clients.os.barbican.ContainerConstraint

Bases: heat.engine.constraints.BaseCustomConstraint

expected_exceptions = (<*class* '*heat.common.exception.EntityNotFound*'>,)

resource_client_name = '*barbican*'

resource_getter_name = '*get_container_by_ref*'

class heat.engine.clients.os.barbican.SecretConstraint

Bases: heat.engine.constraints.BaseCustomConstraint

expected_exceptions = (<*class* '*heat.common.exception.EntityNotFound*'>,)

resource_client_name = '*barbican*'

resource_getter_name = '*get_secret_by_ref*'

The heat.engine.clients.os.ceilometer Module

***class* heat.engine.clients.os.ceilometer.CeilometerClientPlugin(*context*)**

Bases: heat.engine.clients.client_plugin.ClientPlugin

ALARMING = '*alarming*'

METERING = '*metering*'

exceptions_module = *[<module 'ceilometerclient.exc' from '/home/jenkins/workspace/heat-docs-ubuntu-xenial/.tox/venv/local/lib/python2.7/site-packages/ceilometerclient/exc.pyc'>, <module 'ceilometerclient.openstack.common.apiclient.exceptions' from '/home/jenkins/workspace/heat-docs-ubuntu-xenial/.tox/venv/local/lib/python2.7/site-packages/ceilometerclient/openstack/common/apiclient/exceptions.pyc'>]*

is_conflict(*ex*)

is_not_found(*ex*)

is_over_limit(*ex*)

service_types = *['metering', 'alarming']*

The heat.engine.clients.os.cinder Module

***class* heat.engine.clients.os.cinder.BaseCinderConstraint**

Bases: heat.engine.constraints.BaseCustomConstraint

resource_client_name = '*cinder*'

***class* heat.engine.clients.os.cinder.CinderClientPlugin(*context*)**

Bases: heat.engine.clients.client_plugin.ClientPlugin

VOLUME_V2 = '*volumev2*'

VOLUME_V3 = '*volumev3*'

check_attach_volume_complete(*vol_id*)

check_detach_volume_complete(*vol_id*)

exceptions_module = <*module* '*cinderclient.exceptions*' *from* '*/home/jenkins/workspace/heat-docs-ubuntu-xenial/.tox/venv/local/lib/python2.7/site-packages/cinderclient/exceptions.pyc*'>

get_volume(*volume*)

get_volume_api_version()

Returns the most recent API version.

get_volume_backup(*backup*)

get_volume_snapshot(*snapshot*)

get_volume_type(*volume_type*)

has_extension(*alias*)

Check if specific extension is present.

is_conflict(*ex*)

is_not_found(*ex*)

is_over_limit(*ex*)

service_types = ['*volumev2*', '*volumev3*']

class heat.engine.clients.os.cinder.VolumeBackupConstraint

Bases: heat.engine.clients.os.cinder.BaseCinderConstraint

resource_getter_name = '*get_volume_backup*'

class heat.engine.clients.os.cinder.VolumeConstraint

Bases: heat.engine.clients.os.cinder.BaseCinderConstraint

resource_getter_name = '*get_volume*'

class heat.engine.clients.os.cinder.VolumeSnapshotConstraint

Bases: heat.engine.clients.os.cinder.BaseCinderConstraint

resource_getter_name = '*get_volume_snapshot*'

class heat.engine.clients.os.cinder.VolumeTypeConstraint

Bases: heat.engine.clients.os.cinder.BaseCinderConstraint

resource_getter_name = '*get_volume_type*'

The heat.engine.clients.os.designate Module

class heat.engine.clients.os.designate.DesignateClientPlugin(*context*)

Bases: heat.engine.clients.client_plugin.ClientPlugin

DNS = '*dns*'

domain_create(***kwargs*)

domain_update(***kwargs*)

exceptions_module = [<*module* '*designateclient.exceptions*' *from* '*/home/jenkins/workspace/heat-docs-ubuntu-*

xenial/.tox/venv/local/lib/python2.7/site-packages/designateclient/exceptions.pyc'>]

get_domain_id(*domain_id_or_name*)

is_not_found(*ex*)

record_create(***kwargs*)

record_delete(***kwargs*)

record_show(***kwargs*)

record_update(***kwargs*)

service_types = *['dns']*

class heat.engine.clients.os.designate.DesignateDomainConstraint

Bases: heat.engine.constraints.BaseCustomConstraint

resource_client_name = '*designate*'

resource_getter_name = '*get_domain_id*'

The heat.engine.clients.os.glance Module

class heat.engine.clients.os.glance.GlanceClientPlugin(*context*)

Bases: heat.engine.clients.client_plugin.ClientPlugin

IMAGE = '*image*'

V1 = '*1*'

V2 = '*2*'

default_version = '*1*'

exceptions_module = *[<module 'glanceclient.openstack.common.apiclient.exceptions' from '/home/jenkins/workspace/heat-docs-ubuntu-xenial/.tox/venv/local/lib/python2.7/site-packages/glanceclient/openstack/common/apiclient/exceptions.pyc'>, <module 'glanceclient.exc' from '/home/jenkins/workspace/heat-docs-ubuntu-xenial/.tox/venv/local/lib/python2.7/site-*

packages/glanceclient/exc.pyc'>]

find_image_by_name_or_id(*image_identifier*)

Return the ID for the specified image name or identifier.

Parameters: **image_identifier** – image name or a UUID-like identifier

Returns: the id of the requested :image_identifier:

get_image(*image_identifier*)

Return the image object for the specified image name/id.

Parameters: **image_identifier** – image name

Returns: an image object with name/id :image_identifier:

is_conflict(*ex*)

is_not_found(*ex*)

is_over_limit(*ex*)

service_types = *['image']*

supported_versions = *['1', '2']*

class **heat.engine.clients.os.glance.ImageConstraint**

Bases: heat.engine.constraints.BaseCustomConstraint

expected_exceptions = *(<class 'glanceclient.openstack.common.apiclient.exceptions.NotFound'>, <class 'glanceclient.openstack.common.apiclient.exceptions.NoUniqueMatch'>)*

resource_client_name = '*glance*'

resource_getter_name = '*find_image_by_name_or_id*'

The heat.engine.clients.os.heat_plugin Module

class **heat.engine.clients.os.heat_plugin.HeatClientPlugin(*context*)**

Bases: heat.engine.clients.client_plugin.ClientPlugin

CLOUDFORMATION = '*cloudformation*'

ORCHESTRATION = '*orchestration*'

exceptions_module = *<module 'heatclient.exc' from '/home/jenkins/workspace/heat-docs-ubuntu-xenial/.tox/venv/local/lib/python2.7/site-packages/heatclient/exc.pyc'>*

get_cfn_metadata_server_url()

get_heat_cfn_url()

get_heat_url()

get_insecure_option()

get_watch_server_url()

is_conflict(*ex*)

is_not_found(*ex*)

is_over_limit(*ex*)

service_types = *['orchestration', 'cloudformation']*

The heat.engine.clients.os.keystone.heat_keystoneclient Module

Keystone Client functionality for use by resources.

class **heat.engine.clients.os.keystone.heat_keystoneclient.AccessKey**

Bases: `tuple`

AccessKey(id, access, secret)

access

Alias for field number 1

id

Alias for field number 0

secret

Alias for field number 2

***class* heat.engine.clients.os.keystone.heat_keystoneclient.KeystoneClient**

Bases: `object`

Keystone Auth Client.

Delay choosing the backend client module until the client's class needs to be initialized.

***class* heat.engine.clients.os.keystone.heat_keystoneclient.KsClientWrapper(*context*)**

Bases: `object`

Wrap keystone client so we can encapsulate logic used in resources.

Note this is intended to be initialized from a resource on a per-session basis, so the session context is passed in on initialization Also note that an instance of this is created in each request context as part of a lazy-loaded cloud backend and it can be easily referenced in each resource as `self.keystone()`, so there should not be any need to directly instantiate instances of this class inside resources themselves.

auth_ref

auth_token

client

context

create_ec2_keypair(*user_id=None*)

create_stack_domain_project(*stack_id*)

Create a project in the heat stack-user domain.

create_stack_domain_user(*username*, *project_id*, *password=None*)

Create a domain user defined as part of a stack.

The user is defined either via template or created internally by a resource. This user will be added to the heat_stack_user_role as defined in the config, and created in the specified project (which is expected to be in the stack_domain).

Returns the keystone ID of the resulting user.

create_stack_domain_user_keypair(*user_id*, *project_id*)

create_stack_user(*username*, *password=''*)

Create a user defined as part of a stack.

The user is defined either via template or created internally by a resource. This user will be added to the heat_stack_user_role as defined in the config.

Returns the keystone ID of the resulting user.

create_trust_context()

Create a trust using the trustor identity in the current context.

The trust is created with the trustee as the heat service user.

If the current context already contains a trust_id, we do nothing and return the current context.

Returns a context containing the new trust_id.

delete_ec2_keypair(*credential_id=None*, *access=None*, *user_id=None*)

Delete credential containing ec2 keypair.

delete_stack_domain_project(*project_id*)

delete_stack_domain_user(*user_id*, *project_id*)

delete_stack_domain_user_keypair(*user_id*, *project_id*, *credential_id*)

delete_stack_user(*user_id*)

delete_trust(*trust_id*)

Delete the specified trust.

disable_stack_domain_user(*user_id*, *project_id*)

disable_stack_user(*user_id*)

domain_admin_auth

domain_admin_client

enable_stack_domain_user(*user_id*, *project_id*)

enable_stack_user(*user_id*)

get_ec2_keypair(*credential_id=None*, *access=None*, *user_id=None*)

Get an ec2 keypair via v3/credentials, by id or access.

stack_domain

Domain scope data.

This is only used for checking for scoping data, not using the value.

stack_domain_id

stack_domain_user_token(*user_id*, *project_id*, *password*)

Get a token for a stack domain user.

url_for(***kwargs*)

heat.engine.clients.os.keystone.heat_keystoneclient.list_opts()

The heat.engine.clients.os.keystone.keystone_constraints Module

class **heat.engine.clients.os.keystone.keystone_constraints.KeystoneBaseConstraint**

Bases: heat.engine.constraints.BaseCustomConstraint

entity = *None*

resource_client_name = '*keystone*'

validate_with_client(*client*, *resource_id*)

class **heat.engine.clients.os.keystone.keystone_constraints.KeystoneDomainConstraint**

Bases: heat.engine.clients.os.keystone.keystone_constraints.KeystoneBaseConstraint

entity = '*KeystoneDomain*'

resource_getter_name = '*get_domain_id*'

class **heat.engine.clients.os.keystone.keystone_constraints.KeystoneGroupConstraint**

Bases: heat.engine.clients.os.keystone.keystone_constraints.KeystoneBaseConstraint

entity = '*KeystoneGroup*'

resource_getter_name = '*get_group_id*'

class **heat.engine.clients.os.keystone.keystone_constraints.KeystoneProjectCo**

nstraint

Bases: heat.engine.clients.os.keystone.keystone_constraints.KeystoneBaseConstraint

entity = '*KeystoneProject*'
resource_getter_name = '*get_project_id*'

***class* heat.engine.clients.os.keystone.keystone_constraints.KeystoneRegionConstraint**

Bases: heat.engine.clients.os.keystone.keystone_constraints.KeystoneBaseConstraint

entity = '*KeystoneRegion*'
resource_getter_name = '*get_region_id*'

***class* heat.engine.clients.os.keystone.keystone_constraints.KeystoneRoleConstraint**

Bases: heat.engine.clients.os.keystone.keystone_constraints.KeystoneBaseConstraint

entity = '*KeystoneRole*'
resource_getter_name = '*get_role_id*'

***class* heat.engine.clients.os.keystone.keystone_constraints.KeystoneServiceConstraint**

Bases: heat.engine.clients.os.keystone.keystone_constraints.KeystoneBaseConstraint

entity = '*KeystoneService*'
expected_exceptions = *(<class 'heat.common.exception.EntityNotFound'>, <class 'heat.common.exception.KeystoneServiceNameConflict'>)*
resource_getter_name = '*get_service_id*'

***class* heat.engine.clients.os.keystone.keystone_constraints.KeystoneUserConst**

raint

Bases: heat.engine.clients.os.keystone.keystone_constraints.KeystoneBaseConstraint

entity = '*KeystoneUser*'

resource_getter_name = '*get_user_id*'

The heat.engine.clients.os.magnum Module

***class* heat.engine.clients.os.magnum.BaymodelConstraint**

Bases: heat.engine.constraints.BaseCustomConstraint

resource_client_name = '*magnum*'

resource_getter_name = '*get_baymodel*'

***class* heat.engine.clients.os.magnum.MagnumClientPlugin(*context*)**

Bases: heat.engine.clients.client_plugin.ClientPlugin

CONTAINER = '*container-infra*'

get_baymodel(*value*)

is_conflict(*ex*)

is_not_found(*ex*)

is_over_limit(*ex*)

service_types = *['container-infra']*

The heat.engine.clients.os.manila Module

class heat.engine.clients.os.manila.ManilaClientPlugin(*context*)

Bases: heat.engine.clients.client_plugin.ClientPlugin

SHARE = '*share*'

exceptions_module = <*module* '*manilaclient.exceptions*' *from* '*/home/jenkins/workspace/heat-docs-ubuntu-xenial/.tox/venv/local/lib/python2.7/site-packages/manilaclient/exceptions.pyc*'>

get_security_service(*service_identity*)

get_share_network(*share_network_identity*)

get_share_snapshot(*snapshot_identity*)

get_share_type(*share_type_identity*)

is_conflict(*ex*)

is_not_found(*ex*)

is_over_limit(*ex*)

service_types = ['*share*']

class heat.engine.clients.os.manila.ManilaShareBaseConstraint

Bases: heat.engine.constraints.BaseCustomConstraint

expected_exceptions = (<*class* '*heat.common.exception.EntityNotFound*'>, <*class* '*manilaclient.openstack.common.apiclient.exceptions.NoUniqueMatch*'>)

resource_client_name = '*manila*'

class heat.engine.clients.os.manila.ManilaShareNetworkConstraint

Bases: heat.engine.clients.os.manila.ManilaShareBaseConstraint

resource_getter_name = '*get_share_network*'

class heat.engine.clients.os.manila.ManilaShareSnapshotConstraint

Bases: heat.engine.clients.os.manila.ManilaShareBaseConstraint

resource_getter_name = '*get_share_snapshot*'

class heat.engine.clients.os.manila.ManilaShareTypeConstraint

Bases: heat.engine.clients.os.manila.ManilaShareBaseConstraint

resource_getter_name = '*get_share_type*'

The heat.engine.clients.os.mistral Module

class heat.engine.clients.os.mistral.MistralClientPlugin(*context*)

Bases: heat.engine.clients.client_plugin.ClientPlugin

WORKFLOW_V2 = '*workflowv2*'

get_workflow_by_identifier(*workflow_identifier*)

is_conflict(*ex*)

is_not_found(*ex*)

is_over_limit(*ex*)

service_types = *['workflowv2']*

class heat.engine.clients.os.mistral.WorkflowConstraint

Bases: heat.engine.constraints.BaseCustomConstraint

expected_exceptions = *(<class 'heat.common.exception.EntityNotFound'>,)*

resource_client_name = '*mistral*'

resource_getter_name = '*get_workflow_by_identifier*'

The heat.engine.clients.os.monasca Module

class **heat.engine.clients.os.monasca.MonascaClientPlugin(*context*)**

Bases: heat.engine.clients.client_plugin.ClientPlugin

MONITORING = '*monitoring*'

VERSION = '*2_0*'

exceptions_module = *[<module 'monascaclient.exc' from '/home/jenkins/workspace/heat-docs-ubuntu-xenial/.tox/venv/local/lib/python2.7/site-packages/monascaclient/exc.pyc'>]*

get_notification(*notification*)

is_not_found(*ex*)

is_un_processable(*ex*)

service_types = *['monitoring']*

class **heat.engine.clients.os.monasca.MonascaNotificationConstraint**

Bases: heat.engine.constraints.BaseCustomConstraint

resource_client_name = '*monasca*'

resource_getter_name = '*get_notification*'

The heat.engine.clients.os.neutron.lbaas_constraints Module

class **heat.engine.clients.os.neutron.lbaas_constraints.LBaasV2ProviderConstraint**

Bases: heat.engine.clients.os.neutron.neutron_constraints.ProviderConstraint

service_type = '*LOADBALANCERV2*'

class **heat.engine.clients.os.neutron.lbaas_constraints.ListenerConstraint**

Bases: heat.engine.clients.os.neutron.neutron_constraints.NeutronConstraint

extension = '*lbaasv2*'

resource_name = '*listener*'

class **heat.engine.clients.os.neutron.lbaas_constraints.LoadbalancerConstraint**

Bases: heat.engine.clients.os.neutron.neutron_constraints.NeutronConstraint

cmd_resource = '*lbaas_loadbalancer*'

extension = '*lbaasv2*'

resource_name = '*loadbalancer*'

class **heat.engine.clients.os.neutron.lbaas_constraints.PoolConstraint**

Bases: heat.engine.clients.os.neutron.neutron_constraints.NeutronConstraint

cmd_resource = '*lbaas_pool*'

extension = '*lbaasv2*'

resource_name = '*pool*'

The heat.engine.clients.os.neutron.neutron_constraints Module

class **heat.engine.clients.os.neutron.neutron_constraints.AddressScopeConstraint**

Bases: heat.engine.clients.os.neutron.neutron_constraints.NeutronConstraint

extension = '*address-scope*'

resource_name = '*address_scope*'

class **heat.engine.clients.os.neutron.neutron_constraints.LBaasV1ProviderConstraint**

Bases: heat.engine.clients.os.neutron.neutron_constraints.ProviderConstraint

service_type = '*LOADBALANCER*'

class **heat.engine.clients.os.neutron.neutron_constraints.NetworkConstraint**

Bases: heat.engine.constraints.BaseCustomConstraint

expected_exceptions = *(<class 'neutronclient.common.exceptions.NeutronClientException'>, <class 'heat.common.exception.EntityNotFound'>, <class 'heat.common.exception.PhysicalResourceNameAmbiguity'>)*

validate_with_client(*client*, *value*)

class **heat.engine.clients.os.neutron.neutron_constraints.NeutronConstraint**

Bases: heat.engine.constraints.BaseCustomConstraint

cmd_resource = *None*

expected_exceptions = *(<class 'neutronclient.common.exceptions.NeutronClientException'>, <class 'heat.common.exception.EntityNotFound'>)*

extension = *None*

resource_name = *None*

validate_with_client(*client*, *value*)

class heat.engine.clients.os.neutron.neutron_constraints.PortConstraint

Bases: heat.engine.clients.os.neutron.neutron_constraints.NeutronConstraint

resource_name = '*port*'

class heat.engine.clients.os.neutron.neutron_constraints.ProviderConstraint

Bases: heat.engine.constraints.BaseCustomConstraint

expected_exceptions = *(<class 'heat.common.exception.StackValidationFailed'>,)*

service_type = *None*

validate_with_client(*client*, *value*)

class heat.engine.clients.os.neutron.neutron_constraints.QoSPolicyConstraint

Bases: heat.engine.clients.os.neutron.neutron_constraints.NeutronConstraint

cmd_resource = '*qos_policy*'

extension = '*qos*'

resource_name = '*policy*'

class heat.engine.clients.os.neutron.neutron_constraints.RouterConstraint

Bases: heat.engine.clients.os.neutron.neutron_constraints.NeutronConstraint

resource_name = '*router*'

class heat.engine.clients.os.neutron.neutron_constraints.SecurityGroupConstraint

Bases:

heat.engine.clients.os.neutron.neutron_constraints.NeutronConstraint

resource_name = '*security_group*'

class **heat.engine.clients.os.neutron.neutron_constraints.SubnetConstraint**

Bases: heat.engine.clients.os.neutron.neutron_constraints.NeutronConstraint

resource_name = '*subnet*'

class **heat.engine.clients.os.neutron.neutron_constraints.SubnetPoolConstraint**

Bases: heat.engine.clients.os.neutron.neutron_constraints.NeutronConstraint

resource_name = '*subnetpool*'

The heat.engine.clients.os.nova Module

class **heat.engine.clients.os.nova.FlavorConstraint**

Bases: heat.engine.clients.os.nova.NovaBaseConstraint

expected_exceptions = *(<class 'novaclient.exceptions.NotFound'>,)*

resource_getter_name = '*find_flavor_by_name_or_id*'

class **heat.engine.clients.os.nova.HostConstraint**

Bases: heat.engine.clients.os.nova.NovaBaseConstraint

resource_getter_name = '*get_host*'

class **heat.engine.clients.os.nova.KeypairConstraint**

Bases: heat.engine.clients.os.nova.NovaBaseConstraint

resource_getter_name = '*get_keypair*'

validate_with_client(*client*, *key_name*)

class heat.engine.clients.os.nova.NetworkConstraint

Bases: heat.engine.clients.os.nova.NovaBaseConstraint

expected_exceptions = (<*class* '*heat.common.exception.EntityNotFound*'>, <*class* '*heat.common.exception.PhysicalResourceNameAmbiguity*'>)

resource_getter_name = '*get_nova_network_id*'

class heat.engine.clients.os.nova.NovaBaseConstraint

Bases: heat.engine.constraints.BaseCustomConstraint

resource_client_name = '*nova*'

class heat.engine.clients.os.nova.NovaClientPlugin(*context*)

Bases: heat.engine.clients.client_plugin.ClientPlugin

COMPUTE = '*compute*'

absolute_limits(**args*, ***kw*)

Return the absolute limits as a dictionary.

attach_volume(*server_id*, *volume_id*, *device*)

build_userdata(*metadata*, *userdata=None*, *instance_user=None*, *user_data_format='HEAT_CFNTOOLS'*)

Build multipart data blob for CloudInit.

Data blob includes user-supplied Metadata, user data, and the required Heat in-instance configuration.

Parameters:
- **resource** (*heat.engine.Resource*) – the resource implementation
- **userdata** (*str or None*) – user data string
- **instance_user** (*string*) – the user to create on the server
- **user_data_format** (*string*) – Format of user data to return

Returns: multipart mime as a string

check_delete_server_complete(*server_id*)

Wait for server to disappear from Nova.

check_detach_volume_complete(*server_id*, *attach_id*)

Check that nova server lost attachment.

This check is needed for immediate reattachment when updating: there might be some time between cinder marking volume as 'available' and nova removing attachment from its own objects, so we check that nova already knows that the volume is detached.

check_interface_attach(**args*, ***kw*)

check_interface_detach(**args*, ***kw*)

check_rebuild(*server_id*)

Verify that a rebuilding server is rebuilt.

Raise error if it ends up in an ERROR state.

check_resize(*server_id*, *flavor*)

Verify that a resizing server is properly resized.

If that's the case, confirm the resize, if not raise an error.

check_verify_resize(*server_id*)

deferred_server_statuses = ['*BUILD*', '*HARD_REBOOT*', '*PASSWORD*', '*REBOOT*', '*RESCUE*', '*RESIZE*', '*REVERT_RESIZE*', '*SHUTOFF*', '*SUSPENDED*', '*VERIFY_RESIZE*']

detach_volume(*server_id*, *attach_id*)

exceptions_module = <*module* '*novaclient.exceptions*' *from* '*/home/jenkins/workspace/heat-docs-ubuntu-xenial/.tox/venv/local/lib/python2.7/site-packages/novaclient/exceptions.pyc*'>

fetch_server(*server_id*)

Fetch fresh server object from Nova.

Log warnings and return None for non-critical API errors. Use this method in various `check_*_complete` resource methods, where intermittent errors can be tolerated.

find_flavor_by_name_or_id(*flavor*)

Find the specified flavor by name or id.

Parameters: **flavor** – the name of the flavor to find

Returns: the id of :flavor:

get_console_urls(*server*)

Return dict-like structure of server's console urls.

The actual console url is lazily resolved on access.

get_flavor(*flavor_identifier*)

Get the flavor object for the specified flavor name or id.

Parameters: **flavor_identifier** – the name or id of the flavor to find

Returns: a flavor object with name or id :flavor:

get_host(*host_name*)

Get the host id specified by name.

Parameters: **host_name** – the name of host to find

Returns: the list of match hosts

Raises: exception.EntityNotFound

get_ip(*server*, *net_type*, *ip_version*)

Return the server's IP of the given type and version.

get_keypair(*key_name*)

Get the public key specified by :key_name:

Parameters: **key_name** – the name of the key to look for

Returns: the keypair (name, public_key) for :key_name:

Raises: exception.EntityNotFound

get_net_id_by_label(*label*)

get_nova_network_id(*net_identifier*)

get_server(**args*, ***kw*)

Return fresh server object.

Substitutes Nova's NotFound for Heat's EntityNotFound, to be returned to user as HTTP error.

get_status(*server*)

Return the server's status.

Parameters: **server** – server object

Returns: status as a string

has_extension(*alias*)

Check if specific extension is present.

interface_attach(*server_id*, *port_id=None*, *net_id=None*, *fip=None*)

interface_detach(*server_id*, *port_id*)

is_bad_request(*ex*)

is_conflict(*ex*)

is_not_found(*ex*)

is_over_limit(*ex*)

is_unprocessable_entity(*ex*)

meta_serialize(*metadata*)

Serialize non-string metadata values before sending them to Nova.

meta_update(*server*, *metadata*)

Delete/Add the metadata in nova as needed.

rebuild(*server_id*, *image_id*, *password=None*, *preserve_ephemeral=False*)

Rebuild the server and call check_rebuild to verify.

refresh_server(*server*)

Refresh server's attributes.

Also log warnings for non-critical API errors.

rename(*server*, *name*)

Update the name for a server.

resize(*server_id*, *flavor_id*)

Resize the server.

server_to_ipaddress(*server*)

Return the server's IP address, fetching it from Nova.

service_types = *['compute']*

verify_resize(*server_id*)

class **heat.engine.clients.os.nova.ServerConstraint**

Bases: heat.engine.clients.os.nova.NovaBaseConstraint

resource_getter_name = '*get_server*'

The heat.engine.clients.os.sahara Module

class **heat.engine.clients.os.sahara.ImageConstraint**

Bases: heat.engine.clients.os.sahara.SaharaBaseConstraint

expected_exceptions = *(<class 'heat.common.exception.EntityNotFound'>, <class 'heat.common.exception.PhysicalResourceNameAmbiguity'>)*

resource_getter_name = '*get_image_id*'

class **heat.engine.clients.os.sahara.PluginConstraint**

Bases: heat.engine.clients.os.sahara.SaharaBaseConstraint

resource_getter_name = '*get_plugin_id*'

***class* heat.engine.clients.os.sahara.SaharaBaseConstraint**

Bases: heat.engine.constraints.BaseCustomConstraint

resource_client_name = '*sahara*'

***class* heat.engine.clients.os.sahara.SaharaClientPlugin(*context*)**

Bases: heat.engine.clients.client_plugin.ClientPlugin

DATA_PROCESSING = '*data-processing*'

exceptions_module = <*module* '*saharaclient.api.base*' *from* '*/home/jenkins/workspace/heat-docs-ubuntu-xenial/.tox/venv/local/lib/python2.7/site-packages/saharaclient/api/base.pyc*'>

get_image_id(*image_identifier*)

Return the ID for the specified image name or identifier.

Parameters: **image_identifier** – image name or a UUID-like identifier

Returns: the id of the requested :image_identifier:

Raises: exception.EntityNotFound, exception.PhysicalResourceNameAmbiguity

get_image_id_by_name(*image_identifier*)

Return the ID for the specified image name.

Parameters: **image_identifier** – image name

Returns: the id of the requested :image_identifier:

Raises: exception.EntityNotFound, exception.PhysicalResourceNameAmbiguity

get_plugin_id(*plugin_name*)

Get the id for the specified plugin name.

Parameters: **plugin_name** – the name of the plugin to find

Returns: the id of :plugin:

Raises: exception.EntityNotFound

is_conflict(*ex*)

is_not_found(*ex*)

is_not_registered(*ex*)

is_over_limit(*ex*)

service_types = *['data-processing']*

validate_hadoop_version(*plugin_name*, *hadoop_version*)

The heat.engine.clients.os.senlin Module

class heat.engine.clients.os.senlin.ClusterConstraint

Bases: heat.engine.constraints.BaseCustomConstraint

expected_exceptions = *(<class 'openstack.exceptions.HttpException'>,)*

validate_with_client(*client*, *value*)

class heat.engine.clients.os.senlin.PolicyTypeConstraint

Bases: heat.engine.constraints.BaseCustomConstraint

expected_exceptions = *(<class 'heat.common.exception.StackValidationFailed'>,)*

validate_with_client(*client*, *value*)

class heat.engine.clients.os.senlin.ProfileConstraint

Bases: heat.engine.constraints.BaseCustomConstraint

expected_exceptions = *(<class 'openstack.exceptions.HttpException'>,)*

validate_with_client(*client*, *profile*)

class heat.engine.clients.os.senlin.ProfileTypeConstraint

Bases: heat.engine.constraints.BaseCustomConstraint

expected_exceptions = *(<class 'heat.common.exception.StackValidationFailed'>,)*

validate_with_client(*client*, *value*)

class **heat.engine.clients.os.senlin.SenlinClientPlugin(*context*)**

Bases: heat.engine.clients.client_plugin.ClientPlugin

CLUSTERING = '*clustering*'

VERSION = '*1*'

generate_spec(*spec_type*, *spec_props*)

is_bad_request(*ex*)

is_not_found(*ex*)

service_types = *['clustering']*

The heat.engine.clients.os.swift Module

class **heat.engine.clients.os.swift.SwiftClientPlugin(*context*)**

Bases: heat.engine.clients.client_plugin.ClientPlugin

OBJECT_STORE = '*object-store*'

exceptions_module = *<module 'swiftclient.exceptions' from '/home/jenkins/workspace/heat-docs-ubuntu-xenial/.tox/venv/local/lib/python2.7/site-packages/swiftclient/exceptions.pyc'>*

get_signal_url(*container_name*, *obj_name*, *timeout=None*)

Turn on object versioning.

We can use a single TempURL for multiple signals and return a Swift TempURL.

get_temp_url(*container_name*, *obj_name*, *timeout=None*, *method='PUT'*)

Return a Swift TempURL.

is_client_exception(*ex*)

is_conflict(*ex*)

is_not_found(*ex*)

is_over_limit(*ex*)

is_valid_temp_url_path(*path*)

Return True if path is a valid Swift TempURL path, False otherwise.

A Swift TempURL path must: - Be five parts, [‘’, ‘v1’, ‘account’, ‘container’, ‘object’] - Be a v1 request - Have account, container, and object values - Have an object value with more than just ‘/’s

Parameters: **path** (*string*) – The TempURL path

parse_last_modified(*lm*)

Parses the last-modified value.

For example, last-modified values from a swift object header. Returns the datetime.datetime of that value.

Parameters: **lm** (*string*) – The last-modified value (or None)

Returns: An offset-naive UTC datetime of the value (or None)

service_types = *['object-store']*

The heat.engine.clients.os.trove Module

***class* heat.engine.clients.os.trove.FlavorConstraint**

Bases: heat.engine.constraints.BaseCustomConstraint

expected_exceptions = *(<class 'troveclient.openstack.common.apiclient.exceptions.NotFound'>,)*

resource_client_name = *'trove'*

resource_getter_name = *'find_flavor_by_name_or_id'*

***class* heat.engine.clients.os.trove.TroveClientPlugin**(*context*)

Bases: heat.engine.clients.client_plugin.ClientPlugin

DATABASE = '*database*'

exceptions_module = *<module 'troveclient.openstack.common.apiclient.exceptions' from '/home/jenkins/workspace/heat-docs-ubuntu-xenial/.tox/venv/local/lib/python2.7/site-packages/troveclient/openstack/common/apiclient/exceptions.pyc'>*

find_flavor_by_name_or_id(*flavor*)

Find the specified flavor by name or id.

Parameters: **flavor** – the name of the flavor to find

Returns: the id of :flavor:

is_conflict(*ex*)

is_not_found(*ex*)

is_over_limit(*ex*)

service_types = *['database']*

validate_datastore(*datastore_type*, *datastore_version*, *ds_type_key*, *ds_version_key*)

The heat.engine.clients.os.zaqar Module

***class* heat.engine.clients.os.zaqar.ZaqarClientPlugin(*context*)**

Bases: heat.engine.clients.client_plugin.ClientPlugin

DEFAULT_TTL = *3600*

MESSAGING = '*messaging*'

create_for_tenant(*tenant_id*, *token*)

create_from_signed_url(*project_id*, *paths*, *expires*, *methods*, *signature*)

exceptions_module = *<module 'zaqarclient.transport.errors' from '/home/jenkins/workspace/heat-docs-ubuntu-*

xenial/.tox/venv/local/lib/python2.7/site-packages/zaqarclient/transport/errors.pyc'>

`is_not_found`(*ex*)

`service_types` = *['messaging']*

class `heat.engine.clients.os.zaqar.ZaqarEventSink`(*target*, *ttl=None*)

Bases: `object`

`consume`(*context*, *event*)

The `heat.engine.clients.progress` Module

Helper classes that are simple key-value storages meant to be passed between handle_* and check_*_complete, being mutated during subsequent check_*_complete calls.

Some of them impose restrictions on client plugin API, thus they are put in this client-plugin-agnostic module.

class `heat.engine.clients.progress.PoolDeleteProgress`(*task_complete=False*)

Bases: `object`

class `heat.engine.clients.progress.ServerCreateProgress`(*server_id*, *complete=False*)

Bases: `object`

class `heat.engine.clients.progress.ServerDeleteProgress`(*server_id*, *image_id=None*, *image_complete=True*)

Bases: `object`

class `heat.engine.clients.progress.ServerUpdateProgress`(*server_id*,

handler, complete=False, called=False, handler_extra=None, checker_extra=None)

Bases: heat.engine.clients.progress.ServerCreateProgress

Keeps track on particular server update task.

`handler` is a method of client plugin performing required update operation. Its first positional argument must be `server_id` and this method must be resilent to intermittent failures, returning `True` if API was successfully called, `False` otherwise.

If result of API call is asynchronous, client plugin must have corresponding `check_<handler>` method. Its first positional argument must be `server_id` and it must return `True` or `False` indicating completeness of the update operation.

For synchronous API calls, set `complete` attribute of this object to `True`.

`[handler|checker]_extra` arguments, if passed to constructor, should be dictionaries of

```
{'args': tuple(), 'kwargs': dict()}
```

structure and contain parameters with which corresponding `handler` and `check_<handler>` methods of client plugin must be called. `args` is automatically prepended with `server_id`. Missing `args` or `kwargs` are interpreted as empty tuple/dict respectively. Defaults are interpreted as both `args` and `kwargs` being empty.

***class* heat.engine.clients.progress.VolumeAttachProgress(*srv_id, vol_id, device, task_complete=False*)**

Bases: object

***class* heat.engine.clients.progress.VolumeBackupRestoreProgress(*vol_id, backup_id*)**

Bases: object

***class* heat.engine.clients.progress.VolumeDeleteProgress(*task_complete=False*)**

Bases: object

class **heat.engine.clients.progress.VolumeDetachProgress**(***srv_id***, ***vol_id***, ***attach_id***, ***task_complete=False***)

Bases: object

class **heat.engine.clients.progress.VolumeResizeProgress**(***task_complete=False***, ***size=None***)

Bases: object

class **heat.engine.clients.progress.VolumeUpdateAccessModeProgress**(***task_complete=False***, ***read_only=None***)

Bases: object

The heat.engine.constraint.common_constraints Module

class **heat.engine.constraint.common_constraints.CIDRConstraint**

Bases: heat.engine.constraints.BaseCustomConstraint

validate(***value***, ***context***, ***template=None***)

class **heat.engine.constraint.common_constraints.CRONExpressionConstraint**

Bases: heat.engine.constraints.BaseCustomConstraint

validate(***value***, ***context***, ***template=None***)

class **heat.engine.constraint.common_constraints.DNSDomainConstraint**

Bases: heat.engine.constraint.common_constraints.DNSNameConstraint

validate(*value*, *context*)

class heat.engine.constraint.common_constraints.DNSNameConstraint

Bases: heat.engine.constraints.BaseCustomConstraint

validate(*value*, *context*)

class heat.engine.constraint.common_constraints.ExpirationConstraint

Bases: heat.engine.constraints.BaseCustomConstraint

validate(*value*, *context*)

class heat.engine.constraint.common_constraints.IPConstraint

Bases: heat.engine.constraints.BaseCustomConstraint

validate(*value*, *context*, *template=None*)

class heat.engine.constraint.common_constraints.ISO8601Constraint

Bases: heat.engine.constraints.BaseCustomConstraint

validate(*value*, *context*, *template=None*)

class heat.engine.constraint.common_constraints.MACConstraint

Bases: heat.engine.constraints.BaseCustomConstraint

validate(*value*, *context*, *template=None*)

class
heat.engine.constraint.common_constraints.RelativeDNSNameConstraint

Bases: heat.engine.constraint.common_constraints.DNSNameConstraint

validate(*value*, *context*)

class heat.engine.constraint.common_constraints.TestConstraintDelay

Bases: heat.engine.constraints.BaseCustomConstraint

validate_with_client(*client*, *value*)

class heat.engine.constraint.common_constraints.TimezoneConstraint

Bases: heat.engine.constraints.BaseCustomConstraint

validate(*value*, *context*, *template=None*)

The heat.engine.constraint.heat_constraints Module

class **heat.engine.constraint.heat_constraints.ResourceTypeConstraint**

Bases: heat.engine.constraints.BaseCustomConstraint

validate(*value*, *context*, *template=None*)

The heat.engine.constraints Module

class **heat.engine.constraints.AllowedPattern**(*pattern*, *description=None*)

Bases: heat.engine.constraints.Constraint

Constrain values to a predefined regular expression pattern.

Serializes to JSON as:

```
{
    'allowed_pattern': <pattern>,
    'description': <description>
}
```

valid_types = (*'STRING',)*

class **heat.engine.constraints.AllowedValues**(*allowed*, *description=None*)

Bases: heat.engine.constraints.Constraint

Constrain values to a predefined set.

Serializes to JSON as:

```
{
    'allowed_values': [<allowed1>, <allowed2>, ...],
    'description': <description>
}
```

valid_types = *('STRING', 'INTEGER', 'NUMBER', 'BOOLEAN', 'LIST')*

***class* heat.engine.constraints.AnyIndexDict(*value*)**

Bases: _abcoll.Mapping

A Mapping that returns the same value for any integer index.

Used for storing the schema for a list. When converted to a dictionary, it contains a single item with the key '*'.

ANYTHING = '*'

***class* heat.engine.constraints.BaseCustomConstraint**

Bases: object

A base class for validation using API clients.

It will provide a better error message, and reduce a bit of duplication. Subclass must provide expected_exceptions and implement validate_with_client.

error(*value*)

expected_exceptions = *(<class 'heat.common.exception.EntityNotFound'>,)*

resource_client_name = *None*

resource_getter_name = *None*

validate(*value*, *context*, *template=None*)

validate_with_client(*client*, *resource_id*)

***class* heat.engine.constraints.Constraint(*description=None*)**

Bases: _abcoll.Mapping

Parent class for constraints on allowable values for a Property.

Constraints are serializable to dictionaries following the HOT input Parameter constraints schema using dict().

DESCRIPTION = '*description*'

validate(*value*, *schema=None*, *context=None*, *template=None*)

***class* heat.engine.constraints.CustomConstraint(*name*, *description=None*, *environment=None*)**

Bases: heat.engine.constraints.Constraint

A constraint delegating validation to an external class.

custom_constraint

valid_types = (*'STRING'*, *'INTEGER'*, *'NUMBER'*, *'BOOLEAN'*, *'LIST'*)

***class* heat.engine.constraints.Length(*min=None*, *max=None*, *description=None*)**

Bases: heat.engine.constraints.Range

Constrain the length of values within a range.

Serializes to JSON as:

```
{
    'length': {'min': <min>, 'max': <max>},
    'description': <description>
}
```

valid_types = (*'STRING'*, *'LIST'*, *'MAP'*)

***class* heat.engine.constraints.Range(*min=None*, *max=None*, *description=None*)**

Bases: heat.engine.constraints.Constraint

Constrain values within a range.

Serializes to JSON as:

```
{
    'range': {'min': <min>, 'max': <max>},
    'description': <description>
}
```

MAX = '*max*'

MIN = '*min*'

valid_types = *('INTEGER', 'NUMBER')*

***class* heat.engine.constraints.Schema(*data_type*, *description=None*, *default=None*, *schema=None*, *required=False*, *constraints=None*, *label=None*, *immutable=False*)**

Bases: _abcoll.Mapping

Schema base class for validating properties or parameters.

Schema objects are serializable to dictionaries following a superset of the HOT input Parameter schema using dict().

Serialises to JSON in the form:

```
{
    'type': 'list',
    'required': False
    'constraints': [
        {
            'length': {'min': 1},
            'description': 'List must not be empty'
        }
    ],
    'schema': {
        '*': {
            'type': 'string'
        }
    },
    'description': 'An example list property.'
}
```

BOOLEAN = '*Boolean*'

BOOLEAN_TYPE = '*BOOLEAN*'

CONSTRAINTS = '*constraints*'

DEFAULT = '*default*'

DESCRIPTION = '*description*'

IMMUTABLE = '*immutable*'

INTEGER = '*Integer*'

INTEGER_TYPE = '*INTEGER*'

KEYS = *('type', 'description', 'default', 'schema', 'required', 'constraints',*

'*immutable*')

LIST = '*List*'

LIST_TYPE = '*LIST*'

MAP = '*Map*'

MAP_TYPE = '*MAP*'

NUMBER = '*Number*'

NUMBER_TYPE = '*NUMBER*'

REQUIRED = '*required*'

SCHEMA = '*schema*'

STRING = '*String*'

STRING_TYPE = '*STRING*'

TYPE = '*type*'

TYPES = (*'Integer*', '*String*', '*Number*', '*Boolean*', '*Map*', '*List*')

TYPE_KEYS = (*'INTEGER*', '*STRING*', '*NUMBER*', '*BOOLEAN*', '*MAP*', '*LIST*')

set_default(*default=None*)

Set the default value for this Schema object.

static **str_to_num**(*value*)

Convert a string representation of a number into a numeric type.

to_schema_type(*value*)

Returns the value in the schema's data type.

validate(*context=None*)

Validates the schema.

This method checks if the schema itself is valid, and if the default value - if present - complies to the schema's constraints.

validate_constraints(*value*, *context=None*, *skipped=None*, *template=None*)

The heat.engine.dependencies Module

exception **heat.engine.dependencies.CircularDependencyException**(*****kwargs***)

Bases: heat.common.exception.HeatException

msg_fmt = *u'Circular Dependency Found: %(cycle)s'*

class **heat.engine.dependencies.Dependencies**(***edges=None***)

Bases: object

Helper class for calculating a dependency graph.

graph(***reverse=False***)

Return a copy of the underlying dependency graph.

leaves()

Return an iterator over all of the leaf nodes in the graph.

required_by(***last***)

List the keys that require the specified node.

requires(***target***)

List the keys that require the specified node.

roots()

Return an iterator over all of the root nodes in the graph.

translate(***transform***)

Translate all of the nodes using a transform function.

Returns a new Dependencies object.

class **heat.engine.dependencies.Graph**(*****args***)

Bases: collections.defaultdict

A mutable mapping of objects to nodes in a dependency graph.

copy()

Return a copy of the graph.

edges()

Return an iterator over all of the edges in the graph.

map(***func***)

Map the supplied function onto each node in the graph.

Return a dictionary derived from mapping the supplied function onto each node in the graph.

reverse_copy()

Return a copy of the graph with the edges reversed.

static **toposort**(***graph***)

Return a topologically sorted iterator over a dependency graph.

This is a destructive operation for the graph.

class **heat.engine.dependencies.Node**(***requires=None, required_by=None***)

Bases: object

A node in a dependency graph.

copy()

Return a copy of the node.

disjoint()

Return True if this node is both a leaf and a stem.

required_by(*source=None*)

List the keys that require this node, and optionally add new one.

requires(*target=None*)

Add a key that this node requires, and optionally add a new one.

reverse_copy()

Return a copy of the node with the edge directions reversed.

stem()

Return True if this node is a stem (required by nothing).

The heat.engine.environment Module

class **heat.engine.environment.ClassResourceInfo**(*registry*, *path*, *value*)

Bases: heat.engine.environment.ResourceInfo

Store the mapping of resource name to python class implementation.

description = '*Plugin*'

get_class(*files=None*)

class **heat.engine.environment.Environment**(*env=None*, *user_env=True*)

Bases: object

env_as_dict()

Get the entire environment as a dict.

get_class(*resource_type*, *resource_name=None*, *files=None*)

get_class_to_instantiate(*resource_type*, *resource_name=None*)

get_constraint(*name*)

get_event_sinks()

get_resource_info(*resource_type*, *resource_name=None*, *registry_type=None*, *ignore=None*)

get_stack_lifecycle_plugins()

get_types(*cnxt=None*, *support_status=None*, *type_name=None*, *version=None*, *with_description=False*)

load(*env_snippet*)

register_class(*resource_type*, *resource_class*, *path=None*)

register_constraint(*constraint_name*, *constraint*)

register_event_sink(*event_sink_name*, *event_sink_class*)

register_stack_lifecycle_plugin(*stack_lifecycle_name*, *stack_lifecycle_class*)

user_env_as_dict()

Get the environment as a dict, only user-allowed keys.

class **heat.engine.environment.GlobResourceInfo**(*registry*, *path*, *value*)

Bases: heat.engine.environment.MapResourceInfo

Store the mapping (with wild cards) of one resource type to another.

like: OS::Networking::* -> OS::Neutron::*

Also supports many-to-one mapping (mostly useful together with special “OS::Heat::None” resource)

like: OS::* -> OS::Heat::None

description = '*Wildcard Mapping*'

get_resource_info(*resource_type=None*, *resource_name=None*)

matches(*resource_type*)

class **heat.engine.environment.MapResourceInfo**(*registry*, *path*, *value*)

Bases: heat.engine.environment.ResourceInfo

Store the mapping of one resource type to another.

like: OS::Networking::FloatingIp -> OS::Neutron::FloatingIp

description = '*Mapping*'

get_class(*files=None*)

get_resource_info(*resource_type=None*, *resource_name=None*)

***class* heat.engine.environment.ResourceInfo(*registry*, *path*, *value*)**

Bases: object

Base mapping of resource type to implementation.

get_class()

get_class_to_instantiate()

get_resource_info(*resource_type=None*, *resource_name=None*)

matches(*resource_type*)

registry

***class* heat.engine.environment.ResourceRegistry(*global_registry*, *param_defaults*)**

Bases: object

By looking at the environment, find the resource implementation.

as_dict()

Return user resources in a dict format.

get_class(*resource_type*, *resource_name=None*, *files=None*)

get_class_to_instantiate(*resource_type*, *resource_name=None*)

get_resource_info(*resource_type*, *resource_name=None*, *registry_type=None*, *ignore=None*)

Find possible matches to the resource type and name.

Chain the results from the global and user registry to find a match.

get_rsrc_restricted_actions(*resource_name*)

Returns a set of restricted actions.

For a given resource we get the set of restricted actions.

Actions are set in this format via resources:

```
{
    "restricted_actions": [update, replace]
}
```

A restricted_actions value is either update, replace or a list of those values. Resources support wildcard matching. The asterisk sign matches everything.

get_types(*cnxt=None*, *support_status=None*, *type_name=None*, *version=None*, *with_description=False*)

Return a list of valid resource types.

iterable_by(*resource_type*, *resource_name=None*)

load(*json_snippet*)

log_resource_info(*show_all=False*, *prefix=None*)

matches_hook(*resource_name*, *hook*)

Return whether a resource have a hook set in the environment.

For a given resource and a hook type, we check to see if the passed group of resources has the right hook associated with the name.

Hooks are set in this format via resources:

```
{
    "res_name": {
        "hooks": [pre-create, pre-update]
    }, "*_suffix": {
        "hooks": pre-create
```

```
        }, “prefix_*”: {
            “hooks”: pre-update
        }
    }
```

A hook value is either pre-create, pre-update or a list of those values. Resources support wildcard matching. The asterisk sign matches everything.

register_class(*resource_type*, *resource_class*, *path=None*)

remove_item(*info*)

remove_resources_except(*resource_name*)

class **heat.engine.environment.TemplateResourceInfo**(*registry*, *path*, *value*)

Bases: heat.engine.environment.ResourceInfo

Store the info needed to start a TemplateResource.

description = '*Template*'

get_class(*files=None*)

get_class_to_instantiate()

heat.engine.environment.get_child_environment(*parent_env*, *child_params*, *item_to_remove=None*, *child_resource_name=None*)

Build a child environment using the parent environment and params.

This is built from the child_params and the parent env so some resources can use user-provided parameters as if they come from an environment.

- resource_registry must be merged (child env should be loaded after the parent env to take precedence).
- child parameters must overwrite the parent’s as they won’t be relevant in the child template.

If child_resource_name is provided, resources in the registry will be replaced with the contents of the matching child resource plus anything that passes a wildcard match.

heat.engine.environment.is_hook_definition(*key*, *value*)

heat.engine.environment.is_valid_restricted_action(*key*, *value*)

heat.engine.environment.read_global_environment(*env*, *env_dir=None*)

heat.engine.environment.valid_hook_type(*hook*)

heat.engine.environment.valid_restricted_actions(*action*)

The heat.engine.event Module

class **heat.engine.event.Event**(*context*, *stack*, *action*, *status*, *reason*, *physical_resource_id*, *resource_properties*, *resource_name*, *resource_type*, *uuid=None*, *timestamp=None*, *id=None*)

Bases: object

Class representing a Resource state change.

as_dict()

identifier()

Return a unique identifier for the event.

store()

Store the Event in the database.

The heat.engine.function Module

class **heat.engine.function.Function**(*stack*, *fn_name*, *args*)

Bases: object

Abstract base class for template functions.

dep_attrs(*resource_name*)

Return the attributes of the specified resource that are referenced.

Return an iterator over any attributes of the specified resource that this function references.

dependencies(*path*)

result()

Return the result of resolving the function.

Function subclasses must override this method to calculate their results.

stack

validate()

Validate arguments without resolving the function.

Function subclasses must override this method to validate their args.

class **heat.engine.function.Invalid**(*stack*, *fn_name*, *args*)

Bases: heat.engine.function.Function

A function for checking condition functions and to force failures.

This function is used to force failures for functions that are not supported in condition definition.

result()

heat.engine.function.dep_attrs(*snippet*, *resource_name*)

Iterator over dependent attrs of a resource in a template snippet.

The snippet should be already parsed to insert Function objects where appropriate.

Returns: an iterator over the attributes of the specified resource that are referenced in the template snippet.

heat.engine.function.dependencies(*snippet*, *path*='')

Return an iterator over Resource dependencies in a template snippet.

The snippet should be already parsed to insert Function objects where appropriate.

heat.engine.function.resolve(*snippet*)

heat.engine.function.validate(*snippet*)

The heat.engine.hot.functions Module

***class* heat.engine.hot.functions.Digest(*stack*, *fn_name*, *args*)**

Bases: heat.engine.function.Function

A function for performing digest operations.

Takes the form:

```
digest:
  - <algorithm>
  - <value>
```

Valid algorithms are the ones provided by natively by hashlib (md5, sha1, sha224, sha256, sha384, and sha512) or any one provided by OpenSSL.

digest(*algorithm*, *value*)

result()

validate_usage(*args*)

***class* heat.engine.hot.functions.Equals(*stack*, *fn_name*, *args*)**

Bases: heat.engine.function.Function

A function for comparing whether two values are equal.

Takes the form:

```
{ "equals" : ["value_1", "value_2"] }
```

The value can be any type that you want to compare. Returns true if the two values are equal or false if they aren't.

result()

class **heat.engine.hot.functions.GetAtt**(***stack***, ***fn_name***, ***args***)

Bases: heat.engine.hot.functions.GetAttThenSelect

A function for resolving resource attributes.

Takes the form:

```
get_attr:
  - <resource_name>
  - <attribute_name>
  - <path1>
  - ...
```

dep_attrs(***resource_name***)

result()

class **heat.engine.hot.functions.GetAttAllAttributes**(***stack***, ***fn_name***, ***args***)

Bases: heat.engine.hot.functions.GetAtt

A function for resolving resource attributes.

Takes the form:

```
get_attr:
  - <resource_name>
  - <attributes_name>
  - <path1>
  - ...
```

where <attributes_name> and <path1>, ... are optional arguments. If there is no <attributes_name>, result will be dict of all resource's attributes. Else function returns resolved resource's attribute.

dep_attrs(***resource_name***)

Check if there is no attribute_name defined, return empty chain.

result()

class **heat.engine.hot.functions.GetAttThenSelect**(***stack***, ***fn_name***, ***args***)

Bases: heat.engine.cfn.functions.GetAtt

A function for resolving resource attributes.

Takes the form:

```
get_attr:
  - <resource_name>
  - <attribute_name>
  - <path1>
  - ...
```

result()

class **heat.engine.hot.functions.GetFile**(***stack***, ***fn_name***, ***args***)

Bases: heat.engine.function.Function

A function for including a file inline.

Takes the form:

```
get_file: <file_key>
```

And resolves to the content stored in the files dictionary under the given key.

result()

class **heat.engine.hot.functions.GetParam**(***stack***, ***fn_name***, ***args***)

Bases: heat.engine.function.Function

A function for resolving parameter references.

Takes the form:

```
get_param: <param_name>
```

or:

```
get_param:
  - <param_name>
  - <path1>
  - ...
```

result()

***class* heat.engine.hot.functions.Join(*stack*, *fn_name*, *args*)**

Bases: heat.engine.cfn.functions.Join

A function for joining strings.

Takes the form:

```
{ "list_join" : [ "<delim>", [ "<string_1>", "<string_2>", ... ] ] }
```

And resolves to:

```
"<string_1><delim><string_2><delim>..."
```

***class* heat.engine.hot.functions.JoinMultiple(*stack*, *fn_name*, *args*)**

Bases: heat.engine.function.Function

A function for joining one or more lists of strings.

Takes the form:

```
{ "list_join" : [ "<delim>", [ "<string_1>", "<string_2>", ... ] ] }
```

And resolves to:

```
"<string_1><delim><string_2><delim>..."
```

Optionally multiple lists may be specified, which will also be joined.

result()

***class* heat.engine.hot.functions.MapMerge(*stack*, *fn_name*, *args*)**

Bases: heat.engine.function.Function

A function for merging maps.

Takes the form:

```
{ "map_merge" : [{'k1': 'v1', 'k2': 'v2'}, {'k1': 'v2'}] }
```

And resolves to:

```
{'k1': 'v2', 'k2': 'v2'}
```

result()

class heat.engine.hot.functions.MapReplace(*stack*, *fn_name*, *args*)

Bases: heat.engine.function.Function

A function for performing substitutions on maps.

Takes the form:

```
{"map_replace" : [{'k1': 'v1', 'k2': 'v2'},
                  {'keys': {'k1': 'K1'},
                   'values': {'v2': 'V2'}}]}
```

And resolves to:

```
{'K1': 'v1', 'k2': 'V2'}
```

result()

class heat.engine.hot.functions.Removed(*stack*, *fn_name*, *args*)

Bases: heat.engine.function.Function

This function existed in previous versions of HOT, but has been removed.

Check the HOT guide for an equivalent native function.

result()

class heat.engine.hot.functions.Repeat(*stack*, *fn_name*, *args*)

Bases: heat.engine.function.Function

A function for iterating over a list of items.

Takes the form:

```
repeat:
    template:
        <body>
    for_each:
        <var>: <list>
```

The result is a new list of the same size as <list>, where each element is a copy of <body> with any occurrences of <var> replaced with the corresponding item

of <list>.

result()

validate_args()

class **heat.engine.hot.functions.Replace**(***stack***, ***fn_name***, ***args***)

Bases: heat.engine.cfn.functions.Replace

A function for performing string substitutions.

Takes the form:

```
str_replace:
  template: <key_1> <key_2>
  params:
    <key_1>: <value_1>
    <key_2>: <value_2>
    ...
```

And resolves to:

```
"<value_1> <value_2>"
```

This is implemented using Python's str.replace on each key. The order in which replacements are performed is undefined.

class **heat.engine.hot.functions.ReplaceJson**(***stack***, ***fn_name***, ***args***)

Bases: heat.engine.hot.functions.Replace

A function for performing string substitutions.

Behaves the same as Replace, but tolerates non-string parameter values, e.g map/list - these are serialized as json before doing the string substitution.

result()

class **heat.engine.hot.functions.ResourceFacade**(***stack***, ***fn_name***, ***args***)

Bases: heat.engine.cfn.functions.ResourceFacade

A function for retrieving data in a parent provider template.

A function for obtaining data from the facade resource from within the

corresponding provider template.

Takes the form:

```
resource_facade: <attribute_type>
```

where the valid attribute types are “metadata”, “deletion_policy” and “update_policy”.

DELETION_POLICY = '*deletion_policy*'
METADATA = '*metadata*'
UPDATE_POLICY = '*update_policy*'

***class* heat.engine.hot.functions.StrSplit(*stack*, *fn_name*, *args*)**

Bases: heat.engine.function.Function

A function for splitting delimited strings into a list.

Optionally extracting a specific list member by index.

Takes the form:

```
str_split: [delimiter, string, <index> ]
```

or:

```
str_split:
  - delimiter
  - string
  - <index>
```

If <index> is specified, the specified list item will be returned otherwise, the whole list is returned, similar to get_attr with path based attributes accessing lists.

result()

***class* heat.engine.hot.functions.Yaql(*stack*, *fn_name*, *args*)**

Bases: heat.engine.function.Function

A function for executing a yaql expression.

Takes the form:

```
yaql:
    expression:
        <body>
    data:
        <var>: <list>
```

Evaluates expression <body> on the given data.

classmethod `get_yaql_parser`()

`result`()

`validate_args`()

`validate_expression`(***expression***)

The `heat.engine.hot.parameters` Module

class `heat.engine.hot.parameters.HOTParamSchema`(***data_type***, ***description=None***, ***default=None***, ***schema=None***, ***constraints=None***, ***hidden=False***, ***label=None***, ***immutable=False***)

Bases: `heat.engine.parameters.Schema`

HOT parameter schema.

BOOLEAN = '*boolean*'
CONSTRAINTS = '*constraints*'
DEFAULT = '*default*'
DESCRIPTION = '*description*'
HIDDEN = '*hidden*'
IMMUTABLE = '*immutable*'
KEYS = (*'type', 'description', 'default', 'schema', 'constraints', 'hidden', 'label', 'immutable'*)
LABEL = '*label*'
LIST = '*comma_delimited_list*'
MAP = '*json*'

NUMBER = '*number*'

PARAMETER_KEYS = *('type', 'description', 'default', 'schema', 'constraints', 'hidden', 'label', 'immutable')*

SCHEMA = '*schema*'

STRING = '*string*'

TYPE = '*type*'

TYPES = *('string', 'number', 'comma_delimited_list', 'json', 'boolean')*

***classmethod* from_dict(*param_name*, *schema_dict*)**

Return a Parameter Schema object from a legacy schema dictionary.

Parameters: **param_name** (*str*) – name of the parameter owning the schema; used for more verbose logging

***class* heat.engine.hot.parameters.HOTParameters(*stack_identifier*, *tmpl*, *user_params=None*, *param_defaults=None*)**

Bases: heat.engine.parameters.Parameters

PARAM_PROJECT_ID = '*OS::project_id*'

PARAM_REGION = '*OS::region*'

PARAM_STACK_ID = '*OS::stack_id*'

PARAM_STACK_NAME = '*OS::stack_name*'

PSEUDO_PARAMETERS = *('OS::stack_id', 'OS::stack_name', 'OS::region', 'OS::project_id')*

set_stack_id(*stack_identifier*)

Set the StackId pseudo parameter value.

The heat.engine.hot.template Module

***class* heat.engine.hot.template.HOTemplate20130523(*template*, *template_id=None*, *files=None*, *env=None*)**

Bases: heat.engine.template.Template

A Heat Orchestration Template format stack template.

DESCRIPTION = '*description*'

MAPPINGS = '*__undefined__*'

OUTPUTS = '*outputs*'

OUTPUT_DESCRIPTION = '*description*'

OUTPUT_KEYS = *('description', 'value')*

OUTPUT_VALUE = '*value*'

PARAMETERS = '*parameters*'

PARAMETER_GROUPS = '*parameter_groups*'

RESOURCES = '*resources*'

RES_DELETION_POLICY = '*deletion_policy*'

RES_DEPENDS_ON = '*depends_on*'

RES_DESCRIPTION = '*description*'

RES_METADATA = '*metadata*'

RES_PROPERTIES = '*properties*'

RES_TYPE = '*type*'

RES_UPDATE_POLICY = '*update_policy*'

SECTIONS = *('heat_template_version', 'description', 'parameter_groups', 'parameters', 'resources', 'outputs', '__undefined__')*

SECTIONS_NO_DIRECT_ACCESS = *set(['heat_template_version', 'parameters'])*

VERSION = '*heat_template_version*'

add_resource(*definition*, *name=None*)

deletion_policies = *{'Snapshot': 'Snapshot', 'Retain': 'Retain', 'Delete': 'Delete'}*

functions = *{'Fn::MemberListToMap': <class 'heat.engine.cfn.functions.MemberListToMap'>, 'str_replace': <class 'heat.engine.hot.functions.Replace'>, 'get_attr': <class 'heat.engine.hot.functions.GetAttThenSelect'>, 'resource_facade': <class 'heat.engine.hot.functions.ResourceFacade'>, 'Fn::Split': <class 'heat.engine.cfn.functions.Split'>, 'list_join': <class 'heat.engine.hot.functions.Join'>, 'get_param': <class 'heat.engine.hot.functions.GetParam'>, 'get_file': <class 'heat.engine.hot.functions.GetFile'>, 'Ref': <function Ref at 0x7f507f538b90>, 'Fn::Base64': <class 'heat.engine.cfn.functions.Base64'>, 'Fn::ResourceFacade': <class 'heat.engine.cfn.functions.ResourceFacade'>, 'get_resource': <class 'heat.engine.cfn.functions.ResourceRef'>, 'Fn::Select': <class 'heat.engine.cfn.functions.Select'>, 'Fn::Join': <class 'heat.engine.cfn.functions.Join'>, 'Fn::Replace': <class 'heat.engine.cfn.functions.Replace'>, 'Fn::GetAZs': <class*

'*heat.engine.cfn.functions.GetAZs*'>*}*

`get_section_name`(*section*)

`param_schemata`(*param_defaults=None*)

`parameters`(*stack_identifier*, *user_params*, *param_defaults=None*)

`resource_definitions`(*stack*)

classmethod `rsrc_defn_from_snippet`(*name*, *data*)

`validate_resource_definitions`(*stack*)

class `heat.engine.hot.template.HOTemplate20141016`(*template*, *template_id=None*, *files=None*, *env=None*)

Bases: `heat.engine.hot.template.HOTemplate20130523`

`functions` = *{*'*Fn::MemberListToMap*': *<class* '*heat.engine.hot.functions.Removed*'>, '*str_replace*': *<class* '*heat.engine.hot.functions.Replace*'>, '*get_attr*': *<class* '*heat.engine.hot.functions.GetAtt*'>, '*resource_facade*': *<class* '*heat.engine.hot.functions.ResourceFacade*'>, '*Fn::Split*': *<class* '*heat.engine.hot.functions.Removed*'>, '*list_join*': *<class* '*heat.engine.hot.functions.Join*'>, '*get_file*': *<class* '*heat.engine.hot.functions.GetFile*'>, '*get_param*': *<class* '*heat.engine.hot.functions.GetParam*'>, '*Ref*': *<class* '*heat.engine.hot.functions.Removed*'>, '*Fn::Base64*': *<class* '*heat.engine.hot.functions.Removed*'>, '*Fn::ResourceFacade*': *<class* '*heat.engine.hot.functions.Removed*'>, '*get_resource*': *<class* '*heat.engine.cfn.functions.ResourceRef*'>, '*Fn::Select*': *<class* '*heat.engine.cfn.functions.Select*'>, '*Fn::Join*': *<class* '*heat.engine.hot.functions.Removed*'>, '*Fn::Replace*': *<class* '*heat.engine.hot.functions.Removed*'>, '*Fn::GetAZs*': *<class* '*heat.engine.hot.functions.Removed*'>*}*

class `heat.engine.hot.template.HOTemplate20150430`(*template*, *template_id=None*, *files=None*, *env=None*)

Bases: `heat.engine.hot.template.HOTemplate20141016`

`functions` = *{*'*Fn::MemberListToMap*': *<class* '*heat.engine.hot.functions.Removed*'>, '*str_replace*': *<class* '*heat.engine.hot.functions.Replace*'>, '*get_attr*': *<class* '*heat.engine.hot.functions.GetAtt*'>, '*resource_facade*': *<class* '*heat.engine.hot.functions.ResourceFacade*'>, '*digest*': *<class*

'heat.engine.hot.functions.Digest'>, 'Fn::Split': <class
'heat.engine.hot.functions.Removed'>, 'list_join': <class
'heat.engine.hot.functions.Join'>, 'get_file': <class
'heat.engine.hot.functions.GetFile'>, 'get_param': <class
'heat.engine.hot.functions.GetParam'>, 'Ref': <class
'heat.engine.hot.functions.Removed'>, 'Fn::Base64': <class
'heat.engine.hot.functions.Removed'>, 'Fn::ResourceFacade': <class
'heat.engine.hot.functions.Removed'>, 'get_resource': <class
'heat.engine.cfn.functions.ResourceRef'>, 'Fn::Select': <class
'heat.engine.cfn.functions.Select'>, 'Fn::Join': <class
'heat.engine.hot.functions.Removed'>, 'Fn::Replace': <class
'heat.engine.hot.functions.Removed'>, 'Fn::GetAZs': <class
'heat.engine.hot.functions.Removed'>, 'repeat': <class
'heat.engine.hot.functions.Repeat'>}

class heat.engine.hot.template.HOTemplate20151015(*template*, *template_id=None*, *files=None*, *env=None*)

Bases: heat.engine.hot.template.HOTemplate20150430

functions = {'Fn::MemberListToMap': <class
'heat.engine.hot.functions.Removed'>, 'str_replace': <class
'heat.engine.hot.functions.ReplaceJson'>, 'get_attr': <class
'heat.engine.hot.functions.GetAttAllAttributes'>, 'resource_facade':
<class 'heat.engine.hot.functions.ResourceFacade'>, 'digest': <class
'heat.engine.hot.functions.Digest'>, 'Fn::Split': <class
'heat.engine.hot.functions.Removed'>, 'list_join': <class
'heat.engine.hot.functions.JoinMultiple'>, 'get_file': <class
'heat.engine.hot.functions.GetFile'>, 'get_param': <class
'heat.engine.hot.functions.GetParam'>, 'Ref': <class
'heat.engine.hot.functions.Removed'>, 'Fn::Base64': <class
'heat.engine.hot.functions.Removed'>, 'Fn::ResourceFacade': <class
'heat.engine.hot.functions.Removed'>, 'get_resource': <class
'heat.engine.cfn.functions.ResourceRef'>, 'Fn::Select': <class
'heat.engine.hot.functions.Removed'>, 'str_split': <class
'heat.engine.hot.functions.StrSplit'>, 'Fn::Join': <class
'heat.engine.hot.functions.Removed'>, 'Fn::Replace': <class
'heat.engine.hot.functions.Removed'>, 'Fn::GetAZs': <class
'heat.engine.hot.functions.Removed'>, 'repeat': <class
'heat.engine.hot.functions.Repeat'>}

class heat.engine.hot.template.HOTemplate20160408(*template*, *template_id=None*, *files=None*, *env=None*)

Bases: heat.engine.hot.template.HOTemplate20151015

functions = {'Fn::MemberListToMap': <class 'heat.engine.hot.functions.Removed'>, 'str_replace': <class 'heat.engine.hot.functions.ReplaceJson'>, 'get_attr': <class 'heat.engine.hot.functions.GetAttAllAttributes'>, 'resource_facade': <class 'heat.engine.hot.functions.ResourceFacade'>, 'digest': <class 'heat.engine.hot.functions.Digest'>, 'Fn::Split': <class 'heat.engine.hot.functions.Removed'>, 'list_join': <class 'heat.engine.hot.functions.JoinMultiple'>, 'get_file': <class 'heat.engine.hot.functions.GetFile'>, 'get_param': <class 'heat.engine.hot.functions.GetParam'>, 'map_merge': <class 'heat.engine.hot.functions.MapMerge'>, 'Ref': <class 'heat.engine.hot.functions.Removed'>, 'Fn::Base64': <class 'heat.engine.hot.functions.Removed'>, 'Fn::ResourceFacade': <class 'heat.engine.hot.functions.Removed'>, 'get_resource': <class 'heat.engine.cfn.functions.ResourceRef'>, 'Fn::Select': <class 'heat.engine.hot.functions.Removed'>, 'str_split': <class 'heat.engine.hot.functions.StrSplit'>, 'Fn::Join': <class 'heat.engine.hot.functions.Removed'>, 'Fn::Replace': <class 'heat.engine.hot.functions.Removed'>, 'Fn::GetAZs': <class 'heat.engine.hot.functions.Removed'>, 'repeat': <class 'heat.engine.hot.functions.Repeat'>}

class heat.engine.hot.template.HOTemplate20161014(tmpl, template_id=None, files=None, env=None)

Bases: heat.engine.hot.template.HOTemplate20160408

CONDITIONS = 'conditions'

SECTIONS = ('heat_template_version', 'description', 'parameter_groups', 'parameters', 'resources', 'outputs', '__undefined__', 'conditions')

condition_functions = {'get_param': <class 'heat.engine.hot.functions.GetParam'>, 'equals': <class 'heat.engine.hot.functions.Equals'>}

deletion_policies = {'snapshot': 'Snapshot', 'retain': 'Retain', 'delete': 'Delete', 'Delete': 'Delete', 'Snapshot': 'Snapshot', 'Retain': 'Retain'}

functions = {'Ref': <class 'heat.engine.hot.functions.Removed'>, 'Fn::Replace': <class 'heat.engine.hot.functions.Removed'>, 'map_replace': <class 'heat.engine.hot.functions.MapReplace'>, 'digest': <class 'heat.engine.hot.functions.Digest'>, 'get_file': <class 'heat.engine.hot.functions.GetFile'>, 'get_resource': <class

'heat.engine.cfn.functions.ResourceRef'>, 'Fn::Select': <class 'heat.engine.hot.functions.Removed'>, 'Fn::GetAZs': <class 'heat.engine.hot.functions.Removed'>, 'repeat': <class 'heat.engine.hot.functions.Repeat'>, 'str_split': <class 'heat.engine.hot.functions.StrSplit'>, 'yaql': <class 'heat.engine.hot.functions.Yaql'>, 'Fn::MemberListToMap': <class 'heat.engine.hot.functions.Removed'>, 'str_replace': <class 'heat.engine.hot.functions.ReplaceJson'>, 'get_attr': <class 'heat.engine.hot.functions.GetAttAllAttributes'>, 'resource_facade': <class 'heat.engine.hot.functions.ResourceFacade'>, 'list_join': <class 'heat.engine.hot.functions.JoinMultiple'>, 'get_param': <class 'heat.engine.hot.functions.GetParam'>, 'map_merge': <class 'heat.engine.hot.functions.MapMerge'>, 'Fn::Base64': <class 'heat.engine.hot.functions.Removed'>, 'Fn::ResourceFacade': <class 'heat.engine.hot.functions.Removed'>, 'Fn::Split': <class 'heat.engine.hot.functions.Removed'>, 'Fn::Join': <class 'heat.engine.hot.functions.Removed'>}

The heat.engine.lifecycle_plugin Module

***class* heat.engine.lifecycle_plugin.LifecyclePlugin**

Bases: object

Base class for pre-op and post-op work on a stack.

Implementations should extend this class and override the methods.

do_post_op(*cnxt*, *stack*, *current_stack=None*, *action=None*, *is_stack_failure=False*)

Method to be run by heat after stack operations, including failures.

On failure to execute all the registered pre_ops, this method will be called if and only if the corresponding pre_op was successfully called. On failures of the actual stack operation, this method will be called if all the pre operations were

successfully called.

do_pre_op(*cnxt*, *stack*, *current_stack=None*, *action=None*)

Method to be run by heat before stack operations.

get_ordinal()

Get the sort order for pre and post operation execution.

The values returned by get_ordinal are used to create a partial order for pre and post operation method invocations. The default ordinal value of 100 may be overridden. If class1inst.ordinal() < class2inst.ordinal(), then the method on class1inst will be executed before the method on class2inst. If class1inst.ordinal() > class2inst.ordinal(), then the method on class1inst will be executed after the method on class2inst. If class1inst.ordinal() == class2inst.ordinal(), then the order of method invocation is indeterminate.

The heat.engine.notification.autoscaling Module

heat.engine.notification.autoscaling.send(*stack*, *adjustment=None*, *adjustment_type=None*, *capacity=None*, *groupname=None*, *message='error'*, *suffix=None*)

Send autoscaling notifications to the configured notification driver.

The heat.engine.notification.stack Module

heat.engine.notification.stack.send(*stack*)

Send usage notifications to the configured notification driver.

The heat.engine.parameter_groups Module

class **heat.engine.parameter_groups.ParameterGroups**(*tmpl*)

Bases: object

The ParameterGroups specified by the stack's template.

validate()

Validate the parameter group.

Validate that each parameter belongs to only one Parameter Group and that each parameter name in the group references a valid parameter.

The heat.engine.parameters Module

class **heat.engine.parameters.BooleanParam**(*name*, *schema*, *value=None*)

Bases: heat.engine.parameters.Parameter

A template parameter of type “Boolean”.

value()

***class* heat.engine.parameters.CommaDelimitedListParam(*name*, *schema*, *value=None*)**

Bases: heat.engine.parameters.ParsedParameter, _abcoll.Sequence

A template parameter of type “CommaDelimitedList”.

parse(*value*)

value()

***class* heat.engine.parameters.JsonParam(*name*, *schema*, *value=None*)**

Bases: heat.engine.parameters.ParsedParameter

A template parameter who’s value is map or list.

parse(*value*)

value()

***class* heat.engine.parameters.NumberParam(*name*, *schema*, *value=None*)**

Bases: heat.engine.parameters.Parameter

A template parameter of type “Number”.

value()

***class* heat.engine.parameters.Parameter(*name*, *schema*, *value=None*)**

Bases: object

A template parameter.

default()

Return the default value of the parameter.

description()

Return the description of the parameter.

has_default()

Return whether the parameter has a default value.

has_value()

Parameter has a user or default value.

hidden()

Return whether the parameter is hidden.

Hidden parameters should be sanitised in any output to the user.

label()

Return the label or param name.

set_default(*value*)

validate(*validate_value=True*, *context=None*, *template=None*)

Validates the parameter.

This method validates if the parameter's schema is valid, and if the default value - if present - or the user-provided value for the parameter comply with the schema.

value()

Get the parameter value, optionally sanitising it for output.

***class* heat.engine.parameters.Parameters(*stack_identifier*, *tmpl*, *user_params=None*, *param_defaults=None*)**

Bases: _abcoll.Mapping

Parameters of a stack.

The parameters of a stack, with type checking, defaults, etc. specified by the stack's template.

PARAM_REGION = '*AWS::Region*'

PARAM_STACK_ID = '*AWS::StackId*'

PARAM_STACK_NAME = '*AWS::StackName*'

PSEUDO_PARAMETERS = *('AWS::StackId', 'AWS::StackName', 'AWS::Region')*

immutable_params_modified(*new_parameters*, *input_params*)

map(*func*, *filter_func=<function <lambda> at 0x7f507f760938>*)

Map the supplied function onto each Parameter.

Map the supplied function onto each Parameter (with an optional filter function) and return the resulting dictionary.

set_stack_id(*stack_identifier*)

Set the StackId pseudo parameter value.

validate(*validate_value=True*, *context=None*)

Validates all parameters.

This method validates if all user-provided parameters are actually defined in the template, and if all parameters are valid.

***class* heat.engine.parameters.ParsedParameter(*name*, *schema*, *value=None*)**

Bases: heat.engine.parameters.Parameter

A template parameter with cached parsed value.

set_default(*value*)

***class* heat.engine.parameters.Schema(*data_type*, *description=None*, *default=None*, *schema=None*, *constraints=None*, *hidden=False*, *label=None*, *immutable=False*)**

Bases: heat.engine.constraints.Schema

Parameter schema.

BOOLEAN = '*Boolean*'

CONSTRAINTS = '*Constraints*'

DEFAULT = '*Default*'

DESCRIPTION = '*Description*'

HIDDEN = '*NoEcho*'

IMMUTABLE = '*Immutable*'

KEYS = *('Type', 'Description', 'Default', 'Schema', 'Constraints', 'NoEcho', 'Label', 'Immutable')*

LABEL = '*Label*'

LIST = '*CommaDelimitedList*'

MAP = '*Json*'

NUMBER = '*Number*'

PARAMETER_KEYS = *('Type', 'Default', 'NoEcho', 'AllowedValues', 'AllowedPattern', 'MaxLength', 'MinLength', 'MaxValue', 'MinValue', 'Description', 'ConstraintDescription', 'Label')*

SCHEMA = '*Schema*'

STRING = '*String*'

TYPE = '*Type*'

TYPES = *('String', 'Number', 'CommaDelimitedList', 'Json', 'Boolean')*

classmethod `from_dict`(***param_name***, ***schema_dict***)

Return a Parameter Schema object from a legacy schema dictionary.

Parameters: **param_name** (*str*) – name of the parameter owning the schema; used for more verbose logging

static `get_num`(***key***, ***context***)

`set_default`(***default=None***)

`validate_value`(***value***, ***context=None***, ***template=None***)

class `heat.engine.parameters.StringParam`(***name***, ***schema***, ***value=None***)

Bases: `heat.engine.parameters.Parameter`

A template parameter of type “String”.

`value`()

The heat.engine.plugin_manager Module

class **heat.engine.plugin_manager.PluginManager(**extra_packages*)**

Bases: object

A class for managing plugin modules.

map_to_modules(*function*)

Iterate over the results of calling a function on every module.

class **heat.engine.plugin_manager.PluginMapping(*names*, **args*, ***kwargs*)**

Bases: object

A class for managing plugin mappings.

load_all(*plugin_manager*)

Iterate over the mappings from all modules in the plugin manager.

Mappings are returned as a list of (key, value) tuples.

load_from_module(*module*)

Return the mapping specified in the given module.

If no such mapping is specified, an empty dictionary is returned.

The heat.engine.properties Module

class heat.engine.properties.Properties(***schema***, ***data***, ***resolver=<function <lambda> at 0x7f507f761cf8>***, ***parent_name=None***, ***context=None***, ***section=None***)

Bases: _abcoll.Mapping

get_user_value(***key***, ***validate=False***, ***template=None***)

static schema_from_params(***params_snippet***)

Create properties schema from the parameters section of a template.

Parameters: **params_snippet** – parameter definition from a template

Returns: equivalent properties schemata for the specified parameters

classmethod schema_to_parameters_and_properties(***schema***, ***template_type='cfn'***)

Convert a schema to template parameters and properties.

This can be used to generate a provider template that matches the given properties schemata.

Parameters: **schema** – A resource type's properties_schema

Returns: A tuple of params and properties dicts

ex: input: {'foo': {'Type': 'List'}}
output: {'foo': {'Type': 'CommaDelimitedList'}},
{'foo': {'Fn::Split': {'Ref': 'foo'}}}

ex: input: {'foo': {'Type': 'String'}, 'bar': {'Type': 'Map'}}
output: {'foo': {'Type': 'String'}, 'bar': {'Type': 'Json'}},
{'foo': {'Ref': 'foo'}, 'bar': {'Ref': 'bar'}}

validate(***with_value=True***, ***template=None***)

class heat.engine.properties.Property(***schema***, ***name=None***, ***context=None***)

Bases: object

default()

get_value(*value*, *validate=False*, *template=None*)

Get value from raw value and sanitize according to data type.

has_default()

immutable()

implemented()

required()

support_status()

type()

update_allowed()

class **heat.engine.properties.Schema**(*data_type*, *description=None*, *default=None*, *schema=None*, *required=False*, *constraints=None*, *implemented=True*, *update_allowed=False*, *immutable=False*, *support_status=<heat.engine.support.SupportStatus object at 0x7f507f757d10>*, *allow_conversion=False*)

Bases: heat.engine.constraints.Schema

Schema class for validating resource properties.

This class is used for defining schema constraints for resource properties. It inherits generic validation features from the base Schema class and add processing that is specific to resource properties.

CONSTRAINTS = '*constraints*'

DEFAULT = '*default*'

DESCRIPTION = '*description*'

IMMUTABLE = '*immutable*'

KEYS = *('type', 'description', 'default', 'schema', 'required', 'constraints', 'update_allowed', 'immutable')*

REQUIRED = '*required*'

SCHEMA = '*schema*'

TYPE = '*type*'

UPDATE_ALLOWED = '*update_allowed*'

allowed_param_prop_type()

Return allowed type of Property Schema converted from parameter.

Especially, when generating Schema from parameter, Integer Property Schema will be supplied by Number parameter.

classmethod `from_legacy`(***schema_dict***)

Return a Property Schema object from a legacy schema dictionary.

classmethod `from_parameter`(***param***)

Return a Property Schema corresponding to a Parameter Schema.

Convert a parameter schema from a provider template to a property Schema for the corresponding resource facade.

`validate`(***context=None***)

`heat.engine.properties.schemata`(***schema_dicts***)

Return dictionary of Schema objects for given dictionary of schemata.

The input schemata are converted from the legacy (dictionary-based) format to Schema objects where necessary.

The `heat.engine.resource` Module

exception `heat.engine.resource.PollDelay`(***period***)

Bases: `exceptions.Exception`

Exception to delay polling of the resource.

This exception may be raised by a Resource subclass's check_*_complete() methods to indicate that it need not be polled again immediately. If this exception is raised, the check_*_complete() method will not be called again until the nth time that the resource becomes eligible for polling. A PollDelay period of 1 is equivalent to returning False.

***class* heat.engine.resource.Resource(*name*, *definition*, *stack*)**

Bases: object

ACTIONS = (*'INIT'*, *'CREATE'*, *'DELETE'*, *'UPDATE'*, *'ROLLBACK'*, *'SUSPEND'*, *'RESUME'*, *'ADOPT'*, *'SNAPSHOT'*, *'CHECK'*)

ADOPT = *'ADOPT'*

BASE_ATTRIBUTES = (*'show',*)

CHECK = *'CHECK'*

COMPLETE = *'COMPLETE'*

CREATE = *'CREATE'*

DELETE = *'DELETE'*

FAILED = *'FAILED'*

FnBase64(*data*)

For the intrinsic function Fn::Base64.

Parameters: **data** – the input data.

Returns: the Base64 representation of the input data.

FnGetAtt(*key*, **path*)

For the intrinsic function Fn::GetAtt.

Parameters:
- **key** – the attribute key.
- **path** – a list of path components to select from the attribute.

Returns: the attribute value.

FnGetAtts()

For the intrinsic function get_attr which returns all attributes.

Returns: dict of all resource's attributes exclude "show" attribute.

FnGetRefId()

For the intrinsic function Ref.

Results: the id or name of the resource.

INIT = *'INIT'*

IN_PROGRESS = *'IN_PROGRESS'*

RESUME = *'RESUME'*

ROLLBACK = '*ROLLBACK*'

SHOW = '*show*'

SNAPSHOT = '*SNAPSHOT*'

STATUSES = *('IN_PROGRESS', 'FAILED', 'COMPLETE')*

SUSPEND = '*SUSPEND*'

UPDATE = '*UPDATE*'

action_handler_task(*action*, *args=None*, *action_prefix=None*)

A task to call the Resource subclass's handler methods for action.

Calls the handle_<ACTION>() method for the given action and then calls the check_<ACTION>_complete() method with the result in a loop until it returns True. If the methods are not provided, the call is omitted.

Any args provided are passed to the handler.

If a prefix is supplied, the handler method handle_<PREFIX>_<ACTION>() is called instead.

add_dependencies(*deps*)

Add implicit dependencies specific to the resource type.

Some resource types may have implicit dependencies on other resources in the same stack that are not linked by a property value (that would be set using get_resource or get_attr for example, thus creating an explicit dependency). Such dependencies are opaque to the user and should be avoided wherever possible, however in some circumstances they are required due to magic in the underlying API.

The deps parameter is a Dependencies object to which dependency pairs may be added.

add_explicit_dependencies(*deps*)

Add all dependencies explicitly specified in the template.

The deps parameter is a Dependencies object to which dependency pairs are added.

adopt(*resource_data*)

Adopt the existing resource.

Resource subclasses can provide a handle_adopt() method to customise adopt.

attributes_schema = *{}*

base_attributes_schema = *{'show': <heat.engine.attributes.Schema object at 0x7f507d23a390>}*

***static* build_template_dict(*res_name*, *res_type*, *tmpl_type*, *params*, *props*, *outputs*, *description*)**

cancel_grace_period()

ceilometer()

check()

Checks that the physical resource is in its expected state.

Gets the current status of the physical resource and updates the database accordingly. If check is not supported by the resource, default action is to fail and revert the resource's status to its original state with the added message that check was not performed.

cinder()

clear_hook(*hook*)

client(*name=None*, *version=None*)

client_plugin(*name=None*)

create(args*, ***kwargs*)**

Create the resource.

Subclasses should provide a handle_create() method to customise creation.

create_convergence(*template_id*, *resource_data*, *engine_id*, *timeout*)

Creates the resource by invoking the scheduler TaskRunner.

data()

Return the resource data for this resource.

Use methods data_set and data_delete to modify the resource data for this resource.

Returns: a dict representing the resource data for this resource.

data_delete(*key*)

Remove a key from the resource data.

Returns: True if the key existed to delete.

data_set(*key*, *value*, *redact=False*)

Set a key in the resource data.

default_client_name = *None*

delete(**args*, ***kwargs*)

A task to delete the resource.

Subclasses should provide a handle_delete() method to customise deletion.

delete_convergence(*template_id*, *input_data*, *engine_id*, *timeout*)

Destroys the resource if it doesn't belong to given template.

The given template is suppose to be the current template being provisioned.

Also, since this resource is visited as part of clean-up phase, the needed_by should be updated. If this resource was replaced by more recent resource, then delete this and update the replacement resource's needed_by and replaces fields.

delete_snapshot(**args*, ***kwargs*)

dep_attrs(*resource_name*)

destroy(**args*, ***kwargs*)

A task to delete the resource and remove it from the database.

entity = *None*

frozen_definition()

get_attribute(*key*, **path*)

Default implementation for function get_attr and Fn::GetAtt.

This may be overridden by resource plugins to add extra logic specific to the

resource implementation.

get_live_resource_data()

Default implementation; can be overridden by resources.

Get resource data and handle it with exceptions.

get_live_state(*resource_properties*)

Default implementation; should be overridden by resources.

Parameters: **resource_properties** – resource's object of Properties class.

Returns: dict of resource's real state of properties.

get_reference_id()

Default implementation for function get_resource.

This may be overridden by resource plugins to add extra logic specific to the resource implementation.

glance()

handle_adopt(*resource_data=None*)

handle_delete()

Default implementation; should be overridden by resources.

handle_update(*json_snippet*, *tmpl_diff*, *prop_diff*)

has_hook(*hook*)

has_interface(*resource_type*)

Check if resource is mapped to resource_type or is "resource_type".

Check to see if this resource is either mapped to resource_type or is a "resource_type".

has_nested()

Return True if the resource has an existing nested stack.

For most resource types, this will always return False. StackResource

subclasses return True when appropriate. Resource subclasses that may return True must also provide a nested_identifier() method to return the identifier of the nested stack, and a nested() method to return a Stack object for the nested stack.

heat()

identifier()

Return an identifier for this resource.

***classmethod* is_service_available(*context*)**

is_using_neutron()

keystone()

***classmethod* load(*context*, *resource_id*, *is_update*, *data*)**

lock(args*, ***kwds*)**

make_replacement(*new_tmpl_id*)

metadata_get(*refresh=False*)

metadata_set(args*, ***kw*)**

Write new metadata to the database.

The caller may optionally provide a merge_metadata() function, which takes two arguments - the metadata passed to metadata_set() and the current metadata of the resource - and returns the merged metadata to write. If merge_metadata is not provided, the metadata passed to metadata_set() is written verbatim, overwriting any existing metadata.

If a race condition is detected, the write will be retried with the new result of merge_metadata() (if it is supplied) or the verbatim data (if it is not).

metadata_update(*new_metadata=None*)

No-op for resources which don't explicitly override this method.

needs_replace(*after_props*)

Mandatory replace based on certain properties.

needs_replace_with_prop_diff(*changed_properties_set*, *after_props*,

before_props)

Needs replace based on prop_diff.

needs_replace_with_tmpl_diff(***tmpl_diff***)

Needs replace based on tmpl_diff.

neutron()

no_signal_actions = *('**SUSPEND**', '**DELETE**')*

nova()

parse_live_resource_data(***resource_properties***, ***resource_data***)

Default implementation; can be overridden by resources.

Parse resource data for using it in updating properties with live state. :param resource_properties: properties of stored resource plugin. :param resource_data: data from current live state of a resource.

parsed_template(***section=None***, ***default=None***)

Return the parsed template data for the resource.

May be limited to only one section of the data, in which case a default value may also be supplied.

static **pause**()

physical_resource_name()

physical_resource_name_limit = ***255***

physical_resource_name_or_FnGetRefId()

prepare_abandon()

prepare_for_replace()

Prepare resource for replacing.

Some resources requires additional actions before replace them. If resource need to be changed before replacing, this method should be implemented in resource class.

preview()

Default implementation of Resource.preview.

This method should be overridden by child classes for specific behavior.

preview_update(*after*, *before*, *after_props*, *before_props*, *prev_resource*, *check_init_complete=False*)

Simulates update without actually updating the resource.

Raises UpdateReplace, if replacement is required or returns True, if in-place update is required.

static **reduce_physical_resource_name**(*name*, *limit*)

Reduce length of physical resource name to a limit.

The reduced name will consist of the following:

- the first 2 characters of the name
- a hyphen
- the end of the name, truncated on the left to bring the name length within the limit

Parameters:
- **name** – The name to reduce the length of
- **limit** – The max length limit

Returns: A name whose length is less than or equal to the limit

regenerate_info_schema(*definition*)

Default implementation; should be overridden by resources.

Should be overridden by resources that would require schema refresh during update, ex. TemplateResource.

Definition: Resource Definition

reparse(*translate=True*, *client_resolve=True*)

Reparse the resource properties.

Optional translate flag for property translation and client_resolve flag for resolving properties by doing client lookup.

required_by()

List of resources that require this one as a dependency.

Returns a list of names of resources that depend on this resource directly.

required_service_extension = *None*

requires_deferred_auth = *False*

resource_id_set(*inst*)

***classmethod* resource_to_template(*resource_type*, *template_type*='*cfn*')**

Generate a provider template that mirrors the resource.

Parameters:
- **resource_type** – The resource type to be displayed in the template
- **template_type** – the template type to generate, cfn or hot.

Returns: A template where the resource's properties_schema is mapped as parameters, and the resource's attributes_schema is mapped as outputs

restore_prev_rsrc(*convergence=False*)

Restore resource after rollback.

Some resources requires additional actions after rollback. If resource need to be changed during rollback, this method should be implemented in resource class.

resume()

Return a task to resume the resource.

Subclasses should provide a handle_resume() method to implement resume.

rpc_client()

Return a client for making engine RPC calls.

***classmethod* set_needed_by(*db_rsrc*, *needed_by*,**

***expected_engine_id=None*)**

***classmethod* set_requires(*db_rsrc*, *requires*)**

signal(*details=None*, *need_check=True*)

Signal the resource.

Returns True if the metadata for all resources in the stack needs to be regenerated as a result of the signal, False if it should not be.

Subclasses should provide a handle_signal() method to implement the signal. The base-class raise an exception if no handler is implemented.

signal_needs_metadata_updates = *True*

snapshot()

Snapshot the resource and return the created data, if any.

stack

state

Returns state, tuple of action, status.

state_reset()

Reset state to (INIT, COMPLETE).

state_set(*action*, *status*, *reason*='*state changed*')

strict_dependency = *True*

support_status = <*heat.engine.support.SupportStatus object at 0x7f507d23a3d0*>

suspend()

Return a task to suspend the resource.

Subclasses should provide a handle_suspend() method to implement suspend.

swift()

translate_properties(*properties*, *client_resolve=True*)

Translates properties with resource specific rules.

The properties parameter is a properties object and the optional client_resolve flag is to specify whether to do 'RESOLVE' translation with client lookup.

translation_rules(*properties*)

Return specified rules for resource.

trigger_hook(*hook*)

trove()

type()

update(**args*, ***kwargs*)

Return a task to update the resource.

Subclasses should provide a handle_update() method to customise update, the base-class handle_update will fail by default.

update_allowed_properties = *()*

update_convergence(*template_id*, *resource_data*, *engine_id*, *timeout*, *new_stack*)

Update the resource synchronously.

Persist the resource's current_template_id to template_id and resource's requires to list of the required resource ids from the given resource_data and existing resource's requires, then updates the resource by invoking the scheduler TaskRunner.

update_policy_schema = *{}*

update_template_diff(*after*, *before*)

Returns the difference between the before and after json snippets.

If something has been removed in after which exists in before we set it to None.

update_template_diff_properties(*after_props*, *before_props*)

Return changed Properties between the before and after properties.

If any property having immutable as True is updated, raises NotSupported error. If any properties have changed which are not in update_allowed_properties, raises UpdateReplace.

validate()

Validate the resource.

This may be overridden by resource plugins to add extra validation logic specific to the resource implementation.

classmethod **validate_deletion_policy(*policy*)**

validate_template()

Validate structural/syntax aspects of the resource definition.

Resource plugins should not override this, because this interface is expected to be called pre-create so things normally valid in an overridden validate() such as accessing properties may not work.

The heat.engine.rsrc_defn Module

class **heat.engine.rsrc_defn.ResourceDefinition(*name*, *resource_type*, *properties=None*, *metadata=None*, *depends=None*, *deletion_policy=None*, *update_policy=None*, *description=None*)**

Bases: heat.engine.rsrc_defn.ResourceDefinitionCore, _abcoll.Mapping

A resource definition that also acts like a cfn template snippet.

This class exists only for backwards compatibility with existing resource plugins and unit tests; it is deprecated and will be replaced with ResourceDefinitionCore, possibly as soon as the Ocata release.

class **Diff(*old_defn*, *new_defn*)**

Bases: heat.engine.rsrc_defn.Diff, _abcoll.Mapping

A resource definition diff that acts like a cfn template snippet.

This class exists only for backwards compatibility with existing resource plugins and unit tests; it is deprecated and could be removed as soon as the Ocata release. Prefer using the API directly rather than treating the diff as a dict containing the differences between two cfn template snippets.

The heat.engine.scheduler Module

class **heat.engine.scheduler.DependencyTaskGroup(*dependencies*, *task=<function <lambda> at 0x7f507d314500>*, *reverse=False*, *name=None*, *error_wait_time=None*, *aggregate_exceptions=False*)**

Bases: object

Task which manages group of subtasks that have ordering dependencies.

cancel_all(*grace_period=None*)

exception **heat.engine.scheduler.ExceptionGroup(*exceptions=None*)**

Bases: exceptions.Exception

Container for multiple exceptions.

This exception is used by DependencyTaskGroup when the flag aggregate_exceptions is set to True and it's re-raised again when all tasks are finished. This way it can be caught later on so that the individual exceptions can be acted upon.

class **heat.engine.scheduler.TaskRunner(*task*, **args*, ***kwargs*)**

Bases: object

Wrapper for a resumable task (co-routine).

as_task(*timeout=None*, *progress_callback=None*)

Return a task that drives the TaskRunner.

cancel(*grace_period=None*)

Cancel the task and mark it as done.

done()

Return True if the task is complete.

run_to_completion(*wait_time=1*, *progress_callback=None*)

Run the task to completion.

The task will sleep for wait_time seconds between steps. To avoid sleeping, pass None for wait_time.

start(*timeout=None*)

Initialise the task and run its first step.

If a timeout is specified, any attempt to step the task after that number of seconds has elapsed will result in a Timeout being raised inside the task.

started()

Return True if the task has been started.

step()

Run another step of the task.

Return True if the task is complete; False otherwise.

exception **heat.engine.scheduler.TimedCancel**(*task_runner*, *timeout*)

Bases: heat.engine.scheduler.Timeout

trigger(*generator*)

Trigger the timeout on a given generator.

exception **heat.engine.scheduler.Timeout**(*task_runner*, *timeout*)

Bases: exceptions.BaseException

Raised when task has exceeded its allotted (wallclock) running time.

This allows the task to perform any necessary cleanup, as well as use a different exception to notify the controlling task if appropriate. If the task suppresses the exception altogether, it will be cancelled but the controlling task will not be notified of the timeout.

expired()

trigger(*generator*)

Trigger the timeout on a given generator.

heat.engine.scheduler.task_description(*task*)

Return a human-readable string description of a task.

The description is used to identify the task when logging its status.

heat.engine.scheduler.wrappertask(*task*)

Decorator for a task that needs to drive a subtask.

This is essentially a replacement for the Python 3-only "yield from" keyword (PEP 380), using the "yield" keyword that is supported in Python 2. For example:

```
@wrappertask
def parent_task(self):
    self.setup()

    yield self.child_task()

    self.cleanup()
```

The heat.engine.service Module

***class* heat.engine.service.EngineListener**(*host*, *engine_id*, *thread_group_mgr*)

Bases: oslo_service.service.Service

Listen on an AMQP queue named for the engine.

Allows individual engines to communicate with each other for multi-engine support.

ACTIONS = *('stop_stack', 'send')*

SEND = '*send*'

STOP_STACK = '*stop_stack*'

listening(args*, ***kwargs*)**

Respond to a watchdog request.

Respond affirmatively to confirm that the engine performing the action is still alive.

reset(args*, ***kwargs*)**

Reset a service in case it received a SIGHUP.

send(args*, ***kwargs*)**

start(args*, ***kwargs*)**

stop(args*, ***kwargs*)**

Stop a service.

Parameters: graceful – indicates whether to wait for all threads to finish or terminate them instantly

stop_stack(args*, ***kwargs*)**

Stop any active threads on a stack.

wait(args*, ***kwargs*)**

Wait for a service to shut down.

***class* heat.engine.service.EngineService(*host*, *topic*)**

Bases: oslo_service.service.Service

Manages the running instances from creation to destruction.

All the methods in here are called from the RPC backend. This is all done dynamically so if a call is made via RPC that does not have a corresponding method here, an exception will be thrown when it attempts to call into this class. Arguments to these methods are also dynamically added and will be named as keyword arguments by the RPC caller.

RPC_API_VERSION = '*1.32*'

abandon_stack(args*, ***kwargs*)**

Abandon a given stack.

Parameters:
- **cnxt** – RPC context.
- **stack_identity** – Name of the stack you want to abandon.
- **abandon** – Delete Heat stack but not physical resources.

authenticated_to_backend(args*, ***kwargs*)**

Validate the credentials in the RPC context.

Verify that the credentials in the RPC context are valid for the current cloud backend.

check_software_deployment(args*, ***kwargs*)**

count_stacks(args*, ***kwargs*)**

Return the number of stacks that match the given filters.

Parameters:
- **cnxt** – RPC context.
- **filters** – a dict of ATTR:VALUE to match against stacks
- **tenant_safe** – if true, scope the request by the current tenant
- **show_deleted** – if true, count will include the deleted stacks
- **show_nested** – if true, count will include nested stacks
- **show_hidden** – if true, count will include hidden stacks
- **tags** – count stacks containing these tags, combine multiple tags using the boolean AND expression
- **tags_any** – count stacks containing these tags, combine multiple tags using the boolean OR expression
- **not_tags** – count stacks not containing these tags, combine multiple tags using the boolean AND expression
- **not_tags_any** – count stacks not containing these tags, combine multiple tags using the boolean OR expression

Returns: an integer representing the number of matched stacks

create_periodic_tasks(args*, ***kwargs*)**

create_software_config(args*, ***kwargs*)**

create_software_deployment(**args*, ***kwargs*)

create_stack(**args*, ***kwargs*)

Create a new stack using the template provided.

Note that at this stage the template has already been fetched from the heat-api process if using a template-url.

Parameters:

- **cnxt** – RPC context.
- **stack_name** – Name of the stack you want to create.
- **template** – Template of stack you want to create.
- **params** – Stack Input Params
- **files** – Files referenced from the template
- **args** – Request parameters/args passed from API
- **environment_files** (*list or None*) – optional ordered list of environment file names included in the files dict
- **owner_id** – parent stack ID for nested stacks, only expected when called from another heat-engine (not a user option)
- **nested_depth** – the nested depth for nested stacks, only expected when called from another heat-engine
- **user_creds_id** – the parent user_creds record for nested stacks
- **stack_user_project_id** – the parent stack_user_project_id for nested stacks
- **parent_resource_name** – the parent resource name
- **template_id** – the ID of a pre-stored template in the DB

create_watch_data(**args*, ***kwargs*)

Creates data for CloudWatch and WaitConditions.

This could be used by CloudWatch and WaitConditions and treat HA service events like any other CloudWatch.

delete_snapshot(**args*, ***kwargs*)

delete_software_config(**args*, ***kwargs*)

delete_software_deployment(**args*, ***kwargs*)

delete_stack(**args*, ***kwargs*)

Delete a given stack.

Parameters:
- **cnxt** – RPC context.
- **stack_identity** – Name of the stack you want to delete.

describe_stack_resource(*args, **kwargs)

describe_stack_resources(*args, **kwargs)

export_stack(*args, **kwargs)

Exports the stack data json.

Intended to be used to safely retrieve the stack data before performing the abandon action.

Parameters:
- **cnxt** – RPC context.
- **stack_identity** – Name of the stack you want to export.

find_physical_resource(*args, **kwargs)

Return an identifier for the specified resource.

Parameters:
- **cnxt** – RPC context.
- **physical_resource_id** – The physical resource ID to look up.

generate_template(*args, **kwargs)

Generate a template based on the specified type.

Parameters:
- **cnxt** – RPC context.
- **type_name** – Name of the resource type to generate a template for.
- **template_type** – the template type to generate, cfn or hot.

get_environment(*args, **kwargs)

Returns the environment for an existing stack.

Parameters:
- **cnxt** – RPC context
- **stack_identity** – identifies the stack

Return type: dict

get_files(*args, **kwargs)

Returns the files for an existing stack.

Parameters:
- **cnxt** – RPC context
- **stack_identity** – identifies the stack

Return type: dict

get_revision(*args, **kwargs)

get_template(*args, **kwargs)

Get the template.

Parameters:
- **cnxt** – RPC context.
- **stack_identity** – Name of the stack you want to see.

identify_stack(*args, **kwargs)

The full stack identifier for a single, live stack with stack_name.

Parameters:
- **cnxt** – RPC context.
- **stack_name** – Name or UUID of the stack to look up.

list_events(*args, **kwargs)

Lists all events associated with a given stack.

It supports pagination (`limit` and `marker`), sorting (`sort_keys` and `sort_dir`) and filtering(filters) of the results.

Parameters:
- **cnxt** – RPC context.
- **stack_identity** – Name of the stack you want to get events for
- **filters** – a dict with attribute:value to filter the list
- **limit** – the number of events to list (integer or string)
- **marker** – the ID of the last event in the previous page
- **sort_keys** – an array of fields used to sort the list
- **sort_dir** – the direction of the sort ('asc' or 'desc').
- **nested_depth** – Levels of nested stacks to list events for.

list_outputs(**args*, ***kwargs*)

Get a list of stack outputs.

Parameters:
- **cntx** – RPC context.
- **stack_identity** – Name of the stack you want to see.

Returns: list of stack outputs in defined format.

list_resource_types(**args*, ***kwargs*)

Get a list of supported resource types.

Parameters:
- **cnxt** – RPC context.
- **support_status** – Support status of resource type
- **type_name** – Resource type's name (regular expression allowed)
- **heat_version** – Heat version
- **with_description** – Either return resource type description or not

list_services(**args*, ***kwargs*)

list_software_configs(**args*, ***kwargs*)

list_software_deployments(**args*, ***kwargs*)

list_stack_resources(**args*, ***kwargs*)

list_stacks(**args*, ***kwargs*)

Returns attributes of all stacks.

It supports pagination (`limit` and `marker`), sorting (`sort_keys` and `sort_dir`) and filtering (`filters`) of the results.

Parameters:
- **cnxt** – RPC context
- **limit** – the number of stacks to list (integer or string)
- **marker** – the ID of the last item in the previous page
- **sort_keys** – an array of fields used to sort the list
- **sort_dir** – the direction of the sort ('asc' or 'desc')
- **filters** – a dict with attribute:value to filter the list
- **tenant_safe** – if true, scope the request by the current tenant
- **show_deleted** – if true, show soft-deleted stacks
- **show_nested** – if true, show nested stacks

- **show_hidden** – if true, show hidden stacks
- **tags** – show stacks containing these tags, combine multiple tags using the boolean AND expression
- **tags_any** – show stacks containing these tags, combine multiple tags using the boolean OR expression
- **not_tags** – show stacks not containing these tags, combine multiple tags using the boolean AND expression
- **not_tags_any** – show stacks not containing these tags, combine multiple tags using the boolean OR expression

Returns: a list of formatted stacks

list_template_functions(**args*, ***kwargs*)

list_template_versions(**args*, ***kwargs*)

metadata_software_deployments(**args*, ***kwargs*)

preview_stack(**args*, ***kwargs*)

Simulate a new stack using the provided template.

Note that at this stage the template has already been fetched from the heat-api process if using a template-url.

Parameters:

- **cnxt** – RPC context.
- **stack_name** – Name of the stack you want to create.
- **template** – Template of stack you want to create.
- **params** – Stack Input Params
- **files** – Files referenced from the template
- **args** – Request parameters/args passed from API
- **environment_files** (*list or None*) – optional ordered list of environment file names included in the files dict

preview_update_stack(**args*, ***kwargs*)

Shows the resources that would be updated.

The preview_update_stack method shows the resources that would be changed with an update to an existing stack based on the provided template and parameters. See update_stack for description of parameters.

This method *cannot* guarantee that an update will have the actions specified because resource plugins can influence changes/replacements at runtime.

Note that at this stage the template has already been fetched from the heat-api process if using a template-url.

reset(**args*, ***kwargs*)

reset_stack_status(**args*, ***kwargs*)

resource_mark_unhealthy(**args*, ***kwargs*)

Mark the resource as healthy or unhealthy.

> Put the resource in CHECK_FAILED state if 'mark_unhealthy' is true. Put the resource in CHECK_COMPLETE if 'mark_unhealthy' is false and the resource is in CHECK_FAILED state. Otherwise, make no change.

Parameters:
- **mark_unhealthy** – indicates whether the resource is unhealthy.
- **resource_status_reason** – reason for health change.

resource_schema(**args*, ***kwargs*)

Return the schema of the specified type.

Parameters:
- **cnxt** – RPC context.
- **type_name** – Name of the resource type to obtain the schema of.
- **with_description** – Return result with description or not.

resource_signal(**args*, ***kwargs*)

Calls resource's signal for the specified resource.

Parameters: **sync_call** – indicates whether a synchronized call behavior is expected. This is reserved for CFN WaitCondition implementation.

service_manage_cleanup(**args*, ***kwargs*)

service_manage_report(**args*, ***kwargs*)

set_stack_and_resource_to_failed(**args*, ***kwargs*)

set_watch_state(**args*, ***kwargs*)

Temporarily set the state of a given watch.

Parameters:
- **cnxt** – RPC context.
- **watch_name** – Name of the watch.
- **state** – State (must be one defined in WatchRule class.

show_output(**args*, ***kwargs*)

Returns dict with specified output key, value and description.

Parameters:
- **cntx** – RPC context.
- **stack_identity** – Name of the stack you want to see.
- **output_key** – key of desired stack output.

Returns: dict with output key, value and description in defined format.

show_snapshot(*args, **kwargs)

show_software_config(*args, **kwargs)

show_software_deployment(*args, **kwargs)

show_stack(*args, **kwargs)

Return detailed information about one or all stacks.

Parameters:
- **cnxt** – RPC context.
- **stack_identity** – Name of the stack you want to show, or None to show all
- **resolve_outputs** – If True, outputs for given stack/stacks will be resolved

show_watch(*args, **kwargs)

Return the attributes of one watch/alarm.

Parameters:
- **cnxt** – RPC context.
- **watch_name** – Name of the watch you want to see, or None to see all.

show_watch_metric(*args, **kwargs)

Return the datapoints for a metric.

Parameters:
- **cnxt** – RPC context.
- **metric_namespace** – Name of the namespace you want to see, or None to see all.
- **metric_name** – Name of the metric you want to see, or None to see all.

signal_software_deployment(*args, **kwargs)

stack_cancel_update(**args*, ***kwargs*)

Cancel currently running stack update.

Parameters:
- **cnxt** – RPC context.
- **stack_identity** – Name of the stack for which to cancel update.
- **cancel_with_rollback** – Force rollback when cancel update.

stack_check(**args*, ***kwargs*)

Handle request to perform a check action on a stack.

stack_list_snapshots(**args*, ***kwargs*)

stack_restore(**args*, ***kwargs*)

stack_resume(**args*, ***kwargs*)

Handle request to perform a resume action on a stack.

stack_snapshot(**args*, ***kwargs*)

stack_suspend(**args*, ***kwargs*)

Handle request to perform suspend action on a stack.

start(**args*, ***kwargs*)

stop(**args*, ***kwargs*)

update_software_deployment(**args*, ***kwargs*)

update_stack(**args*, ***kwargs*)

Update an existing stack based on the provided template and params.

Note that at this stage the template has already been fetched from the heat-api process if using a template-url.

Parameters:
- **cnxt** – RPC context.
- **stack_identity** – Name of the stack you want to create.
- **template** – Template of stack you want to create.
- **params** – Stack Input Params
- **files** – Files referenced from the template
- **args** – Request parameters/args passed from API
- **environment_files** (*list or None*) – optional ordered list of environment file names included in the files dict
- **template_id** – the ID of a pre-stored template in the DB

`validate_template`(**args*, ***kwargs*)

Check the validity of a template.

Checks, so far as we can, that a template is valid, and returns information about the parameters suitable for producing a user interface through which to specify the parameter values.

Parameters:
- **cnxt** – RPC context.
- **template** – Template of stack you want to create.
- **params** – Stack Input Params
- **files** – Files referenced from the template
- **environment_files** (*list or None*) – optional ordered list of environment file names included in the files dict
- **show_nested** – if True, any nested templates will be checked
- **ignorable_errors** – List of error_code to be ignored as part of validation

`wait`(**args*, ***kwargs*)

Wait for a service to shut down.

class `heat.engine.service.ThreadGroupManager`

Bases: `object`

`add_msg_queue`(*stack_id*, *msg_queue*)

`add_timer`(*stack_id*, *func*, **args*, ***kwargs*)

Define a periodic task in the stack threadgroups.

The task is run in a separate greenthread.

Periodicity is cfg.CONF.periodic_interval

remove_msg_queue(*gt*, *stack_id*, *msg_queue*)

send(*stack_id*, *message*)

start(*stack_id*, *func*, **args*, ***kwargs*)

Run the given method in a sub-thread.

start_with_acquired_lock(*stack*, *lock*, *func*, **args*, ***kwargs*)

Run the given method in a sub-thread with an existing stack lock.

Release the provided lock when the thread finishes.

Parameters:
- **stack** (*heat.engine.parser.Stack*) – Stack to be operated on
- **lock** (*heat.engine.stack_lock.StackLock*) – The acquired stack lock
- **func** (*function or instancemethod*) – Callable to be invoked in sub-thread
- **args** – Args to be passed to func
- **kwargs** – Keyword-args to be passed to func

start_with_lock(*cnxt*, *stack*, *engine_id*, *func*, **args*, ***kwargs*)

Run the method in sub-thread after acquiring the stack lock.

Release the lock when the thread finishes.

Parameters:
- **cnxt** – RPC context
- **stack** (*heat.engine.parser.Stack*) – Stack to be operated on
- **engine_id** – The UUID of the engine/worker acquiring the lock
- **func** (*function or instancemethod*) – Callable to be invoked in sub-thread
- **args** – Args to be passed to func
- **kwargs** – Keyword-args to be passed to func.

stop(*stack_id*, *graceful=False*)

Stop any active threads on a stack.

stop_timers(*stack_id*)

The heat.engine.service_software_config Module

class **heat.engine.service_software_config.SoftwareConfigService**(*threads=1000*)

Bases: oslo_service.service.Service

check_software_deployment(*cnxt*, *deployment_id*, *timeout*)

create_software_config(*cnxt*, *group*, *name*, *config*, *inputs*, *outputs*, *options*)

create_software_deployment(*cnxt*, *server_id*, *config_id*, *input_values*, *action*, *status*, *status_reason*, *stack_user_project_id*, *deployment_id=None*)

delete_software_config(*cnxt*, *config_id*)

delete_software_deployment(*cnxt*, *deployment_id*)

list_software_configs(*cnxt*, *limit=None*, *marker=None*, *tenant_safe=True*)

list_software_deployments(*cnxt*, *server_id*)

metadata_software_deployments(*cnxt*, *server_id*)

show_software_config(*cnxt*, *config_id*)

show_software_deployment(*cnxt*, *deployment_id*)

signal_software_deployment(*cnxt*, *deployment_id*, *details*, *updated_at*)

update_software_deployment(*cnxt*, *deployment_id*, *config_id*, *input_values*, *output_values*, *action*, *status*, *status_reason*, *updated_at*)

The heat.engine.service_stack_watch Module

class **heat.engine.service_stack_watch.StackWatch(*thread_group_mgr*)**

Bases: object

check_stack_watches(*sid*)

periodic_watcher_task(*sid*)

Evaluate all watch-rules defined for stack ID.

Periodic task, created for each stack, triggers watch-rule evaluation for all rules defined for the stack sid = stack ID.

start_watch_task(*stack_id*, *cnxt*)

The heat.engine.stack Module

exception **heat.engine.stack.ForcedCancel(*with_rollback=True*)**

Bases: exceptions.Exception

Exception raised to cancel task execution.

class **heat.engine.stack.Stack(*context*, *stack_name*, *tmpl*, *stack_id=None*, *action=None*, *status=None*, *status_reason=''*, *timeout_mins=None*, *resolve_data=True*, *disable_rollback=True*, *parent_resource=None*, *owner_id=None*, *adopt_stack_data=None*, *stack_user_project_id=None*, *created_time=None*, *updated_time=None*, *user_creds_id=None*, *tenant_id=None*, *use_stored_context=False*, *username=None*, *nested_depth=0*, *strict_validate=True*, *convergence=False*, *current_traversal=None*, *tags=None*, *prev_raw_template_id=None*, *current_deps=None*, *cache_data=None*, *resource_validate=True*,

service_check_defer=False, deleted_time=None)

Bases: _abcoll.Mapping

ACTIONS = (*'CREATE', 'DELETE', 'UPDATE', 'ROLLBACK', 'SUSPEND', 'RESUME', 'ADOPT', 'SNAPSHOT', 'CHECK', 'RESTORE'*)

ADOPT = '*ADOPT*'

CHECK = '*CHECK*'

COMPLETE = '*COMPLETE*'

CREATE = '*CREATE*'

DELETE = '*DELETE*'

FAILED = '*FAILED*'

IN_PROGRESS = '*IN_PROGRESS*'

RESTORE = '*RESTORE*'

RESUME = '*RESUME*'

ROLLBACK = '*ROLLBACK*'

SNAPSHOT = '*SNAPSHOT*'

STATUSES = (*'IN_PROGRESS', 'FAILED', 'COMPLETE'*)

SUSPEND = '*SUSPEND*'

UPDATE = '*UPDATE*'

access_allowed(*credential_id, resource_name*)

Is credential_id authorised to access resource by resource_name.

add_resource(*resource*)

Insert the given resource into the stack.

adopt(**args, **kwargs*)

Adopt existing resources into a new stack.

cache_data_reference_id(*resource_name*)

cache_data_resource_all_attributes(*resource_name*)

cache_data_resource_attribute(*resource_name, attribute_key*)

check(**args, **kwargs*)

converge_stack(**args, **kwargs*)

Update the stack template and trigger convergence for resources.

convergence_dependencies

create(**args*, ***kwargs*)

Create the stack and all of the resources.

create_stack_user_project_id(**args*, ***kwargs*)

db_active_resources_get()

db_resource_get(*name*)

delete(**args*, ***kwargs*)

Delete all of the resources, and then the stack itself.

The action parameter is used to differentiate between a user initiated delete and an automatic stack rollback after a failed create, which amount to the same thing, but the states are recorded differently.

Note abandon is a delete where all resources have been set to a RETAIN deletion policy, but we also don't want to delete anything required for those resources, e.g the stack_user_project.

delete_snapshot(**args*, ***kwargs*)

Remove a snapshot from the backends.

dependencies

dispatch_event(*ev*)

env

This is a helper to allow resources to access stack.env.

get_availability_zones()

static **get_dep_attrs**(*resources*, *outputs*, *resource_name*, *value_sec*)

Return the attributes of the specified resource that are referenced.

Return an iterator over any attributes of the specified resource that are referenced.

get_kwargs_for_cloning(*keep_status=False*, *only_db=False*)

Get common kwargs for calling Stack() for cloning.

The point of this method is to reduce the number of places that we need to update when a kwarg to Stack.__init__() is modified. It is otherwise easy to forget an option and cause some unexpected error if this option is lost.

Note: - This doesn't return the args(name, template) but only the kwargs. - We often want to start 'fresh' so don't want to maintain the old

> status, action and status_reason.

- We sometimes only want the DB attributes.

has_cache_data(*resource_name*)

has_timed_out()

Returns True if this stack has timed-out.

identifier()

Return an identifier for this stack.

iter_resources(*nested_depth=0*, *filters=None*)

Iterates over all the resources in a stack.

Iterating includes nested stacks up to nested_depth levels below.

classmethod **load**(*context*, *stack_id=None*, *stack=None*, *show_deleted=True*, *use_stored_context=False*, *force_reload=False*, *cache_data=None*, *resolve_data=True*, *service_check_defer=False*, *resource_validate=True*)

Retrieve a Stack from the database.

classmethod **load_all**(*context*, *limit=None*, *marker=None*, *sort_keys=None*, *sort_dir=None*, *filters=None*, *tenant_safe=True*, *show_deleted=False*, *resolve_data=True*, *show_nested=False*, *show_hidden=False*, *tags=None*, *tags_any=None*, *not_tags=None*, *not_tags_any=None*)

mark_complete()

Mark the update as complete.

This currently occurs when all resources have been updated; there may still be resources being cleaned up, but the Stack should now be in service.

object_path_in_stack()

Return stack resources and stacks in path from the root stack.

If this is not nested return (None, self), else return stack resources and stacks in path from the root stack and including this stack.

Returns: a list of (stack_resource, stack) tuples.

output(args*, ***kwargs*)**

Get the value of the specified stack output.

parent_resource

Dynamically load up the parent_resource.

Note: this should only be used by “Fn::ResourceFacade”

path_in_stack()

Return tuples of names in path from the root stack.

If this is not nested return (None, self.name), else return tuples of names (stack_resource.name, stack.name) in path from the root stack and including this stack.

Returns: a list of (string, string) tuples.

persist_state_and_release_lock(*engine_id*)

Persist stack state to database and release stack lock

prepare_abandon(args*, ***kwargs*)**

preview_resources()

Preview the stack with all of the resources.

purge_db()

Cleanup database after stack has completed/failed.

- Delete the resources from DB.

2. If the stack failed, update the current_traversal to empty string so that the resource workers bail out. 3. Delete previous raw template if stack completes successfully. 4. Deletes all sync points. They are no longer needed after stack

> has completed/failed.

- Delete the stack if the action is DELETE.

register_access_allowed_handler(*credential_id*, *handler*)

Register an authorization handler function.

Register a function which determines whether the credentials with a given ID can have access to a named resource.

remove_resource(*resource_name*)

Remove the resource with the specified name.

requires_deferred_auth()

Determine whether to perform API requests with deferred auth.

Returns whether this stack may need to perform API requests during its lifecycle using the configured deferred authentication method.

reset_dependencies()

reset_resource_attributes()

resolve_static_data(*snippet*, *path*='')

resource_by_refid(*refid*)

Return the resource in this stack with the specified refid.

Returns: resource in this stack with the specified refid, or None if not found.

resource_get(*name*)

Return a stack resource, even if not in the current template.

resources

restart_resource(*resource_name*)

Restart the resource specified by resource_name.

stop resource_name and all that depend on it start resource_name and all that depend on it

restore(**args*, ***kwargs*)

Restore the given snapshot.

Invokes handle_restore on all resources.

resume(**args*, ***kwargs*)

Resume the stack.

Invokes handle_resume for all stack resources.

Waits for all resources to become RESUME_COMPLETE then declares the stack RESUME_COMPLETE. Note the default implementation for all resources is to do nothing other than move to RESUME_COMPLETE, so the resources must implement handle_resume for this to have any effect.

rollback()

root_stack_id()

set_parent_stack(*parent_stack*)

set_stack_user_project_id(*project_id*)

snapshot(**args*, ***kwargs*)

Snapshot the stack, invoking handle_snapshot on all resources.

stack_task(**args*, ***kwargs*)

A task to perform an action on the stack.

All of the resources are traversed in forward or reverse dependency order.

Parameters:
- **action** – action that should be executed with stack resources
- **reverse** – define if action on the resources need to be executed in reverse order (resources - first and then res dependencies)
- **post_func** – function that need to be executed after

action complete on the stack :param aggregate_exceptions: define if

exceptions should be aggregated :param pre_completion_func: function that need to be executed right before action completion. Uses stack ,action, status and reason as input parameters

state

Returns state, tuple of action, status.

state_set(**args*, ***kwargs*)

Update the stack state.

store(**args*, ***kwargs*)

Store the stack in the database and return its ID.

If self.id is set, we update the existing stack.

stored_context()

supports_check_action()

suspend(**args*, ***kwargs*)

Suspend the stack.

Invokes handle_suspend for all stack resources.

Waits for all resources to become SUSPEND_COMPLETE then declares the stack SUSPEND_COMPLETE. Note the default implementation for all resources is to do nothing other than move to SUSPEND_COMPLETE, so the resources must implement handle_suspend for this to have any effect.

tags

time_elapsed()

Time elapsed in seconds since the stack operation started.

time_remaining()

Time left before stack times out.

timeout_secs()

Return the stack action timeout in seconds.

total_resources(*stack_id=None*)

Return the total number of resources in a stack.

Includes nested stacks below.

update(**args*, ***kwargs*)

Update the stack.

Compare the current stack with newstack, and where necessary create/update/delete the resources until this stack aligns with newstack.

Note update of existing stack resources depends on update being implemented in the underlying resource types

Update will fail if it exceeds the specified timeout. The default is 60 minutes, set in the constructor

update_task(**args*, ***kwargs*)

validate(**args*, ***kwargs*)

Validates the stack.

worker_client

Return a client for making engine RPC calls.

heat.engine.stack.reset_state_on_error(*func*)

The heat.engine.stack_lock Module

class **heat.engine.stack_lock.StackLock(*context*, *stack_id*, *engine_id*)**

Bases: object

acquire(*retry=True*)

Acquire a lock on the stack.

Parameters: **retry** (*boolean*) – When True, retry if lock was released while stealing.

get_engine_id()

release()

Release a stack lock.

thread_lock(args*, ***kwds*)**

Acquire a lock and release it only if there is an exception.

The release method still needs to be scheduled to be run at the end of the thread using the Thread.link method.

Parameters: **retry** (*boolean*) – When True, retry if lock was released while stealing.

try_acquire()

Try to acquire a stack lock.

Don’t raise an ActionInProgress exception or try to steal lock.

try_thread_lock(args*, ***kwds*)**

Similar to thread_lock, but acquire the lock using try_acquire.

Only release it upon any exception after a successful acquisition.

The heat.engine.support Module

class **heat.engine.support.SupportStatus(*status='SUPPORTED', message=None, version=None, previous_status=None*)**

Bases: object

to_dict()

validate()

heat.engine.support.is_valid_status(*status*)

The heat.engine.sync_point Module

heat.engine.sync_point.create(*context, entity_id, traversal_id, is_update, stack_id*)

Creates a sync point entry in DB.

heat.engine.sync_point.delete_all(*context, stack_id, traversal_id*)

Deletes all sync points of a stack associated with a traversal_id.

heat.engine.sync_point.deserialize_input_data(*db_input_data*)

heat.engine.sync_point.get(*context, entity_id, traversal_id, is_update*)

Retrieves a sync point entry from DB.

heat.engine.sync_point.make_key(components*)**

heat.engine.sync_point.serialize_input_data(*input_data*)

heat.engine.sync_point.sync(*cnxt, entity_id, current_traversal, is_update, propagate, predecessors, new_data*)

heat.engine.sync_point.update_input_data(*context*, *entity_id*, *current_traversal*, *is_update*, *atomic_key*, *input_data*)

The heat.engine.template Module

class **heat.engine.template.Template**(*template*, *template_id=None*, *files=None*, *env=None*)

Bases: _abcoll.Mapping

A stack template.

add_resource(*definition*, *name=None*)

Add a resource to the template.

The resource is passed as a ResourceDefinition object. If no name is specified, the name from the ResourceDefinition should be used.

condition_functions = *{}*

classmethod **create_empty_template**(*version=('heat_template_version', '2015-04-30')*, *from_template=None*)

Create an empty template.

Creates a new empty template with given version. If version is not provided, a new empty HOT template of version “2015-04-30” is returned.

Parameters: **version** – A tuple containing version header of the template: version key and value. E.g. (“heat_template_version”, “2015-04-30”) :returns: A new empty template.

files

functions = *{}*

get_section_name(*section*)

Return a correct section name.

***classmethod* load(*context*, *template_id*, *t=None*)**

Retrieve a Template with the given ID from the database.

param_schemata(*param_defaults=None*)

Return a dict of parameters.Schema objects for the parameters.

parameters(*stack_identifier*, *user_params*, *param_defaults=None*)

Return a parameters.Parameters object for the stack.

parse(*stack*, *snippet*, *path=''*)

parse_condition(*stack*, *snippet*)

remove_all_resources()

Remove all the resources from the template.

remove_resource(*name*)

Remove a resource from the template.

resource_definitions(*stack*)

Return a dictionary of ResourceDefinition objects.

store(*context*)

Store the Template in the database and return its ID.

validate()

Validate the template.

Validates the top-level sections of the template as well as syntax inside select sections. Some sections are not checked here but in code parts that are responsible for working with the respective sections (e.g. parameters are check by parameters schema class).

validate_resource_definitions(*stack*)

Check section's type of ResourceDefinitions.

classmethod **validate_resource_key_type(*key*, *valid_types*, *typename*, *allowed_keys*, *rsrc_name*, *rsrc_data*)**

Validation type of the specific resource key.

Used in validate_resource_definition and check correctness of key's type.

validate_section(*section*, *sub_section*, *data*, *allowed_keys*)

The heat.engine.template_files Module

class **heat.engine.template_files.ReadOnlyDict**

Bases: dict

class **heat.engine.template_files.TemplateFiles(*files*)**

Bases: _abcoll.Mapping

store(*ctxt*)

update(*files*)

The heat.engine.timestamp Module

class **heat.engine.timestamp.Timestamp(*db_fetch*, *attribute*)**

Bases: object

A descriptor for writing a timestamp to the database.

The heat.engine.translation Module

***class* heat.engine.translation.TranslationRule(*properties*, *rule*, *translation_path*, *value=None*, *value_name=None*, *value_path=None*, *client_plugin=None*, *finder=None*, *entity=None*)**

Bases: object

Translating mechanism one properties to another.

Mechanism uses list of rules, each defines by this class, and can be executed. Working principe: during resource creating after properties defining resource take list of rules, specified by method translation_rules, which should be overloaded for each resource, if it's needed, and execute each rule using translate_properties method. Next operations are allowed:

- **ADD. This rule allows to add some value to list-type properties. Only**

  list-type values can be added to such properties. Using for other cases is prohibited and will be returned with error.

- **REPLACE. This rule allows to replace some property value to another. Used**

  for all types of properties. Note, that if property has list type, then value will be replaced for all elements of list, where it needed. If element in such property must be replaced by value of another element of this property, value_name must be defined.

- **DELETE. This rule allows to delete some property. If property has list**

  type, then deleting affects value in all list elements.

- **RESOLVE. This rule allows to resolve some property using client and**

  the finder function. Finders may require an additional entity key.

ADD = '*Add*'

DELETE = '*Delete*'

REPLACE = '*Replace*'

RESOLVE = '*Resolve*'

RULE_KEYS = *('Add', 'Replace', 'Delete', 'Resolve')*

execute_rule(*client_resolve=True*)

translate_property(*path*, *data*, *return_value=False*, *value=None*, *value_data=None*, *value_key=None*, *client_resolve=True*)

validate()

The heat.engine.update Module

***class* heat.engine.update.StackUpdate(*existing_stack*, *new_stack*, *previous_stack*, *rollback=False*)**

Bases: object

A Task to perform the update of an existing stack to a new template.

dependencies()

Return the Dependencies graph for the update.

Returns a Dependencies object representing the dependencies between update operations to move from an existing stack definition to a new one.

preview()

The heat.engine.watchrule Module

***class* heat.engine.watchrule.WatchRule(*context*, *watch_name*, *rule*, *stack_id=None*, *state='NODATA'*, *wid=None*, *watch_data=None*, *last_evaluated=datetime.datetime(2016, 8, 5, 3, 24, 20, 664)*)**

Bases: object

ACTION_MAP = *{'NODATA': 'InsufficientDataActions', 'ALARM': 'AlarmActions', 'NORMAL': 'OKActions'}*

ALARM = '*ALARM*'

CEILOMETER_CONTROLLED = '*CEILOMETER_CONTROLLED*'

NODATA = '*NODATA*'

NORMAL = '*NORMAL*'

SUSPENDED = '*SUSPENDED*'

WATCH_STATES = *('ALARM', 'NORMAL', 'NODATA', 'SUSPENDED', 'CEILOMETER_CONTROLLED')*

create_watch_data(*data*)

created_at = *None*

destroy()

Delete the watchrule from the database.

do_Average()

do_Maximum()

do_Minimum()

do_SampleCount()

Count all samples within the specified period.

do_Sum()

do_data_cmp(*data*, *threshold*)

evaluate()

get_alarm_state()

get_details()

classmethod **load**(*context*, *watch_name=None*, *watch=None*)

Load the watchrule object.

The object can be loaded either from the DB by name or from an existing DB object.

rule_actions(*new_state*)

run_rule()

set_watch_state(*state*)

Temporarily set the watch state.

Returns: list of functions to be scheduled in the stack ThreadGroup for the specified state.

state_set(*state*)

Persistently store the watch state.

store()

Store the watchrule in the database and return its ID.

If self.id is set, we update the existing rule.

updated_at = *None*

heat.engine.watchrule.rule_can_use_sample(*wr*, *stats_data*)

The heat.engine.worker Module

class **heat.engine.worker.WorkerService**(*host*, *topic*, *engine_id*, *thread_group_mgr*)

Bases: oslo_service.service.Service

Service that has 'worker' actor in convergence.

This service is dedicated to handle internal messages to the 'worker' (a.k.a. 'converger') actor in convergence. Messages on this bus will use the 'cast' rather than 'call' method to anycast the message to an engine that will handle it asynchronously. It won't wait for or expect replies from these messages.

RPC_API_VERSION = '*1.3*'

cancel_check_resource(**args*, ***kwargs*)

Cancel check_resource for given stack.

All the workers running for the given stack will be cancelled.

check_resource(**args*, ***kwargs*)

Process a node in the dependency graph.

The node may be associated with either an update or a cleanup of its associated resource.

reset(**args*, ***kwargs*)

Reset a service in case it received a SIGHUP.

start(**args*, ***kwargs*)

stop(**args*, ***kwargs*)

wait(**args*, ***kwargs*)

Wait for a service to shut down.

The heat.hacking.checks Module

heat.hacking.checks.check_python3_no_iteritems(*logical_line*)

heat.hacking.checks.check_python3_no_iterkeys(*logical_line*)

heat.hacking.checks.check_python3_no_itervalues(*logical_line*)

heat.hacking.checks.factory(*register*)

heat.hacking.checks.no_log_warn(*logical_line*)

Disallow 'LOG.warn('

https://bugs.launchpad.net/tempest/+bug/1508442

Heat301

The heat.httpd.heat_api Module

WSGI script for heat-api.

Script for running heat-api under Apache2.

heat.httpd.heat_api.init_application()

The heat.httpd.heat_api_cfn Module

WSGI script for heat-api-cfn.

Script for running heat-api-cfn under Apache2.

heat.httpd.heat_api_cfn.init_application()

The heat.httpd.heat_api_cloudwatch Module

WSGI script for heat-api-cloudwatch.

Script for running heat-api-cloudwatch under Apache2.

heat.httpd.heat_api_cloudwatch.init_application()

The heat.objects.base Module

Heat common internal object model

***class* heat.objects.base.HeatObject(*context=None*, ***kwargs*)**

Bases: oslo_versionedobjects.base.VersionedObject

OBJ_PROJECT_NAMESPACE = '*heat*'

VERSION = '*1.0*'

***class* heat.objects.base.HeatObjectRegistry**

Bases: oslo_versionedobjects.base.VersionedObjectRegistry

The heat.objects.event Module

Event object.

***class* heat.objects.event.Event(*context=None*, ***kwargs*)**

Bases: heat.objects.base.HeatObject, oslo_versionedobjects.base.VersionedObjectDictCompat

***classmethod* count_all_by_stack(*context*, *stack_id*)**

***classmethod* create(*context*, *values*)**

**fields = {'*resource_name*': *String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True)*, '*stack_id*': *String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False)*, '*resource_status_reason*': *String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True)*, '*physical_resource_id*': *String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True)*, '*resource_properties*': *Json(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True)*, '*updated_at*': *DateTime(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True)*, '*resource_type*': *String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True)*, '*resource_status*': *String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True)*, '*id*': *Integer(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False)*, '*resource_action*': *String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True)*, '*uuid*': *String(default=<class*

'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'created_at': DateTime(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False)}

classmethod `get_all`(***context***)

classmethod `get_all_by_stack`(***context***, ***stack_id***, ****kwargs***)

classmethod `get_all_by_tenant`(***context***, ****kwargs***)

classmethod `get_by_id`(***context***, ***event_id***)

`identifier`(***stack_identifier***)

Return a unique identifier for the event.

The `heat.objects.fields` Module

class `heat.objects.fields.Json`

Bases: `oslo_versionedobjects.fields.FieldType`

`coerce`(***obj***, ***attr***, ***value***)

`from_primitive`(***obj***, ***attr***, ***value***)

`to_primitive`(***obj***, ***attr***, ***value***)

class `heat.objects.fields.JsonField`(****kwargs***)

Bases: `oslo_versionedobjects.fields.AutoTypedField`

`AUTO_TYPE` = ***<heat.objects.fields.Json object at 0x7f507d2ed050>***

class `heat.objects.fields.ListField`(****kwargs***)

Bases: `oslo_versionedobjects.fields.AutoTypedField`

`AUTO_TYPE` = ***<oslo_versionedobjects.fields.List object at 0x7f507d2ed150>***

The heat.objects.raw_template Module

RawTemplate object.

class **heat.objects.raw_template.RawTemplate**(***context=None***, *****kwargs***)

Bases: heat.objects.base.HeatObject, oslo_versionedobjects.base.VersionedObjectDictCompat, oslo_versionedobjects.base.ComparableVersionedObject

VERSION = '*1.1*'

classmethod **create**(***context***, ***values***)

classmethod **delete**(***context***, ***template_id***)

classmethod **encrypt_hidden_parameters**(***tmpl***)

environment

fields = *{'files': Json(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'files_id': Integer(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'environment': Json(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False), 'template': Json(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False), 'id': Integer(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False)}*

files

files_id

classmethod **get_by_id**(***context***, ***template_id***)

id

template

classmethod **update_by_id**(***context***, ***template_id***, ***values***)

The heat.objects.raw_template_files Module

RawTemplateFiles object.

class **heat.objects.raw_template_files.RawTemplateFiles(*context=None*, ***kwargs*)**

Bases: heat.objects.base.HeatObject, oslo_versionedobjects.base.VersionedObjectDictCompat, oslo_versionedobjects.base.ComparableVersionedObject

VERSION = '*1.0*'

classmethod **create(*context*, *values*)**

fields = *{'files': Json(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False), 'id': Integer(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False)}*

files

id

The heat.objects.resource Module

Resource object.

class **heat.objects.resource.Resource(*context=None*, ***kwargs*)**

Bases: heat.objects.base.HeatObject, oslo_versionedobjects.base.VersionedObjectDictCompat, oslo_versionedobjects.base.ComparableVersionedObject

classmethod **create(*context*, *values*)**

classmethod `delete`(*context*, *resource_id*)

static `encrypt_properties_data`(*data*)

classmethod `exchange_stacks`(*context*, *resource_id1*, *resource_id2*)

`fields` = *{'status': String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'updated_at': DateTime(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'current_template_id': Integer(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False), 'root_stack_id': String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'data': List(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'replaces': Integer(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'replaced_by': Integer(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'id': Integer(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False), 'engine_id': String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'name': String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'status_reason': String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'needed_by': List(default=None,nullable=True), 'stack_id': String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False), 'physical_resource_id': String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'atomic_key': Integer(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'rsrc_metadata': Json(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'properties_data_encrypted': Boolean(default=False,nullable=False), 'properties_data': Json(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'requires': List(default=None,nullable=True), 'action': String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'uuid': String(default=<class*

'*oslo_versionedobjects.fields.UnspecifiedDefault*'>,*nullable=False*),
'*created_at*': *DateTime(default=<class*
'*oslo_versionedobjects.fields.UnspecifiedDefault*'>,*nullable=False*)}

classmethod `get_all`(***context***)

classmethod `get_all_active_by_stack`(***context***, ***stack_id***)

classmethod `get_all_by_root_stack`(***context***, ***stack_id***, ***filters***)

classmethod `get_all_by_stack`(***context***, ***stack_id***, ***filters=None***)

classmethod `get_by_name_and_stack`(***context***, ***resource_name***, ***stack_id***)

classmethod `get_by_physical_resource_id`(***context***,
physical_resource_id)

classmethod `get_obj`(***context***, ***resource_id***, ***refresh=False***)

classmethod `purge_deleted`(***context***, ***stack_id***)

`refresh`()

`select_and_update`(***values***, ***expected_engine_id=None***, ***atomic_key=0***)

`update_and_save`(***values***)

classmethod `update_by_id`(***context***, ***resource_id***, ***values***)

`update_metadata`(***metadata***)

`heat.objects.resource.retry_on_conflict`(***func***)

The heat.objects.resource_data Module

ResourceData object.

class `heat.objects.resource_data.ResourceData`(***context=None***, *****kwargs***)

Bases: `heat.objects.base.HeatObject`, `oslo_versionedobjects.base.VersionedObjectDictCompat`, `oslo_versionedobjects.base.ComparableVersionedObject`

classmethod `delete`(***resource***, ***key***)

`fields` = {'*redact*': *Boolean(default=<class*

'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'decrypt_method': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'updated_at': DateTime(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'key':
String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'value': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'id':
Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'created_at': DateTime(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'resource_id': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False)}

classmethod `get_all`(***resource***, ****args***, *****kwargs***)

classmethod `get_by_key`(***context***, ***resource_id***, ***key***)

classmethod `get_obj`(***resource***, ***key***)

classmethod `get_val`(***resource***, ***key***)

classmethod `set`(***resource***, ***key***, ***value***, ****args***, *****kwargs***)

The `heat.objects.service` Module

Service object.

class `heat.objects.service.Service`(***context=None***, *****kwargs***)

Bases: `heat.objects.base.HeatObject`, `oslo_versionedobjects.base.VersionedObjectDictCompat`, `oslo_versionedobjects.base.ComparableVersionedObject`

classmethod `create`(***context***, ***values***)

classmethod `delete`(***context***, ***service_id***, ***soft_delete=True***)

`fields` = *{'created_at': DateTime(default=<class*
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'report_interval': Integer(default=<class

'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'engine_id': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'deleted_at': DateTime(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'updated_at': DateTime(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'topic': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'hostname': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False), 'id':
String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'host': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'binary': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False)}

classmethod `get_all`(*context*)

classmethod `get_all_by_args`(*context*, *host*, *binary*, *hostname*)

classmethod `get_by_id`(*context*, *service_id*)

classmethod `update_by_id`(*context*, *service_id*, *values*)

The `heat.objects.snapshot` Module

Snapshot object.

class `heat.objects.snapshot.Snapshot`(*context=None*, ***kwargs*)

Bases: `heat.objects.base.HeatObject`, `oslo_versionedobjects.base.VersionedObjectDictCompat`, `oslo_versionedobjects.base.ComparableVersionedObject`

classmethod `create`(*context*, *values*)

classmethod `delete`(*context*, *snapshot_id*)

`fields` = *{'stack_id': String(default=<class*
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'status': String(default=<class

'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'status_reason': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'tenant': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'name': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'data': Json(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'id':
String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'created_at': DateTime(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'updated_at': DateTime(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True)}

classmethod `get_all`(***context***, ***stack_id***)

classmethod `get_snapshot_by_stack`(***context***, ***snapshot_id***, ***stack***)

classmethod `update`(***context***, ***snapshot_id***, ***values***)

The heat.objects.software_config Module

SoftwareConfig object.

class `heat.objects.software_config.SoftwareConfig`(***context=None***, *****kwargs***)

Bases: `heat.objects.base.HeatObject`, `oslo_versionedobjects.base.VersionedObjectDictCompat`, `oslo_versionedobjects.base.ComparableVersionedObject`

classmethod `create`(***context***, ***values***)

classmethod `delete`(***context***, ***config_id***)

`fields` = {*'config': Json(default=<class*
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),

'updated_at': DateTime(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'tenant': String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'id': String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False), 'group': String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'name': String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'created_at': DateTime(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False)}

classmethod `get_all`(***context***, ****kwargs***)

classmethod `get_by_id`(***context***, ***config_id***)

The heat.objects.software_deployment Module

SoftwareDeployment object.

class `heat.objects.software_deployment.SoftwareDeployment`(***context=None***, ****kwargs***)

Bases: `heat.objects.base.HeatObject`, `oslo_versionedobjects.base.VersionedObjectDictCompat`, `oslo_versionedobjects.base.ComparableVersionedObject`

classmethod `create`(***context***, ***values***)

classmethod `delete`(***context***, ***deployment_id***)

`fields` = *{'status': String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'input_values': Json(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'status_reason': String(default=<class*

'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'output_values': Json(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'server_id': String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False), 'updated_at': DateTime(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'config': Object(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False), 'tenant': String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False), 'config_id': String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False), 'id': String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False), 'action': String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'stack_user_project_id': String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'created_at': DateTime(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False)}

classmethod `get_all`(***context, server_id=None***)

classmethod `get_by_id`(***context, deployment_id***)

classmethod `update_by_id`(***context, deployment_id, values***)

Note this is a bit unusual as it returns the object.

Other update_by_id methods return a bool (was it updated).

The `heat.objects.stack` Module

Stack object.

class `heat.objects.stack.Stack`(***context=None, **kwargs***)

Bases: `heat.objects.base.HeatObject`, `oslo_versionedobjects.base.VersionedObjectDictCompat`, `oslo_versionedobjects.base.ComparableVersionedObject`

classmethod `count_all`(*context*, ***kwargs*)

classmethod `count_total_resources`(*context*, *stack_id*)

classmethod `create`(*context*, *values*)

classmethod `delete`(*context*, *stack_id*)

classmethod `encrypt_hidden_parameters`(*tmpl*)

`fields` = {'*current_traversal*': *String(default=<class* '*oslo_versionedobjects.fields.UnspecifiedDefault*'>,*nullable=False),* '*status*': *String(default=<class* '*oslo_versionedobjects.fields.UnspecifiedDefault*'>,*nullable=True),* '*updated_at*': *DateTime(default=<class* '*oslo_versionedobjects.fields.UnspecifiedDefault*'>,*nullable=True),* '*parent_resource_name*': *String(default=<class* '*oslo_versionedobjects.fields.UnspecifiedDefault*'>,*nullable=True),* '*prev_raw_template_id*': *Integer(default=<class* '*oslo_versionedobjects.fields.UnspecifiedDefault*'>,*nullable=False),* '*prev_raw_template*': *Object(default=<class* '*oslo_versionedobjects.fields.UnspecifiedDefault*'>,*nullable=False),* '*backup*': *Boolean(default=<class* '*oslo_versionedobjects.fields.UnspecifiedDefault*'>,*nullable=False),* '*tenant*': *String(default=<class* '*oslo_versionedobjects.fields.UnspecifiedDefault*'>,*nullable=True),* '*raw_template*': *Object(default=<class* '*oslo_versionedobjects.fields.UnspecifiedDefault*'>,*nullable=False),* '*raw_template_id*': *Integer(default=<class* '*oslo_versionedobjects.fields.UnspecifiedDefault*'>,*nullable=False),* '*id*': *String(default=<class* '*oslo_versionedobjects.fields.UnspecifiedDefault*'>,*nullable=False),* '*timeout*': *Integer(default=<class* '*oslo_versionedobjects.fields.UnspecifiedDefault*'>,*nullable=True),* '*stack_user_project_id*': *String(default=<class* '*oslo_versionedobjects.fields.UnspecifiedDefault*'>,*nullable=True),* '*name*': *String(default=<class* '*oslo_versionedobjects.fields.UnspecifiedDefault*'>,*nullable=False),* '*owner_id*': *String(default=<class* '*oslo_versionedobjects.fields.UnspecifiedDefault*'>,*nullable=True),* '*status_reason*': *String(default=<class* '*oslo_versionedobjects.fields.UnspecifiedDefault*'>,*nullable=True),* '*convergence*': *Boolean(default=<class* '*oslo_versionedobjects.fields.UnspecifiedDefault*'>,*nullable=False),* '*username*': *String(default=<class*

'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'current_deps': Json(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'disable_rollback': Boolean(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'deleted_at': DateTime(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'user_creds_id': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'action': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'nested_depth': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'created_at': DateTime(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False)}

classmethod `get_all`(***context***, ***limit=None***, ***sort_keys=None***, ***marker=None***, ***sort_dir=None***, ***filters=None***, ***tenant_safe=True***, ***show_deleted=False***, ***show_nested=False***, ***show_hidden=False***, ***tags=None***, ***tags_any=None***, ***not_tags=None***, ***not_tags_any=None***)

classmethod `get_all_by_owner_id`(***context***, ***owner_id***)

classmethod `get_by_id`(***context***, ***stack_id***, ****kwargs***)

classmethod `get_by_name`(***context***, ***stack_name***)

classmethod `get_by_name_and_owner_id`(***context***, ***stack_name***, ***owner_id***)

classmethod `get_root_id`(***context***, ***stack_id***)

classmethod `get_status`(***context***, ***stack_id***)

Return action and status for the given stack.

`identifier`()

Return an identifier for this stack.

classmethod `persist_state_and_release_lock`(***context***, ***stack_id***, ***engine_id***, ***values***)

`refresh`()

classmethod `select_and_update`(***context***, ***stack_id***, ***values***, ***exp_trvsl=None***)

Update the stack by selecting on traversal ID.

Uses UPDATE ... WHERE (compare and swap) to catch any concurrent update problem.

If the stack is found with given traversal, it is updated.

If there occurs a race while updating, only one will succeed and other will get return value of False.

tags

update_and_save(*values*)

classmethod **update_by_id**(*context*, *stack_id*, *values*)

Update and return (boolean) if it was updated.

Note: the underlying stack_update filters by current_traversal and stack_id.

The heat.objects.stack_lock Module

StackLock object.

class **heat.objects.stack_lock.StackLock**(*context=None*, ***kwargs*)

Bases: heat.objects.base.HeatObject, oslo_versionedobjects.base.VersionedObjectDictCompat, oslo_versionedobjects.base.ComparableVersionedObject

classmethod **create**(*context*, *stack_id*, *engine_id*)

fields = {'*stack_id*': *String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False)*, '*engine_id*': *String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True)*, '*created_at*': *DateTime(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False)*, '*updated_at*': *DateTime(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True)*}

classmethod **get_engine_id**(*context*, *stack_id*)

classmethod release(*context*, *stack_id*, *engine_id*)

classmethod steal(*context*, *stack_id*, *old_engine_id*, *new_engine_id*)

The heat.objects.stack_tag Module

StackTag object.

class heat.objects.stack_tag.StackTag(*context=None*, ***kwargs*)

Bases: heat.objects.base.HeatObject, oslo_versionedobjects.base.VersionedObjectDictCompat, oslo_versionedobjects.base.ComparableVersionedObject

fields = {'*stack_id*': *String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False)*, '*updated_at*': *DateTime(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True)*, '*id*': *Integer(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False)*, '*created_at*': *DateTime(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False)*, '*tag*': *String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True)*}

classmethod get_obj(*context*, *tag*)

class heat.objects.stack_tag.StackTagList(**args*, ***kwargs*)

Bases: heat.objects.base.HeatObject, oslo_versionedobjects.base.ObjectListBase

classmethod delete(*context*, *stack_id*)

fields = {'*objects*': *List(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False)*}

classmethod from_db_object(*context*, *db_tags*)

classmethod get(*context*, *stack_id*)

classmethod set(*context*, *stack_id*, *tags*)

The heat.objects.sync_point Module

SyncPoint object.

class **heat.objects.sync_point.SyncPoint**(***context=None***, *****kwargs***)

Bases: heat.objects.base.HeatObject, oslo_versionedobjects.base.VersionedObjectDictCompat, oslo_versionedobjects.base.ComparableVersionedObject

classmethod **create**(***context***, ***values***)

classmethod **delete_all_by_stack_and_traversal**(***context***, ***stack_id***, ***traversal_id***)

fields = ***{'stack_id': String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False), 'atomic_key': Integer(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False), 'created_at': DateTime(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False), 'entity_id': String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False), 'input_data': Json(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'is_update': Boolean(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False), 'traversal_id': String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False), 'updated_at': DateTime(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True)}***

classmethod **get_by_key**(***context***, ***entity_id***, ***traversal_id***, ***is_update***)

classmethod **update_input_data**(***context***, ***entity_id***, ***traversal_id***, ***is_update***, ***atomic_key***, ***input_data***)

The heat.objects.user_creds Module

UserCreds object.

class **heat.objects.user_creds.UserCreds**(***context=None***, *****kwargs***)

Bases: heat.objects.base.HeatObject, oslo_versionedobjects.base.VersionedObjectDictCompat, oslo_versionedobjects.base.ComparableVersionedObject

auth_url

classmethod **create**(***context***)

created_at

decrypt_method

classmethod **delete**(***context***, ***user_creds_id***)

fields = *{'password': String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'decrypt_method': String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'updated_at': DateTime(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'tenant': String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'auth_url': String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'trust_id': String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'id': String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False), 'trustor_user_id': String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'username': String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'region_name': String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'created_at': DateTime(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),*

'tenant_id': String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True)}

classmethod `get_by_id`(*context*, *user_creds_id*)

id

password

region_name

tenant

tenant_id

trust_id

trustor_user_id

updated_at

username

The `heat.objects.watch_data` Module

WatchData object.

class `heat.objects.watch_data.WatchData`(*context=None*, ***kwargs*)

Bases: `heat.objects.base.HeatObject`, `oslo_versionedobjects.base.VersionedObjectDictCompat`

classmethod `create`(*context*, *values*)

`fields` = *{'watch_rule_id': String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False), 'data': Json(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'id': Integer(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False), 'created_at': DateTime(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False), 'updated_at': DateTime(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True)}*

classmethod `get_all`(*context*)

classmethod `get_all_by_watch_rule_id`(*context*, *watch_rule_id*)

The heat.objects.watch_rule Module

WatchRule object.

class **heat.objects.watch_rule.WatchRule(*context=None*, ***kwargs*)**

Bases: heat.objects.base.HeatObject, oslo_versionedobjects.base.VersionedObjectDictCompat

classmethod **create(*context*, *values*)**

classmethod **delete(*context*, *watch_id*)**

fields = *{'state': String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'stack_id': String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False), 'last_evaluated': DateTime(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'updated_at': DateTime(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'name': String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'rule': Json(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'id': Integer(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False), 'created_at': DateTime(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False), 'watch_data': List(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False)}*

classmethod **get_all(*context*)**

classmethod **get_all_by_stack(*context*, *stack_id*)**

classmethod **get_by_id(*context*, *rule_id*)**

classmethod **get_by_name(*context*, *watch_rule_name*)**

classmethod **update_by_id(*context*, *watch_id*, *values*)**

The heat.rpc.api Module

The heat.rpc.client Module

Client side of the heat engine RPC API.

class **heat.rpc.client.EngineClient**

Bases: object

Client side of the heat engine rpc API.

API version history:

```
1.0 - Initial version.
1.1 - Add support_status argument to list_resource_types()
1.4 - Add support for service list
1.9 - Add template_type option to generate_template()
1.10 - Add support for software config list
1.11 - Add support for template versions list
1.12 - Add with_detail option for stack resources list
1.13 - Add support for template functions list
1.14 - Add cancel_with_rollback option to stack_cancel_update
1.15 - Add preview_update_stack() call
1.16 - Adds version, type_name to list_resource_types()
1.17 - Add files to validate_template
1.18 - Add show_nested to validate_template
1.19 - Add show_output and list_outputs for returning stack outputs
1.20 - Add resolve_outputs to stack show
1.21 - Add deployment_id to create_software_deployment
1.22 - Add support for stack export
1.23 - Add environment_files to create/update/preview/validate
1.24 - Adds ignorable_errors to validate_template
1.25 - list_stack_resource filter update
1.26 - Add mark_unhealthy
1.27 - Add check_software_deployment
1.28 - Add get_environment call
1.29 - Add template_id to create_stack/update_stack
1.30 - Add possibility to resource_type_* return descriptions
1.31 - Add nested_depth to list_events, when nested_depth is specified
```

```
       add root_stack_id to response
1.32 - Add get_files call
```

BASE_RPC_API_VERSION = '*1.0*'

abandon_stack(*ctxt*, *stack_identity*)

Deletes a given stack but resources would not be deleted.

Parameters:
- **ctxt** – RPC context.
- **stack_identity** – Name of the stack you want to abandon.

authenticated_to_backend(*ctxt*)

Validate the credentials in the RPC context.

Verify that the credentials in the RPC context are valid for the current cloud backend.

Parameters: **ctxt** – RPC context.

call(*ctxt*, *msg*, *version=None*, *timeout=None*)

cast(*ctxt*, *msg*, *version=None*)

check_software_deployment(*cnxt*, *deployment_id*, *timeout*)

count_stacks(*ctxt*, *filters=None*, *tenant_safe=True*, *show_deleted=False*, *show_nested=False*, *show_hidden=False*, *tags=None*, *tags_any=None*, *not_tags=None*, *not_tags_any=None*)

Returns the number of stacks that match the given filters.

Parameters:
- **ctxt** – RPC context.
- **filters** – a dict of ATTR:VALUE to match against stacks
- **tenant_safe** – if true, scope the request by the current tenant
- **show_deleted** – if true, count will include the deleted stacks
- **show_nested** – if true, count will include nested stacks
- **show_hidden** – if true, count will include hidden stacks
- **tags** – count stacks containing these tags, combine multiple tags using the boolean AND expression
- **tags_any** – count stacks containing these tags, combine multiple tags using the boolean OR expression
- **not_tags** – count stacks not containing these tags, combine

multiple tags using the boolean AND expression
- **not_tags_any** – count stacks not containing these tags, combine multiple tags using the boolean OR expression

Returns: an integer representing the number of matched stacks

create_software_config(*cnxt*, *group*, *name*, *config*, *inputs=None*, *outputs=None*, *options=None*)

create_software_deployment(*cnxt*, *server_id*, *config_id=None*, *input_values=None*, *action='INIT'*, *status='COMPLETE'*, *status_reason=''*, *stack_user_project_id=None*, *deployment_id=None*)

create_stack(*ctxt*, *stack_name*, *template*, *params*, *files*, *args*, *environment_files=None*)

Creates a new stack using the template provided.

Note that at this stage the template has already been fetched from the heat-api process if using a template-url.

Parameters:
- **ctxt** – RPC context.
- **stack_name** – Name of the stack you want to create.
- **template** – Template of stack you want to create.
- **params** – Stack Input Params/Environment
- **files** – files referenced from the environment.
- **args** – Request parameters/args passed from API
- **environment_files** (*list or None*) – optional ordered list of environment file names included in the files dict

create_watch_data(*ctxt*, *watch_name*, *stats_data*)

Creates data for CloudWatch and WaitConditions.

This could be used by CloudWatch and WaitConditions and treat HA service events like any other CloudWatch.

Parameters:
- **ctxt** – RPC context.
- **watch_name** – Name of the watch/alarm
- **stats_data** – The data to post.

delete_snapshot(*cnxt*, *stack_identity*, *snapshot_id*)

delete_software_config(*cnxt*, *config_id*)

delete_software_deployment(*cnxt*, *deployment_id*)

delete_stack(*ctxt*, *stack_identity*, *cast=True*)

Deletes a given stack.

Parameters:
- **ctxt** – RPC context.
- **stack_identity** – Name of the stack you want to delete.
- **cast** – cast the message or use call (default: True)

describe_stack_resource(*ctxt*, *stack_identity*, *resource_name*, *with_attr=False*)

Get detailed resource information about a particular resource.

Parameters:
- **ctxt** – RPC context.
- **stack_identity** – Name of the stack.
- **resource_name** – the Resource.

describe_stack_resources(*ctxt*, *stack_identity*, *resource_name*)

Get detailed resource information about one or more resources.

Parameters:
- **ctxt** – RPC context.
- **stack_identity** – Name of the stack.
- **resource_name** – the Resource.

export_stack(*ctxt*, *stack_identity*)

Exports the stack data in JSON format.

Parameters:
- **ctxt** – RPC context.
- **stack_identity** – Name of the stack you want to export.

find_physical_resource(*ctxt*, *physical_resource_id*)

Return an identifier for the resource.

Parameters:
- **ctxt** – RPC context.
- **physcial_resource_id** – The physical resource ID to look up.

generate_template(*ctxt*, *type_name*, *template_type='cfn'*)

Generate a template based on the specified type.

Parameters:
- **ctxt** – RPC context.
- **type_name** – The resource type name to generate a template for.
- **template_type** – the template type to generate, cfn or hot.

`get_environment`(*context*, *stack_identity*)

Returns the environment for an existing stack.

Parameters:
- **context** – RPC context
- **stack_identity** – identifies the stack

Return type: dict

`get_files`(*context*, *stack_identity*)

Returns the files for an existing stack.

Parameters:
- **context** – RPC context
- **stack_identity** – identifies the stack

Return type: dict

`get_revision`(*ctxt*)

`get_template`(*ctxt*, *stack_identity*)

Get the template.

Parameters:
- **ctxt** – RPC context.
- **stack_name** – Name of the stack you want to see.

`identify_stack`(*ctxt*, *stack_name*)

Returns the full stack identifier for a single, live stack.

Parameters:
- **ctxt** – RPC context.
- **stack_name** – Name of the stack you want to see, or None to see all

ignore_error_named(*error*, *name*)

Raises the error unless its local name matches the supplied name.

Parameters:
- **error** – Remote raised error to derive the local name from.
- **name** – Name to compare local name to.

list_events(*ctxt*, *stack_identity*, *filters=None*, *limit=None*, *marker=None*, *sort_keys=None*, *sort_dir=None*, *nested_depth=None*)

Lists all events associated with a given stack.

It supports pagination (`limit` and `marker`), sorting (`sort_keys` and `sort_dir`) and filtering(filters) of the results.

Parameters:
- **ctxt** – RPC context.
- **stack_identity** – Name of the stack you want to get events for
- **filters** – a dict with attribute:value to filter the list
- **limit** – the number of events to list (integer or string)
- **marker** – the ID of the last event in the previous page
- **sort_keys** – an array of fields used to sort the list
- **sort_dir** – the direction of the sort ('asc' or 'desc').
- **nested_depth** – Levels of nested stacks to list events for.

list_outputs(*cntx*, *stack_identity*)

list_resource_types(*ctxt*, *support_status=None*, *type_name=None*, *heat_version=None*, *with_description=False*)

Get a list of valid resource types.

Parameters:
- **ctxt** – RPC context.
- **support_status** – Support status of resource type
- **type_name** – Resource type's name (regular expression allowed)
- **heat_version** – Heat version
- **with_description** – Either return resource type description or not

list_services(*cnxt*)

list_software_configs(*cnxt*, *limit=None*, *marker=None*,

***tenant_safe=True*)**

list_software_deployments(*cnxt*, *server_id=None*)

list_stack_resources(*ctxt*, *stack_identity*, *nested_depth=0*, *with_detail=False*, *filters=None*)

List the resources belonging to a stack.

Parameters:
- **ctxt** – RPC context.
- **stack_identity** – Name of the stack.
- **nested_depth** – Levels of nested stacks of which list resources.
- **with_detail** – show detail for resources in list.
- **filters** – a dict with attribute:value to search the resources

list_stacks(*ctxt*, *limit=None*, *marker=None*, *sort_keys=None*, *sort_dir=None*, *filters=None*, *tenant_safe=True*, *show_deleted=False*, *show_nested=False*, *show_hidden=False*, *tags=None*, *tags_any=None*, *not_tags=None*, *not_tags_any=None*)

Returns attributes of all stacks.

It supports pagination (`limit` and `marker`), sorting (`sort_keys` and `sort_dir`) and filtering (`filters`) of the results.

Parameters:
- **ctxt** – RPC context.
- **limit** – the number of stacks to list (integer or string)
- **marker** – the ID of the last item in the previous page
- **sort_keys** – an array of fields used to sort the list
- **sort_dir** – the direction of the sort (‘asc’ or ‘desc’)
- **filters** – a dict with attribute:value to filter the list
- **tenant_safe** – if true, scope the request by the current tenant
- **show_deleted** – if true, show soft-deleted stacks
- **show_nested** – if true, show nested stacks
- **show_hidden** – if true, show hidden stacks
- **tags** – show stacks containing these tags, combine multiple tags using the boolean AND expression
- **tags_any** – show stacks containing these tags, combine multiple tags using the boolean OR expression
- **not_tags** – show stacks not containing these tags, combine

multiple tags using the boolean AND expression
- **not_tags_any** – show stacks not containing these tags, combine multiple tags using the boolean OR expression

Returns: a list of stacks

`list_template_functions`(*ctxt*, *template_version*)

Get a list of available functions in a given template.

Parameters: **ctxt** – RPC context

:param template_name : name of the template which function list you want to get

`list_template_versions`(*ctxt*)

Get a list of available template versions.

Parameters: **ctxt** – RPC context.

`local_error_name`(*error*)

Returns the name of the error with any _Remote postfix removed.

Parameters: **error** – Remote raised error to derive the name from.

static `make_msg`(*method*, ***kwargs*)

`metadata_software_deployments`(*cnxt*, *server_id*)

`preview_stack`(*ctxt*, *stack_name*, *template*, *params*, *files*, *args*, *environment_files=None*)

Simulates a new stack using the provided template.

Note that at this stage the template has already been fetched from the heat-api process if using a template-url.

Parameters:
- **ctxt** – RPC context.
- **stack_name** – Name of the stack you want to create.
- **template** – Template of stack you want to create.
- **params** – Stack Input Params/Environment
- **files** – files referenced from the environment.
- **args** – Request parameters/args passed from API
- **environment_files** (*list or None*) – optional ordered list of environment file names included in the files dict

preview_update_stack(*ctxt*, *stack_identity*, *template*, *params*, *files*, *args*, *environment_files=None*)

Returns the resources that would be changed in an update.

Based on the provided template and parameters.

Requires RPC version 1.15 or above.

Parameters:
- **ctxt** – RPC context.
- **stack_identity** – Name of the stack you wish to update.
- **template** – New template for the stack.
- **params** – Stack Input Params/Environment
- **files** – files referenced from the environment.
- **args** – Request parameters/args passed from API
- **environment_files** (*list or None*) – optional ordered list of environment file names included in the files dict

resource_mark_unhealthy(*ctxt*, *stack_identity*, *resource_name*, *mark_unhealthy*, *resource_status_reason=None*)

Mark the resource as unhealthy or healthy.

Parameters:
- **ctxt** – RPC context.
- **stack_identity** – Name of the stack.
- **resource_name** – the Resource.
- **mark_unhealthy** – indicates whether the resource is unhealthy.
- **resource_status_reason** – reason for health change.

resource_schema(*ctxt*, *type_name*, *with_description=False*)

Get the schema for a resource type.

Parameters:
- **ctxt** – RPC context.
- **with_description** – Return resource with description or not.

resource_signal(*ctxt*, *stack_identity*, *resource_name*, *details*, *sync_call=False*)

Generate an alarm on the resource.

Parameters:
- **ctxt** – RPC context.
- **stack_identity** – Name of the stack.
- **resource_name** – the Resource.
- **details** – the details of the signal.

set_watch_state(*ctxt*, *watch_name*, *state*)

Temporarily set the state of a given watch.

Parameters:
- **ctxt** – RPC context.
- **watch_name** – Name of the watch
- **state** – State (must be one defined in WatchRule class)

show_output(*cntx*, *stack_identity*, *output_key*)

show_snapshot(*cnxt*, *stack_identity*, *snapshot_id*)

show_software_config(*cnxt*, *config_id*)

show_software_deployment(*cnxt*, *deployment_id*)

show_stack(*ctxt*, *stack_identity*, *resolve_outputs=True*)

Returns detailed information about one or all stacks.

Parameters:
- **ctxt** – RPC context.
- **stack_identity** – Name of the stack you want to show, or None to

show all :param resolve_outputs: If True, stack outputs will be resolved

show_watch(*ctxt*, *watch_name*)

Returns the attributes of one watch/alarm.

The show_watch method returns the attributes of one watch or all watches if no watch_name is passed.

Parameters:
- **ctxt** – RPC context.
- **watch_name** – Name of the watch/alarm you want to see, or None to see all

show_watch_metric(*ctxt*, *metric_namespace=None*, *metric_name=None*)

Returns the datapoints for a metric.

The show_watch_metric method returns the datapoints associated with a

specified metric, or all metrics if no metric_name is passed.

Parameters:
- **ctxt** – RPC context.
- **metric_namespace** – Name of the namespace you want to see, or None to see all
- **metric_name** – Name of the metric you want to see, or None to see all

signal_software_deployment(*cnxt*, *deployment_id*, *details*, *updated_at=None*)

stack_cancel_update(*ctxt*, *stack_identity*, *cancel_with_rollback=True*)

stack_check(*ctxt*, *stack_identity*)

stack_list_snapshots(*cnxt*, *stack_identity*)

stack_restore(*cnxt*, *stack_identity*, *snapshot_id*)

stack_resume(*ctxt*, *stack_identity*)

stack_snapshot(*ctxt*, *stack_identity*, *name*)

stack_suspend(*ctxt*, *stack_identity*)

update_software_deployment(*cnxt*, *deployment_id*, *config_id=None*, *input_values=None*, *output_values=None*, *action=None*, *status=None*, *status_reason=None*, *updated_at=None*)

update_stack(*ctxt*, *stack_identity*, *template*, *params*, *files*, *args*, *environment_files=None*)

Updates an existing stack based on the provided template and params.

Note that at this stage the template has already been fetched from the heat-api process if using a template-url.

Parameters:
- **ctxt** – RPC context.
- **stack_name** – Name of the stack you want to create.
- **template** – Template of stack you want to create.
- **params** – Stack Input Params/Environment
- **files** – files referenced from the environment.
- **args** – Request parameters/args passed from API
- **environment_files** (*list or None*) – optional ordered list of environment file names included in the files dict

validate_template(*ctxt*, *template*, *params=None*, *files=None*, *environment_files=None*, *show_nested=False*, *ignorable_errors=None*)

Uses the stack parser to check the validity of a template.

Parameters:

- **ctxt** – RPC context.
- **template** – Template of stack you want to create.
- **params** – Stack Input Params/Environment
- **files** – files referenced from the environment/template.
- **environment_files** – ordered list of environment file names included in the files dict
- **show_nested** – if True nested templates will be validated
- **ignorable_errors** – List of error_code to be ignored as part of

validation

The heat.rpc.listener_client Module

Client side of the heat worker RPC API.

class **heat.rpc.listener_client.EngineListenerClient(*engine_id*)**

Bases: object

Client side of the heat listener RPC API.

API version history:

1.0 - Initial version.

BASE_RPC_API_VERSION = '*1.0*'

is_alive(*ctxt*)

The heat.rpc.worker_api Module

The heat.rpc.worker_client Module

Client side of the heat worker RPC API.

class **heat.rpc.worker_client.WorkerClient**

Bases: object

Client side of the heat worker RPC API.

API version history:

```
1.0 - Initial version.
1.1 - Added check_resource.
1.2 - Add adopt data argument to check_resource.
1.3 - Added cancel_check_resource API.
```

BASE_RPC_API_VERSION = '*1.0*'

cancel_check_resource(*ctxt*, *stack_id*, *engine_id*)

Send check-resource cancel message.

Sends a cancel message to given heat engine worker.

cast(*ctxt*, *msg*, *version=None*)

check_resource(*ctxt*, *resource_id*, *current_traversal*, *data*, *is_update*, *adopt_stack_data*)

***static* make_msg(*method*, ***kwargs*)**

The heat.scaling.cooldown Module

***class* heat.scaling.cooldown.CooldownMixin**

Bases: object

Utility class to encapsulate Cooldown related logic.

This class is shared between AutoScalingGroup and ScalingPolicy. This logic includes both cooldown timestamp comparing and scaling in progress checking.

The heat.scaling.lbutils Module

heat.scaling.lbutils.reload_loadbalancers(*group*, *load_balancers*, *exclude=None*)

Notify the LoadBalancer to reload its config.

This must be done after activation (instance in ACTIVE state), otherwise the instances' IP addresses may not be available.

The heat.scaling.rolling_update Module

heat.scaling.rolling_update.needs_update(*targ_capacity*, *curr_capacity*, *num_up_to_date*)

Return whether there are more batch updates to do.

Inputs are the target size for the group, the current size of the group, and the number of members that already have the latest definition.

heat.scaling.rolling_update.next_batch(*targ_capacity*, *curr_capacity*, *num_up_to_date*, *batch_size*, *min_in_service*)

Return details of the next batch in a batched update.

The result is a tuple containing the new size of the group and the number of members that may receive the new definition (by a combination of creating new members and updating existing ones).

Inputs are the target size for the group, the current size of the group, the number

of members that already have the latest definition, the batch size, and the minimum number of members to keep in service during a rolling update.

The heat.scaling.scalingutil Module

heat.scaling.scalingutil.calculate_new_capacity(*current*, *adjustment*, *adjustment_type*, *min_adjustment_step*, *minimum*, *maximum*)

Calculates new capacity from the given adjustments.

Given the current capacity, calculates the new capacity which results from applying the given adjustment of the given adjustment-type. The new capacity will be kept within the maximum and minimum bounds.

The heat.scaling.template Module

heat.scaling.template.make_template(*resource_definitions*, *version=('heat_template_version', '2015-04-30')*, *child_env=None*)

Return a Template object containing the given resource definitions.

By default, the template will be in the HOT format. A different format can be specified by passing a (version_type, version_string) tuple matching any of the available template format plugins.

heat.scaling.template.member_definitions(*old_resources*, *new_definition*, *num_resources*, *num_new*, *get_new_id*, *customise=<function _identity at 0x7f507cfd0140>*)

Iterate over resource definitions for a scaling group

Generates the definitions for the next change to the scaling group. Each item is a

(name, definition) tuple.

The input is a list of (name, definition) tuples for existing resources in the group, sorted in the order that they should be replaced or removed (i.e. the resource that should be the first to be replaced (on update) or removed (on scale down) appears at the beginning of the list.) New resources are added or old resources removed as necessary to ensure a total of num_resources.

The number of resources to have their definition changed to the new one is controlled by num_new. This value includes any new resources to be added, with any shortfall made up by modifying the definitions of existing resources.

The `heat.version` Module

Index

Symbols | **A** | **B** | **C** | **D** | **E** | **F** | **G** | **H** | **I** | **J** | **K** | **L** | **M** | **N** | **O** | **P** | **Q** | **R** | **S** | **T** | **U** | **V** | **W** | **Y** | **Z**

Symbols

--config-dir
- **heat-api command line option**
- **heat-api-cfn command line option**
- **heat-api-cloudwatch command line option**
- **heat-engine command line option**

--config-dir <DIR>
- **heat-keystone-setup-domain command line option**

--config-file
- **heat-api command line option**
- **heat-api-cfn command line option**
- **heat-api-cloudwatch command line option**
- **heat-engine command line option**

--stack-user-domain-name <DOMAIN>
- **heat-keystone-setup-domain command line option**

--version
- **heat-api command line option**
- **heat-api-cfn command line option**
- **heat-api-cloudwatch command line option**
- **heat-engine command line option**
- **heat-keystone-setup-domain command line option**

-h, --help
- **heat-db-setup command line option**
- **heat-keystone-setup-domain command line option**

--config-file <PATH>
heat-keystone-setup-domain command line option
--stack-domain-admin <USERNAME>
heat-keystone-setup-domain command line option
--stack-domain-admin-password <PASSWORD>
heat-keystone-setup-domain command line option
-p, --password
heat-db-setup command line option
-r, --rootpw
heat-db-setup command line option
-y, --yes
heat-db-setup command line option

A

abandon() (heat.api.openstack.v1.stacks.StackController method)
abandon_stack() (heat.engine.service.EngineService method)
(heat.rpc.client.EngineClient method)
absolute_limits() (heat.engine.clients.os.nova.NovaClientPlugin method)
access (heat.engine.clients.os.keystone.heat_keystoneclient.AccessKey attribute)
access_allowed() (heat.engine.stack.Stack method)
AccessKey (class in heat.engine.clients.os.keystone.heat_keystoneclient)
acquire() (heat.engine.stack_lock.StackLock method)
action() (heat.api.openstack.v1.actions.ActionController method)
action_handler_task() (heat.engine.resource.Resource method)
ACTION_MAP
allowed_param_prop_type() (heat.engine.properties.Schema method)
AllowedPattern (class in heat.engine.constraints)
AllowedValues (class in heat.engine.constraints)
AlreadyExistsError
ALTERNATE_VERSION (heat.engine.cfn.template.CfnTemplateBase attribute)
AnyIndexDict (class in heat.engine.constraints)
ANYTHING (heat.engine.constraints.AnyIndexDict attribute)
AodhClientPlugin (class in heat.engine.clients.os.aodh)
API server
args() (heat.api.openstack.v1.stacks.InstantiationData method)
as_dict() (heat.engine.environment.ResourceRegistry method)
(heat.engine.event.Event method)

(heat.engine.watchrule.WatchRule attribute)
ActionController (class in heat.api.openstack.v1.actions)
ACTIONS (heat.api.openstack.v1.actions.ActionController attribute)
(heat.engine.resource.Resource attribute)
(heat.engine.service.EngineListener attribute)
(heat.engine.stack.Stack attribute)
ADD (heat.engine.translation.TranslationRule attribute)
add_dependencies() (heat.engine.resource.Resource method)
add_explicit_dependencies() (heat.engine.resource.Resource method)
add_msg_queue() (heat.engine.service.ThreadGroupManager method)
add_resource() (heat.engine.cfn.template.CfnTemplateBase method)
(heat.engine.hot.template.HOTemplate20130523 method)
(heat.engine.stack.Stack method)
(heat.engine.template.Template method)
add_timer() (heat.engine.service.ThreadGroupManager method)
AddressScopeConstraint (class in heat.engine.clients.os.neutron.neutron_constraints)
ADOPT (heat.engine.resource.Resource attribute)
(heat.engine.stack.Stack attribute)
adopt() (heat.engine.resource.Resource method)
(heat.engine.stack.Stack method)
as_output() (heat.engine.attributes.Attribute method)
as_outputs() (heat.engine.attributes.Attributes static method)
as_task() (heat.engine.scheduler.TaskRunner method)
attach_volume() (heat.engine.clients.os.nova.NovaClientPlugin method)
Attribute (class in heat.engine.attributes)
Attributes (class in heat.engine.attributes)
attributes_schema (heat.engine.resource.Resource attribute)
auth_ref (heat.engine.clients.os.keystone.heat_keystoneclient.KsClientWrapper attribute)
auth_token (heat.engine.clients.os.keystone.heat_keystoneclient.KsClientWrapper attribute)
auth_url (heat.objects.user_creds.UserCreds attribute)
authenticated_to_backend() (heat.engine.service.EngineService method)
(heat.rpc.client.EngineClient method)
AUTO_TYPE (heat.objects.fields.JsonField attribute)
(heat.objects.fields.ListField attribute)

ALARM (heat.engine.watchrule.WatchRule attribute)
ALARMING (heat.engine.clients.os.aodh.AodhClientPlugin attribute)
(heat.engine.clients.os.ceilometer.CeilometerClientPlugin attribute)

B

BarbicanClientPlugin (class in heat.engine.clients.os.barbican)
Base64 (class in heat.engine.cfn.functions)
BASE_ATTRIBUTES (heat.engine.resource.Resource attribute)
base_attributes_schema (heat.engine.resource.Resource attribute)
BASE_RPC_API_VERSION (heat.rpc.client.EngineClient attribute)
(heat.rpc.listener_client.EngineListenerClient attribute)
(heat.rpc.worker_client.WorkerClient attribute)
BaseCinderConstraint (class in heat.engine.clients.os.cinder)
BaseCustomConstraint (class in heat.engine.constraints)
BaymodelConstraint (class in heat.engine.clients.os.magnum)
BOOLEAN (heat.engine.attributes.Schema attribute)
(heat.engine.constraints.Schema attribute)
(heat.engine.hot.parameters.HOTParamSchema attribute)
(heat.engine.parameters.Schema attribute)
BOOLEAN_TYPE (heat.engine.constraints.Schema attribute)
BooleanParam (class in heat.engine.parameters)
build_info() (heat.api.openstack.v1.build_info.BuildInfoController method)
build_resource_description() (in module heat.engine.api)
build_template_dict() (heat.engine.resource.Resource static method)
build_userdata() (heat.engine.clients.os.nova.NovaClientPlugin method)
BuildInfoController (class in heat.api.openstack.v1.build_info)

C

cache_data_reference_id() (heat.engine.stack.Stack method)
cache_data_resource_all_attributes() (heat.engine.stack.Stack method)
cache_data_resource_attribute() (heat.engine.stack.Stack method)
CACHE_LOCAL (heat.engine.attributes.Schema attribute)
CACHE_MODES (heat.engine.attributes.Schema attribute)
CACHE_NONE (heat.engine.attributes.Schema attribute)
calculate_new_capacity() (in module heat.scaling.scalingutil)
call() (heat.rpc.client.EngineClient method)
cancel() (heat.engine.scheduler.TaskRunner method)
cancel_all() (heat.engine.scheduler.DependencyTaskGroup method)
cancel_check_resource() (heat.engine.worker.WorkerService method)
    (heat.rpc.worker_client.WorkerClient method)
cancel_grace_period() (heat.engine.resource.Resource method)
CANCEL_UPDATE
CLOUDFORMATION (heat.engine.clients.os.heat_plugin.HeatClientPlugin attribute)
ClusterConstraint (class in heat.engine.clients.os.senlin)
CLUSTERING (heat.engine.clients.os.senlin.SenlinClientPlugin attribute)
cmd_resource (heat.engine.clients.os.neutron.lbaas_constraints.LoadbalancerConstraint attribute)
    (heat.engine.clients.os.neutron.lbaas_constraints.PoolConstraint attribute)
    (heat.engine.clients.os.neutron.neutron_constraints.NeutronConstraint attribute)
    (heat.engine.clients.os.neutron.neutron_constraints.QoSPolicyConstraint attribute)
code (heat.api.aws.exception.AlreadyExistsError attribute)
    (heat.api.aws.exception.HeatAPIException attribute)
    (heat.api.aws.exception.HeatAPINotImplementedError attribute)
    (heat.api.aws.exception.HeatAccessDeniedError attribute)
    (heat.api.aws.exception.HeatActionInProgressError attribute)
    (heat.api.aws.exception.HeatIncompleteSignatureError attribute)
    (heat.api.aws.exception.HeatInternalFailureError attribute)
    (heat.api.aws.exception.HeatInvalidActionError attribute)
    (heat.api.aws.exception.HeatInvalidClientTokenIdError attribute)

(heat.api.openstack.v1.actions.ActionController attribute)
cancel_update() (heat.api.cfn.v1.stacks.StackController method)
cast() (heat.rpc.client.EngineClient method)
    (heat.rpc.worker_client.WorkerClient method)
ceilometer() (heat.engine.resource.Resource method)
CEILOMETER_CONTROLLED (heat.engine.watchrule.WatchRule attribute)
CeilometerClientPlugin (class in heat.engine.clients.os.ceilometer)
CFN
CfnTemplate (class in heat.engine.cfn.template)
CfnTemplateBase (class in heat.engine.cfn.template)
CHECK (heat.api.openstack.v1.actions.ActionController attribute)
    (heat.engine.resource.Resource attribute)
    (heat.engine.stack.Stack attribute)
check() (heat.engine.check_resource.CheckResource method)
    (heat.engine.resource.Resource method)
    (heat.engine.stack.Stack method)
check_attach_volume_complete() (heat.engine.clients.os.cinder.CinderClientPlugin method)
check_create_complete() (built-in
(heat.api.aws.exception.HeatInvalidParameterCombinationError attribute)
(heat.api.aws.exception.HeatInvalidParameterValueError attribute)
(heat.api.aws.exception.HeatInvalidQueryParameterError attribute)
(heat.api.aws.exception.HeatMalformedQueryStringError attribute)
(heat.api.aws.exception.HeatMissingActionError attribute)
(heat.api.aws.exception.HeatMissingAuthenticationTokenError attribute)
(heat.api.aws.exception.HeatMissingParameterError attribute)
(heat.api.aws.exception.HeatOptInRequiredError attribute)
(heat.api.aws.exception.HeatRequestExpiredError attribute)
(heat.api.aws.exception.HeatRequestLimitExceeded attribute)
(heat.api.aws.exception.HeatServiceUnavailableError attribute)
(heat.api.aws.exception.HeatSignatureError attribute)
(heat.api.aws.exception.HeatThrottlingError attribute)
coerce() (heat.objects.fields.Json method)
collection() (in module heat.api.openstack.v1.views.stacks_view)
CommaDelimitedListParam (class in heat.engine.parameters)
COMPLETE (heat.engine.resource.Resource attribute)
    (heat.engine.stack.Stack attribute)
COMPUTE (heat.engine.clients.os.nova.NovaClientPlugin attribute)
condition_functions (heat.engine.cfn.template.CfnTemplate attribute)
    (heat.engine.hot.template.HOTemplate20161014 attribute)

function)
check_delete_complete() (built-in function)
check_delete_server_complete() (heat.engine.clients.os.nova.NovaClientPlugin method)
check_delete_snapshot_complete() (built-in function)
check_detach_volume_complete() (heat.engine.clients.os.cinder.CinderClientPlugin method)
(heat.engine.clients.os.nova.NovaClientPlugin method)
check_interface_attach() (heat.engine.clients.os.nova.NovaClientPlugin method)
check_interface_detach() (heat.engine.clients.os.nova.NovaClientPlugin method)
check_python3_no_iteritems() (in module heat.hacking.checks)
check_python3_no_iterkeys() (in module heat.hacking.checks)
check_python3_no_itervalues() (in module heat.hacking.checks)
check_rebuild() (heat.engine.clients.os.nova.NovaClientPlugin method)
check_resize() (heat.engine.clients.os.nova.NovaClientPlugin method)
check_resource() (heat.engine.worker.WorkerService method)
(heat.rpc.worker_client.WorkerClient method)
check_resource_cleanup() (in module heat.engine.check_resource)
check_resource_update() (in module heat.engine.check_resource)
(heat.engine.template.Template attribute)
CONDITIONS (heat.engine.cfn.template.CfnTemplate attribute)
(heat.engine.hot.template.HOTemplate20161014 attribute)
Constraint (class in heat.engine.constraints)
CONSTRAINTS (heat.engine.constraints.Schema attribute)
(heat.engine.hot.parameters.HOTParamSchema attribute)
(heat.engine.parameters.Schema attribute)
(heat.engine.properties.Schema attribute)
construct_input_data() (in module heat.engine.check_resource)
consume() (heat.engine.clients.os.zaqar.ZaqarEventSink method)
CONTAINER (heat.engine.clients.os.magnum.MagnumClientPlugin attribute)
ContainerConstraint (class in heat.engine.clients.os.barbican)
context (heat.engine.clients.client_plugin.ClientPlugin attribute)
(heat.engine.clients.os.keystone.heat_keystoneclient.KsClientWrapper attribute)
Controller (class in heat.api.versions)
converge_stack() (heat.engine.stack.Stack method)
convergence_dependencies (heat.engine.stack.Stack attribute)
CooldownMixin (class in heat.scaling.cooldown)
copy() (heat.engine.dependencies.Graph

check_resume_complete() (built-in function)
check_software_deployment() (heat.engine.service.EngineService method)
(heat.engine.service_software_config.SoftwareConfigService method)
(heat.rpc.client.EngineClient method)
check_stack_complete() (in module heat.engine.check_resource)
check_stack_watches() (heat.engine.service_stack_watch.StackWatch method)
check_suspend_complete() (built-in function)
check_update_complete() (built-in function)
check_verify_resize() (heat.engine.clients.os.nova.NovaClientPlugin method)
CheckResource (class in heat.engine.check_resource)
CIDRConstraint (class in heat.engine.constraint.common_constraints)
cinder() (heat.engine.resource.Resource method)
CinderClientPlugin (class in heat.engine.clients.os.cinder)
CircularDependencyException
ClassResourceInfo (class in heat.engine.environment)
clear_hook() (heat.engine.resource.Resource method)
client (heat.engine.clients.os.keystone.heat_keystoneclient.KsClientWra method)
(heat.engine.dependencies.Node method)
count_all() (heat.objects.stack.Stack class method)
count_all_by_stack() (heat.objects.event.Event class method)
count_stacks() (heat.engine.service.EngineService method)
(heat.rpc.client.EngineClient method)
count_total_resources() (heat.objects.stack.Stack class method)
CREATE (heat.engine.resource.Resource attribute)
(heat.engine.stack.Stack attribute)
create() (heat.api.cfn.v1.stacks.StackController method)
(heat.api.openstack.v1.software_configs.SoftwareConfigController method)
(heat.api.openstack.v1.software_deployments.SoftwareDeploymentController method)
(heat.api.openstack.v1.stacks.StackController method)
(heat.api.openstack.v1.stacks.StackSerializer method)
(heat.engine.resource.Resource method)
(heat.engine.stack.Stack method)
(heat.objects.event.Event class method)
(heat.objects.raw_template.RawTemplate class method)
(heat.objects.raw_template_files.RawTemplateFiles class method)
(heat.objects.resource.Resource class method)
(heat.objects.service.Service class method)
(heat.objects.snapshot.Snapshot class

pper attribute)
client()
(heat.engine.clients.client_plugin.ClientPlugin method)
(heat.engine.resource.Resource method)
client_plugin()
(heat.engine.resource.Resource method)
ClientPlugin (class in heat.engine.clients.client_plugin)
clients
(heat.engine.clients.client_plugin.ClientPlugin attribute)
method)
(heat.objects.software_config.SoftwareConfig class method)
(heat.objects.software_deployment.SoftwareDeployment class method)
(heat.objects.stack.Stack class method)
(heat.objects.stack_lock.StackLock class method)
(heat.objects.sync_point.SyncPoint class method)
(heat.objects.user_creds.UserCreds class method)
(heat.objects.watch_data.WatchData class method)
(heat.objects.watch_rule.WatchRule class method)
(in module heat.engine.sync_point)
create_certificate()
(heat.engine.clients.os.barbican.BarbicanClientPlugin method)
create_convergence()
(heat.engine.resource.Resource method)
create_ec2_keypair()
(heat.engine.clients.os.keystone.heat_keystoneclient.KsClientWrapper method)
create_empty_template()
(heat.engine.template.Template class method)
create_for_tenant()
(heat.engine.clients.os.zaqar.ZaqarClientPlugin method)
create_from_signed_url()
(heat.engine.clients.os.zaqar.ZaqarClientPlugin method)
create_generic_container()
(heat.engine.clients.os.barbican.BarbicanClientPlugin method)
create_or_update()
(heat.api.cfn.v1.stacks.StackController method)
CREATE_OR_UPDATE_ACTION
(heat.api.cfn.v1.stacks.StackController

attribute)
create_periodic_tasks()
(heat.engine.service.EngineService
method)
create_resource() (in module
heat.api.cfn.v1.signal)
(in module heat.api.cfn.v1.stacks)
(in module heat.api.cloudwatch.watch)
(in module
heat.api.openstack.v1.actions)
(in module
heat.api.openstack.v1.build_info)
(in module
heat.api.openstack.v1.events)
(in module
heat.api.openstack.v1.resources)
(in module
heat.api.openstack.v1.services)
(in module
heat.api.openstack.v1.software_configs
)
(in module
heat.api.openstack.v1.software_deploy
ments)
(in module
heat.api.openstack.v1.stacks)
create_rsa()
(heat.engine.clients.os.barbican.Barbican
ClientPlugin method)
create_software_config()
(heat.engine.service.EngineService
method)
(heat.engine.service_software_config.S
oftwareConfigService method)
(heat.rpc.client.EngineClient method)
create_software_deployment()
(heat.engine.service.EngineService
method)
(heat.engine.service_software_config.S
oftwareConfigService method)
(heat.rpc.client.EngineClient method)
CREATE_STACK

(heat.api.cfn.v1.stacks.StackController attribute)
create_stack() (heat.engine.service.EngineService method)
(heat.rpc.client.EngineClient method)
create_stack_domain_project() (heat.engine.clients.os.keystone.heat_keystoneclient.KsClientWrapper method)
create_stack_domain_user() (heat.engine.clients.os.keystone.heat_keystoneclient.KsClientWrapper method)
create_stack_domain_user_keypair() (heat.engine.clients.os.keystone.heat_keystoneclient.KsClientWrapper method)
create_stack_user() (heat.engine.clients.os.keystone.heat_keystoneclient.KsClientWrapper method)
create_stack_user_project_id() (heat.engine.stack.Stack method)
create_trust_context() (heat.engine.clients.os.keystone.heat_keystoneclient.KsClientWrapper method)
create_watch_data() (heat.engine.service.EngineService method)
(heat.engine.watchrule.WatchRule method)
(heat.rpc.client.EngineClient method)
created_at (heat.engine.watchrule.WatchRule attribute)
(heat.objects.user_creds.UserCreds attribute)
CRONExpressionConstraint (class in heat.engine.constraint.common_constraints)
custom_constraint (heat.engine.constraints.CustomConstraint attribute)
CustomConstraint (class in

heat.engine.constraints)

D

data()
(heat.engine.resource.Resource method)
data_delete()
(heat.engine.resource.Resource method)
DATA_PROCESSING
(heat.engine.clients.os.sahara.SaharaClientPlugin attribute)
data_set()
(heat.engine.resource.Resource method)
DATABASE
(heat.engine.clients.os.trove.TroveClientPlugin attribute)
db_active_resources_get()
(heat.engine.stack.Stack method)
db_resource_get()
(heat.engine.stack.Stack method)
decrypt_method
(heat.objects.user_creds.UserCreds attribute)
DEFAULT
(heat.engine.constraints.Schema attribute)
(heat.engine.hot.parameters.HOTParamSchema attribute)
(heat.engine.parameters.Schema attribute)
(heat.engine.properties.Schema attribute)
default()
(heat.api.cfn.v1.stacks.StackController method)
(heat.api.openstack.v1.software_configs.SoftwareConfigControll
describe_alarms()
(heat.api.cloudwatch.watch.WatchController method)
describe_alarms_for_metric()
(heat.api.cloudwatch.watch.WatchController method)
describe_stack_resource()
(heat.api.cfn.v1.stacks.StackController method)
(heat.engine.service.EngineService method)
(heat.rpc.client.EngineClient method)
describe_stack_resources()
(heat.api.cfn.v1.stacks.StackController method)
(heat.engine.service.EngineService method)
(heat.rpc.client.EngineClient method)
DESCRIPTION
(heat.engine.attributes.Schema attribute)
(heat.engine.cfn.template.CfnTemplateBase attribute)
(heat.engine.constraints.Constraint attribute)
(heat.engine.constraints.Schema attribute)
description
(heat.engine.environment.ClassResourceInfo attribute)
(heat.engine.environment.GlobResourceInfo attribute)
(heat.engine.environment.MapResourceInfo attribute)
(heat.engine.environment.TemplateResourceInfo attribute)

er method)
(heat.api.openstack.v1.software_deployments.SoftwareDeploymentController method)
(heat.api.openstack.v1.stacks.StackController method)
(heat.engine.parameters.Parameter method)
(heat.engine.properties.Property method)
default_client_name (heat.engine.resource.Resource attribute)
DEFAULT_TTL (heat.engine.clients.os.zaqar.ZaqarClientPlugin attribute)
default_version (heat.engine.clients.client_plugin.ClientPlugin attribute)
(heat.engine.clients.os.aodh.AodhClientPlugin attribute)
(heat.engine.clients.os.glance.GlanceClientPlugin attribute)
deferred_server_statuses (heat.engine.clients.os.nova.NovaClientPlugin attribute)
DELETE (heat.engine.resource.Resource attribute)
(heat.engine.stack.Stack attribute)
(heat.engine.translation.TranslationRule attribute)
delete() (heat.api.cfn.v1.stacks.StackController method)
(heat.api.openstack.v1.software_configs.SoftwareConfigController method)
(heat.api.openstack.v1.software_deployments.SoftwareDeploymentController method)
DESCRIPTION (heat.engine.hot.parameters.HOTParamSchema attribute)
(heat.engine.hot.template.HOTemplate20130523 attribute)
(heat.engine.parameters.Schema attribute)
(heat.engine.properties.Schema attribute)
description() (heat.engine.parameters.Parameter method)
deserialize_input_data() (in module heat.engine.sync_point)
DesignateClientPlugin (class in heat.engine.clients.os.designate)
DesignateDomainConstraint (class in heat.engine.clients.os.designate)
destroy() (heat.engine.resource.Resource method)
(heat.engine.watchrule.WatchRule method)
detach_volume() (heat.engine.clients.os.nova.NovaClientPlugin method)
detail() (heat.api.openstack.v1.stacks.StackController method)
Digest (class in heat.engine.hot.functions)
digest() (heat.engine.hot.functions.Digest method)
disable_alarm_actions() (heat.api.cloudwatch.watch.WatchController method)
disable_stack_domain_user() (heat.engine.clients.os.keystone.heat_keystoneclient.KsClientWrapper method)
disable_stack_user() (heat.engine.clients.os.keystone.heat_k

(heat.api.openstack.v1.stacks.StackController method)
(heat.engine.resource.Resource method)
(heat.engine.stack.Stack method)
(heat.objects.raw_template.RawTemplate class method)
(heat.objects.resource.Resource class method)
(heat.objects.resource_data.ResourceData class method)
(heat.objects.service.Service class method)
(heat.objects.snapshot.Snapshot class method)
(heat.objects.software_config.SoftwareConfig class method)
(heat.objects.software_deployment.SoftwareDeployment class method)
(heat.objects.stack.Stack class method)
(heat.objects.stack_tag.StackTagList class method)
(heat.objects.user_creds.UserCreds class method)
(heat.objects.watch_rule.WatchRule class method)
delete_alarms() (heat.api.cloudwatch.watch.WatchController method)
delete_all() (in module heat.engine.sync_point)
delete_all_by_stack_and_traversal() (heat.objects.sync_point.SyncPoint class method)
delete_convergence() (heat.engine.resource.Resource method)
delete_ec2_keypair() (heat.engine.clients.os.keystoneclient.KsClientWrapper method)
disjoint() (heat.engine.dependencies.Node method)
dispatch_event() (heat.engine.stack.Stack method)
DNS (heat.engine.clients.os.designate.DesignateClientPlugin attribute)
DNSDomainConstraint (class in heat.engine.constraint.common_constraints)
DNSNameConstraint (class in heat.engine.constraint.common_constraints)
do_Average() (heat.engine.watchrule.WatchRule method)
do_data_cmp() (heat.engine.watchrule.WatchRule method)
do_Maximum() (heat.engine.watchrule.WatchRule method)
do_Minimum() (heat.engine.watchrule.WatchRule method)
do_post_op() (heat.engine.lifecycle_plugin.LifecyclePlugin method)
do_pre_op() (heat.engine.lifecycle_plugin.LifecyclePlugin method)
do_SampleCount() (heat.engine.watchrule.WatchRule method)
do_Sum() (heat.engine.watchrule.WatchRule method)
does_endpoint_exist() (heat.engine.clients.client_plugin.ClientPlugin method)

(heat.engine.clients.os.keystone.heat_keystoneclient.KsClientWrapper method)
delete_snapshot()
(heat.api.openstack.v1.stacks.StackController method)
(heat.engine.resource.Resource method)
(heat.engine.service.EngineService method)
(heat.engine.stack.Stack method)
(heat.rpc.client.EngineClient method)
delete_software_config()
(heat.engine.service.EngineService method)
(heat.engine.service_software_config.SoftwareConfigService method)
(heat.rpc.client.EngineClient method)
delete_software_deployment()
(heat.engine.service.EngineService method)
(heat.engine.service_software_config.SoftwareConfigService method)
(heat.rpc.client.EngineClient method)
delete_stack()
(heat.engine.service.EngineService method)
(heat.rpc.client.EngineClient method)
delete_stack_domain_project()
(heat.engine.clients.os.keystone.heat_keystoneclient.KsClientWrapper method)
delete_stack_domain_user()
(heat.engine.clients.os.keystone.heat_keystoneclient.KsClientWrapper method)
domain_admin_auth
(heat.engine.clients.os.keystone.heat_keystoneclient.KsClientWrapper attribute)
domain_admin_client
(heat.engine.clients.os.keystone.heat_keystoneclient.KsClientWrapper attribute)
domain_create()
(heat.engine.clients.os.designate.DesignateClientPlugin method)
domain_update()
(heat.engine.clients.os.designate.DesignateClientPlugin method)
done()
(heat.engine.scheduler.TaskRunner method)
DynamicSchemeAttributes (class in heat.engine.attributes)

er method)
delete_stack_domain_user_keypair
()
(heat.engine.clients.os.keystone.h
eat_keystoneclient.KsClientWrapp
er method)
delete_stack_user()
(heat.engine.clients.os.keystone.h
eat_keystoneclient.KsClientWrapp
er method)
delete_trust()
(heat.engine.clients.os.keystone.h
eat_keystoneclient.KsClientWrapp
er method)
deletion_policies
(heat.engine.cfn.template.CfnTem
plateBase attribute)
(heat.engine.hot.template.HOTe
mplate20130523 attribute)
(heat.engine.hot.template.HOTe
mplate20161014 attribute)
DELETION_POLICY
(heat.engine.cfn.functions.Resour
ceFacade attribute)
(heat.engine.hot.functions.Reso
urceFacade attribute)
dep_attrs()
(heat.engine.cfn.functions.GetAtt
method)
(heat.engine.function.Function
method)
(heat.engine.hot.functions.GetA
tt method)
(heat.engine.hot.functions.GetA
ttAllAttributes method)
(heat.engine.resource.Resource
method)
(in module
heat.engine.function)
Dependencies (class in
heat.engine.dependencies)
dependencies

(heat.engine.stack.Stack attribute)
dependencies() (heat.engine.cfn.functions.GetAtt method)
(heat.engine.cfn.functions.ResourceRef method)
(heat.engine.function.Function method)
(heat.engine.update.StackUpdate method)
(in module heat.engine.function)
Dependency
DependencyTaskGroup (class in heat.engine.scheduler)
describe() (heat.api.cfn.v1.stacks.StackController method)
describe_alarm_history() (heat.api.cloudwatch.watch.WatchController method)

E

EC2Token (class in heat.api.aws.ec2token)
EC2Token_filter_factory() (in module heat.api.aws.ec2token)
edges() (heat.engine.dependencies.Graph method)
enable_alarm_actions() (heat.api.cloudwatch.watch.WatchController method)
enable_stack_domain_user() (heat.engine.clients.os.keystone.heat_keystoneclient.KsClientWrapper method)
enable_stack_user()
error() (heat.engine.constraints.BaseCustomConstraint method)
error_map (heat.api.middleware.fault.FaultWrapper attribute)
estimate_template_cost() (heat.api.cfn.v1.stacks.StackController method)
evaluate() (heat.engine.watchrule.WatchRule method)
Event (class in heat.engine.event)
(class in heat.objects.event)
EventController (class in

(heat.engine.clients.os.keystone.heat_keystoneclient.KsClientWrapper method)
encrypt_hidden_parameters() (heat.objects.raw_template.RawTemplate class method)
(heat.objects.stack.Stack class method)
encrypt_properties_data() (heat.objects.resource.Resource static method)
EngineClient (class in heat.rpc.client)
EngineListener (class in heat.engine.service)
EngineListenerClient (class in heat.rpc.listener_client)
EngineService (class in heat.engine.service)
entity
(heat.engine.clients.os.keystone.keystone_constraints.KeystoneBaseConstraint attribute)
(heat.engine.clients.os.keystone.keystone_constraints.KeystoneDomainConstraint attribute)
(heat.engine.clients.os.keystone.keystone_constraints.KeystoneGroupConstraint attribute)
(heat.engine.clients.os.keystone.keystone_constraints.KeystoneProjectConstraint attribute)
(heat.engine.clients.os.keystone.keystone_constraints.KeystoneRegionConstraint attribute)
(heat.engine.clients.os.keystone.keystone_constraints.KeystoneRoleConstraint attribute)
(heat.engine.clients.os.keystone.keystone_constraints.KeystoneServiceConstraint attribute)
(heat.engine.clients.os.keystone
heat.api.openstack.v1.events)
events_list()
(heat.api.cfn.v1.stacks.StackController method)
ExceptionGroup
exceptions_module
(heat.engine.clients.client_plugin.ClientPlugin attribute)
(heat.engine.clients.os.aodh.AodhClientPlugin attribute)
(heat.engine.clients.os.ceilometer.CeilometerClientPlugin attribute)
(heat.engine.clients.os.cinder.CinderClientPlugin attribute)
(heat.engine.clients.os.designate.DesignateClientPlugin attribute)
(heat.engine.clients.os.glance.GlanceClientPlugin attribute)
(heat.engine.clients.os.heat_plugin.HeatClientPlugin attribute)
(heat.engine.clients.os.manila.ManilaClientPlugin attribute)
(heat.engine.clients.os.monasca.MonascaClientPlugin attribute)
(heat.engine.clients.os.nova.NovaClientPlugin attribute)
(heat.engine.clients.os.sahara.SaharaClientPlugin attribute)
(heat.engine.clients.os.swift.SwiftClientPlugin attribute)
(heat.engine.clients.os.trove.TroveClientPlugin attribute)
(heat.engine.clients.os.zaqar.ZaqarClientPlugin attribute)
exchange_stacks()
(heat.objects.resource.Resource class method)
execute_rule()
(heat.engine.translation.TranslationRule method)
expected_exceptions
(heat.engine.clients.os.barbican.Contain

e.keystone_constraints.KeystoneUserConstraint attribute)
(heat.engine.resource.Resource attribute)
env (heat.engine.stack.Stack attribute)
env_as_dict() (heat.engine.environment.Environment method)
Environment (class in heat.engine.environment)
environment (heat.objects.raw_template.RawTemplate attribute)
environment() (heat.api.openstack.v1.stacks.InstantiationData method)
(heat.api.openstack.v1.stacks.StackController method)
environment_files() (heat.api.openstack.v1.stacks.InstantiationData method)
Equals (class in heat.engine.hot.functions)
err_type (heat.api.aws.exception.HeatAPIException attribute)
(heat.api.aws.exception.HeatAPINotImplementedError attribute)
(heat.api.aws.exception.HeatInternalFailureError attribute)
(heat.api.aws.exception.HeatServiceUnavailableError attribute)
erConstraint attribute)
(heat.engine.clients.os.barbican.SecretConstraint attribute)
(heat.engine.clients.os.glance.ImageConstraint attribute)
(heat.engine.clients.os.keystone.keystone_constraints.KeystoneServiceConstraint attribute)
(heat.engine.clients.os.manila.ManilaShareBaseConstraint attribute)
(heat.engine.clients.os.mistral.WorkflowConstraint attribute)
(heat.engine.clients.os.neutron.neutron_constraints.NetworkConstraint attribute)
(heat.engine.clients.os.neutron.neutron_constraints.NeutronConstraint attribute)
(heat.engine.clients.os.neutron.neutron_constraints.ProviderConstraint attribute)
(heat.engine.clients.os.nova.FlavorConstraint attribute)
(heat.engine.clients.os.nova.NetworkConstraint attribute)
(heat.engine.clients.os.sahara.ImageConstraint attribute)
(heat.engine.clients.os.senlin.ClusterConstraint attribute)
(heat.engine.clients.os.senlin.PolicyTypeConstraint attribute)
(heat.engine.clients.os.senlin.ProfileConstraint attribute)
(heat.engine.clients.os.senlin.ProfileTypeConstraint attribute)
(heat.engine.clients.os.trove.FlavorConstraint attribute)
(heat.engine.constraints.BaseCustomConstraint attribute)
ExpirationConstraint (class in heat.engine.constraint.common_constraints)

expired()
(heat.engine.scheduler.Timeout method)
explanation
(heat.api.aws.exception.AlreadyExistsError attribute)
(heat.api.aws.exception.HeatAPIException attribute)
(heat.api.aws.exception.HeatAPINotImplementedError attribute)
(heat.api.aws.exception.HeatAccessDeniedError attribute)
(heat.api.aws.exception.HeatActionInProgressError attribute)
(heat.api.aws.exception.HeatIncompleteSignatureError attribute)
(heat.api.aws.exception.HeatInternalFailureError attribute)
(heat.api.aws.exception.HeatInvalidActionError attribute)
(heat.api.aws.exception.HeatInvalidClientTokenIdError attribute)
(heat.api.aws.exception.HeatInvalidParameterCombinationError attribute)
(heat.api.aws.exception.HeatInvalidParameterValueError attribute)
(heat.api.aws.exception.HeatInvalidQueryParameterError attribute)
(heat.api.aws.exception.HeatMalformedQueryStringError attribute)
(heat.api.aws.exception.HeatMissingActionError attribute)
(heat.api.aws.exception.HeatMissingAuthenticationTokenError attribute)
(heat.api.aws.exception.HeatMissingParameterError attribute)
(heat.api.aws.exception.HeatOptInRequiredError attribute)
(heat.api.aws.exception.HeatRequestExpiredError attribute)
(heat.api.aws.exception.HeatRequestLimitExceeded attribute)
(heat.api.aws.exception.HeatServiceU

navailableError attribute)
(heat.api.aws.exception.HeatSignatureError attribute)
(heat.api.aws.exception.HeatThrottlingError attribute)
export()
(heat.api.openstack.v1.stacks.StackController method)
export_stack()
(heat.engine.service.EngineService method)
(heat.rpc.client.EngineClient method)
extension
(heat.engine.clients.os.neutron.lbaas_constraints.ListenerConstraint attribute)
(heat.engine.clients.os.neutron.lbaas_constraints.LoadbalancerConstraint attribute)
(heat.engine.clients.os.neutron.lbaas_constraints.PoolConstraint attribute)
(heat.engine.clients.os.neutron.neutron_constraints.AddressScopeConstraint attribute)
(heat.engine.clients.os.neutron.neutron_constraints.NeutronConstraint attribute)
(heat.engine.clients.os.neutron.neutron_constraints.QoSPolicyConstraint attribute)
extract_args() (in module heat.engine.api)
extract_param_list() (in module heat.api.aws.utils)
extract_param_pairs() (in module heat.api.aws.utils)

F

factory() (in module heat.hacking.checks)
format_resource_properties() (in module heat.engine.api)

FAILED (heat.engine.resource.Resource attribute)
(heat.engine.stack.Stack attribute)
Fault (class in heat.api.middleware.fault)
FaultWrapper (class in heat.api.middleware.fault)
fetch_server() (heat.engine.clients.os.nova.NovaClientPlugin method)
fields (heat.objects.event.Event attribute)
(heat.objects.raw_template.RawTemplate attribute)
(heat.objects.raw_template_files.RawTemplateFiles attribute)
(heat.objects.resource.Resource attribute)
(heat.objects.resource_data.ResourceData attribute)
(heat.objects.service.Service attribute)
(heat.objects.snapshot.Snapshot attribute)
(heat.objects.software_config.SoftwareConfig attribute)
(heat.objects.software_deployment.SoftwareDeployment attribute)
(heat.objects.stack.Stack attribute)
(heat.objects.stack_lock.StackLock attribute)
(heat.objects.stack_tag.StackTag attribute)
(heat.objects.stack_tag.StackTagList attribute)
(heat.objects.sync_point.SyncPoint attribute)
(heat.objects.user_creds.UserCreds attribute)
(heat.objects.watch_data.WatchData attribute)
format_response() (in module heat.api.aws.utils)
format_snapshot() (in module heat.engine.api)
format_software_config() (in module heat.engine.api)
format_software_deployment() (in module heat.engine.api)
format_stack() (in module heat.api.openstack.v1.views.stacks_view)
(in module heat.engine.api)
format_stack_db_object() (in module heat.engine.api)
format_stack_output() (in module heat.engine.api)
format_stack_outputs() (in module heat.engine.api)
format_stack_preview() (in module heat.engine.api)
format_stack_resource() (in module heat.engine.api)
format_validate_parameter() (in module heat.engine.api)
format_watch() (in module heat.engine.api)
format_watch_data() (in module heat.engine.api)
from_attribute() (heat.engine.attributes.Schema class method)
from_db_object() (heat.objects.stack_tag.StackTagList class method)
from_dict() (heat.engine.hot.parameters.HOTParamSchema class method)
(heat.engine.parameters.Schema class method)
from_legacy() (heat.engine.properties.Schema class method)

(heat.objects.watch_rule.WatchRule attribute)
files (heat.engine.template.Template attribute)
(heat.objects.raw_template.RawTemplate attribute)
(heat.objects.raw_template_files.RawTemplateFiles attribute)
files()
(heat.api.openstack.v1.stacks.InstantiationData method)
(heat.api.openstack.v1.stacks.StackController method)
files_id
(heat.objects.raw_template.RawTemplate attribute)
find_flavor_by_name_or_id()
(heat.engine.clients.os.nova.NovaClientPlugin method)
(heat.engine.clients.os.trove.TroveClientPlugin method)
find_image_by_name_or_id()
(heat.engine.clients.os.glance.GlanceClientPlugin method)
find_physical_resource()
(heat.engine.service.EngineService method)
(heat.rpc.client.EngineClient method)
FindInMap (class in heat.engine.cfn.functions)
FlavorConstraint (class in heat.engine.clients.os.nova)
(class in heat.engine.clients.os.trove)
FnBase64()
(heat.engine.resource.Resource method)
FnGetAtt()
(heat.engine.resource.Resource method)
FnGetAtts()
from_parameter()
(heat.engine.properties.Schema class method)
from_primitive()
(heat.objects.fields.Json method)
frozen_definition()
(heat.engine.resource.Resource method)
Function (class in heat.engine.function)
functions
(heat.engine.cfn.template.CfnTemplateBase attribute)
(heat.engine.cfn.template.HeatTemplate attribute)
(heat.engine.hot.template.HOTemplate20130523 attribute)
(heat.engine.hot.template.HOTemplate20141016 attribute)
(heat.engine.hot.template.HOTemplate20150430 attribute)
(heat.engine.hot.template.HOTemplate20151015 attribute)
(heat.engine.hot.template.HOTemplate20160408 attribute)
(heat.engine.hot.template.HOTemplate20161014 attribute)
(heat.engine.template.Template attribute)

(heat.engine.resource.Resource method)
FnGetRefId() (heat.engine.resource.Resource method)
ForcedCancel
format_event() (in module heat.api.openstack.v1.events)
(in module heat.engine.api)
format_notification_body() (in module heat.engine.api)
format_resource() (in module heat.api.openstack.v1.resources)
format_resource_attributes() (in module heat.engine.api)

G

generate_spec() (heat.engine.clients.os.senlin.SenlinClientPlugin method)
generate_template() (heat.api.openstack.v1.stacks.StackController method)
(heat.engine.service.EngineService method)
(heat.rpc.client.EngineClient method)
get() (heat.objects.stack_tag.StackTagList class method)
(in module heat.engine.sync_point)
get_alarm_state() (heat.engine.watchrule.WatchRule method)
get_all() (heat.objects.event.Event class method)
(heat.objects.resource.Resource class method)
(heat.objects.resource_data.ResourceData class method)
(heat.objects.stack_tag.StackTag class method)
get_metric_statistics() (heat.api.cloudwatch.watch.WatchController method)
get_net_id_by_label() (heat.engine.clients.os.nova.NovaClientPlugin method)
get_notification() (heat.engine.clients.os.monasca.MonascaClientPlugin method)
get_nova_network_id() (heat.engine.clients.os.nova.NovaClientPlugin method)
get_num() (heat.engine.parameters.Schema static method)
get_obj() (heat.objects.resource.Resource class method)

ceData class method)
(heat.objects.service.Service class method)
(heat.objects.snapshot.Snapshot class method)
(heat.objects.software_config.SoftwareConfig class method)
(heat.objects.software_deployment.SoftwareDeployment class method)
(heat.objects.stack.Stack class method)
(heat.objects.watch_data.WatchData class method)
(heat.objects.watch_rule.WatchRule class method)
get_all_active_by_stack()
(heat.objects.resource.Resource class method)
get_all_by_args()
(heat.objects.service.Service class method)
get_all_by_owner_id()
(heat.objects.stack.Stack class method)
get_all_by_root_stack()
(heat.objects.resource.Resource class method)
get_all_by_stack()
(heat.objects.event.Event class method)
(heat.objects.resource.Resource class method)
(heat.objects.watch_rule.WatchRule class method)
get_all_by_tenant()
(heat.objects.event.Event class method)
get_all_by_watch_rule_id()
(heat.objects.watch_data.WatchData class method)
get_allowed_params() (in module heat.api.openstack.v1.util)
get_ordinal()
(heat.engine.lifecycle_plugin.LifecyclePlugin method)
get_param_value() (in module heat.api.aws.utils)
get_plugin_id()
(heat.engine.clients.os.sahara.SaharaClientPlugin method)
get_reference_id()
(heat.engine.resource.Resource method)
get_resource_info()
(heat.engine.environment.Environment method)
(heat.engine.environment.GlobResourceInfo method)
(heat.engine.environment.MapResourceInfo method)
(heat.engine.environment.ResourceInfo method)
(heat.engine.environment.ResourceRegistry method)
get_revision()
(heat.engine.service.EngineService method)
(heat.rpc.client.EngineClient method)
get_root_id()
(heat.objects.stack.Stack class method)
get_rsrc_restricted_actions()
(heat.engine.environment.ResourceRegistry method)
get_secret_by_ref()
(heat.engine.clients.os.barbican.BarbicanClientPlugin method)
get_section_name()
(heat.engine.cfn.template.CfnTemplateBase method)
(heat.engine.hot.template.HOTemplate20130523 method)
(heat.engine.template.Template

get_attribute() (heat.engine.resource.Resource method)
get_availability_zones() (heat.engine.stack.Stack method)
get_baymodel() (heat.engine.clients.os.magnum.MagnumClientPlugin method)
get_by_id() (heat.objects.event.Event class method)
(heat.objects.raw_template.RawTemplate class method)
(heat.objects.service.Service class method)
(heat.objects.software_config.SoftwareConfig class method)
(heat.objects.software_deployment.SoftwareDeployment class method)
(heat.objects.stack.Stack class method)
(heat.objects.user_creds.UserCreds class method)
(heat.objects.watch_rule.WatchRule class method)
get_by_key() (heat.objects.resource_data.ResourceData class method)
(heat.objects.sync_point.SyncPoint class method)
get_by_name() (heat.objects.stack.Stack class method)
(heat.objects.watch_rule.WatchRule class method)
get_by_name_and_owner_id() (heat.objects.stack.Stack class method)
get_by_name_and_stack() (heat.objects.resource.Resource class method)
get_by_physical_resource_id() (heat.objects.resource.Resource class method)
get_security_service() (heat.engine.clients.os.manila.ManilaClientPlugin method)
get_server() (heat.engine.clients.os.nova.NovaClientPlugin method)
get_share_network() (heat.engine.clients.os.manila.ManilaClientPlugin method)
get_share_snapshot() (heat.engine.clients.os.manila.ManilaClientPlugin method)
get_share_type() (heat.engine.clients.os.manila.ManilaClientPlugin method)
get_signal_url() (heat.engine.clients.os.swift.SwiftClientPlugin method)
get_snapshot_by_stack() (heat.objects.snapshot.Snapshot class method)
get_stack_lifecycle_plugins() (heat.engine.environment.Environment method)
get_status() (heat.engine.clients.os.nova.NovaClientPlugin method)
(heat.objects.stack.Stack class method)
get_temp_url() (heat.engine.clients.os.swift.SwiftClientPlugin method)
get_template() (heat.api.cfn.v1.stacks.StackController method)
(heat.engine.service.EngineService method)
(heat.rpc.client.EngineClient method)
get_types() (heat.engine.environment.Environm

method)
get_cfn_metadata_server_url()
(heat.engine.clients.os.heat_plugin.HeatClientPlugin method)
get_child_environment() (in module heat.engine.environment)
get_class()
(heat.engine.environment.ClassResourceInfo method)
(heat.engine.environment.Environment method)
(heat.engine.environment.MapResourceInfo method)
(heat.engine.environment.ResourceInfo method)
(heat.engine.environment.ResourceRegistry method)
(heat.engine.environment.TemplateResourceInfo method)
get_class_to_instantiate()
(heat.engine.environment.Environment method)
(heat.engine.environment.ResourceInfo method)
(heat.engine.environment.ResourceRegistry method)
(heat.engine.environment.TemplateResourceInfo method)
get_collection_links() (in module heat.api.openstack.v1.views.views_common)
get_console_urls()
(heat.engine.clients.os.nova.NovaClientPlugin method)
get_constraint()
(heat.engine.environment.Environment method)
get_container_by_ref()
(heat.engine.clients.os.barbican.BarbicanClientPlugin method)
get_dep_attrs()
(heat.engine.stack.Stack static method)
get_cfn_metadata_server_url()
(heat.engine.clients.os.heat_plugin.HeatClientPlugin method)
get_child_environment() (in module heat.engine.environment)
get_class()
(heat.engine.environment.ClassResourceInfo method)
(heat.engine.environment.Environment method)
(heat.engine.environment.MapResourceInfo method)
(heat.engine.environment.ResourceInfo method)
(heat.engine.environment.ResourceRegistry method)
(heat.engine.environment.TemplateResourceInfo method)
get_class_to_instantiate()
(heat.engine.environment.Environment method)
(heat.engine.environment.ResourceInfo method)
(heat.engine.environment.ResourceRegistry method)
(heat.engine.environment.TemplateResourceInfo method)
get_collection_links() (in module heat.api.openstack.v1.views.views_common)
get_console_urls()
(heat.engine.clients.os.nova.NovaClientPlugin method)
get_constraint()
(heat.engine.environment.Environment method)
get_container_by_ref()
(heat.engine.clients.os.barbican.BarbicanClientPlugin method)
get_dep_attrs()
(heat.engine.stack.Stack static method)
get_unserialized_body()
(heat.api.aws.exception.HeatAPIException method)
get_user_value()
(heat.engine.properties.Properties method)
get_val()
(heat.objects.resource_data.ResourceData class method)
get_value()
(heat.engine.properties.Property method)
get_volume()
(heat.engine.clients.os.cinder.CinderClientPlugin method)
get_volume_api_version()
(heat.engine.clients.os.cinder.CinderClientPlugin method)
get_volume_backup()
(heat.engine.clients.os.cinder.CinderClientPlugin method)
get_volume_snapshot()
(heat.engine.clients.os.cinder.CinderClientPlugin method)
get_volume_type()
(heat.engine.clients.os.cinder.CinderClientPlugin method)
get_watch_server_url()
(heat.engine.clients.os.heat_plugin.HeatClientPlugin method)
get_workflow_by_identifier()
(heat.engine.clients.os.mistral.MistralClientPlugin method)
get_yaql_parser()
(heat.engine.hot.functions.Yaql class method)
GetAtt (class in heat.engine.cfn.functions)
(class in

method)
get_details()
(heat.engine.watchrule.WatchRule
method)
get_domain_id()
(heat.engine.clients.os.designate.Desi
gnateClientPlugin method)
get_ec2_keypair()
(heat.engine.clients.os.keystone.heat_
keystoneclient.KsClientWrapper
method)
get_engine_id()
(heat.engine.stack_lock.StackLock
method)
(heat.objects.stack_lock.StackLock
class method)
get_environment()
(heat.engine.service.EngineService
method)
(heat.rpc.client.EngineClient
method)
get_event_sinks()
(heat.engine.environment.Environmen
t method)
get_files()
(heat.engine.service.EngineService
method)
(heat.rpc.client.EngineClient
method)
get_flavor()
(heat.engine.clients.os.nova.NovaClie
ntPlugin method)
get_heat_cfn_url()
(heat.engine.clients.os.heat_plugin.He
atClientPlugin method)
get_heat_url()
(heat.engine.clients.os.heat_plugin.He
atClientPlugin method)
get_host()
(heat.engine.clients.os.nova.NovaClie
ntPlugin method)
get_href()
heat.engine.hot.functions)
GetAttAllAttributes (class in
heat.engine.hot.functions)
GetAttThenSelect (class in
heat.engine.hot.functions)
GetAZs (class in
heat.engine.cfn.functions)
GetFile (class in
heat.engine.hot.functions)
GetParam (class in
heat.engine.hot.functions)
glance()
(heat.engine.resource.Resource
method)
GlanceClientPlugin (class in
heat.engine.clients.os.glance)
global_index()
(heat.api.openstack.v1.software_con
figs.SoftwareConfigController
method)
(heat.api.openstack.v1.stacks.Sta
ckController method)
GlobResourceInfo (class in
heat.engine.environment)
Graph (class in
heat.engine.dependencies)
graph()
(heat.engine.dependencies.Depende
ncies method)

(heat.api.versions.Controller method)
get_image()
(heat.engine.clients.os.glance.Glance ClientPlugin method)
get_image_id()
(heat.engine.clients.os.sahara.Sahara ClientPlugin method)
get_image_id_by_name()
(heat.engine.clients.os.sahara.Sahara ClientPlugin method)
get_insecure_option()
(heat.engine.clients.os.heat_plugin.HeatClientPlugin method)
get_ip()
(heat.engine.clients.os.nova.NovaClientPlugin method)
get_keypair()
(heat.engine.clients.os.nova.NovaClientPlugin method)
get_kwargs_for_cloning()
(heat.engine.stack.Stack method)
get_live_resource_data()
(heat.engine.resource.Resource method)
get_live_state()
(heat.engine.resource.Resource method)

H

handle_adopt()
(heat.engine.resource.Resource method)
handle_create() (built-in function)
handle_delete() (built-in function)
(heat.engine.resource.Resource method)
handle_delete_snapshot() (built-in function)
handle_resume() (built-in function)
heat.engine.plugin_manager (module)
heat.engine.properties (module)
heat.engine.resource (module)
heat.engine.rsrc_defn (module)
heat.engine.scheduler (module)
heat.engine.service (module)
heat.engine.service_software_config (module)
heat.engine.service_stack_watch

handle_snapshot_delete() (built-in function)
handle_suspend() (built-in function)
handle_update() (built-in function)
(heat.engine.resource.Resource method)
has_cache_data() (heat.engine.stack.Stack method)
has_default() (heat.engine.parameters.Parameter method)
(heat.engine.properties.Property method)
has_extension() (heat.engine.clients.os.cinder.CinderClientPlugin method)
(heat.engine.clients.os.nova.NovaClientPlugin method)
has_hook() (heat.engine.resource.Resource method)
has_interface() (heat.engine.resource.Resource method)
has_nested() (heat.engine.resource.Resource method)
has_timed_out() (heat.engine.stack.Stack method)
has_value() (heat.engine.parameters.Parameter method)
Heat Orchestration Template
heat() (heat.engine.resource.Resource method)
heat-api command line option
--config-dir
--config-file
--version
heat-api-cfn command line option
--config-dir
--config-file
(module)
heat.engine.stack (module)
heat.engine.stack_lock (module)
heat.engine.support (module)
heat.engine.sync_point (module)
heat.engine.template (module)
heat.engine.template_files (module)
heat.engine.timestamp (module)
heat.engine.translation (module)
heat.engine.update (module)
heat.engine.watchrule (module)
heat.engine.worker (module)
heat.hacking.checks (module)
heat.httpd.heat_api (module)
heat.httpd.heat_api_cfn (module)
heat.httpd.heat_api_cloudwatch (module)
heat.objects.base (module)
heat.objects.event (module)
heat.objects.fields (module)
heat.objects.raw_template (module)
heat.objects.raw_template_files (module)
heat.objects.resource (module)
heat.objects.resource_data (module)
heat.objects.service (module)
heat.objects.snapshot (module)
heat.objects.software_config (module)
heat.objects.software_deployment (module)
heat.objects.stack (module)
heat.objects.stack_lock (module)
heat.objects.stack_tag (module)
heat.objects.sync_point (module)
heat.objects.user_creds (module)
heat.objects.watch_data (module)
heat.objects.watch_rule (module)
heat.rpc.api (module)
heat.rpc.client (module)
heat.rpc.listener_client (module)
heat.rpc.worker_api (module)

--version
heat-api-cloudwatch command line option
--config-dir
--config-file
--version
heat-db-setup command line option
-h, --help
-p, --password
-r, --rootpw
-y, --yes
heat-engine command line option
--config-dir
--config-file
--version
heat-keystone-setup-domain command line option
--config-dir <DIR>
--config-file <PATH>
--stack-domain-admin <USERNAME>
--stack-domain-admin-password <PASSWORD>
--stack-user-domain-name <DOMAIN>
--version
-h, --help
heat.api.aws.ec2token (module)
heat.api.aws.exception (module)
heat.api.aws.utils (module)
heat.api.cfn.v1.signal (module)
heat.api.cfn.v1.stacks (module)
heat.api.cfn.versions (module)
heat.api.cloudwatch.watch (module)
heat.api.middleware.fault (module)
heat.api.middleware.ssl (module)
heat.api.middleware.version_negotiation (module)
heat.api.openstack.v1.actions (module)
heat.api.openstack.v1.build_info (module)
heat.rpc.worker_client (module)
heat.scaling.cooldown (module)
heat.scaling.lbutils (module)
heat.scaling.rolling_update (module)
heat.scaling.scalingutil (module)
heat.scaling.template (module)
heat.version (module)
HeatAccessDeniedError
HeatActionInProgressError
HeatAPIException
HeatAPINotImplementedError
HeatClientPlugin (class in heat.engine.clients.os.heat_plugin)
HeatIncompleteSignatureError
HeatInternalFailureError
HeatInvalidActionError
HeatInvalidClientTokenIdError
HeatInvalidParameterCombinationError
HeatInvalidParameterValueError
HeatInvalidQueryParameterError
HeatMalformedQueryStringError
HeatMissingActionError
HeatMissingAuthenticationTokenError
HeatMissingParameterError
HeatObject (class in heat.objects.base)
HeatObjectRegistry (class in heat.objects.base)
HeatOptInRequiredError
HeatRequestExpiredError
HeatRequestLimitExceeded
HeatServiceUnavailableError
HeatSignatureError
HeatTemplate (class in heat.engine.cfn.template)
HeatThrottlingError
HIDDEN (heat.engine.hot.parameters.HOTParamSchema attribute)
(heat.engine.parameters.Schema

heat.api.openstack.v1.events (module)
heat.api.openstack.v1.resources (module)
heat.api.openstack.v1.services (module)
heat.api.openstack.v1.software_configs (module)
heat.api.openstack.v1.software_deployments (module)
heat.api.openstack.v1.stacks (module)
heat.api.openstack.v1.util (module)
heat.api.openstack.v1.views.stacks_view (module)
heat.api.openstack.v1.views.views_common (module)
heat.api.openstack.versions (module)
heat.api.versions (module)
heat.engine.api (module)
heat.engine.attributes (module)
heat.engine.cfn.functions (module)
heat.engine.cfn.template (module)
heat.engine.check_resource (module)
heat.engine.clients.client_plugin (module)
heat.engine.clients.os.aodh (module)
heat.engine.clients.os.barbican (module)
heat.engine.clients.os.ceilometer (module)
heat.engine.clients.os.cinder (module)
heat.engine.clients.os.designate (module)
heat.engine.clients.os.glance (module)
heat.engine.clients.os.heat_plugin (module)
heat.engine.clients.os.keystone.heat_keystoneclient (module)
heat.engine.clients.os.keystone.keystone_constraints (module)
heat.engine.clients.os.magnum
attribute)
hidden() (heat.engine.parameters.Parameter method)
HostConstraint (class in heat.engine.clients.os.nova)
HOT
HOTemplate20130523 (class in heat.engine.hot.template)
HOTemplate20141016 (class in heat.engine.hot.template)
HOTemplate20150430 (class in heat.engine.hot.template)
HOTemplate20151015 (class in heat.engine.hot.template)
HOTemplate20160408 (class in heat.engine.hot.template)
HOTemplate20161014 (class in heat.engine.hot.template)
HOTParameters (class in heat.engine.hot.parameters)
HOTParamSchema (class in heat.engine.hot.parameters)

(module)
heat.engine.clients.os.manila (module)
heat.engine.clients.os.mistral (module)
heat.engine.clients.os.monasca (module)
heat.engine.clients.os.neutron.lbaas_constraints (module)
heat.engine.clients.os.neutron.neutron_constraints (module)
heat.engine.clients.os.nova (module)
heat.engine.clients.os.sahara (module)
heat.engine.clients.os.senlin (module)
heat.engine.clients.os.swift (module)
heat.engine.clients.os.trove (module)
heat.engine.clients.os.zaqar (module)
heat.engine.clients.progress (module)
heat.engine.constraint.common_constraints (module)
heat.engine.constraint.heat_constraints (module)
heat.engine.constraints (module)
heat.engine.dependencies (module)
heat.engine.environment (module)
heat.engine.event (module)
heat.engine.function (module)
heat.engine.hot.functions (module)
heat.engine.hot.parameters (module)
heat.engine.hot.template (module)
heat.engine.lifecycle_plugin (module)
heat.engine.notification.autoscaling (module)
heat.engine.notification.stack (module)
heat.engine.parameter_groups (module)
heat.engine.parameters (module)

- id (heat.engine.clients.os.keystone.heat_keystoneclient.AccessKey attribute)
  - (heat.objects.raw_template.RawTemplate attribute)
  - (heat.objects.raw_template_files.RawTemplateFiles attribute)
  - (heat.objects.user_creds.UserCreds attribute)
- identified_stack() (in module heat.api.openstack.v1.util)
- identifier()
  - (heat.engine.event.Event method)
  - (heat.engine.resource.Resource method)
  - (heat.engine.stack.Stack method)
  - (heat.objects.event.Event method)
  - (heat.objects.stack.Stack method)
- identify_stack()
  - (heat.engine.service.EngineService method)
  - (heat.rpc.client.EngineClient method)
- ignore_conflict_and_not_found (heat.engine.clients.client_plugin.ClientPlugin attribute)
- ignore_error_named() (heat.rpc.client.EngineClient method)
- ignore_not_found (heat.engine.clients.client_plugin.ClientPlugin attribute)
- IMAGE (heat.engine.clients.os.glance.Gla
- interface_detach() (heat.engine.clients.os.nova.NovaClientPlugin method)
- Invalid (class in heat.engine.function)
- invalidate() (heat.engine.clients.client_plugin.ClientPlugin method)
- IPConstraint (class in heat.engine.constraint.common_constraints)
- is_alive() (heat.rpc.listener_client.EngineListenerClient method)
- is_bad_request() (heat.engine.clients.os.nova.NovaClientPlugin method)
  - (heat.engine.clients.os.senlin.SenlinClientPlugin method)
- is_client_exception() (heat.engine.clients.client_plugin.ClientPlugin method)
  - (heat.engine.clients.os.swift.SwiftClientPlugin method)
- is_conflict() (heat.engine.clients.client_plugin.ClientPlugin method)
  - (heat.engine.clients.os.aodh.AodhClientPlugin method)
  - (heat.engine.clients.os.ceilometer.CeilometerClientPlugin method)
  - (heat.engine.clients.os.cinder.CinderClientPlugin method)
  - (heat.engine.clients.os.glance.GlanceClientPlugin method)
  - (heat.engine.clients.os.heat_plugin.HeatClientPlugin method)
  - (heat.engine.clients.os.magnum.MagnumClientPlugin method)

nceClientPlugin attribute)
ImageConstraint (class in heat.engine.clients.os.glance)
(class in heat.engine.clients.os.sahara)
IMMUTABLE (heat.engine.constraints.Schema attribute)
(heat.engine.hot.parameters.HOTParamSchema attribute)
(heat.engine.parameters.Schema attribute)
(heat.engine.properties.Schema attribute)
immutable() (heat.engine.properties.Property method)
immutable_params_modified() (heat.engine.parameters.Parameters method)
implemented() (heat.engine.properties.Property method)
IN_PROGRESS (heat.engine.resource.Resource attribute)
(heat.engine.stack.Stack attribute)
index() (heat.api.openstack.v1.events.EventController method)
(heat.api.openstack.v1.resources.ResourceController method)
(heat.api.openstack.v1.services.ServiceController method)
(heat.api.openstack.v1.software_configs.SoftwareConfigController method)
(heat.api.openstack.v1.software_deployments.SoftwareDeploymentController method)
(heat.api.openstack.v1.stacks.S
(heat.engine.clients.os.manila.ManilaClientPlugin method)
(heat.engine.clients.os.mistral.MistralClientPlugin method)
(heat.engine.clients.os.nova.NovaClientPlugin method)
(heat.engine.clients.os.sahara.SaharaClientPlugin method)
(heat.engine.clients.os.swift.SwiftClientPlugin method)
(heat.engine.clients.os.trove.TroveClientPlugin method)
is_hook_definition() (in module heat.engine.environment)
is_not_found() (heat.engine.clients.client_plugin.ClientPlugin method)
(heat.engine.clients.os.aodh.AodhClientPlugin method)
(heat.engine.clients.os.barbican.BarbicanClientPlugin method)
(heat.engine.clients.os.ceilometer.CeilometerClientPlugin method)
(heat.engine.clients.os.cinder.CinderClientPlugin method)
(heat.engine.clients.os.designate.DesignateClientPlugin method)
(heat.engine.clients.os.glance.GlanceClientPlugin method)
(heat.engine.clients.os.heat_plugin.HeatClientPlugin method)
(heat.engine.clients.os.magnum.MagnumClientPlugin method)
(heat.engine.clients.os.manila.ManilaClientPlugin method)
(heat.engine.clients.os.mistral.MistralClientPlugin method)
(heat.engine.clients.os.monasca.MonascaClientPlugin method)
(heat.engine.clients.os.nova.NovaClientPlugin method)
(heat.engine.clients.os.sahara.Sahara

tackController method)
INIT (heat.engine.resource.Resource attribute)
init_application() (in module heat.httpd.heat_api)
(in module heat.httpd.heat_api_cfn)
(in module heat.httpd.heat_api_cloudwatch)
Input parameters
InstantiationData (class in heat.api.openstack.v1.stacks)
INTEGER (heat.engine.attributes.Schema attribute)
(heat.engine.constraints.Schema attribute)
INTEGER_TYPE (heat.engine.constraints.Schema attribute)
interface_attach() (heat.engine.clients.os.nova.NovaClientPlugin method)
ClientPlugin method)
(heat.engine.clients.os.senlin.SenlinClientPlugin method)
(heat.engine.clients.os.swift.SwiftClientPlugin method)
(heat.engine.clients.os.trove.TroveClientPlugin method)
(heat.engine.clients.os.zaqar.ZaqarClientPlugin method)
is_not_registered() (heat.engine.clients.os.sahara.SaharaClientPlugin method)
is_over_limit() (heat.engine.clients.client_plugin.ClientPlugin method)
(heat.engine.clients.os.aodh.AodhClientPlugin method)
(heat.engine.clients.os.ceilometer.CeilometerClientPlugin method)
(heat.engine.clients.os.cinder.CinderClientPlugin method)
(heat.engine.clients.os.glance.GlanceClientPlugin method)
(heat.engine.clients.os.heat_plugin.HeatClientPlugin method)
(heat.engine.clients.os.magnum.MagnumClientPlugin method)
(heat.engine.clients.os.manila.ManilaClientPlugin method)
(heat.engine.clients.os.mistral.MistralClientPlugin method)
(heat.engine.clients.os.nova.NovaClientPlugin method)
(heat.engine.clients.os.sahara.SaharaClientPlugin method)
(heat.engine.clients.os.swift.SwiftClientPlugin method)
(heat.engine.clients.os.trove.TroveClientPlugin method)
is_service_available() (heat.engine.resource.Resource class method)

is_un_processable() (heat.engine.clients.os.monasca.MonascaClientPlugin method)
is_unprocessable_entity() (heat.engine.clients.os.nova.NovaClientPlugin method)
is_using_neutron() (heat.engine.resource.Resource method)
is_valid_restricted_action() (in module heat.engine.environment)
is_valid_status() (in module heat.engine.support)
is_valid_temp_url_path() (heat.engine.clients.os.swift.SwiftClientPlugin method)
ISO8601Constraint (class in heat.engine.constraint.common_constraints)
iter_resources() (heat.engine.stack.Stack method)
iterable_by() (heat.engine.environment.ResourceRegistry method)

J

Join (class in heat.engine.cfn.functions)
(class in heat.engine.hot.functions)
JoinMultiple (class in heat.engine.hot.functions)
Json (class in heat.objects.fields)
JsonField (class in heat.objects.fields)
JsonParam (class in heat.engine.parameters)

K

KEY_MANAGER (heat.engine.clients.os.barbican.BarbicanClientPlugin attribute)
KeypairConstraint (class in heat.engine.clients.os.keystone.keystone_constraints)
KeystoneGroupConstraint (class in heat.engine.clients.os.keystone.keystone_constraints)

heat.engine.clients.os.nova)
KEYS (heat.engine.attributes.Schema attribute)
(heat.engine.constraints.Schema attribute)
(heat.engine.hot.parameters.HOTParamSchema attribute)
(heat.engine.parameters.Schema attribute)
(heat.engine.properties.Schema attribute)
keystone() (heat.engine.resource.Resource method)
KeystoneBaseConstraint (class in heat.engine.clients.os.keystone.keystone_constraints)
KeystoneClient (class in heat.engine.clients.os.keystone.heat_keystoneclient)
KeystoneDomainConstraint (class in heat.engine.clients.os.keystone.keystone_constraints)
KeystoneProjectConstraint (class in heat.engine.clients.os.keystone.keystone_constraints)
KeystoneRegionConstraint (class in heat.engine.clients.os.keystone.keystone_constraints)
KeystoneRoleConstraint (class in heat.engine.clients.os.keystone.keystone_constraints)
KeystoneServiceConstraint (class in heat.engine.clients.os.keystone.keystone_constraints)
KeystoneUserConstraint (class in heat.engine.clients.os.keystone.keystone_constraints)
KsClientWrapper (class in heat.engine.clients.os.keystone.heat_keystoneclient)

L

LABEL (heat.engine.hot.parameters.HOTParamSchema attribute)
(heat.engine.parameters.Schema attribute)
label() (heat.engine.parameters.Parameter method)
LBaasV1ProviderConstraint (class in heat.engine.clients.os.neutron.neutron_constraints)
LBaasV2ProviderConstraint (class in heat.engine.clients.os.neutron.lbaas_constraints)
leaves()
list_stack_resources() (heat.api.cfn.v1.stacks.StackController method)
(heat.engine.service.EngineService method)
(heat.rpc.client.EngineClient method)
list_stacks() (heat.engine.service.EngineService method)
(heat.rpc.client.EngineClient method)
list_template_functions() (heat.api.openstack.v1.stacks.StackController method)

(heat.engine.dependencies.Dependencies method)
Length (class in heat.engine.constraints)
LifecyclePlugin (class in heat.engine.lifecycle_plugin)
LIST (heat.engine.attributes.Schema attribute)
(heat.engine.constraints.Schema attribute)
(heat.engine.hot.parameters.HOTParamSchema attribute)
(heat.engine.parameters.Schema attribute)
list()
(heat.api.cfn.v1.stacks.StackController method)
list_events()
(heat.engine.service.EngineService method)
(heat.rpc.client.EngineClient method)
list_metrics()
(heat.api.cloudwatch.watch.WatchController method)
list_opts() (in module heat.api.aws.ec2token)
(in module heat.api.middleware.ssl)
(in module heat.engine.clients.os.keystone.heat_keystoneclient)
list_outputs()
(heat.api.openstack.v1.stacks.StackController method)
(heat.engine.service.EngineService method)
(heat.rpc.client.EngineClient method)
list_resource_types()
(heat.api.openstack.v1.stacks.StackController method)
(heat.engine.service.EngineService method)
(heat.rpc.client.EngineClient method)
list_template_versions()
(heat.api.openstack.v1.stacks.StackController method)
(heat.engine.service.EngineService method)
(heat.rpc.client.EngineClient method)
LIST_TYPE
(heat.engine.constraints.Schema attribute)
ListenerConstraint (class in heat.engine.clients.os.neutron.lbaas_constraints)
listening()
(heat.engine.service.EngineListener method)
ListField (class in heat.objects.fields)
load()
(heat.engine.environment.Environment method)
(heat.engine.environment.ResourceRegistry method)
(heat.engine.resource.Resource class method)
(heat.engine.stack.Stack class method)
(heat.engine.template.Template class method)
(heat.engine.watchrule.WatchRule class method)
load_all()
(heat.engine.plugin_manager.PluginMapping method)
(heat.engine.stack.Stack class method)
load_from_module()
(heat.engine.plugin_manager.PluginMapping method)

method)
(heat.rpc.client.EngineClient method)
list_services() (heat.engine.service.EngineService method)
(heat.rpc.client.EngineClient method)
list_snapshots() (heat.api.openstack.v1.stacks.StackController method)
list_software_configs() (heat.engine.service.EngineService method)
(heat.engine.service_software_config.SoftwareConfigService method)
(heat.rpc.client.EngineClient method)
list_software_deployments() (heat.engine.service.EngineService method)
(heat.engine.service_software_config.SoftwareConfigService method)
(heat.rpc.client.EngineClient method)
load_resource() (in module heat.engine.check_resource)
LoadbalancerConstraint (class in heat.engine.clients.os.neutron.lbaas_constraints)
local_error_name() (heat.rpc.client.EngineClient method)
lock() (heat.engine.resource.Resource method)
log_resource_info() (heat.engine.environment.ResourceRegistry method)
lookup() (heat.api.openstack.v1.stacks.StackController method)

M

MACConstraint (class in heat.engine.constraint.common_constraints)
MagnumClientPlugin (class in heat.engine.clients.os.magnum)
make_key() (in module heat.engine.sync_point)
make_link() (in module heat.api.openstack.v1.util)
make_msg() (heat.rpc.client.EngineClient static method)
matches() (heat.engine.environment.GlobResourceInfo method)
(heat.engine.environment.ResourceInfo method)
matches_hook() (heat.engine.environment.ResourceRegistry method)
MAX (heat.engine.constraints.Range attribute)
member_definitions() (in module heat.scaling.template)

(heat.rpc.worker_client.WorkerClient static method)
make_replacement() (heat.engine.resource.Resource method)
make_template() (in module heat.scaling.template)
make_url() (in module heat.api.openstack.v1.util)
ManilaClientPlugin (class in heat.engine.clients.os.manila)
ManilaShareBaseConstraint (class in heat.engine.clients.os.manila)
ManilaShareNetworkConstraint (class in heat.engine.clients.os.manila)
ManilaShareSnapshotConstraint (class in heat.engine.clients.os.manila)
ManilaShareTypeConstraint (class in heat.engine.clients.os.manila)
MAP (heat.engine.attributes.Schema attribute)
(heat.engine.constraints.Schema attribute)
(heat.engine.hot.parameters.HOTParamSchema attribute)
(heat.engine.parameters.Schema attribute)
map() (heat.engine.dependencies.Graph method)
(heat.engine.parameters.Parameters method)
map_remote_error() (in module heat.api.aws.exception)
map_to_modules() (heat.engine.plugin_manager.PluginManager method)
MAP_TYPE (heat.engine.constraints.Schema attribute)
MemberListToMap (class in heat.engine.cfn.functions)
MESSAGING (heat.engine.clients.os.zaqar.ZaqarClientPlugin attribute)
meta_serialize() (heat.engine.clients.os.nova.NovaClientPlugin method)
meta_update() (heat.engine.clients.os.nova.NovaClientPlugin method)
Metadata
METADATA (heat.engine.cfn.functions.ResourceFacade attribute)
(heat.engine.hot.functions.ResourceFacade attribute)
Metadata service
metadata() (heat.api.openstack.v1.resources.ResourceController method)
(heat.api.openstack.v1.software_deployments.SoftwareDeploymentController method)
metadata_get() (heat.engine.resource.Resource method)
metadata_set() (heat.engine.resource.Resource method)
metadata_software_deployments() (heat.engine.service.EngineService method)
(heat.engine.service_software_config.SoftwareConfigService method)
(heat.rpc.client.EngineClient method)
metadata_update() (heat.engine.resource.Resource method)
METERING (heat.engine.clients.os.ceilometer.CeilometerClientPlugin attribute)
MIN (heat.engine.constraints.Range attribute)
MistralClientPlugin (class in

attribute)
MapMerge (class in heat.engine.hot.functions)
MAPPINGS (heat.engine.cfn.template.CfnTemplateBase attribute)
(heat.engine.hot.template.HOTemplate20130523 attribute)
MapReplace (class in heat.engine.hot.functions)
MapResourceInfo (class in heat.engine.environment)
mark_complete() (heat.engine.stack.Stack method)
mark_unhealthy() (heat.api.openstack.v1.resources.ResourceController method)
heat.engine.clients.os.mistral)
MonascaClientPlugin (class in heat.engine.clients.os.monasca)
MonascaNotificationConstraint (class in heat.engine.clients.os.monasca)
MONITORING (heat.engine.clients.os.monasca.MonascaClientPlugin attribute)
msg_fmt (heat.engine.dependencies.CircularDependencyException attribute)
Multi-region

N

needs_replace() (heat.engine.resource.Resource method)
needs_replace_with_prop_diff() (heat.engine.resource.Resource method)
needs_replace_with_tmpl_diff() (heat.engine.resource.Resource method)
needs_update() (in module heat.scaling.rolling_update)
Nested resource
Nested stack
NetworkConstraint (class in heat.engine.clients.os.neutron.neutron_constraints)
(class in heat.engine.clients.os.nova)
neutron() (heat.engine.resource.Resource
no_signal_actions (heat.engine.resource.Resource attribute)
NODATA (heat.engine.watchrule.WatchRule attribute)
Node (class in heat.engine.dependencies)
NORMAL (heat.engine.watchrule.WatchRule attribute)
Nova Instance metadata
nova() (heat.engine.resource.Resource method)
NovaBaseConstraint (class in heat.engine.clients.os.nova)
NovaClientPlugin (class in heat.engine.clients.os.nova)
NUMBER (heat.engine.constraints.Schema

method)
NeutronConstraint (class in heat.engine.clients.os.neutron.neutron_constraints)
next_batch() (in module heat.scaling.rolling_update)
no_log_warn() (in module heat.hacking.checks)
attribute)
(heat.engine.hot.parameters.HOTParamSchema attribute)
(heat.engine.parameters.Schema attribute)
NUMBER_TYPE (heat.engine.constraints.Schema attribute)
NumberParam (class in heat.engine.parameters)

O

OBJ_PROJECT_NAMESPACE (heat.objects.base.HeatObject attribute)
object_path_in_stack() (heat.engine.stack.Stack method)
OBJECT_STORE (heat.engine.clients.os.swift.SwiftClientPlugin attribute)
OpenStack Orchestrate ORCHESTRATION (heat.engine.clients.os.heat_plugin.HeatClientPlugin attribute)
output() (heat.engine.stack.Stack method)
OUTPUT_DESCRIPTION (heat.engine.cfn.template.CfnTemplateBase attribute)
(heat.engine.hot.template.HOTemplate20130523 attribute)
OUTPUT_KEYS (heat.engine.cfn.template.CfnTemplateBase attribute)
(heat.engine.hot.template.HOTemplate20130523 attribute)
OUTPUT_VALUE (heat.engine.cfn.template.CfnTemplateBase attribute)
(heat.engine.hot.template.HOTemplate20130523 attribute)
Outputs
OUTPUTS (heat.engine.cfn.template.CfnTemplateBase attribute)
(heat.engine.hot.template.HOTemplate20130523 attribute)

P

PARAM_ENVIRONMENT (heat.api.openstack.v1.stacks.InstantiationData attribute)
PARAM_ENVIRONMENT_FILES (heat.api.openstack.v1.stacks.InstantiationData attribute)
PARAM_FILES (heat.api.openstack.v1.stacks.InstantiationData attribute)
PARAM_PROJECT_ID (heat.engine.hot.parameters.HOTParameters attribute)
PARAM_REGION (heat.engine.hot.parameters.HOTParameters attribute)
    (heat.engine.parameters.Parameters attribute)
param_schemata() (heat.engine.cfn.template.CfnTemplateBase method)
    (heat.engine.hot.template.HOTemplate20130523 method)
    (heat.engine.template.Template method)
PARAM_STACK_ID (heat.engine.hot.parameters.HOTParameters attribute)
    (heat.engine.parameters.Parameters attribute)
PARAM_STACK_NAME (heat.api.openstack.v1.stacks.InstantiationData attribute)
    (heat.engine.hot.parameters.HOTParameters attribute)
    (heat.engine.parameters.Parameters attribute)
PARAM_TEMPLATE (heat.api.openstack.v1.stacks.InstantiationData attribute)
physical_resource_name_limit (heat.engine.resource.Resource attribute)
physical_resource_name_or_FnGetRefId() (heat.engine.resource.Resource method)
PluginConstraint (class in heat.engine.clients.os.sahara)
PluginManager (class in heat.engine.plugin_manager)
PluginMapping (class in heat.engine.plugin_manager)
policy_enforce() (in module heat.api.openstack.v1.util)
PolicyTypeConstraint (class in heat.engine.clients.os.senlin)
PollDelay
PoolConstraint (class in heat.engine.clients.os.neutron.lbaas_constraints)
PoolDeleteProgress (class in heat.engine.clients.progress)
PortConstraint (class in heat.engine.clients.os.neutron.neutron_constraints)
prepare_abandon() (heat.engine.resource.Resource method)
    (heat.engine.stack.Stack method)
prepare_args() (heat.api.openstack.v1.stacks.StackController method)
prepare_for_replace() (heat.engine.resource.Resource method)
preview()

onData attribute)
PARAM_TEMPLATE_URL
(heat.api.openstack.v1.stacks.InstantiationData attribute)
PARAM_USER_PARAMS
(heat.api.openstack.v1.stacks.InstantiationData attribute)
Parameter (class in heat.engine.parameters)
PARAMETER_GROUPS
(heat.engine.hot.template.HOTemplate20130523 attribute)
PARAMETER_KEYS
(heat.engine.hot.parameters.HOTParamSchema attribute)
(heat.engine.parameters.Schema attribute)
ParameterGroups (class in heat.engine.parameter_groups)
Parameters
(class in heat.engine.parameters)
PARAMETERS
(heat.engine.cfn.template.CfnTemplateBase attribute)
(heat.engine.hot.template.HOTemplate20130523 attribute)
parameters()
(heat.engine.cfn.template.CfnTemplateBase method)
(heat.engine.hot.template.HOTemplate20130523 method)
(heat.engine.template.Template method)
ParamRef (class in heat.engine.cfn.functions)
PARAMS
(heat.api.openstack.v1.stacks.InstantiationData attribute)
parent_resource
(heat.engine.stack.Stack attribute)
parse()
(heat.engine.parameters.CommaDelimitedStringParameter method)
(heat.engine.parameters.Schema method)
(heat.engine.properties.Properties method)
(heat.engine.properties.Schema method)
(heat.engine.resource.Resource method)
(heat.engine.template.Template method)
(heat.engine.template_files.TemplateFiles method)
(heat.engine.template_files.TemplateFiles.FileContentGetter method)
(heat.engine.template_files.TemplateFiles.FileMapping method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment.Segment.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment.Segment.Segment.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment.Segment.Segment.Segment.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment.Segment.Segment.Segment.Segment.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment.Segment.Segment.Segment.Segment.Segment.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment method)
(heat.engine.template_files.TemplateFiles.FileMapping.Segment

edListParam method)
(heat.engine.parameters.JsonParam method)
(heat.engine.template.Template method)
parse_condition()
(heat.engine.template.Template method)
parse_error_check()
(heat.api.openstack.v1.stacks.InstantiationData static method)
parse_last_modified()
(heat.engine.clients.os.swift.SwiftClientPlugin method)
parse_live_resource_data()
(heat.engine.resource.Resource method)
parsed_template()
(heat.engine.resource.Resource method)
ParsedParameter (class in heat.engine.parameters)
password
(heat.objects.user_creds.UserCreds attribute)
path_in_stack() (heat.engine.stack.Stack method)
pause() (heat.engine.resource.Resource static method)
periodic_watcher_task()
(heat.engine.service_stack_watch.StackWatch method)
persist_state_and_release_lock()
(heat.engine.stack.Stack method)
(heat.objects.stack.Stack class method)
physical_resource_name()
(heat.engine.resource.Resource method)
heat.engine.properties)
Provider resource
Provider template
ProviderConstraint (class in heat.engine.clients.os.neutron.neutron_constraints)
PSEUDO_PARAMETERS
(heat.engine.hot.parameters.HOTParameters attribute)
(heat.engine.parameters.Parameters attribute)
purge_db()
(heat.engine.stack.Stack method)
purge_deleted()
(heat.objects.resource.Resource class method)
put_metric_alarm()
(heat.api.cloudwatch.watch.WatchController method)
put_metric_data()
(heat.api.cloudwatch.watch.WatchController method)

Q

QoSPolicyConstraint (class in

heat.engine.clients.os.neutron.neutron_constraints)

R

Range (class in heat.engine.constraints)
RawTemplate (class in heat.objects.raw_template)
RawTemplateFiles (class in heat.objects.raw_template_files)
read_global_environment() (in module heat.engine.environment)
ReadOnlyDict (class in heat.engine.template_files)
rebuild() (heat.engine.clients.os.nova.NovaClientPlugin method)
record_create() (heat.engine.clients.os.designate.DesignateClientPlugin method)
record_delete() (heat.engine.clients.os.designate.DesignateClientPlugin method)
record_show() (heat.engine.clients.os.designate.DesignateClientPlugin method)
record_update() (heat.engine.clients.os.designate.DesignateClientPlugin method)
reduce_physical_resource_name() (heat.engine.resource.Resource static method)
Ref() (in module heat.engine.cfn.functions)
reformat_dict_keys() (in module heat.api.aws.utils)
refresh() (heat.objects.resource.Resource method)
(heat.objects.stack.Stack resize() (heat.engine.clients.os.nova.NovaClientPlugin method)
RESOLVE (heat.engine.translation.TranslationRule attribute)
resolve() (in module heat.engine.function)
resolve_static_data() (heat.engine.stack.Stack method)
Resource
(class in heat.engine.resource)
(class in heat.objects.resource)
Resource attribute
Resource group
Resource Metadata
Resource plugin
Resource property
Resource provider
resource_by_refid() (heat.engine.stack.Stack method)
resource_client_name (heat.engine.clients.os.barbican.ContainerConstraint attribute)
(heat.engine.clients.os.barbican.SecretConstraint attribute)
(heat.engine.clients.os.cinder.BaseCinderConstraint attribute)
(heat.engine.clients.os.designate.DesignateDomainConstraint attribute)
(heat.engine.clients.os.glance.ImageConstraint attribute)
(heat.engine.clients.os.keystone.keystone_constraints.KeystoneBaseConstraint attribute)
(heat.engine.clients.os.magnum.Bay

method)
refresh_server()
(heat.engine.clients.os.nova.NovaClientPlugin method)
regenerate_info_schema()
(heat.engine.resource.Resource method)
region_name
(heat.objects.user_creds.UserCreds attribute)
register_access_allowed_handler()
(heat.engine.stack.Stack method)
register_class()
(heat.engine.environment.Environment method)
(heat.engine.environment.ResourceRegistry method)
register_constraint()
(heat.engine.environment.Environment method)
register_event_sink()
(heat.engine.environment.Environment method)
register_stack_lifecycle_plugin()
(heat.engine.environment.Environment method)
registry
(heat.engine.environment.ResourceInfo attribute)
RelativeDNSNameConstraint (class in heat.engine.constraint.common_constraints)
release()
(heat.engine.stack_lock.StackLock method)
(heat.objects.stack_lock.StackLock class method)
reload_loadbalancers() (in module heat.scaling.lbutils)
remove_all_resources()
(heat.engine.template.Template method)
modelConstraint attribute)
(heat.engine.clients.os.manila.ManilaShareBaseConstraint attribute)
(heat.engine.clients.os.mistral.WorkflowConstraint attribute)
(heat.engine.clients.os.monasca.MonascaNotificationConstraint attribute)
(heat.engine.clients.os.nova.NovaBaseConstraint attribute)
(heat.engine.clients.os.sahara.SaharaBaseConstraint attribute)
(heat.engine.clients.os.trove.FlavorConstraint attribute)
(heat.engine.constraints.BaseCustomConstraint attribute)
resource_definitions()
(heat.engine.cfn.template.CfnTemplateBase method)
(heat.engine.hot.template.HOTemplate20130523 method)
(heat.engine.template.Template method)
resource_get()
(heat.engine.stack.Stack method)
resource_getter_name
(heat.engine.clients.os.barbican.ContainerConstraint attribute)
(heat.engine.clients.os.barbican.SecretConstraint attribute)
(heat.engine.clients.os.cinder.VolumeBackupConstraint attribute)
(heat.engine.clients.os.cinder.VolumeConstraint attribute)
(heat.engine.clients.os.cinder.VolumeSnapshotConstraint attribute)
(heat.engine.clients.os.cinder.VolumeTypeConstraint attribute)
(heat.engine.clients.os.designate.DesignateDomainConstraint attribute)
(heat.engine.clients.os.glance.ImageConstraint attribute)
(heat.engine.clients.os.keystone.key

method)
remove_item()
(heat.engine.environment.ResourceRegistry method)
remove_msg_queue()
(heat.engine.service.ThreadGroupManager method)
remove_resource()
(heat.engine.stack.Stack method)
(heat.engine.template.Template method)
remove_resources_except()
(heat.engine.environment.ResourceRegistry method)
Removed (class in heat.engine.hot.functions)
rename()
(heat.engine.clients.os.nova.NovaClientPlugin method)
reparse()
(heat.engine.resource.Resource method)
Repeat (class in heat.engine.hot.functions)
Replace (class in heat.engine.cfn.functions)
(class in heat.engine.hot.functions)
REPLACE
(heat.engine.translation.TranslationRule attribute)
ReplaceJson (class in heat.engine.hot.functions)
REQUEST_SCOPE
(heat.api.openstack.v1.actions.ActionController attribute)
(heat.api.openstack.v1.build_info.BuildInfoController attribute)
(heat.api.openstack.v1.events.EventController attribute)
(heat.api.openstack.v1.resources.ResourceController attribute)
stone_constraints.KeystoneDomainConstraint attribute)
(heat.engine.clients.os.keystone.keystone_constraints.KeystoneGroupConstraint attribute)
(heat.engine.clients.os.keystone.keystone_constraints.KeystoneProjectConstraint attribute)
(heat.engine.clients.os.keystone.keystone_constraints.KeystoneRegionConstraint attribute)
(heat.engine.clients.os.keystone.keystone_constraints.KeystoneRoleConstraint attribute)
(heat.engine.clients.os.keystone.keystone_constraints.KeystoneServiceConstraint attribute)
(heat.engine.clients.os.keystone.keystone_constraints.KeystoneUserConstraint attribute)
(heat.engine.clients.os.magnum.BaymodelConstraint attribute)
(heat.engine.clients.os.manila.ManilaShareNetworkConstraint attribute)
(heat.engine.clients.os.manila.ManilaShareSnapshotConstraint attribute)
(heat.engine.clients.os.manila.ManilaShareTypeConstraint attribute)
(heat.engine.clients.os.mistral.WorkflowConstraint attribute)
(heat.engine.clients.os.monasca.MonascaNotificationConstraint attribute)
(heat.engine.clients.os.nova.FlavorConstraint attribute)
(heat.engine.clients.os.nova.HostConstraint attribute)
(heat.engine.clients.os.nova.KeypairConstraint attribute)
(heat.engine.clients.os.nova.NetworkConstraint attribute)
(heat.engine.clients.os.nova.ServerConstraint attribute)

(heat.api.openstack.v1.services.ServiceController attribute)
(heat.api.openstack.v1.software_configs.SoftwareConfigController attribute)
(heat.api.openstack.v1.software_deployments.SoftwareDeploymentController attribute)
(heat.api.openstack.v1.stacks.StackController attribute)
REQUIRED (heat.engine.constraints.Schema attribute)
(heat.engine.properties.Schema attribute)
required() (heat.engine.properties.Property method)
required_by() (heat.engine.dependencies.Dependencies method)
(heat.engine.dependencies.Node method)
(heat.engine.resource.Resource method)
required_service_extension (heat.engine.resource.Resource attribute)
requires() (heat.engine.dependencies.Dependencies method)
(heat.engine.dependencies.Node method)
requires_deferred_auth (heat.engine.resource.Resource attribute)
requires_deferred_auth() (heat.engine.stack.Stack method)
RES_DELETION_POLICY (heat.engine.cfn.template.CfnTemplateBase attribute)
(heat.engine.hot.template.HOTemplate attribute)
(heat.engine.clients.os.sahara.ImageConstraint attribute)
(heat.engine.clients.os.sahara.PluginConstraint attribute)
(heat.engine.clients.os.trove.FlavorConstraint attribute)
(heat.engine.constraints.BaseCustomConstraint attribute)
resource_id_set() (heat.engine.resource.Resource method)
resource_mark_unhealthy() (heat.engine.service.EngineService method)
(heat.rpc.client.EngineClient method)
resource_name (heat.engine.clients.os.neutron.lbaas_constraints.ListenerConstraint attribute)
(heat.engine.clients.os.neutron.lbaas_constraints.LoadbalancerConstraint attribute)
(heat.engine.clients.os.neutron.lbaas_constraints.PoolConstraint attribute)
(heat.engine.clients.os.neutron.neutron_constraints.AddressScopeConstraint attribute)
(heat.engine.clients.os.neutron.neutron_constraints.NeutronConstraint attribute)
(heat.engine.clients.os.neutron.neutron_constraints.PortConstraint attribute)
(heat.engine.clients.os.neutron.neutron_constraints.QoSPolicyConstraint attribute)
(heat.engine.clients.os.neutron.neutron_constraints.RouterConstraint attribute)
(heat.engine.clients.os.neutron.neutron_constraints.SecurityGroupConstr

mplate20130523 attribute)
RES_DEPENDS_ON (heat.engine.cfn.template.CfnTemplateBase attribute)
(heat.engine.hot.template.HOTemplate20130523 attribute)
RES_DESCRIPTION (heat.engine.cfn.template.CfnTemplateBase attribute)
(heat.engine.hot.template.HOTemplate20130523 attribute)
RES_METADATA (heat.engine.cfn.template.CfnTemplateBase attribute)
(heat.engine.hot.template.HOTemplate20130523 attribute)
RES_PROPERTIES (heat.engine.cfn.template.CfnTemplateBase attribute)
(heat.engine.hot.template.HOTemplate20130523 attribute)
RES_TYPE (heat.engine.cfn.template.CfnTemplateBase attribute)
(heat.engine.hot.template.HOTemplate20130523 attribute)
RES_UPDATE_POLICY (heat.engine.cfn.template.CfnTemplateBase attribute)
(heat.engine.hot.template.HOTemplate20130523 attribute)
reset() (heat.engine.service.EngineListener method)
(heat.engine.service.EngineService method)
(heat.engine.worker.WorkerService method)
reset_dependencies() (heat.engine.stack.Stack method)
reset_resolved_values() (heat.engine.attributes.Attributes
aint attribute)
(heat.engine.clients.os.neutron.neutron_constraints.SubnetConstraint attribute)
(heat.engine.clients.os.neutron.neutron_constraints.SubnetPoolConstraint attribute)
resource_schema() (heat.api.openstack.v1.stacks.StackController method)
(heat.engine.service.EngineService method)
(heat.rpc.client.EngineClient method)
resource_signal() (heat.engine.service.EngineService method)
(heat.rpc.client.EngineClient method)
resource_to_template() (heat.engine.resource.Resource class method)
ResourceController (class in heat.api.openstack.v1.resources)
ResourceData (class in heat.objects.resource_data)
ResourceDefinition (class in heat.engine.rsrc_defn)
ResourceDefinition.Diff (class in heat.engine.rsrc_defn)
ResourceFacade (class in heat.engine.cfn.functions)
(class in heat.engine.hot.functions)
ResourceInfo (class in heat.engine.environment)
ResourceRef (class in heat.engine.cfn.functions)
ResourceRegistry (class in heat.engine.environment)
RESOURCES (heat.engine.cfn.template.CfnTemplateBase attribute)

method)
reset_resource_attributes()
(heat.engine.stack.Stack method)
reset_stack_status()
(heat.engine.service.EngineService
method)
reset_state_on_error() (in module
heat.engine.stack)
(heat.engine.hot.template.HOTempla
te20130523 attribute)
resources (heat.engine.stack.Stack
attribute)
ResourceTypeConstraint (class in
heat.engine.constraint.heat_constraint
s)
restart_resource()
(heat.engine.stack.Stack method)
RESTORE (heat.engine.stack.Stack
attribute)
restore() (heat.engine.stack.Stack
method)
restore_prev_rsrc()
(heat.engine.resource.Resource
method)
restore_snapshot()
(heat.api.openstack.v1.stacks.StackCo
ntroller method)
result()
(heat.engine.cfn.functions.Base64
method)
(heat.engine.cfn.functions.FindInMa
p method)
(heat.engine.cfn.functions.GetAZs
method)
(heat.engine.cfn.functions.GetAtt
method)
(heat.engine.cfn.functions.Join
method)
(heat.engine.cfn.functions.MemberLi
stToMap method)
(heat.engine.cfn.functions.ParamRef
method)
(heat.engine.cfn.functions.Replace
method)
(heat.engine.cfn.functions.Resource
Facade method)
(heat.engine.cfn.functions.Resource
Ref method)
(heat.engine.cfn.functions.Select
method)

(heat.engine.cfn.functions.Split method)
(heat.engine.function.Function method)
(heat.engine.function.Invalid method)
(heat.engine.hot.functions.Digest method)
(heat.engine.hot.functions.Equals method)
(heat.engine.hot.functions.GetAtt method)
(heat.engine.hot.functions.GetAttAllAttributes method)
(heat.engine.hot.functions.GetAttThenSelect method)
(heat.engine.hot.functions.GetFile method)
(heat.engine.hot.functions.GetParam method)
(heat.engine.hot.functions.JoinMultiple method)
(heat.engine.hot.functions.MapMerge method)
(heat.engine.hot.functions.MapReplace method)
(heat.engine.hot.functions.Removed method)
(heat.engine.hot.functions.Repeat method)
(heat.engine.hot.functions.ReplaceJson method)
(heat.engine.hot.functions.StrSplit method)
(heat.engine.hot.functions.Yaql method)
RESUME (heat.api.openstack.v1.actions.ActionController attribute)
(heat.engine.resource.Resource attribute)
(heat.engine.stack.Stack attribute)

resume()
(heat.engine.resource.Resource
method)
(heat.engine.stack.Stack method)
retry_if_connection_err() (in module
heat.engine.clients.client_plugin)
retry_if_result_is_false() (in module
heat.engine.clients.client_plugin)
retry_on_conflict() (in module
heat.objects.resource)
reverse_copy()
(heat.engine.dependencies.Graph
method)
(heat.engine.dependencies.Node
method)
ROLLBACK
(heat.engine.resource.Resource
attribute)
(heat.engine.stack.Stack attribute)
rollback() (heat.engine.stack.Stack
method)
root_stack_id()
(heat.engine.stack.Stack method)
roots()
(heat.engine.dependencies.Dependenci
es method)
RouterConstraint (class in
heat.engine.clients.os.neutron.neutron
_constraints)
RPC_API_VERSION
(heat.engine.service.EngineService
attribute)
(heat.engine.worker.WorkerService
attribute)
rpc_client()
(heat.engine.resource.Resource
method)
rsrc_defn_from_snippet()
(heat.engine.hot.template.HOTemplate
20130523 class method)
rule_actions()
(heat.engine.watchrule.WatchRule

method)
rule_can_use_sample() (in module heat.engine.watchrule)
RULE_KEYS (heat.engine.translation.TranslationRule attribute)
run_rule() (heat.engine.watchrule.WatchRule method)
run_to_completion() (heat.engine.scheduler.TaskRunner method)

S

SaharaBaseConstraint (class in heat.engine.clients.os.sahara)
SaharaClientPlugin (class in heat.engine.clients.os.sahara)
Schema (class in heat.engine.attributes)
(class in heat.engine.constraints)
(class in heat.engine.parameters)
(class in heat.engine.properties)
SCHEMA (heat.engine.constraints.Schema attribute)
(heat.engine.hot.parameters.HOTParamSchema attribute)
(heat.engine.parameters.Schema attribute)
(heat.engine.properties.Schema attribute)
schema_from_outputs() (heat.engine.attributes.Attributes static method)
schema_from_params() (heat.engine.properties.Properties static method)
schema_to_parameters_and_properties() (heat.engine.properties.Properties class method)
Split (class in heat.engine.cfn.functions)
ssl_options (heat.api.aws.ec2token.EC2Token attribute)
SSLMiddleware (class in heat.api.middleware.ssl)
Stack
(class in heat.engine.stack)
(class in heat.objects.stack)
stack (heat.engine.function.Function attribute)
(heat.engine.resource.Resource attribute)
Stack resource
stack_cancel_update() (heat.engine.service.EngineService method)
(heat.rpc.client.EngineClient method)
stack_check() (heat.engine.service.EngineService method)
(heat.rpc.client.EngineClient

method)
schemata() (in module heat.engine.attributes)
(in module heat.engine.properties)
secret (heat.engine.clients.os.keystone.heat_keystoneclient.AccessKey attribute)
SecretConstraint (class in heat.engine.clients.os.barbican)
SECTIONS (heat.engine.cfn.template.CfnTemplate attribute)
(heat.engine.cfn.template.CfnTemplateBase attribute)
(heat.engine.hot.template.HOTemplate20130523 attribute)
(heat.engine.hot.template.HOTemplate20161014 attribute)
SECTIONS_NO_DIRECT_ACCESS (heat.engine.cfn.template.CfnTemplateBase attribute)
(heat.engine.hot.template.HOTemplate20130523 attribute)
SecurityGroupConstraint (class in heat.engine.clients.os.neutron.neutron_constraints)
Select (class in heat.engine.cfn.functions)
select_and_update() (heat.objects.resource.Resource method)
(heat.objects.stack.Stack class method)
select_from_attribute() (in module heat.engine.attributes)
SEND (heat.engine.service.EngineListener attribute)
send() (heat.engine.service.EngineListener method)
(heat.engine.service.ThreadGroupManager method)
stack_domain (heat.engine.clients.os.keystone.heat_keystoneclient.KsClientWrapper attribute)
stack_domain_id (heat.engine.clients.os.keystone.heat_keystoneclient.KsClientWrapper attribute)
stack_domain_user_token() (heat.engine.clients.os.keystone.heat_keystoneclient.KsClientWrapper method)
stack_list_snapshots() (heat.engine.service.EngineService method)
(heat.rpc.client.EngineClient method)
stack_name() (heat.api.openstack.v1.stacks.InstantiationData method)
stack_restore() (heat.engine.service.EngineService method)
(heat.rpc.client.EngineClient method)
stack_resume() (heat.engine.service.EngineService method)
(heat.rpc.client.EngineClient method)
stack_snapshot() (heat.engine.service.EngineService method)
(heat.rpc.client.EngineClient method)
stack_suspend() (heat.engine.service.EngineService method)
(heat.rpc.client.EngineClient method)
stack_task()

nager method)
(in module heat.engine.notification.autoscaling)
(in module heat.engine.notification.stack)
SenlinClientPlugin (class in heat.engine.clients.os.senlin)
serialize_input_data() (in module heat.engine.sync_point)
server_to_ipaddress() (heat.engine.clients.os.nova.NovaClientPlugin method)
ServerConstraint (class in heat.engine.clients.os.nova)
ServerCreateProgress (class in heat.engine.clients.progress)
ServerDeleteProgress (class in heat.engine.clients.progress)
ServerUpdateProgress (class in heat.engine.clients.progress)
Service (class in heat.objects.service)
service_manage_cleanup() (heat.engine.service.EngineService method)
service_manage_report() (heat.engine.service.EngineService method)
service_type (heat.engine.clients.os.neutron.lbaas_constraints.LBaasV2ProviderConstraint attribute)
(heat.engine.clients.os.neutron.neutron_constraints.LBaasV1ProviderConstraint attribute)
(heat.engine.clients.os.neutron.neutron_constraints.ProviderConstraint attribute)
service_types (heat.engine.clients.client_plugin.ClientPlugin attribute)
(heat.engine.clients.os.aodh.AodhClientPlugin attribute)
(heat.engine.stack.Stack method)
StackController (class in heat.api.cfn.v1.stacks)
(class in heat.api.openstack.v1.stacks)
StackLock (class in heat.engine.stack_lock)
(class in heat.objects.stack_lock)
StackSerializer (class in heat.api.openstack.v1.stacks)
StackTag (class in heat.objects.stack_tag)
StackTagList (class in heat.objects.stack_tag)
StackUpdate (class in heat.engine.update)
StackWatch (class in heat.engine.service_stack_watch)
start() (heat.engine.scheduler.TaskRunner method)
(heat.engine.service.EngineListener method)
(heat.engine.service.EngineService method)
(heat.engine.service.ThreadGroupManager method)
(heat.engine.worker.WorkerService method)
start_watch_task() (heat.engine.service_stack_watch.StackWatch method)
start_with_acquired_lock() (heat.engine.service.ThreadGroupManager method)
start_with_lock() (heat.engine.service.ThreadGroupManager method)
started() (heat.engine.scheduler.TaskRunner method)

(heat.engine.clients.os.barbican.BarbicanClientPlugin attribute)
(heat.engine.clients.os.ceilometer.CeilometerClientPlugin attribute)
(heat.engine.clients.os.cinder.CinderClientPlugin attribute)
(heat.engine.clients.os.designate.DesignateClientPlugin attribute)
(heat.engine.clients.os.glance.GlanceClientPlugin attribute)
(heat.engine.clients.os.heat_plugin.HeatClientPlugin attribute)
(heat.engine.clients.os.magnum.MagnumClientPlugin attribute)
(heat.engine.clients.os.manila.ManilaClientPlugin attribute)
(heat.engine.clients.os.mistral.MistralClientPlugin attribute)
(heat.engine.clients.os.monasca.MonascaClientPlugin attribute)
(heat.engine.clients.os.nova.NovaClientPlugin attribute)
(heat.engine.clients.os.sahara.SaharaClientPlugin attribute)
(heat.engine.clients.os.senlin.SenlinClientPlugin attribute)
(heat.engine.clients.os.swift.SwiftClientPlugin attribute)
(heat.engine.clients.os.trove.TroveClientPlugin attribute)
(heat.engine.clients.os.zaqar.ZaqarClientPlugin attribute)
ServiceController (class in heat.api.openstack.v1.services)
set()
(heat.objects.resource_data.ResourceData class method)
(heat.objects.stack_tag.StackTagList class method)
set_alarm_state()
(heat.api.cloudwatch.watch.WatchController method)
state
(heat.engine.resource.Resource attribute)
(heat.engine.stack.Stack attribute)
state_reset()
(heat.engine.resource.Resource method)
state_set()
(heat.engine.resource.Resource method)
(heat.engine.stack.Stack method)
(heat.engine.watchrule.WatchRule method)
STATUSES
(heat.engine.resource.Resource attribute)
(heat.engine.stack.Stack attribute)
steal()
(heat.objects.stack_lock.StackLock class method)
stem()
(heat.engine.dependencies.Node method)
step()
(heat.engine.scheduler.TaskRunner method)
stop()
(heat.engine.service.EngineListener method)
(heat.engine.service.EngineService method)
(heat.engine.service.ThreadGroupManager method)
(heat.engine.worker.WorkerService method)
STOP_STACK
(heat.engine.service.EngineListener attribute)
stop_stack()

set_default() (heat.engine.constraints.Schema method)
(heat.engine.parameters.Parameter method)
(heat.engine.parameters.ParsedParameter method)
(heat.engine.parameters.Schema method)
set_needed_by() (heat.engine.resource.Resource class method)
set_parent_stack() (heat.engine.stack.Stack method)
set_requires() (heat.engine.resource.Resource class method)
set_stack_and_resource_to_failed() (heat.engine.service.EngineService method)
set_stack_id() (heat.engine.hot.parameters.HOTParameters method)
(heat.engine.parameters.Parameters method)
set_stack_user_project_id() (heat.engine.stack.Stack method)
set_watch_state() (heat.engine.service.EngineService method)
(heat.engine.watchrule.WatchRule method)
(heat.rpc.client.EngineClient method)
SHARE (heat.engine.clients.os.manila.ManilaClientPlugin attribute)
SHOW (heat.engine.resource.Resource attribute)
show() (heat.api.openstack.v1.events.EventController method)
(heat.api.openstack.v1.resources.Res
(heat.engine.service.EngineListener method)
stop_timers() (heat.engine.service.ThreadGroupManager method)
store() (heat.engine.event.Event method)
(heat.engine.stack.Stack method)
(heat.engine.template.Template method)
(heat.engine.template_files.TemplateFiles method)
(heat.engine.watchrule.WatchRule method)
stored_context() (heat.engine.stack.Stack method)
str_to_num() (heat.engine.constraints.Schema static method)
strict_dependency (heat.engine.resource.Resource attribute)
STRING (heat.engine.attributes.Schema attribute)
(heat.engine.constraints.Schema attribute)
(heat.engine.hot.parameters.HOTParamSchema attribute)
(heat.engine.parameters.Schema attribute)
STRING_TYPE (heat.engine.constraints.Schema attribute)
StringParam (class in heat.engine.parameters)
StrSplit (class in heat.engine.hot.functions)
SubnetConstraint (class in heat.engine.clients.os.neutron.neutron_constraints)

ourceController method)
(heat.api.openstack.v1.software_configs.SoftwareConfigController method)
(heat.api.openstack.v1.software_deployments.SoftwareDeploymentController method)
(heat.api.openstack.v1.stacks.StackController method)
show_output()
(heat.api.openstack.v1.stacks.StackController method)
(heat.engine.service.EngineService method)
(heat.rpc.client.EngineClient method)
show_snapshot()
(heat.api.openstack.v1.stacks.StackController method)
(heat.engine.service.EngineService method)
(heat.rpc.client.EngineClient method)
show_software_config()
(heat.engine.service.EngineService method)
(heat.engine.service_software_config.SoftwareConfigService method)
(heat.rpc.client.EngineClient method)
show_software_deployment()
(heat.engine.service.EngineService method)
(heat.engine.service_software_config.SoftwareConfigService method)
(heat.rpc.client.EngineClient method)
show_stack()
(heat.engine.service.EngineService method)
(heat.rpc.client.EngineClient method)
show_watch()
(heat.engine.service.EngineService method)
(heat.rpc.client.EngineClient method)
show_watch_metric()
(heat.engine.service.EngineService method)
SubnetPoolConstraint (class in heat.engine.clients.os.neutron.neutron_constraints)
support_status
(heat.engine.resource.Resource attribute)
support_status()
(heat.engine.attributes.Attribute method)
(heat.engine.properties.Property method)
supported_versions
(heat.engine.clients.client_plugin.ClientPlugin attribute)
(heat.engine.clients.os.aodh.AodhClientPlugin attribute)
(heat.engine.clients.os.glance.GlanceClientPlugin attribute)
supports_check_action()
(heat.engine.stack.Stack method)
SupportStatus (class in heat.engine.support)
SUSPEND
(heat.api.openstack.v1.actions.ActionController attribute)
(heat.engine.resource.Resource attribute)
(heat.engine.stack.Stack attribute)
suspend()
(heat.engine.resource.Resource method)
(heat.engine.stack.Stack method)
SUSPENDED
(heat.engine.watchrule.WatchRule attribute)
swift()
(heat.engine.resource.Resource method)
SwiftClientPlugin (class in heat.engine.clients.os.swift)

method)
(heat.rpc.client.EngineClient method)
signal()
(heat.api.cfn.v1.signal.SignalController method)
(heat.api.openstack.v1.resources.ResourceController method)
(heat.engine.resource.Resource method)
signal_needs_metadata_updates
(heat.engine.resource.Resource attribute)
signal_software_deployment()
(heat.engine.service.EngineService method)
(heat.engine.service_software_config.SoftwareConfigService method)
(heat.rpc.client.EngineClient method)
SignalController (class in heat.api.cfn.v1.signal)
Snapshot (class in heat.objects.snapshot)
SNAPSHOT
(heat.engine.resource.Resource attribute)
(heat.engine.stack.Stack attribute)
snapshot()
(heat.api.openstack.v1.stacks.StackController method)
(heat.engine.resource.Resource method)
(heat.engine.stack.Stack method)
SoftwareConfig (class in heat.objects.software_config)
SoftwareConfigController (class in heat.api.openstack.v1.software_configs)
SoftwareConfigService (class in heat.engine.service_software_config)
SoftwareDeployment (class in heat.objects.software_deployment)
SoftwareDeploymentController (class in heat.api.openstack.v1.software_deploy
sync() (in module heat.engine.sync_point)
SyncPoint (class in heat.objects.sync_point)

ments)

T

tags (heat.engine.stack.Stack attribute)
(heat.objects.stack.Stack attribute)
task_description() (in module heat.engine.scheduler)
TaskRunner (class in heat.engine.scheduler)
Template (class in heat.engine.template)
template (heat.objects.raw_template.RawTemplate attribute)
Template resource
template() (heat.api.openstack.v1.stacks.InstantiationData method)
(heat.api.openstack.v1.stacks.StackController method)
TemplateFiles (class in heat.engine.template_files)
TemplateResourceInfo (class in heat.engine.environment)
tenant (heat.objects.user_creds.UserCreds attribute)
tenant_id (heat.objects.user_creds.UserCreds attribute)
TestConstraintDelay (class in heat.engine.constraint.common_constraints)
thread_lock() (heat.engine.stack_lock.StackLock method)
ThreadGroupManager (class in heat.engine.service)
time_elapsed() (heat.engine.stack.Stack method)
to_primitive() (heat.objects.fields.Json method)
to_schema_type() (heat.engine.constraints.Schema method)
toposort() (heat.engine.dependencies.Graph static method)
total_resources() (heat.engine.stack.Stack method)
translate() (heat.engine.dependencies.Dependencies method)
translate_filters() (in module heat.engine.api)
translate_properties() (heat.engine.resource.Resource method)
translate_property() (heat.engine.translation.TranslationRule method)
translation_rules() (heat.engine.resource.Resource method)
TranslationRule (class in heat.engine.translation)
trigger() (heat.engine.scheduler.TimedCancel method)
(heat.engine.scheduler.Timeout method)
trigger_hook() (heat.engine.resource.Resource method)

time_remaining() (heat.engine.stack.Stack method)
TimedCancel
Timeout
timeout_secs() (heat.engine.stack.Stack method)
Timestamp (class in heat.engine.timestamp)
TimezoneConstraint (class in heat.engine.constraint.common_constraints)
title
(heat.api.aws.exception.AlreadyExistsError attribute)
(heat.api.aws.exception.HeatAPIException attribute)
(heat.api.aws.exception.HeatAPINotImplementedError attribute)
(heat.api.aws.exception.HeatAccessDeniedError attribute)
(heat.api.aws.exception.HeatActionInProgressError attribute)
(heat.api.aws.exception.HeatIncompleteSignatureError attribute)
(heat.api.aws.exception.HeatInternalFailureError attribute)
(heat.api.aws.exception.HeatInvalidActionError attribute)
(heat.api.aws.exception.HeatInvalidClientTokenIdError attribute)
(heat.api.aws.exception.HeatInvalidParameterCombinationError attribute)
(heat.api.aws.exception.HeatInvalidParameterValueError attribute)
(heat.api.aws.exception.HeatInvalidQueryParameterError attribute)
(heat.api.aws.exception.HeatMalformedQueryStringError attribute)
(heat.api.aws.exception.HeatMissingActionError attribute)
(heat.api.aws.exception.HeatMissingAuthenticationTokenError attribute)
trove()
(heat.engine.resource.Resource method)
TroveClientPlugin (class in heat.engine.clients.os.trove)
trust_id
(heat.objects.user_creds.UserCreds attribute)
trustor_user_id
(heat.objects.user_creds.UserCreds attribute)
try_acquire()
(heat.engine.stack_lock.StackLock method)
try_thread_lock()
(heat.engine.stack_lock.StackLock method)
TYPE
(heat.engine.attributes.Schema attribute)
(heat.engine.constraints.Schema attribute)
(heat.engine.hot.parameters.HOTParamSchema attribute)
(heat.engine.parameters.Schema attribute)
(heat.engine.properties.Schema attribute)
type()
(heat.engine.properties.Property method)
(heat.engine.resource.Resource method)
TYPE_KEYS
(heat.engine.constraints.Schema attribute)
TYPES
(heat.engine.attributes.Schema attribute)
(heat.engine.constraints.Schema attribute)
(heat.engine.hot.parameters.

(heat.api.aws.exception.HeatMissingParameterError attribute)
(heat.api.aws.exception.HeatOptInRequiredError attribute)
(heat.api.aws.exception.HeatRequestExpiredError attribute)
(heat.api.aws.exception.HeatRequestLimitExceeded attribute)
(heat.api.aws.exception.HeatServiceUnavailableError attribute)
(heat.api.aws.exception.HeatSignatureError attribute)
(heat.api.aws.exception.HeatThrottlingError attribute)
to_dict()
(heat.engine.support.SupportStatus method)

U

UPDATE
(heat.engine.resource.Resource attribute)
(heat.engine.stack.Stack attribute)
update()
(heat.api.cfn.v1.stacks.StackController method)
(heat.api.openstack.v1.software_deployments.SoftwareDeploymentController method)
(heat.api.openstack.v1.stacks.StackController method)
(heat.engine.resource.Resource method)
(heat.engine.stack.Stack method)
(heat.engine.template_files.TemplateFiles method)
(heat.objects.snapshot.Snapshot class method)
UPDATE_ALLOWED
HOTParamSchema attribute)
(heat.engine.parameters.Schema attribute)
update_software_deployment()
(heat.engine.service.EngineService method)
(heat.engine.service_software_config.SoftwareConfigService method)
(heat.rpc.client.EngineClient method)
UPDATE_STACK
(heat.api.cfn.v1.stacks.StackController attribute)
update_stack()
(heat.engine.service.EngineService method)
(heat.rpc.client.EngineClient method)
update_task()
(heat.engine.stack.Stack method)
update_template_diff()
(heat.engine.resource.Resource

(heat.engine.properties.Schema attribute)
update_allowed() (heat.engine.properties.Property method)
update_allowed_properties (heat.engine.resource.Resource attribute)
update_and_save() (heat.objects.resource.Resource method)
(heat.objects.stack.Stack method)
update_by_id() (heat.objects.raw_template.RawTemplate class method)
(heat.objects.resource.Resource class method)
(heat.objects.service.Service class method)
(heat.objects.software_deployment.SoftwareDeployment class method)
(heat.objects.stack.Stack class method)
(heat.objects.watch_rule.WatchRule class method)
update_convergence() (heat.engine.resource.Resource method)
update_input_data() (heat.objects.sync_point.SyncPoint class method)
(in module heat.engine.sync_point)
update_metadata() (heat.objects.resource.Resource method)
update_patch() (heat.api.openstack.v1.stacks.StackController method)
UPDATE_POLICY (heat.engine.cfn.functions.ResourceFacade attribute)
(heat.engine.hot.functions.Resource method)
update_template_diff_properties() (heat.engine.resource.Resource method)
update_waitcondition() (heat.api.cfn.v1.signal.SignalController method)
updated_at (heat.engine.watchrule.WatchRule attribute)
(heat.objects.user_creds.UserCreds attribute)
url_for() (heat.engine.clients.client_plugin.ClientPlugin method)
(heat.engine.clients.os.keystone.heat_keystoneclient.KsClientWrapper method)
User data
user_env_as_dict() (heat.engine.environment.Environment method)
UserCreds (class in heat.objects.user_creds)
username (heat.objects.user_creds.UserCreds attribute)

Facade attribute)
update_policy_schema (heat.engine.resource.Resource attribute)

V

V1 (heat.engine.clients.os.glance.GlanceClientPlugin attribute)
V2 (heat.engine.clients.os.aodh.AodhClientPlugin attribute)
(heat.engine.clients.os.glance.GlanceClientPlugin attribute)
valid_hook_type() (in module heat.engine.environment)
valid_restricted_actions() (in module heat.engine.environment)
valid_types (heat.engine.constraints.AllowedPattern attribute)
(heat.engine.constraints.AllowedValues attribute)
(heat.engine.constraints.CustomConstraint attribute)
(heat.engine.constraints.Length attribute)
(heat.engine.constraints.Range attribute)
validate() (heat.engine.cfn.functions.GetAtt method)
(heat.engine.constraint.common_constraints.CIDRConstraint method)
(heat.engine.constraint.common_constraints.CRONExpressionConstraint method)
(heat.engine.constraint.common_constraints.TestConstraintDelay method)
(heat.engine.constraints.BaseCustomConstraint method)
validate_with_client() (heat.engine.clients.os.keystone.keystone_constraints.KeystoneBaseConstraint method)
(heat.engine.clients.os.neutron.neutron_constraints.NetworkConstraint method)
(heat.engine.clients.os.neutron.neutron_constraints.NeutronConstraint method)
(heat.engine.clients.os.neutron.neutron_constraints.ProviderConstraint method)
(heat.engine.clients.os.nova.KeypairConstraint method)
(heat.engine.clients.os.senlin.ClusterConstraint method)
(heat.engine.clients.os.senlin.PolicyTypeConstraint method)
(heat.engine.clients.os.senlin.ProfileConstraint method)
(heat.engine.clients.os.senlin.ProfileTypeConstraint method)
value() (heat.engine.parameters.BooleanParam method)
(heat.engine.parameters.CommaDelimitedListParam method)

constraints.DNSDomainConstraint method)
(heat.engine.constraint.common_constraints.DNSNameConstraint method)
(heat.engine.constraint.common_constraints.ExpirationConstraint method)
(heat.engine.constraint.common_constraints.IPConstraint method)
(heat.engine.constraint.common_constraints.ISO8601Constraint method)
(heat.engine.constraint.common_constraints.MACConstraint method)
(heat.engine.constraint.common_constraints.RelativeDNSNameConstraint method)
(heat.engine.constraint.common_constraints.TimezoneConstraint method)
(heat.engine.constraint.heat_constraints.ResourceTypeConstraint method)
(heat.engine.constraints.BaseCustomConstraint method)
(heat.engine.constraints.Constraint method)
(heat.engine.constraints.Schema method)
(heat.engine.function.Function method)
(heat.engine.parameter_groups.ParameterGroups method)
(heat.engine.parameters.Parameter method)
(heat.engine.parameters.Parameters method)
(heat.engine.properties.Properties method)
(heat.engine.properties.Schema method)
(heat.engine.parameters.JsonParam method)
(heat.engine.parameters.NumberParam method)
(heat.engine.parameters.Parameter method)
(heat.engine.parameters.StringParam method)
verify_resize() (heat.engine.clients.os.nova.NovaClientPlugin method)
VERSION (heat.engine.cfn.template.CfnTemplateBase attribute)
(heat.engine.clients.os.monasca.MonascaClientPlugin attribute)
(heat.engine.clients.os.senlin.SenlinClientPlugin attribute)
(heat.engine.hot.template.HOTemplate20130523 attribute)
(heat.objects.base.HeatObject attribute)
(heat.objects.raw_template.RawTemplate attribute)
(heat.objects.raw_template_files.RawTemplateFiles attribute)
VersionNegotiationFilter (class in heat.api.middleware.version_negotiation)
VOLUME_V2 (heat.engine.clients.os.cinder.CinderClientPlugin attribute)
VOLUME_V3 (heat.engine.clients.os.cinder.CinderClientPlugin attribute)
VolumeAttachProgress (class in heat.engine.clients.progress)
VolumeBackupConstraint (class in heat.engine.clients.os.cinder)
VolumeBackupRestoreProgress (class in heat.engine.clients.progress)
VolumeConstraint (class in

method)
(heat.engine.resource.Resource method)
(heat.engine.stack.Stack method)
(heat.engine.support.SupportStatus method)
(heat.engine.template.Template method)
(heat.engine.translation.TranslationRule method)
(in module heat.engine.function)
validate_args() (heat.engine.hot.functions.Repeat method)
(heat.engine.hot.functions.Yaql method)
validate_constraints() (heat.engine.constraints.Schema method)
validate_datastore() (heat.engine.clients.os.trove.TroveClientPlugin method)
validate_deletion_policy() (heat.engine.resource.Resource class method)
validate_expression() (heat.engine.hot.functions.Yaql method)
validate_hadoop_version() (heat.engine.clients.os.sahara.SaharaClientPlugin method)
validate_resource_definitions() (heat.engine.cfn.template.CfnTemplateBase method)
(heat.engine.hot.template.HOTemplate20130523 method)
(heat.engine.template.Template method)
validate_resource_key_type() (heat.engine.template.Template class method)
validate_section() heat.engine.clients.os.cinder)
VolumeDeleteProgress (class in heat.engine.clients.progress)
VolumeDetachProgress (class in heat.engine.clients.progress)
VolumeResizeProgress (class in heat.engine.clients.progress)
VolumeSnapshotConstraint (class in heat.engine.clients.os.cinder)
VolumeTypeConstraint (class in heat.engine.clients.os.cinder)
VolumeUpdateAccessModeProgress (class in heat.engine.clients.progress)

(heat.engine.template.Template method)
validate_template() (heat.api.cfn.v1.stacks.StackController method)
(heat.api.openstack.v1.stacks.StackController method)
(heat.engine.resource.Resource method)
(heat.engine.service.EngineService method)
(heat.rpc.client.EngineClient method)
validate_usage() (heat.engine.hot.functions.Digest method)
validate_value() (heat.engine.parameters.Schema method)

W

Wait condition
wait() (heat.engine.service.EngineListener method)
(heat.engine.service.EngineService method)
(heat.engine.worker.WorkerService method)
WATCH_STATES (heat.engine.watchrule.WatchRule attribute)
WatchController (class in heat.api.cloudwatch.watch)
WatchData (class in heat.objects.watch_data)
WatchRule (class in heat.engine.watchrule)
worker_client (heat.engine.stack.Stack attribute)
WorkerClient (class in heat.rpc.worker_client)
WorkerService (class in heat.engine.worker)
WORKFLOW_V2 (heat.engine.clients.os.mistral.MistralClientPlugin attribute)
WorkflowConstraint (class in heat.engine.clients.os.mistral)
wrappertask() (in module heat.engine.scheduler)

(class in heat.objects.watch_rule)

Y

Yaql (class in heat.engine.hot.functions)

Z

ZaqarClientPlugin (class in heat.engine.clients.os.zaqar)

ZaqarEventSink (class in heat.engine.clients.os.zaqar)

Python Module Index

h

h

heat

heat.api.aws.ec2token

heat.api.aws.exception

heat.api.aws.utils

heat.api.cfn.v1.signal

heat.api.cfn.v1.stacks

heat.api.cfn.versions

heat.api.cloudwatch.watch

heat.api.middleware.fault

heat.api.middleware.ssl

heat.api.middleware.version_negotiation

heat.api.openstack.v1.actions

heat.api.openstack.v1.build_info

heat.api.openstack.v1.events

heat.api.openstack.v1.resources

heat.api.openstack.v1.services

heat.api.openstack.v1.software_configs

heat.api.openstack.v1.software_deployments

heat.api.openstack.v1.stacks

heat.api.openstack.v1.util

heat.api.openstack.v1.views.stacks_view

heat.api.openstack.v1.views.views_common

heat.api.openstack.versions

heat.api.versions

heat.engine.api

heat.engine.attributes

heat.engine.cfn.functions

heat.engine.cfn.template

heat.engine.check_resource

heat.engine.clients.client_plugin

heat.engine.clients.os.aodh

heat.engine.clients.os.barbican

heat.engine.clients.os.ceilometer

heat.engine.clients.os.cinder

heat.engine.clients.os.designate

heat.engine.clients.os.glance

heat.engine.clients.os.heat_plugin

heat.engine.clients.os.keystone.heat_keystoneclient

heat.engine.clients.os.keystone.keystone_constraints

heat.engine.clients.os.magnum

heat.engine.clients.os.manila

heat.engine.clients.os.mistral

heat.engine.clients.os.monasca

heat.engine.clients.os.neutron.lbaas_constraints

heat.engine.clients.os.neutron.neutron_constraints

heat.engine.clients.os.nova

heat.engine.clients.os.sahara

heat.engine.clients.os.senlin

heat.engine.clients.os.swift

heat.engine.clients.os.trove

heat.engine.clients.os.zaqar

heat.engine.clients.progress

heat.engine.constraint.common_constraints

heat.engine.constraint.heat_constraints

heat.engine.constraints

heat.engine.dependencies

heat.engine.environment

heat.engine.event

heat.engine.function

heat.engine.hot.functions

heat.engine.hot.parameters

heat.engine.hot.template

heat.engine.lifecycle_plugin

heat.engine.notification.autoscaling

heat.engine.notification.stack

heat.engine.parameter_groups

heat.engine.parameters

heat.engine.plugin_manager

heat.engine.properties

heat.engine.resource

heat.engine.rsrc_defn

heat.engine.scheduler

heat.engine.service

heat.engine.service_software_config

heat.engine.service_stack_watch

heat.engine.stack

heat.engine.stack_lock

heat.engine.support

heat.engine.sync_point

heat.engine.template

heat.engine.template_files

heat.engine.timestamp

heat.engine.translation

heat.engine.update

heat.engine.watchrule

heat.engine.worker

heat.hacking.checks

heat.httpd.heat_api

heat.httpd.heat_api_cfn

heat.httpd.heat_api_cloudwatch

heat.objects.base

heat.objects.event

heat.objects.fields

heat.objects.raw_template

heat.objects.raw_template_files

heat.objects.resource

heat.objects.resource_data

heat.objects.service

heat.objects.snapshot

heat.objects.software_config

heat.objects.software_deployment

heat.objects.stack

heat.objects.stack_lock

heat.objects.stack_tag

heat.objects.sync_point

heat.objects.user_creds

heat.objects.watch_data

heat.objects.watch_rule

heat.rpc.api

heat.rpc.client

heat.rpc.listener_client

heat.rpc.worker_api

heat.rpc.worker_client

heat.scaling.cooldown

heat.scaling.lbutils

heat.scaling.rolling_update

heat.scaling.scalingutil

heat.scaling.template

heat.version