ERICSSON

OpenStack Block Storage API in CEE

Cloud Execution Environment

INTERWORK DESCRIPTION

1/155 19-CRA 119 1863/5 Uen B

Contents

1 Introduction

This document is an introduction to the use of the OpenStack Application Programming Interface (API) component ‘‘block storage’’ in the Cloud Execution Environment (CEE).

While the aim of the document is primarily to present the block storage API in CEE, it also contains descriptive information about CEE.

In this document, the term Logical Unit Number (LUN) is used both for the logical unit number and for the logical unit.

1.1 API Versions

OpenStack Block Storage API v3 is supported and recommended. Block Storage API v3 is functionally identical to Block Storage API v2.

OpenStack Block Storage API v2 is supported.

OpenStack Block Storage API v1 is supported, but deprecated.

1.2 Document References

This section lists the official OpenStack API references.

1.2.1 API Design Base Reference

For the description of the API operations and extensions of block storage, refer to the section ‘‘Block Storage API v2’’ in the *OpenStack API Complete Reference*.

Note: The date on the front page shows the date on which the PDF document was generated. The date of the latest change of the actual content can be different.

This is a stored copy of the OpenStack API Reference document version that was the base for the development of this CEE version. All references to the OpenStack API made in this document are based on this specific document version. The document can include API extensions that are not supported in this CEE version. Refer to Section 2.2 on page 2 for details.

The OpenStack Block Storage API in CEE is based on OpenStack Mitaka.

2 Supported Operations

The following sections contain the API operations and API extensions that are supported in CEE.

2.1 Basic OpenStack Operations

For the detailed description of basic block storage API operations, refer to the section ‘‘Block Storage API v2’’ in the *OpenStack API Complete Reference*.

2.1.1 Limitations

For CEE-specific limitations and recommendations, see Section 4 on page 4.

2.2 OpenStack Extensions

No OpenStack API extensions are used in CEE.

3 Ericsson Extensions

There are no Ericsson API extensions that are specific to CEE.

4 Limitations and Recommendations

This section describes CEE specific limitations and recommendations.

4.1 General Limitations

Note: If EMC² VNX Storage system is used, other limitations also apply. Refer to Section 4.2 on page 5.

If EMC² ScaleIO Storage solution is used, other limitations also apply. Refer to Section 4.3 on page 7.

4.1.1 OpenStack Block Storage Volumes Only Supported with EMC VNX or EMC ScaleIO

OpenStack Block Storage volumes are only supported when an EMC VNX or an EMC ScaleIO is present in the system. The corresponding OpenStack Block Storage services are active even if an EMC VNX or an EMC ScaleIO is not present.

4.1.2 OpenStack Block Storage Volume Migration Not Supported

The migration of OpenStack Block Storage volume between different storage back ends is not supported, as multiple storage back ends are not supported, see Section 4.1.3 on page 4.

4.1.3 Multiple Storage Back-End Configuration Not Supported

The configuration of multiple storage back ends is not possible.

4.1.4 Quota Sets Extension Do Not Validate Tenant-ID

OpenStack Block Storage quota management commands (`quota-show, quota-update, quota-defaults, quota-usage`) require a `tenant_id` as command argument. These commands can also be executed with a `tenant_name` as command argument without receiving an appropriate error message, but in this case the desired effect will not be achieved.

If a name or ID is given that does not correspond to a tenant in Keystone, the command does not show an appropriate error message. When updating the quotas for a non-existing tenant, the new quotas are stored in the OpenStack Block Storage database where the given `tenant_id` is used as `project_id`.

Refer to the section “Block Storage API v2” in the *OpenStack API Complete Reference* for further information on quota sets extension.

4.1.5 Multiple Storage Pool Configuration

By default one storage pool is configured for one Cinder back end, so that only that storage pool is used.

4.1.6 Backup of OpenStack Block Storage Volume Not Supported

Backup actions and creating a backup of an existing OpenStack Block Storage volume are not supported.

Refer to the section “Block Storage API v2” in the *OpenStack API Complete Reference* for further information on backups and backup actions.

4.2 Limitations with EMC VNX

This section list limitations that apply if EMC[2] VNX Storage system is used, in addition to the limitations listed in Section 4.1 on page 4.

4.2.1 Snapshots of OpenStack Block Storage Volumes Require EMC License

The EMC VNX Snapshot enabler (part of EMC Local Protection Suite SW license) must be installed on the EMC VNX to use the snapshot feature of OpenStack Block Storage. Refer to *VNX5400 SW Installation* for further information on how to install enablers.

Refer to the section “Block Storage API v2” in the *OpenStack API Complete Reference* for further information on snapshots.

4.2.2 Creating Volume from an Existing Volume Requires EMC License

The EMC VNX Snapshot enabler (part of EMC Local Protection Suite SW license), must be installed on the EMC VNX. An OpenStack Block Storage Volume can then be created from another volume, with, for example:

```
cinder create --source-volid <sourceVolumeId> <size>
```

This is because snapshots are used to perform the volume creation operation.

Refer to *VNX5400 SW Installation* for further information on how to install enablers.

Refer to the section “Block Storage API v2” in the *OpenStack API Complete Reference* for further information on operation `create volume`.

4.2.3 Extending Thick Volumes with Snapshots Not Supported

The EMC VNX does not support extending the size of a thick volume for which snapshots exist. Attempts to extend such a volume result in a status change of the volume, to `error_extending`.

A short summary on thick and thin provisioned volumes can be found in Section 5.2.1 on page 10.

Refer to the section ‘‘Block Storage API v2’’ in the *OpenStack API Complete Reference* for further information on volume snapshots.

4.2.4 API Request Load Limit

The number of concurrent API requests towards the OpenStack Block Storage API cannot exceed 300. This corresponds to all OpenStack Block Storage operations, such as volume creation, deletion, volume attachment, detachment, and status.

4.2.5 Caching for iSCSI Ports

The EMC VNX Cinder driver caches the iSCSI ports information. After changing the iSCSI port configurations, the administrator must restart the OpenStack Block Storage service or wait 5 minutes before performing any volume attachment. Otherwise, the attachment can fail because old iSCSI port configurations were used. The caching for iSCSI ports is not configurable.

4.2.6 Insufficient OpenStack Block Storage Error Messages

When only using thick provisioned LUNs, OpenStack Block Storage does not provide an error message. This occurs when creating a volume with a size that exceeds the available remaining storage capacity on the EMC VNX. Instead, the state of the volume is listed as `ERROR`, without further information.

Over-provisioning by using thin provisioned LUNs is possible, but it must be handled with special care. When the consumed space in the storage pool exceeds a certain threshold (70% per default), the creation of new volumes fails.

A short summary on thick and thin provisioned volumes can be found in Section 5.2.1 on page 10.

Refer to the section ‘‘Block Storage API v2’’ in the *OpenStack API Complete Reference* for further information on operation `create volume`.

4.2.7 Quality of Service Specifications Extension Not Supported

The requests `create`, `list`, `show details`, `associate`, `disassociate`, and `delete` for quality of service (QoS) specifications are not supported, these methods are not included in the EMC VNX Cinder driver.

Refer to the section "Block Storage API v2" in the *OpenStack API Complete Reference* for further information on QoS specifications.

4.2.8 Multiattach Option Not Supported

The body parameter `multiattach` for the operation `create volume` is not supported, as this method is not included in the EMC VNX Cinder driver. Refer to section "Block Storage API v2" in the *OpenStack API Complete Reference* for further information on operation `create volume`.

4.2.9 Volume Replication Not Supported

The parameter `source_replica` for the operation `create volume` is not supported, as this method is not included in the EMC VNX Cinder driver. Refer to section "Block Storage API v2" in the *OpenStack API Complete Reference* for further information on operation `create volume`.

4.3 Limitations with EMC ScaleIO

This section list limitations that apply if EMC² ScaleIO Storage solution is used, in addition to the limitations listed in Section 4.1 on page 4.

4.3.1 Supported operations

The ScaleIO Cinder driver supports the following operations:

- Create volume
- Delete volume
- Attach volume
- Detach volume
- Create snapshot
- Delete snapshot
- Create volume from snapshot
- Copy image to volume
- Copy volume to image

- Extend volume

4.3.2 Volume Limitations

ScaleIO volume size is limited to a basic granularity of 8 GiB. If volume size is not a multiple of 8 GiB, the size is rounded up. For example: a request to create a volume of 10 GiB will create a volume of 16 GiB. In OpenStack, the volume will be displayed with its requested size (10 GiB) instead of its actual size (16 GiB). The limitation also applies to the printout of the command `cinder quota-usage <tenant_id>`.

4.3.3 Snapshots

The ScaleIO storage system enables the user to take snapshots of existing volumes, up to 31 per volume. The number of volumes and snapshots combined belonging to one volume tree is limited to a maximum of **32**.

For example, the operations `create volume > snapshot volume > create volume from snapshot` use ScaleIO snapshots all the way, that is, the second volume created from the snapshot does not start a new volume tree, all volumes belong to the same, original volume tree.

4.4 Recommendations

Not applicable.

5 Additional Information

5.1 General Information

5.1.1 Quota Sets Extension for Block Storage

To prevent system capacities from being exhausted, OpenStack comes with strict operational limits. For block storage, only 10 volumes and 1000 GB per tenant are allowed by default. You must have administrator rights to apply any changes to the settings and carefully consider system capabilities before doing so. Check also section ‘‘Block Storage API v2’’ in the *OpenStack API Complete Reference* for further information on limits. To check and change these quotas, use the following commands.

Note: Quotas for tenants can also be managed with Atlas.

View all tenants:

```
openstack project list
```

List the default quotas for a tenant:

```
cinder quota-defaults TenantID
```

Show the current use of a per-tenant quota:

```
cinder quota-usage TenantID
```

Update a particular quota value:

```
cinder quota-update --QuotaName NewValue TenantID
```

For example:

```
cinder quota-update --volumes NewValue TenantID
cinder quota-usage TenantID
```

```
+-----------+--------+----------+-------+
|    Type   | In_use | Reserved | Limit |
+-----------+--------+----------+-------+
| gigabytes |   0    |    0     |  1000 |
| snapshots |   0    |    0     |   10  |
|  volumes  |   0    |    0     |   10  |
+-----------+--------+----------+-------+
```

5.2 EMC VNX-specific Information

5.2.1 Specify Extra Spec Keys

This section describes extra spec keys that are supported by the VNX Cinder Driver. Extra spec keys can be used to set the following types of volume properties:

- Provisioning type, see Section 5.2.1.1 on page 10
- Storage tiering support, see Section 5.2.1.2 on page 12
- Fully Automated Storage Tiering (FAST) cache support, see Section 5.2.1.3 on page 12

5.2.1.1 Provisioning Type

OpenStack Block Storage volumes correspond to LUNs on the EMC VNX. These LUNs can be provisioned with the following values:

thick Default option.

The volume is fully provisioned. Thick provisioned LUNs always utilize the full storage space on the EMC VNX corresponding to the size of the OpenStack Block Storage volume.

thin The volume is virtually provisioned. Thin provisioned LUNs only occupy the storage space used within the LUN.

deduplicated The volume is virtually provisioned and deduplication is enabled on it.

compressed The volume is virtually provisioned and compression is enabled on it.

By default the EMC VNX Cinder driver creates thick provisioned LUNs.

To enable OpenStack Block Storage volumes which are backed by other LUN provision types, extra specs must be defined in OpenStack Block Storage before first use.

Create Thin Provisioned LUN

Thin provisioning allows for making more efficient use of the available physical storage resources through over-provisioning. The downside of over-provisioning is that tenant I/O operations can fail if the available physical storage capacity is exceeded while virtual storage still seems to be available.

The example below shows how to create a thin provisioned volume:

```
cinder --os-username <OS-Username> --os-tenant-name
<OS-Tenant-Name> type-create "ThinVolume"

cinder --os-username <OS-Username> --os-tenant-name
<OS-Tenant-Name> type-key "ThinVolume" set storagetype:provi
sioning=thin
```

Note: Using thin provisioned LUNs requires a corresponding enabler to be installed on the EMC VNX. Refer to *VNX5400 SW Installation* for further information on how to install enablers.

After defining these extra specs, OpenStack Block Storage volumes can be created by providing the additional **--volume_type=*ThinVolume*** command option.

Create Deduplicated LUN

The system level deduplication settings can be configured at the VNX. To create a deduplicated volume, activate the VNX deduplication license on VNX first, and use key `deduplication_support=True` to let the Block Storage scheduler find a volume back end that manages a VNX with the deduplication license activated.

The example below shows how to create a deduplicated volume:

```
cinder --os-username <OS-Username> --os-tenant-name
<OS-Tenant-Name> type-create "DeduplicatedVolume"

cinder --os-username <OS-Username> --os-tenant-name
<OS-Tenant-Name> type-key "DeduplicatedVolume" set storagetyp
e:provisioning=deduplicated deduplication_support=True
```

Create Compressed LUN

The system level compression settings can be configured at the VNX. To create a compressed volume, activate the VNX compression license on VNX first, and use key `compression_support=True` to let the Block Storage scheduler find a volume back end that manages a VNX with the compression license activated.

The example below shows how to create a compressed volume:

```
cinder --os-username <OS-Username> --os-tenant-name
<OS-Tenant-Name> type-create "CompressedVolume"

cinder --os-username <OS-Username> --os-tenant-name
<OS-Tenant-Name> type-key "CompressedVolume" set storagetype:
provisioning=compressed compression_support=True
```

Note: VNX does not support snapshots created on a compressed volume. The operation fails and OpenStack shows the snapshot in `error` state.

Refer to section "Block Storage API v2" in the *OpenStack API Complete Reference* for further information on creating volume types.

5.2.1.2 Storage Tiering Support

VNX supports fully automated storage tiering which requires the FAST license activated on the VNX. Use the extra spec key `storagetype:tiering` to set the tiering policy of a volume and use the key `fast_support=True` to let the Block Storage scheduler find a volume back end that manages a VNX with the FAST license activated.

The following values are supported:

- `StartHighThenAuto` (default option)
- `Auto`
- `HighestAvailable`
- `LowestAvailable`
- `NoMovement`

The example below shows how to create a volume with tiering policy:

```
cinder --os-username <OS-Username> --os-tenant-name
<OS-Tenant-Name> type-create "AutoTieringVolume"

cinder --os-username <OS-Username> --os-tenant-name
<OS-Tenant-Name> type-key "AutoTieringVolume" set
storagetype:tiering=Auto fast_support=True

cinder --os-username <OS-Username> --os-tenant-name
<OS-Tenant-Name> type-create "ThinVolumeOnLowestAvaibleTier"

cinder --os-username <OS-Username> --os-tenant-name
<OS-Tenant-Name> type-key "CompressedVolumeOnLowestAvaibleTier"
set storagetype:provisioning=thin storagetype:tiering=Auto
fast_support=True
```

Note: Tiering policy cannot be set for a deduplicated volume. Check storage pool properties on VNX for the tiering policy of a deduplicated volume.

5.2.1.3 FAST Cache Support

VNX has a FAST cache feature which requires the FAST cache license to be activated on the VNX. Use the extra spec key `fast_cache_enabled` to choose whether to create a volume on the volume back end which

manages a pool with FAST cache enabled. The value of the extra spec key `fast_cache_enabled` is either `true` or `false`.

When creating a volume, if the key `fast_cache_enabled` is set in the volume type, the volume is created by a back end which manages a pool with FAST cache enabled.

5.2.2 Force Deletion of LUNs in Storage Groups

Owing to OpenStack time-out issues it can happen that LUNs corresponding to some available volumes remain in an EMC VNX. Users are allowed to delete the available volumes in this situation. The EMC VNX Cinder driver moves the LUN out of storage groups. The EMC VNX Cinder driver then deletes the LUN, if the user tries to delete the volume whose corresponding LUN remained in storage groups in the EMC VNX.

5.3 EMC ScaleIO-specific Information

5.3.1 Provisioning Type

The ScaleIO Cinder driver supports creation of thin provisioned volumes as well as thick provisioning. If the provisioning type is not specified, the default value of `thick` is used.