

# Deployment Guide for Cloud Execution Environment (CEE)

Virtual Multimedia Resource Function

Installation Instructions

## **Copyright**

© Ericsson AB 2016–2019. All rights reserved. No part of this document may be reproduced in any form without the written permission of the copyright owner.

## **Disclaimer**

The contents of this document are subject to revision without notice due to continued progress in methodology, design and manufacturing. Ericsson shall have no liability for any error or damage of any kind resulting from the use of this document.

## **Trademark List**

All trademarks mentioned herein are the property of their respective owners. These are shown in the document *Trademark Information*.



# Contents

<b>1</b>	<b>About This Document</b>	<b>1</b>
<b>2</b>	<b>vMRF Deployment Principles for CEE</b>	<b>2</b>
<b>3</b>	<b>vMRF Deployment Process for CEE</b>	<b>3</b>
<b>4</b>	<b>Prerequisites for vMRF Deployment</b>	<b>7</b>
4.1	Download and Extract vMRF Software Delivery Package	7
<b>5</b>	<b>vMRF Deployment Preparations for the Cloud Administrator</b>	<b>8</b>
5.1	Cloud Hardware and Software Preparation and Configuration	8
5.2	Create Flavor	12
5.3	Create Network Topology	14
<b>6</b>	<b>vMRF Deployment for the End User</b>	<b>16</b>
6.1	Set Up Command-Line Access and Security	16
6.2	Define Anti-Affinity Policy	17
6.3	Create Subnets	17
6.4	Configure Shared Storage	18
6.5	Prepare Deployment Parameters	20
6.6	Upload the vMRF Image	23
6.7	Instantiate and Check vMRF with One VM	24
6.8	Provide Initial Configuration	26
6.9	Check vMRF Status	26
6.10	Scale Out to Full VNF Size	27
	<b>Reference List</b>	<b>28</b>





# 1 About This Document

This document describes vMRF deployment on a Cloud Execution Environment (CEE) cloud service. CEE is based on OpenStack, with additional features that expand its flexibility of use and meet the needs of telecommunication service providers.

The following user roles are distinguished in this document:

**Cloud Administrator**

The cloud administrator is the cloud service provider who delivers the cloud service to the end user. The cloud administrator must fulfill certain prerequisites before the end user can start deploying vMRF.

**End User**

The end user is the vMRF operator and deployment responsible, who is assumed to be a cloud service consumer on a CEE cloud service. The end user is also referred to as a tenant.

## 2 vMRF Deployment Principles for CEE

If the hardware and software requirements are met, and after the needed configurations in CEE are done, vMRF is instantiated.

vMRF can contain one or more Virtual Network Functions (VNF), for example, one or more Virtual Multimedia Resource Functions.

A single VNF contains multiple Virtual Machines (VMs). See [Figure 1](#) for an example overview of vMRF deployment with two vMRF VNFs.

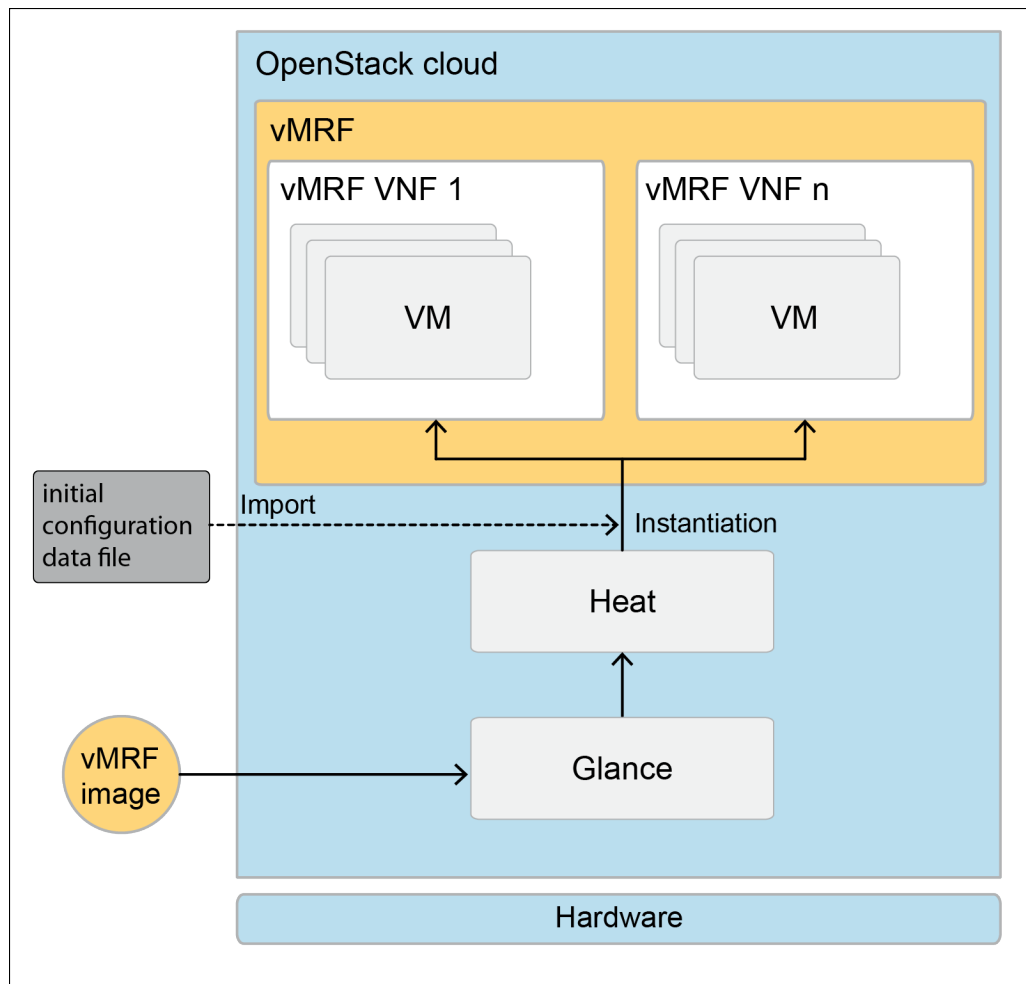


Figure 1 vMRF Deployment



## 3 vMRF Deployment Process for CEE

The vMRF deployment process consists of preparations and basic configuration of the cloud environment, and the actual instantiation of one or more VNF instances.

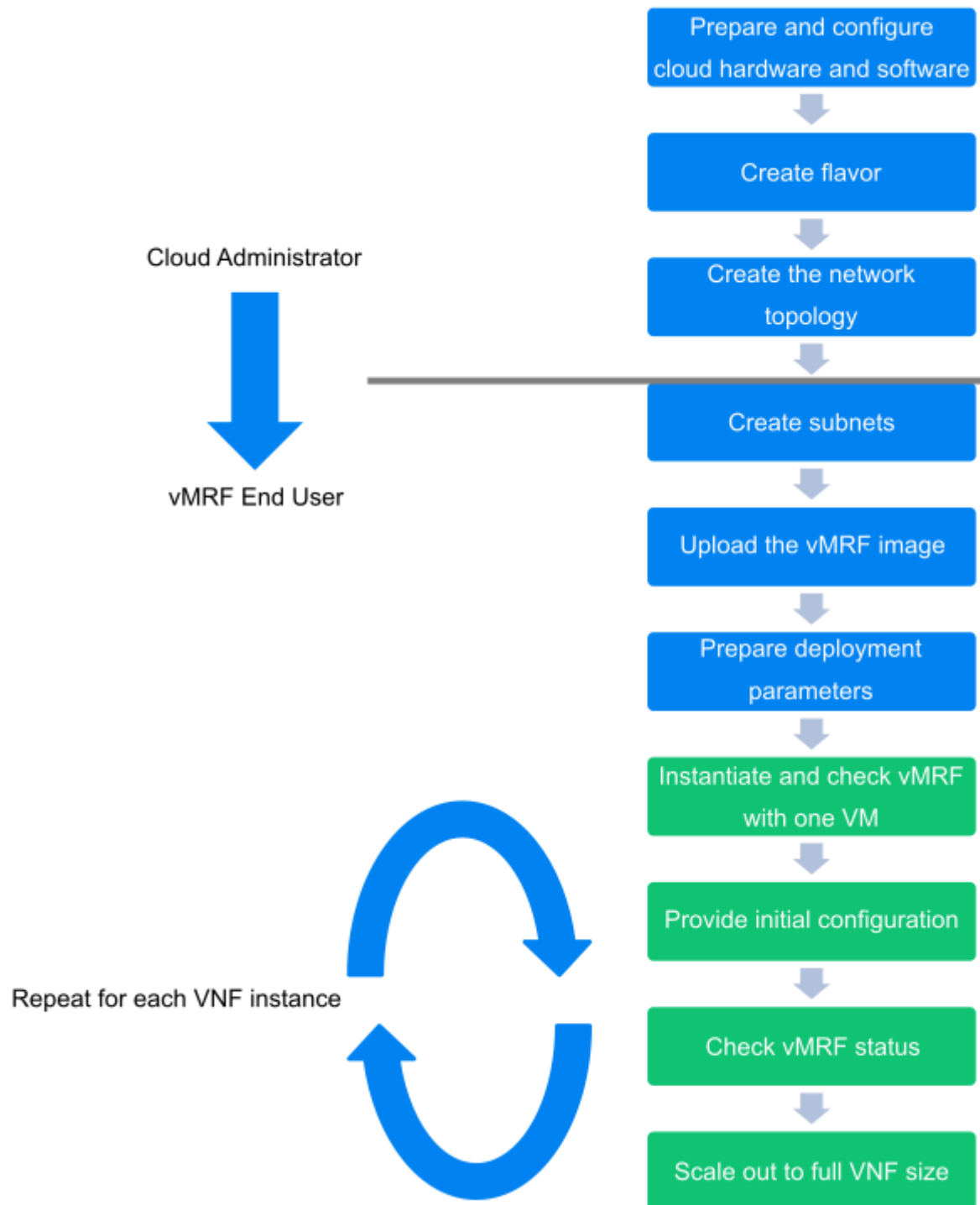


Figure 2 vMRF Deployment Process

1. Prepare the cloud environment to run vMRF





This set of steps is done by the cloud administrator.

a. Prepare and configure cloud hardware and software

This step involves checking that the necessary hardware exists, and making hardware-related configuration in CEE, in the hypervisor and in the host Operating System so that the requirements listed in [Prerequisites for vMRF Deployment](#) on page 7 are fulfilled.

b. Create a flavor

Flavors in OpenStack are virtual resource templates that define RAM size, disk size, number of CPU cores, and so on, for running VNFs. For vMRF, a flavor must be used or created that has at least the minimum requirements.

c. Create the network topology

This step involves ensuring that the required networks to which the VNF connects are in place.

2. Deploy and check vMRF

This set of steps is done by the end user.

a. Download and extract the vMRF software delivery package

The vMRF software delivery package contains the Virtual Machine Disk (VMDK) image file and the Heat Orchestration Template (HOT) files. The vMRF software delivery package must be extracted to a place where the files can be accessed from CEE.

b. Upload the vMRF image

The extracted VMDK image file must be uploaded using OpenStack Glance.

**Note:** When using Cinder block storage, the Cinder volume is created from the Glance image during VNF stack creation. The Glance image is not to be deleted during VNF lifetime, as it is used for Cinder volume creation for VMs during scaling.

c. Prepare Initial Configuration Data

Initial configuration data means the data needed for vMRF to start processing traffic. If you are importing this initial configuration data during deployment, you must prepare it so that it matches your environment. For other options, see [Initial Configuration Guide](#).

d. Prepare Deployment Parameters

The correct values of the deployment HOT template parameters must be specified in the `example_environment.yaml` file provided in the vMRF software delivery package.



e. Instantiate and Check vMRF with One VM

The vMRF instantiation is done by creating a stack in OpenStack Heat. This step is repeated for each VNF instance that needs to be created.

During instantiation, initial configuration data can also be imported into the VNF. This makes it possible for the VNF to have all necessary configuration for processing traffic right after being created.

**Note:** If initial configuration data is not imported during instantiation, it must be performed after deployment. This is also recommended when deploying the VNF for the first time. For more information, see the [Initial Configuration Guide](#).

f. Check vMRF status

It is recommended to run a status check on the newly deployed vMRF.

g. Scale Out to Full VNF Size

After the VNF is verified with one VM, you can scale out to the full size of the VNF by updating the vMRF stack.



## 4 Prerequisites for vMRF Deployment

Before the end user can deploy and use vMRF, the cloud administrator must ensure that the environment fulfills hardware, software, and network requirements. The main requirements are listed in [vMRF Infrastructure Requirements](#).

### 4.1 Download and Extract vMRF Software Delivery Package

Before the deployment, the end user must download and extract the vMRF software delivery package. Both the end user and the cloud administrator must have access to the proper example files in the package.

#### Steps

1. Download the vMRF software delivery package to a computer from which the OpenStack clients are reachable.
2. Extract the vMRF software delivery package.
3. Check that the following files exist after extracting the vMRF software delivery package:

- vMRF image (`vmrf-image-corei7-mrsv.vmdk`)
- Deployment HOT file (`vmrf.yaml` file)
- Example files, including an example environment `yaml` file

**Note:** The `yaml` files included in the `examples` folder of the software delivery package serve as examples for a possible network configuration. They can be modified to match your network environment.

If Cinder block storage is used, `vmrf.yaml` and `scaling_group.yaml` must be edited to boot from volume. These HOT files contain sections for booting from image and volume. For booting from volume, the image booting sections must be commented out, and volume booting sections must be uncommented.

For deployment with the platform automatic IP address configuration option, use the HOT and `yaml` files from the `cee` folder of the software delivery package. Using this method, IP addresses are assigned by the virtualization infrastructure.

4. Ensure that all the extracted files are in the same folder.



## 5 vMRF Deployment Preparations for the Cloud Administrator

The procedures for vMRF deployment preparation must be performed by the cloud administrator to prepare the cloud environment for running vMRF. The procedures described in this section serve as examples only to demonstrate how to fulfill the vMRF requirements.

### 5.1 Cloud Hardware and Software Preparation and Configuration

Preparation for vMRF deployment starts by checking that the necessary hardware exists, and making hardware-related configurations in the VIM, in the hypervisor, and in the host Operating System.

#### 5.1.1 Define an Availability Zone for vMRF

This procedure describes how to define an availability zone for vMRF in OpenStack. The created vMRF availability zone is input for the vMRF deployment, as the deployment file contains a corresponding parameter. The procedure uses an example scenario of creating a host aggregate with an availability zone for vMRF, called MRSv-zone, and adding two hosts to it, called node-1 and node-2. The aggregate also gets metadata so that the aggregate can be bound to the vMRF flavor as described in [Create Flavor](#) on page 12.

The procedure uses the OpenStack Nova command-line client. For information on managing host aggregates in the OpenStack dashboard, see the [OpenStack Admin User Guide](#).

#### Steps

1. Create a new host aggregate called MRSv with an availability zone called MRSv-zone using the following command:  
  
**nova aggregate-create MRSv MRSv-zone**
2. Add the nodes to be used for vMRF to the MRSv host aggregate using the following command:  
  
**nova aggregate-add-host MRSv node-1 nova aggregate-add-host MRSv node-2**
3. Using the following command, specify metadata for the MRSv host aggregate so that it can be bound to a flavor:



```
nova aggregate-set-metadata MRSv key=999
```

## 5.1.2 Configure CPU Core Isolation for the vMRF Compute Hosts

Configure core isolation so that host OS processes are excluded from those cores that are used by vMRF. This is required to ensure good and predictable performance.

**Note:** This procedure is only needed if core isolation for the compute hosts is not already configured in CEE. Proceed only if the `isolcpus` option is **not** visible in the output of the following command: `cat /proc/cmdline | grep isolcpus`.

### Steps

Do the following on each node added to the MRSv host aggregate:

1. Check the number of cores and sockets on the compute host with the following command:

```
lscpu -p
```

2. Based on the command printout, identify the first two physical CPU cores from the same Non-Uniform Memory Access (NUMA) node as the virtual switch.

As an example, consider a single-socket compute host, with a 10-core CPU. Hyperthreading is enabled, so each physical core runs two threads, giving in total 20 logical processors. Suppose that these 20 logical processors are distributed in the following way:

NUMA node	Logical processors in the <code>lscpu -p</code> printout
0	0–9, 10–19

Suppose that the virtual switch uses NUMA node 0. This means that the first two physical cores (first four logical processors) on NUMA node 0 can be identified as follows: 0, 1, 10, 11.

3. Isolate all CPUs other than the ones identified on the NUMA node. To do this, add the `isolcpus` option to the `GRUB_CMDLINE_LINUX_DEFAULT` parameter in the `/etc/default/grub` file. If the file does not exist, create it.

### Example

Example of an existing file before adding the `isolcpus` option:

```
cat /etc/default/grub
GRUB_TIMEOUT=10 GRUB_CMDLINE_LINUX_DEFAULT="nomdmonddf nomdmonisw"
```



### Example

Example of an existing file after adding the `isolcpus` option:

```
cat /etc/default/grub
GRUB_TIMEOUT=10 GRUB_CMDLINE_LINUX_DEFAULT="nomdmonddf nomdmonisw isolcpus=2-9,12-19"
```

### Example

Example of a newly created file with the `isolcpus` option:

```
cat /etc/default/grub
GRUB_CMDLINE_LINUX_DEFAULT="isolcpus=2-9,12-19"
```

4. Run the `update-grub` command.
5. Reboot the compute host.
6. Check that the `isolcpus` option is visible in the `/proc/cmdline` file:

### Example

```
cat /proc/cmdline | grep isolcpus
BOOT_IMAGE=/vmlinuz-3.13.0-79-generic root=/dev/mapper/os-root ro console=tty0 net.ifna →
mes=0 biosdevname=0 rootdelay=90 nomodeset root=UUID=201cf72b-dcb7-4721-9e63-b9f83cd022 →
fb isolcpus=2-9,12-19
```

## 5.1.3

### Configure CPU Settings for the vMRF Compute Hosts

This procedure describes how to configure CPU settings for vMRF that ensure the best possible computing environment for media stream processing.

For more information on the configuration options in the procedure, see the [OpenStack configuration overview](#).

**Note:** This procedure is only needed if CPU settings for the compute hosts are not already configured in CEE.

Do the following on each node added to the MRSv host aggregate:

#### Steps

1. Set the following in the `/etc/nova/nova.conf` file:

```
cpu_allocation_ratio = 1.0
```

**Note:** Make sure that `cpu_allocation_ratio = 1.0` is included in the `Default` section of the file.

```
cpu_mode=host-passthrough
```

2. Restart the Nova Compute service using the following command:



```
initctl restart nova-compute
```

### 5.1.4 Configure CPU Frequency Scaling for the vMRF Compute Hosts

This procedure describes how to configure the CPU frequency scaling governor on compute hosts for higher performance. It is recommended to use the setting below for about 20–30% additional capacity for transcoding.

#### Steps

1. Set the CPU frequency governor to performance on each node added to the MRSv host aggregate:

```
echo performance | tee /sys/devices/system/cpu/cpu*/cpufreq/  
scaling_governor >/dev/null
```

### 5.1.5 Configure NTP Servers

This procedure describes how to specify the external NTP servers for vMRF. External NTP servers are needed to keep the vMRF VMs synchronized, which ensures that quality of service expectations are met.

The following procedure must be performed on each node added to the MRSv host aggregate.

**Note:** This procedure is only needed if NTP servers are not already configured in CEE.

#### Prerequisites

- One or more NTP servers are available and you know their IP address or host name.

#### Steps

1. Add the following line in the `/etc/ntp.conf` file for each NTP server that vMRF uses:

```
server <NTP_server_IP_address_or_host_name>
```

Example

```
$ cat /etc/ntp.conf  
# /etc/ntp.conf, configuration for ntpd; see ntp.conf(5) for h  
elp
```

```
driftfile /var/lib/ntp/ntp.drift
```

```
# Enable this if you want statistics to be logged.
```



```
#statsdir /var/log/ntpstats/

statistics loopstats peerstats clockstats
filegen loopstats file loopstats type day enable
filegen peerstats file peerstats type day enable
filegen clockstats file clockstats type day enable

# Specify one or more NTP servers.
server 198.51.100.0
server time1.example.com
server time2.example.com
```

2. Reboot the compute host, and ensure that the ntp service is running.

### 5.1.6 Configure Project and Users

This procedure describes how to create a new project for vMRF and how to add members. The procedure uses the OpenStack dashboard.

#### Steps

1. In the **Identity** tab, create a new project and add the member users.
2. Select **Identity > Projects > Manage Members** for the new project.
3. In the **Project Members** list, add the admin user to the list, and add the admin role for the admin user.
4. Inform the end user of the name of the new project.

## 5.2 Create Flavor

This procedure describes how to create a flavor, that is, a virtual hardware template, used for vMRF VMs. The procedure describes how to use the OpenStack Nova command-line client to create a minimum size flavor for vMRF based on the minimum system requirements.

Flavors can also be created using the OpenStack dashboard, as described in the [OpenStack Admin User Guide](#).

**Note:** To deploy without hyperthreading, CEE version R.6.3.1 or higher is required.

#### Steps

1. Use the following command to create the flavor named `MRSv_flavor` with the ID `auto`, with 6144 MB of memory, 6 GB root disk space, and eight virtual CPUs:





```
nova flavor-create MRSv_flavor auto 6144 6 8
```

**Note:** In the case of deployment with hyperthreading, it is recommended to specify an even number of vCPUs in a flavor. The maximum number of vCPUs per a vMRF VM is 34.

If using Cinder block storage, specify 0 GB of root disk space in the flavor. Volume size is defined at the creation of Cinder volume in `vmrf.yaml` and `scaling_group.yaml`.

2. Use the following command to set the memory page size:

```
nova flavor-key MRSv_flavor set hw:mem_page_size=1048576
```

3. Use the following command to set the number of CPU sockets for the flavor:

```
nova flavor-key MRSv_flavor set hw:cpu_sockets=1
```

**Note:** This setting is needed to ensure that vCPUs correspond to physical CPUs of the same socket.

4. Use the following command to set CPU pinning for the flavor:

```
nova flavor-key MRSv_flavor set hw:cpu_policy=dedicated
```

5. Set CPU thread policy for the flavor:

- In the case of deployment with hyperthreading, use the following command:

```
nova flavor-key MRSv_flavor set hw:cpu_thread_policy=prefer
```

- In the case of deployment without hyperthreading, use the following command:

```
nova flavor-key MRSv_flavor set  
hw:cpu_thread_policy=isolate
```

6. Use the following command to bind the flavor to the host aggregate by adding the defined key:

```
nova flavor-key MRSv_flavor set  
aggregate_instance_extra_specs:key=999
```

7. Configure watchdog action as follows:

- In the case of deployment with watchdog action set to hard reset, use the following command:

```
nova flavor-key MRSv_flavor set hw:watchdog_action=reset
```

**Note:** The `watchdog_enable` deployment parameter must be set to 1 in the `example_environment.yaml` file.



- In the case of deployment without configured watchdog action, continue with the next step.
  - 8. Inform the personnel who are doing the vMRF instantiation of the name of the flavor.
- Note:** The reason for this step is that the name of the flavor is a parameter in the environment HOT file that is used for the instantiation.

## 5.3 Create Network Topology

The vMRF VNF instance connects to networks. The networks listed in [Table 1](#) must be created already before the VNF instance can be deployed, since the HOT template uses them as input parameters.

Table 1 vMRF Networks

Network Type
O&M
H.248 signaling towards SGCs
User plane towards media networks

### Steps

1. Using the network plan, create the required networks listed above, if they do not exist.

To create networks by launching Heat stacks, use the yaml files provided in the vMRF software delivery package. The vMRF software delivery package is available for the end user after download, in the folder where it was extracted to. To create networks by launching Heat stacks, do the following:

  - a. Change the example yaml files provided in the vMRF software delivery package to match your network environment.

**Note:** The yaml files included in the examples folder of the software delivery package serve as examples for a possible network configuration. They can be modified to match your network environment.
  - b. Log in to the CLI as the admin user for the vMRF project.

**Note:** Due to dashboard limitations, the CLI must be used in this procedure. For the movable IP address feature to be set up properly, the O&M network stack must be created as the administrator inside the vMRF project.
  - c. Create the network stacks using the following commands:



```
heat stack-create <O&M_network_stack_name> -e  
example_environment.yaml -f management_network.yaml
```

```
heat stack-create <signaling, user plane, and reserved  
network stack name> -e example_environment.yaml -f  
traffic_signaling_networks_admin.yaml
```

- d. Check that the stacks got created successfully using the following command:

```
heat stack-list
```

- e. Check that the networks got created successfully using the following command:

```
neutron net-list
```

- 2. Inform the personnel who are doing the vMRF instantiation of the names of all the networks created in the previous step.

**Note:** The reason for this step is that the names of the networks are parameters in the environment HOT file that is used for the instantiation.



## 6 vMRF Deployment for the End User

After the deployment preparations are completed by the cloud administrator, the end user can start vMRF deployment.

### 6.1 Set Up Command-Line Access and Security

Command-line access is mandatory for some of the vMRF deployment activities due to dashboard limitations.

To be able to log in to the vMRF instance after it is instantiated, the proper access and security configurations must be in place.

#### 6.1.1 Set Up the OpenStack Heat Command-Line Client

Each OpenStack component provides a command-line client, which enables access to the component API through commands. For example, the Heat component provides a heat command-line client.

##### Steps

1. Install the OpenStack Heat command-line client according to the [OpenStack End User Guide](#).
2. Set up CLI access according to the [OpenStack End User Guide](#).

#### 6.1.2 Set Up Access and Security for Instances

To be able to connect to your cloud instances using SSH, the proper access and security configurations must be in place.

##### Steps

1. Log in to the dashboard.
2. Configure the security group for the project. Since the default security group does not allow any ingress connections, either modify the default security group, or create a new one. The rules must allow ingress connections for all the vMRF protocols and ports listed in vMRF Security Management. For information on managing security groups, see the [OpenStack End User Guide](#).
3. Follow the instructions to add or import a key pair in the [OpenStack End User Guide](#).



The name of the SSH key that you added or imported is needed at VNF instantiation.

**Note:** SSH host keys for the VMs of the VNF cluster are generated automatically by the first VM during the deployment process, and copied to all other VMs.

The fingerprints of generated SSH host keys are visible on the **Stack Overview** page of the OpenStack dashboard after the deployment of vMRF.

## 6.2 Define Anti-Affinity Policy

By default, HOT files specify an anti-affinity policy for all the VMs in the VNF. The anti-affinity policy defines the placement of VMs on the compute hosts of the cloud environment. It is recommended to place only one media plane VM on the same compute host, unless dimensioning with the Ericsson dimensioning tool shows that multiple VMs can be allocated on the same host for the specific traffic profile (as described in [vMRF Infrastructure Requirements](#)).

If several VMs are allocated to the same host, `scheduler_hints` row from the `scaling_group.yaml` needs to be commented out, as seen in the following example.

```
scaled_vm:
  type: OS::Nova::Server
  properties:
    name: { get_param: scaled_vm_name }
    image: { get_param: mrf_image }
    flavor: { get_param: mrf_flavor }
    networks:
      - port: { get_resource: internal_port }
      - port: { get_resource: management_port }
      - port: { get_resource: signaling_port }
      - port: { get_resource: media_port }
    availability_zone: { get_param: availability_zone }
    config_drive: True
    user_data_format: RAW
    user_data: { get_resource: pl_init }
    metadata: { "ha-policy": "managed-on-host" }
    scheduler_hints:
#      group: { get_param: server_group }
```

## 6.3 Create Subnets

Subnets must be created within the networks created by the cloud administrator. The `example_environment.yaml` file provided in the vMRF software delivery package must match your network environment. Heat stack creation must be done in the CLI due to dashboard limitations.



## Steps

### In the CLI, do the following:

1. Change the example yaml files provided in the vMRF software delivery package to match your network environment.

**Note:** The yaml files included in the `examples` folder of the software delivery package serve as examples for a possible network configuration. They can be modified to match your network environment.

2. Create the subnets as a Heat stack using the following command:

```
heat stack-create <subnet_stack_name> -e
example_environment.yaml -f traffic_signaling_subnets.yaml
```

3. Check that the stack got created successfully using the following command:

```
heat stack-list
```

4. Check that the subnets got created successfully using the following command:

```
neutron subnet-list
```

As an alternative, log in to the dashboard and check that the subnets are visible in the **Network Topology** view.

## 6.4 Configure Shared Storage

**Note:** This procedure is optional. Perform the steps only if shared storage is used.

The external shared storage allows for the storage of the following files from each VM on a remote server:

- Log files from the `/var/log` directory, including journal log files
- Crash dump files
- Configuration backup files created by the automatic backup and restore function

vMRF connects to the server with SSHFS, mounts a specified directory path, and creates a subfolder for the cluster, and subfolders for the files for each VM in the cluster. This ensures that logs and other shared files of different VMs and VNF instances do not get mixed up.

For authentication, an SSH key pair has to be created. This key pair can be cluster-specific, or common for all clusters.



The following parameters must be prepared and included in the `example_environment.yaml` file parameter list or the vApp property list during deployment:

- Storage server username
- Storage server address (IP address and port number)
- Storage server path
- Storage server SSH private key and fingerprint

### Prerequisites

- The private and public SSH keys are generated.

### Steps

1. Generate the private SSH key parameter value by replacing the end of line characters with the string `"\n"` and including the key data string between single quotation mark (') characters using the following command:

```
echo "'$(awk 'BEGIN{ORS="\n"} {print $0}' .ssh/id_rsa)'"
```

**Note:** The resulting string must be added in OpenStack-based deployments as the `shared_storage_ssh_private_key` value in the `example_environment.yaml` file or, in VMware-based deployments as vApp property.

2. Generate the SSH fingerprint parameter value from the public SSH key by replacing the end of line characters with the string `"\n"` using the following command:

```
echo "'$(ssh-keyscan -p <server_port> <server_host> |awk 'BEGIN{ORS="\n"} {print $0}')"'
```

**Note:** The resulting string must be added in OpenStack-based deployments as the `shared_storage_server_fingerprint_key` value in the `example_environment.yaml` file or, in VMware-based deployments as vApp property.

3. On the shared storage server, append the public key to the authorized keys file using the following command:

```
cat .ssh/<public_key_file_name> >> .ssh/<authorized_keys_file_name>
```

— RELATED INFORMATION —

[6.5 Prepare Deployment Parameters on page 20](#)



## 6.5 Prepare Deployment Parameters

1. Ensure that the correct values of the following deployment HOT template parameters are specified in the `example_environment.yaml` file provided in the vMRF software delivery package:

HOT Parameter Name	Description
admin_username	The user name, password, and SSH key of the emergency user. <b>The default password must be changed by supplying a new password hash in this parameter.</b> <sup>(1)</sup>
admin_password_hash	
admin_authorized_key	
mrf_flavor	Virtual hardware template used by VMs of the cluster.
vnf_name	Name of the VNF instance
external_net_name	Internal and external networks used by the VNF.
management_net_name	
signaling_net_name	
media_net_name	
mrf_image	The VNF image file name.
payload_instance_count	The number of VMs to be created.
mrf_config	<p>If you are importing initial configuration data during deployment, you must give a value to this parameter. The parameter contains the content of the <code>tar.gz</code> file that contains the vMRF configuration that will be imported during system startup, in Base64 encoding. The input can be generated using the following command:</p> <pre><b>base64 -w 0 &lt;file_path&gt;</b></pre> <p>If no value is provided, initial configuration must be provided after instantiation either manually or from an exported configuration file.</p>
shared_storage_server_username <sup>(2)</sup>	User name for the external shared storage server.
shared_storage_ssh_private_key <sup>(2)</sup>	The SSH private key for external shared storage in one line. The input can be generated by replacing the end of line characters with the string <code>"\n"</code> and including the key data string between single quotation mark (') characters using the following command: <code>echo "\$(<code>awk 'BEGIN{ORS="\n"} {print \$0}' .ssh/id_rsa</code>)"</code> .





HOT Parameter Name	Description
shared_storage_server_path <sup>(2)</sup>	The external shared storage POSIX path pointing to a directory on the shared storage server.
shared_storage_server_ip <sup>(2)</sup>	The shared storage server IP address.
shared_storage_server_port <sup>(2)</sup>	The shared storage server port.
shared_storage_server_fingerprint <sup>(2)</sup>	<p>The shared storage server fingerprint in one line. The fingerprint can be generated by replacing the end of line characters with the string "\n" using the following command:</p> <pre>echo "'\$(ssh-keyscan -p &lt;server_port&gt; &lt;server_host&gt;  awk 'BEGIN{ORS="\n"} {print \$0}' )'"</pre> <p>The input can be generated by replacing the end of line characters with the string "\n".</p>
announcement_storage_server_username <sup>(3)</sup>	User name for the external announcement storage server. The default value is announcements.
announcement_storage_ssh_private_key <sup>(3)</sup>	<p>The SSH private key for external announcement storage in one line. The input can be generated by replacing the end of line characters with the string "\n" and including the key data string between single quotation mark (') characters using the following command: <code>echo "'\$(awk 'BEGIN{ORS="\n"} {print \$0}' .ssh/id_rsa)'"</code>.</p>
announcement_storage_server_path <sup>(3)</sup>	The external announcement storage POSIX path pointing to a directory on the announcement storage server. The default path is announcement_storage.
announcement_storage_server_ip <sup>(3)</sup>	The announcement storage server IP address.
announcement_storage_server_port <sup>(3)</sup>	The announcement storage server port. The default port number is 22.
announcement_storage_server_fingerprint <sup>(3)</sup>	<p>The announcement storage server fingerprint in one line. The fingerprint can be created using the following command:</p> <pre>ssh-keyscan -p &lt;server_port&gt; &lt;server_host&gt;</pre> <p>The input can be generated by replacing the end of line characters with the string "\n".</p>



HOT Parameter Name	Description
pm_data_monitoring_hosts_ip_address	Optional parameter. IP address of PM data monitoring host.
pm_data_monitoring_hosts_ip_address	Optional parameter. Listening port of PM data monitoring on host.
availability_zone	Availability zone to use for vMRF instances. Default is the OpenStack 'nova' zone which consists of all compute nodes.
watchdog_interval	Watchdog ping interval in seconds. Min.: 1, Max.: 58. Default: 1.
watchdog_enable	Parameter for enabling watchdog. Min.: 0 (Watchdog feature disabled), Max.: 1 (Watchdog feature enabled). Default: 0.
ntp_server_1	Parameter for configuring NTP server IP address on the guest OS side VNF. <sup>(4)</sup>
ntp_server_2	

- (1) Password change for the emergency user is supported only during deployment.
- (2) This parameter needs to be defined only if the optional shared storage or the optional announcement storage is used. Leave it blank otherwise. For more information on the shared storage feature, see [vMRF Overview](#).
- (3) This parameter needs to be defined only if the optional announcement storage server is used. For local storage of announcement files, leave it blank. In this case the vMRF VMs will start up configured for local storage of announcement files. For more information on the shared storage feature, see [vMRF Overview](#).
- (4) NtpServer MO is not used by vMRF.

2. Add any missing deployment parameters and their default values to the `example_environment.yaml` file.

To generate a new password hash for the parameter `admin_password_hash`, run the following command: `mkpasswd -m sha-512 <new_emergency_user_password>`.

The following is an example of the additional parameters that can be appended to the file. Comments are included for guidance.

### Example

```
#Examples for additional deployment parameters, change values according to your environment
mrs_v_image: "MRSv1_1"
#Include the following line only if the emergency user is not "mrsv-admin"
admin_username: "mymrsv-admin"
#Replace the password hash on the following line with the actual password hash for the emergency user
admin_password_hash: "$6$asYHS3nJ8$6dwcdDEoTM029Ff0e4Ejxa4HKKS0LkzhAdGPLLoQ..."
admin_authorized_key: "mykey"
#Replace the value on the following line with the base64 command output on the path to the tar.gz file
mrs_v_config: "H4sICcipGFcC/2JhY2t1cF8x0DA0MTZfNC50YXIA7V1bc+I4Fs5zV+1/oPp1F..."
shared_storage_server_username: "myuser"
shared_storage_ssh_private_key: '-----BEGIN RSA PRIVATE KEY-----
\nMIICXQIBAAKBgQDF60kPVTtczzKVAKQuQw+/zWYgJgMr7GuU75/YR+6NhHVVs9\nDhLQuW9yB53intRX8DHiG+1Zk20ng51GR→
Gw1SmIT8uqJA2xYtbYrcFPvBVyW0BN3\nKyCNeBZogcN6YDb442w/2H/NvA5EfSUNrhTdoUhPY7XDHvs8oRAjySZEJQIDAQAB\nAo→
GBAMT2Narwozbla+
2MRV7fCPTYfbnan3Zgkm5uILlBmbvaezzX56UHEGpvYB0W\nnv2T7s04gV0Qxjq8KLCKasDphdVydHnroLC/dW85mQzdS3rvoXTv0p→
xD9ys1XL9Ji\nFXUMm+vY0ireglqS0TRCmsFTETVsm7jYst72bdKhpwFsgTYxAkEA5gTs23J+q20+\nSd0erxGks7vyoJoZRVLdXs→'
```



```
NfIu012qcGpAe3G2
EPx36RNWIK1TZgBtJ0McyXK9gF\nBEfmX+VBmwJBANxGLLNvVxEn/Lmf0ITRoHjF8gG1xxLFZr5no7uCVftMTkxGa+DR\nlaGdiUg>
V00pwJ0cnYN5+fljphWr00K2zT8CQQDjLD4ZxgygljYmKvX6nzxBXwCT\n6V3H/70umFpfh0IK/ycp3YzUd5oz9ybGzuDsyUwJ6>
4meN9cp+7ceX5ne2
69AkAa\nncQi+0n1Ac7w4xXAjTbCKrtEZbKzWvT6Ru2q56naDrNbVhsM9GY/PBjtg+2fDgP3\nhn0XDv0hWUA1WDjT0+DjAkBu+yu>
5aEoq2idarCXI3UVK8PPLAfczXKwjJL2N3zMB\n6j6U14mJzY/wT7IC/qzyMbMcY5h9I6w6IqBM9hBon6U\n
-----END RSA PRIVATE KEY-----\n'
shared_storage_server_path: "/home/myuser/mrsv"
shared_storage_server_ip: "192.0.2.7"
shared_storage_server_port: "22"
shared_storage_server_fingerprint: "12:f8:7e:78:61:b4:bf:e2:de:24:15:96:4e:d4:72:53"
announcement_storage_server_username: "announcement_user"
announcement_storage_ssh_private_key: "announcementStoragePrivKey"
announcement_storage_server_path: "/home/myuser/announcement_storage"
announcement_storage_server_ip: "192.0.2.9"
announcement_storage_server_port: "22"
announcement_storage_server_fingerprint: '131.160.94.101 ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzZDHAyNTYAAAIbmlzdHAyNTYAAABBBKFUShoQyxnvofm2fv0g0q8/MQ37wKdT7zuQPNfV16p3Fv0B+vRDRNGUMV/erDZPvZgTQDb>
iBpg9fB1LfeFmH8Y=\n13=>
1.160.94.101 ssh-ed25519 AAAAC3NzaC11ZDI1NTE5AAAAIHVgKowKydpJ3zs990CFW/gfDXG5173Zgk+yecHAjn0FU\n'
pm_data_monitoring_hosts_ip_address: "192.0.2.8"
pm_data_monitoring_hosts_port: "8000"
#Include the following line only if the availability zone is not "nova"
availability_zone: "mymrsvzone"
```

If importing initial configuration data during deployment:

3. Ensure that the initial configuration data matches your environment.

## 6.6 Upload the vMRF Image

This procedure describes how to upload the vMRF image to the CEE cloud, using the vMRF image file.

### Steps

1. Create the vMRF image.

In the CLI, use the following command:

```
glance image-create --name vMRF-<product_revision> --
visibility=public --disk-format vmdk --container-format bare
--file vmrf-image-corei7-mrsv.vmdk
```

In the dashboard, follow the [OpenStack End User Guide](#), and fill the image properties according to [Table 2](#).

Table 2 vMRF Image Properties for OpenStack Dashboard

Parameter	Description
Name	Enter a name for the image. It is recommended to include the vMRF product version in the name.
Description	Enter a brief description of the image.
Image Source	Choose <b>Image File</b> .
Image File or Image Location	Browse for the vMRF image file on your file system and add it.



Parameter	Description
Format	Choose <b>VMDK</b> .
Architecture	Enter <b>x86_64</b> .
Minimum Disk (GB) and Minimum RAM (MB)	Leave these fields empty.
Public	Select this check box, so that the image is public to all users with access to the current project.
Protected	Do not select this check box.

2. Check that the image exists using the following command:

```
glance image-list --name vMRF-<product_revision>
```

## 6.7 Instantiate and Check vMRF with One VM

This procedure describes how to instantiate a vMRF VNF instance in the OpenStack cloud, using the OpenStack Glance image and the HOT files. Instantiate the VNF with one VM and verify that VM before scaling out to the full size of the VNF.

The VNF instance must be created in the CLI due to dashboard limitations.

### Prerequisites

- The vMRF VMs to be scaled-out are configured as MRFP (Media Resource Function Processor) nodes in the MTAS.

A vMRF VM identifies itself to the MTAS with a Message Id (MId) that contains the vMRF VM signaling IP address and SCTP port. Typically, the whole range of signaling IP addresses (the signaling subnet) configured in the OpenStack for the new vMRF VNF, is configured as MRFP nodes in the MTAS.

For more information on adding an MRFP node in MTAS, see section Add MRFP in *MTAS Media Control Management Guide*.

### Steps

1. Log in to the OpenStack CLI.
2. Instantiate vMRF by creating the vMRF stack in Heat using the `heat stack-create` command. The following is an example of the `heat stack-create` command when all parameters have been added to the `example_environment.yaml` file:



```
heat stack-create <stack_name> -f vmrf.yaml -e
example_environment.yaml -P payload_instance_count=1
```

**Note:** Ensure that you do not use dots (".") in the stack name. The stack name must not be longer than 52 characters.

When using Cinder block storage, the Cinder volume is created from the Glance image during VNF stack creation. The Glance image is not to be deleted during VNF lifetime, as it is used for Cinder volume creation for VMs during scaling.

3. Check that the stack exists using the following command:

```
heat stack-show <stack_name>
```

As an alternative, log in to the dashboard and check that the VM belonging to the VNF is visible in the **Network Topology** view.

4. In the OpenStack dashboard **Stack Overview** page, check the outputs section for the vMRF VNF O&M IP address.
5. Open an SSH connection to the O&M IP address of the vMRF VNF instance using the following command:

```
ssh -A -i <admin_username>@<O&M_IP_address>
```

6. Run the following command:

```
verify_vmrf_cluster_status.py
```

7. Check that all components are in the correct state. The following is an example output from a VNF that is operating correctly:

### Example

```
$ verify_vmrf_node_status.py
Running command: "verify_vmrf_node_status.py" on host: 192.168.0.3 (fi2-vmrf)
eth0: OK
eth1: OK
eth2: OK
SC role: ACTIVE
RED role: ACTIVE
CoreMW: OK
COM: OK, RUNNING
MrfDirector: OK, RUNNING
CliDaemon: OK
IpPipeline: OK
TC-MPD: OK
MrfAgent: OK
CloudInit: OK
ConfigImport: OK
SEC-CERT: OK
neighbourdetection: OK
LM: OK
```

**Note:** The MrfDirector and COM components are in the OK state only for the VM whose SC role is ACTIVE, that is, the active System Controller (SC) VM. In all other VMs, these components are in the OK, NOT RUNNING state, which is normal behavior.



## 6.8 Provide Initial Configuration

For the VNF to start handling traffic, configuration data must be provided. This procedure describes how to provide initial configuration data for the VNF if it was not already injected using the `mrf_config` HOT parameter during instantiation.

### Steps

1. Provide configuration data for the VNF.
  - If this is the first deployment of the VNF, or configuration data is not available at this point, perform initial configuration, as described in section [Manual Configuration of the Initial Configuration Guide](#).
  - If configuration data is available, either from a backup of an older VNF, or prepared by Ericsson, import configuration data, as described in section [Importing Configuration Data from NETCONF Template of the Initial Configuration Guide](#).

## 6.9 Check vMRF Status

This procedure describes how to verify the vMRF deployment. The status check involves running a vMRF command.

### Steps

1. In the OpenStack dashboard **Stack Overview** page, check the outputs section for the vMRF VNF O&M IP address.
2. Open an SSH connection to the O&M IP address of the vMRF VNF instance using the following command:

```
ssh -A -i <private key .pem file> <user ID>@<O&M IP address>
```

3. Run the following command to check the status of VMs in the cluster:

```
verify_vmrf_cluster_status.py
```

- If the VMs are operating correctly, the OK status is displayed in the command printout. You can exit this procedure.
  - If there are VMs with faulty components, a list of faulty VMs and detailed component information of the cluster is displayed. Continue with the next step.
4. Check all components with erroneous state. For specific trouble cases and remedies, refer to the [vMRF Troubleshooting Guideline](#).



**Note:** The MrfDirector and COM components are in the OK state only for the VM whose SC role is ACTIVE, that is, the active System Controller (SC) VM. In all other VMs, these components are in the OK, NOT RUNNING state, which is normal behavior.

## 6.10 Scale Out to Full VNF Size

This procedure describes how to scale out to the full size of the VNF by updating the vMRF stack.

### Steps

1. Log in to the OpenStack CLI.
2. Use the `heat stack-update` command to increase the size of the VNF.

```
heat stack-update <stack_name> -x -P  
payload_instance_count=<new_number_of_VMs>
```

**Note:** This command only updates the `payload_instance_count` parameter, it is not required to specify the environment file.

3. Log in to the dashboard and check that the VMs belonging to the VNF are visible in the **Network Topology** view.



## Reference List

- [1] *MTAS Media Control Management Guide*, 19/1553-AVA 901 29/9