

# Deployment Guide for OpenStack Virtual Multimedia Resource Function

---

## Installation Instructions

**Copyright**

© Ericsson AB 2016, 2017. 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.



# Contents

<b>1</b>	<b>About This Document</b>	<b>1</b>
<b>2</b>	<b>vMRF Deployment Principles for OpenStack</b>	<b>2</b>
<b>3</b>	<b>vMRF Deployment Process for OpenStack</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	Prepare and Configure Cloud Hardware and Software	8
5.2	Create Flavor	13
5.3	Configure Heat Stack Domain Users	14
5.4	Create Network Topology	15
<b>6</b>	<b>vMRF Deployment for the End User</b>	<b>17</b>
6.1	Set Up Command-Line Access and Security	17
6.2	Create Subnets	18
6.3	Upload the vMRF Image	18
6.4	Prepare Initial Configuration Data (Optional)	19
6.5	Prepare Deployment Parameters	23
6.6	Instantiate and Check vMRF with One VM	26
6.7	Scale Out to Full VNF Size	28
6.8	Check vMRF Status	28





# 1 About This Document

This document describes vMRF deployment on an OpenStack cloud service.

The following user roles are distinguished in this document:

**End User**

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

**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.

## 2 vMRF Deployment Principles for OpenStack

If the hardware and software requirements are met, and after the needed configurations in OpenStack are done, vMRF is instantiated using the OpenStack Heat orchestration engine.

vMRF can contain one or more Virtual Network Functions (VNF), for example, one or more Virtual Multimedia Resource Functions (vMRF). Once the vMRF image is onboarded, any number of vMRF VNFs can be instantiated by performing the same single action in the OpenStack Heat component.

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

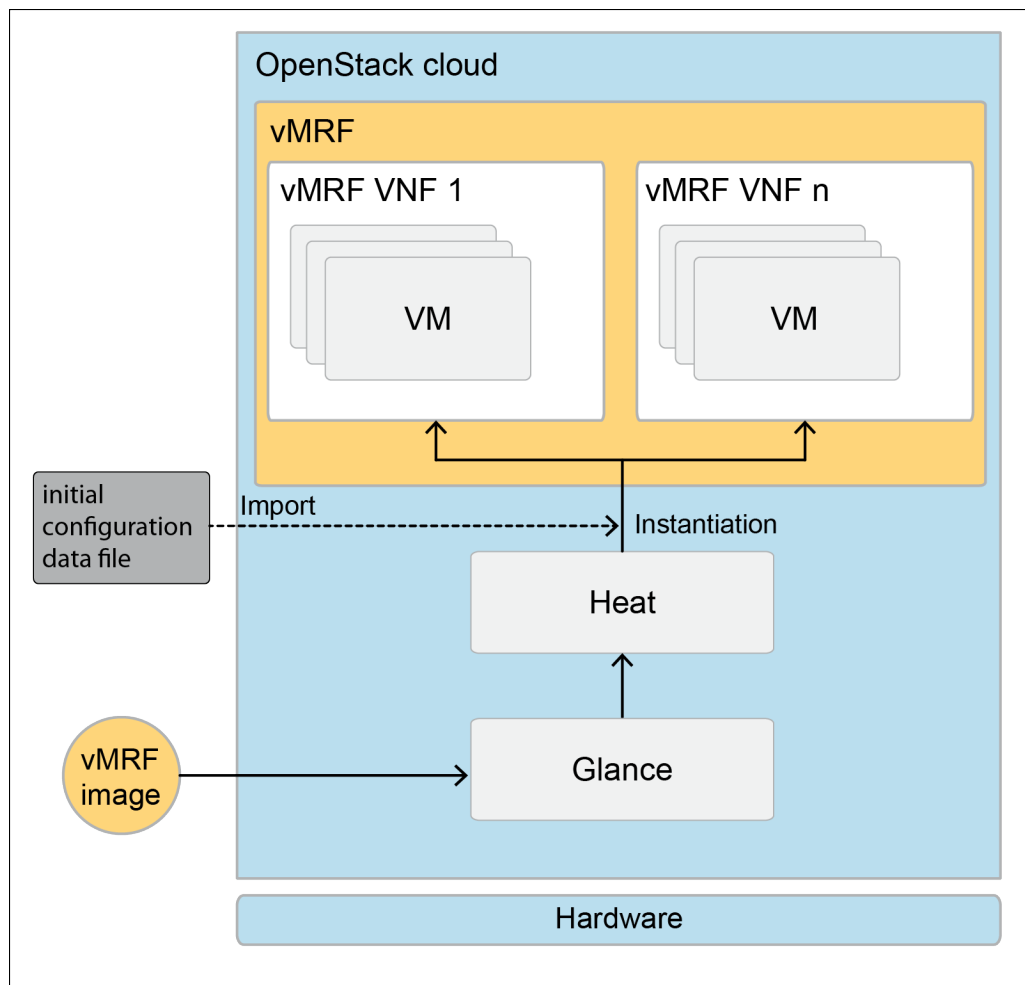
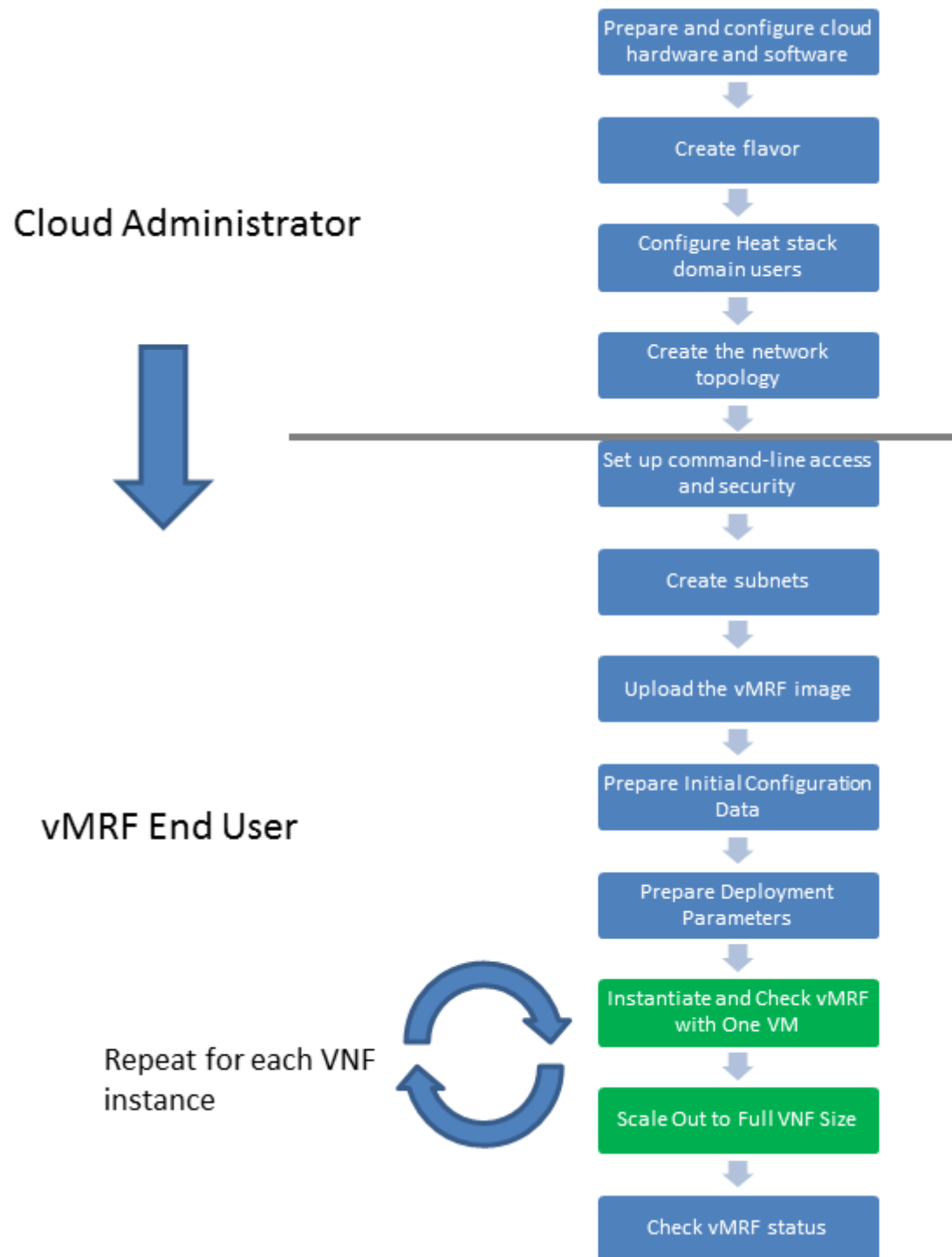


Figure 1 vMRF Deployment



## 3 vMRF Deployment Process for OpenStack

The vMRF deployment process consists of preparations and basic configuration of the cloud environment, and the actual instantiation of one or more vMRF 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 OpenStack, 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. Configure Heat stack domain users

Heat stack domain user configuration is needed for the vMRF instantiation by Heat. Depending on the OpenStack provider, it is possible that the Heat stack domain user configuration does not exist by default.

d. 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. Set up command-line access and security

This step involves configuring the security group for the project and setting up access to the Heat command-line client and to the vMRF VNF instance.

b. 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 OpenStack.

c. Upload the vMRF image

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

d. 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, refer to *Initial Configuration Guide*.



e. 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.

f. 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. For more information, refer to the *Initial Configuration Guide*.

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.

h. Check vMRF status

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



## 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
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 Prepare and Configure Cloud Hardware and Software

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

#### 5.1.1 Define an Availability Zone for vMRF

This procedure must be performed to fulfill the requirement that an vMRF compute host must only run vMRF VMs. The requirement means that other VMs are not allowed to execute on the same physical CPUs as vMRF VMs to ensure that quality of service expectations are met.

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 13.

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

Do the following:

#### 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
```

- 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.

### Steps

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

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

```
lscpu -p
```

- 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 two-socket compute host, with a 10-core CPU in each socket (20 physical cores in total). The NUMA topology is such that each socket corresponds to a NUMA node (0 and 1). In addition, hyperthreading is enabled, so each physical core runs two threads, giving in total 40 logical processors. Suppose that these 40 logical processors are distributed in the following way for the two NUMA nodes:

NUMA node	Logical processors in the <code>lscpu -p</code> printout
0	0-9, 20-29
1	10-19, 30-39

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, 20, 21.

- Isolate all CPUs other than the ones identified on the same 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 of an existing file before adding the `isolcpus` option:



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

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

```
cat /etc/default/grub
GRUB_TIMEOUT=10
GRUB_CMDLINE_LINUX_DEFAULT="nomdmonddf nomdmonisw
isolcpus=2-9,22-29" →
```

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

```
cat /etc/default/grub
GRUB_CMDLINE_LINUX_DEFAULT="isolcpus=2-9,22-29"
```

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

```
cat /proc/cmdline | grep isolcpus
BOOT_IMAGE=/vmlinuz-3.13.0-79-generic root=/dev/ →
mapper/os-root ro console=tty0 net.ifnames=0 →
biosdevname=0 rootdelay=90 nomodeset →
root=UUID=201cf72b-dcb7-4721-9e63-b9f83cd022fb →
isolcpus=2-9,22-29
```

### 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.

Prerequisites:

- The OpenStack environment contains software with the following or newer versions:
  - QEMU emulator version 2.2.0
  - libvirt (libvirt) 1.2.12

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. To deploy without hyperthreading in OpenStack Kilo, add the following to the `Default` section in the `/etc/nova/nova.conf` file, otherwise continue with [Step 3](#):

```
vcpu_pin_set=<list of physical cores to be used for MRSv VMs>
```

**Note:** Make sure that you exclude the cores isolated for the host OS processes in [Configure CPU Core Isolation for the vMRF Compute Hosts](#) on page 9.

Example for the `vcpu_pin_set` command based on the core isolation shown in [Configure CPU Core Isolation for the vMRF Compute Hosts](#) on page 9:

```
vcpu_pin_set=2-9
```

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

```
initctl restart nova-compute
```

For more information, refer to the [OpenStack configuration overview](#).

### 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.

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

## Steps

1. Set the CPU frequency governor to `performance`:

```
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.

Prerequisites:

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

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

#### Steps

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

**<NTP server IP address or host name server <>>**

The following is an example of an `/etc/ntp.conf` file:

#### Example

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

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

Do the following:

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 using the `hw:cpu_thread_policy` parameter, as described in [Step 5](#), OpenStack Mitaka or newer version is required.

## Steps

1. Use the following command to create the flavor named "MRSv\_flavor" with the example ID "99", with 6144 MB of memory, 4 GB root disk space, and eight virtual CPUs:

```
nova flavor-create MRSv_flavor 99 6144 4 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.

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. 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 Configure Heat Stack Domain Users

Depending on the OpenStack provider, it is possible that the Heat stack domain user configuration does not exist by default. This procedure describes how to check and set the Heat stack domain user configuration.

Do the following:

### Steps

1. Check that the following parameters have a value in the `heat.conf` file on the controller:
  - `stack_domain_admin_password=<password>`
  - `stack_domain_admin=<domain admin ID>`
  - `stack_user_domain=<domain ID>`



### Result

If the parameters have a value, the stack domain user configuration exists, and you can quit this procedure. If the parameters do not have a value, continue with the next step.

2. Do the stack domain user configuration according to the [OpenStack Cloud Administrator Guide](#).

## 5.4 Create Network Topology

The vMRF VNF instance connects to networks. The networks 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
Network reserved for future use

Do the following:

### Steps

1. Using the network plan, create the required networks listed in vMRF, 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, as described before. To create networks by launching Heat stacks, do the following:

- a. Change the `example_environment.yaml` file provided in the vMRF software delivery package 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 in [Table 1](#).

**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.

Do the following:

##### 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, refer to 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 Create Subnets

Subnets must be created within the networks created by the cloud administrator in [Create Network Topology](#) on page 15. 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_environment.yaml` file provided in the vMRF software delivery package 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.

## 6.3 Upload the vMRF Image

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

Do the following:



## 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 [unresolved external reference].

*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.
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>
```

## 6.4 Prepare Initial Configuration Data (Optional)

Initial configuration data means the data needed for vMRF to start processing traffic. This procedure describes how to prepare initial configuration data if you are importing it during deployment. For other options, refer to *Initial Configuration Guide*.

Initial configuration data is packed into a `tar.gz` file. An example initial configuration file is part of the vMRF software delivery package, with the name `example_config.tar.gz`. You must manage the content of the `tar.gz` file



included in the vMRF software delivery package using a Python script that is also delivered in the software delivery package. To run the script, you need Python installed on a Linux computer.

The `tar.gz` file contains the following:

- An XML file that contains MO configuration values. The example `tar.gz` file contains an example XML file with the name `config.xml`. You must edit the XML file with values that match your environment, as described in the procedure below. As an optional service, Ericsson can provide a file that is already edited to match your environment.
- Node credentials: You must add the node credentials to the `tar.gz` file, as described in the procedure below.
- Trusted certificates: You must add the trusted certificates to the `tar.gz` file, as described in the procedure below.
- `manifest.yaml`: The file contains configuration metadata for the import and the contents of `/usr/share/image/package_build_data` for troubleshooting.

## 6.4.1 Prepare the Initial Configuration Tar File

### Steps

To prepare initial configuration data, do the following:

1. If the configuration XML file is already edited to match your environment, go to [Step 4](#). Otherwise, extract the initial configuration XML file from the tar file for editing using the `vmrs_config_update.py` script:

```
vmrs_config_update.py -c example_config.tar.gz -gxf  
config.xml
```

2. Edit the XML file with specific values for your environment, as described in [Edit the Initial Configuration XML File](#) on page 21.
3. Pack the XML file back into the tar file using the `vmrs_config_update.py` script:

```
vmrs_config_update.py -c example_config.tar.gz -sxf  
config.xml
```

4. Add the node credentials and trusted certificates to the tar file, and set the LDAP password using the `vmrs_config_update.py` script:

```
vmrs_config_update.py -c example_config.tar.gz -anc  
key.pem cert.pem -atc <trusted_certificate>.pem ldap  
<password>
```





5. If the initial configuration is not a configuration backup of an older VNF, configure security as described in *vMRF Security Management*. Skip the installing node credentials and trusted certificates configuration steps, as they were already performed during the configuration data import procedure.

## 6.4.2 Edit the Initial Configuration XML File

For a list of MOs and attributes to configure, refer to the *Initial Configuration Guide*.

**Note:** The XML document is case-sensitive, pay attention when editing MO names and attributes.

### Steps

1. Search for the *MediaResourceFunction* element and check the IP version. Edit the attribute, if needed.

XML Line to Check or Edit	Description
<medialpVersion>IPV4</medialpVersion>	The IP versions used in the network (IPv4, IPv6, or both). IPv4 is the default value.

2. Search for the *MrfH248Control* element inside *MediaResourceFunction*.

- a. Check, or edit, if needed, `localPortNumber`.

XML Line to Check or Edit	Description
<localPortNumber>2944</localPortNumber>	The local signaling SCTP port.

- b. Create or edit the *MrfH248Interface* instance. There must be at least one H.248 interface instance configured for the vMRF controlling server.

XML Line to Check or Edit	Description
<MrfH248Interface> <mrh248InterfaceId>3</mrh248InterfaceId> <remoteIpAddress>10.50.186.161</remoteIpAddress> <administrativeState>UNLOCKED</administrativeState> <remotePortNumber>2944</remotePortNumber>	The vMRF controlling server settings.



XML Line to Check or Edit	Description
<code>&lt;/MrfH248Interface&gt;</code>	

3. Search for the `Announcements` element inside `MediaResourceFunction`.

a. Check `defaultLanguageCode`.

XML Line to Check or Edit	Description
<code>&lt;defaultLanguageCode&gt;en-EN&lt;/defaultLanguageCode&gt;</code>	The default language selection for announcements.

b. Create and edit `BasicAnnouncement` elements. Note, that files named with the `fileNames` attribute must be stored in the announcement server, otherwise the import fails.

XML Line to Check or Edit	Description
<code>&lt;BasicAnnouncements&gt;</code> <code>&lt;basicAnnouncementsId&gt;1&lt;/basicAnnouncementsId&gt;</code> <code>&lt;BasicAnnouncement&gt;</code> <code>&lt;basicAnnouncementId&gt;1615&lt;/basicAnnouncementId&gt;</code> <code>&lt;duration&gt;10000&lt;/duration&gt;</code> <code>&lt;iteration&gt;20&lt;/iteration&gt;</code> <code>&lt;languageCode&gt;en-GB&lt;/languageCode&gt;</code> <code>&lt;fileName&gt;phr_sub_busy.wav&lt;/fileName&gt;</code> <code>&lt;announcementId&gt;1615&lt;/announcementId&gt;</code> <code>&lt;userLabel&gt;Basic announcement subscriber busy&lt;/userLabel&gt;</code> <code>&lt;filePath&gt;basic&lt;/filePath&gt;</code>	Settings for basic announcement duration, iteration, language, and file name and path.

4. Search for the `SctpProfile` element, and check or edit the following attributes:

XML Line to Check or Edit	Description
<code>&lt;dscp&gt;40&lt;/dscp&gt;</code> <code>&lt;heartbeatInterval&gt;30000&lt;/heartbeatInterval&gt;</code> <code>&lt;initRto&gt;3000&lt;/initRto&gt;</code>	SCTP settings.



XML Line to Check or Edit	Description
<code>&lt;maxBurst&gt;4&lt;/maxBurst&gt;</code> <code>&lt;maxRto&gt;60000&lt;/maxRto&gt;</code> <code>&lt;maxInitRt&gt;8&lt;/maxInitRt&gt;</code> <code>&lt;minRto&gt;1000&lt;/minRto&gt;</code> <code>&lt;sackTimer&gt;40&lt;/sackTimer&gt;</code> <code>&lt;assocMaxRtx&gt;10&lt;/assocMaxRtx&gt;</code> <code>&lt;pathMaxRtx&gt;5&lt;/pathMaxRtx&gt;</code> <code>&lt;primaryPathMaxRtx&gt;0&lt;/primaryPathMaxRtx&gt;</code> <code>&lt;cookieLife&gt;60&lt;/cookieLife&gt;</code> <code>&lt;alphaIndex&gt;ALPHA_1_8TH&lt;/alphaIndex&gt;</code> <code>&lt;betaIndex&gt;BETA_1_4TH&lt;/betaIndex&gt;</code>	

```

<Transport>
  <transportId>1</transportId>
  <SctpProfile xmlns="urn:com:ericsson:ecim:SctpProfileMOM">
    <sctpProfileId>1</sctpProfileId>
    <dscp>40</dscp>
    <heartbeatInterval>30000</heartbeatInterval>
    <initRto>3000</initRto>
    <maxBurst>4</maxBurst>
    <maxRto>60000</maxRto>
    <maxInitRt>8</maxInitRt>
    <minRto>1000</minRto>
    <sackTimer>40</sackTimer>
    <assocMaxRtx>10</assocMaxRtx>
    <pathMaxRtx>5</pathMaxRtx>
    <primaryPathMaxRtx>0</primaryPathMaxRtx>
    <cookieLife>60</cookieLife>
    <alphaIndex>ALPHA_1_8TH</alphaIndex>
    <betaIndex>BETA_1_4TH</betaIndex>
  </SctpProfile>
</Transport>

```

## 6.5 Prepare Deployment Parameters

This procedure describes how to prepare the parameters for deployment.

### Steps

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
<code>admin_username</code>	The user name, password, and SSH key of the emergency user. <b>The default password must be changed by</b>
<code>admin_password_hash</code>	



HOT Parameter Name	Description
admin_authorized_key	<b>supplying a new password hash in this parameter.</b> <sup>(1)</sup> For information on the SSH key, see <a href="#">Set Up Access and Security for Instances</a> on page 17.
mrsv_flavor	See <a href="#">Create Flavor</a> on page 13.
external_net_name	See <a href="#">Create Network Topology</a> on page 15.
management_net_name	
rsrv_net_name	
signaling_net_name	
media_net_name	
mrsv_image	See <a href="#">Upload the vMRF Image</a> on page 18.
payload_instance_count	The number of additional VMs to be created. Ensure that this parameter is set to 0.
mrsv_config	If you are importing initial configuration data during deployment, you must give a value to this parameter. The parameter contains the path to 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 with the following command:  <b>base64 -w 0 &lt;file path&gt;</b>
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> .
shared_storage_server_path <sup>(2)</sup>	The external shared storage path.
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>	The shared storage server fingerprint in one line. The fingerprint can be created with the following command:  <b>ssh-keyscan -p &lt;server_port&gt;                   &lt;server_host&gt;</b>



HOT Parameter Name	Description
	The input can be generated by replacing the end of line characters with the string "\n".
announcement_storage_server_username <sup>(3)</sup>	User name for the external announcement storage server.  The default value is <code>announcements</code> .
announcement_storage_ssh_private_key <sup>(3)</sup>	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".
announcement_storage_server_path <sup>(3)</sup>	The external announcement storage path.  The default path is <code>announcement_storage</code> .
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>	The announcement storage server fingerprint in one line. The fingerprint can be created with the following command:  <b>ssh-keyscan -p &lt;server_port&gt; &lt;server_host&gt;</b>  The input can be generated by replacing the end of line characters with the string "\n".
pm_data_monitoring_hosts_ip_address	Optional parameter. IP address of PM data monitoring host
pm_data_monitoring_hosts_port	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.

(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 announcement storage is used. Leave it blank otherwise. For more information on the shared storage feature, refer to vMRF Overview.

(3) This parameter needs to be defined only if the 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, refer to vMRF Overview.



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
  mrsv_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$6dwcdDEoTM029FfOe4Ejxa4HKKS0LkzhAdGPLLoQ..."
  admin_authorized_key: "mykey"
#Replace the value on the following line with the base64 command output on the path to the tar.gz →
file
  mrsv_config: "H4sICCIpGFcC/2JhY2t1cF8xODA0MTZfNC50YXIA7V1bc+I4Fs5zV+1/oPp1F..."
  shared_storage_server_username: "myuser"
  shared_storage_ssh_private_key: "myprivkey"
  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: "12:f8:7f:78:61:b4:bf:e2:de:44:15:96:4e:d4:72:99"
  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"
```

3. If you are importing initial configuration data during deployment, ensure that it matches your environment, as described in [Prepare Initial Configuration Data \(Optional\)](#) on page 19.

## 6.6 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.

### 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
```

**Note:** Ensure that you do not use dots (".") in the stack name.

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 <private key .pem file>
<admin_username>@<O&M IP address>
```

The *<private key .pem file>* is the private key file of the user, and it corresponds to the public key file added to the Compute database in [Set Up Access and Security for Instances](#) on page 17.

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_cluster_status.py
Running Command: "verify_vmrf_cluster_status.py" on host: 192.168.0.3 (fi2-vmrf)
eth0: OK
eth1: OK
eth2: OK
SC role: ACTIVE
CoreMW: OK
COM: OK, RUNNING
MrfDirector: OK, RUNNING
CliDaemon: OK
IpPipeline: OK
TC-MPD: OK
MrfAgent: OK
CloudInit: OK
SEC-CERT: OK
neighbourdetection: 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.



## 6.7 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.

**Note:** When using the command, all deployment parameters must have the same values as during stack creation. To ensure this, the same `example_environment.yaml` file must be used, and only the `payload_instance_count` parameter needs to be specified. Any deviation from the original values can impact the traffic.

The following example shows how to use the command to increase the size of the VNF while using the `example_environment.yaml` file:

```
heat stack-update <stack_name> -e
example_environment.yaml -f vmrf.yaml -P
payload_instance_count=<new number of VMs>
```

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

## 6.8 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>
```

The `<private key .pem file>` is the private key file of the user, and it corresponds to the public key file added to the Compute database in [Set Up Access and Security for Instances](#) on page 17.

3. Run the following command:





### **verify\_vmrf\_cluster\_status.py**

If any of the components is in a state other than OK, continue with the next step.

4. Run the following command:

### **cluster run verify\_vmrf\_node\_status.py**

Check that all components are in the correct state. In the case of any problems, refer to the *vMRF Troubleshooting Guideline*. The following is an example output from a VNF with three VMs that is operating correctly:

#### **Example**

```
$ cluster run verify_vmrf_node_status.py
Running command: "verify_vmrf_node_status.py" on host: mrsv-vmrf.novalocal
    eth0: OK
    eth1: OK
    eth2: OK
    SC role: ACTIVE
    CoreMW: OK
        COM: OK, RUNNING
    MrfDirector: OK, RUNNING
    CliDaemon: OK
    IpPipeline: OK
    TC-MPD: OK
    MrfAgent: OK
    CloudInit: OK
    SEC-CERT: OK
    neighbourdetection: OK

Running command: "verify_vmrf_node_status.py" on host: mrsv-vmrf-0.novalocal
    eth0: OK
    eth1: OK
    eth2: OK
    SC role: STANDBY
    CoreMW: OK
        COM: OK, NOT RUNNING
    MrfDirector: OK, NOT RUNNING
    CliDaemon: OK
    IpPipeline: OK
    TC-MPD: OK
    MrfAgent: OK
    CloudInit: OK
    SEC-CERT: OK
    neighbourdetection: OK

Running command: "verify_vmrf_node_status.py" on host: mrsv-vmrf-1.novalocal
    eth0: OK
    eth1: OK
    eth2: OK
    SC role: QUIESCED
    CoreMW: OK
        COM: OK, NOT RUNNING
    MrfDirector: OK, NOT RUNNING
    CliDaemon: OK
    IpPipeline: OK
    TC-MPD: OK
    MrfAgent: OK
    CloudInit: OK
    SEC-CERT: OK
    neighbourdetection: 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.