

Overview of VM Life Cycle Management

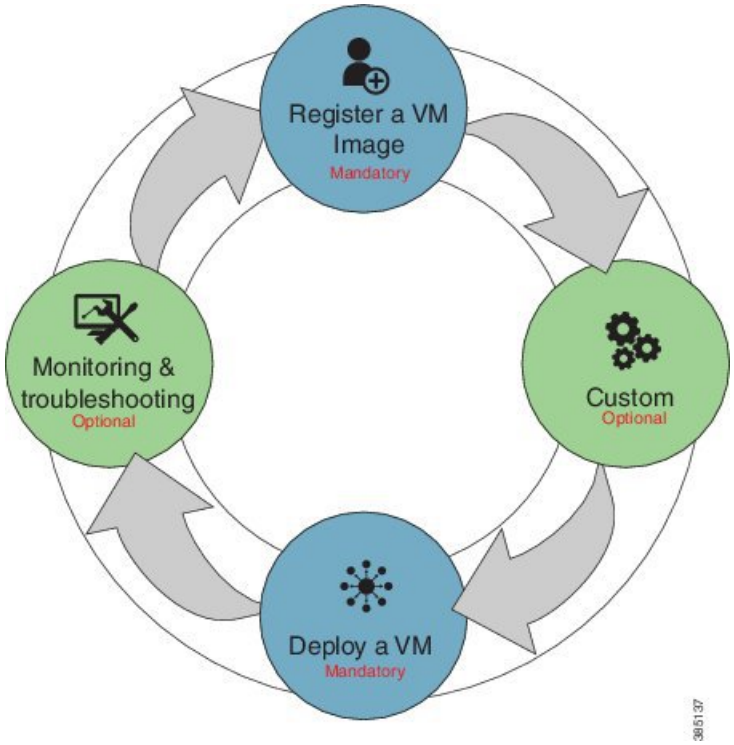
VM life cycle management refers to the entire process of registering, deploying, updating, monitoring VMs, and getting them service chained as per your requirements. You can perform these tasks and more using a set of REST APIs or NETCONF commands or the Cisco Enterprise NFVIS portal.

- Workflow of VM Life Cycle Management, on page 1
- Uploading VM Images to an NFVIS Server, on page 3
- VM Bootstrap Configuration Options with a VM Deployment, on page 4
- OpenStack Configuration Drive Support for Third Party VMs, on page 5
- Performing Resource Verification, on page 6
- Configuring Management IP Subnet, on page 7
- VM States, on page 7

Workflow of VM Life Cycle Management

The following diagram depicts the basic workflow of the VM life cycle management using REST APIs:

Figure 1: VM Life Cycle Management



1. **Register a VM Image—**To create a VM instance, you must register the source image in the NFVIS image repository. Registering the image is a one-time activity. The source image is expected to be hosted on a remote server. In the Registration API you specify the URL on the remote server, where the file is located. When invoked, device pulls the API info, completes the file transfer and registration. Once registered, you can create one or more VM instances. Remote servers supported for hosting the source images (.tar.gz files) are HTTP/HTTPS servers. Starting with NFVIS 4.12.1 release, remote servers can also be FTP servers or SCP servers.

2. **Customizing the VM—**After registering a VM image, you can optionally create a custom profile or flavor for the VM image if the profiles defined in the image file do not match your requirement. The flavor creation option lets you provide specific profiling details for a VM image, such as the virtual CPU on which the VM will run, and the amount of virtual memory the VM will consume.

   Depending on the topology requirement, you can create additional networks and bridges to attach the VM to during deployment.

3. **Deploy a VM—** A VM can be deployed using the deployment API. The deployment API allows you to provide values to the parameters that are passed to the system during deployment. Depending on the VM you are deploying, some parameters are mandatory and others optional.

4. **Manage and Monitor a VM—**You can monitor a VM using APIs and commands that enable you to get the VM status and debug logs. Using VM management APIs, you can start, stop, or reboot a VM, and view statistics for a VM such as CPU usage.

   A VM can also be managed by changing or updating its profile. You can change a VM's profile to one of the existing profiles in the image file; alternatively, you can create a new custom profile for the VM.

   The vNICs on a deployed VM can also be added or updated.

Note Before performing the VM life cycle management tasks, you will have to upload the VM images to the NFVIS server or http/s server.

For details on APIs, see the VM Lifecycle Management APIs chapter in the *API Reference for Cisco Enterprise Network Function Virtualization Infrastructure Software*.

Uploading VM Images to an NFVIS Server

You can upload VM images to an NFVIS server in the following ways. The files are copied to the default location (/data/intdatastore/uploads) on the host server.

- Copy the images from your local system to the NFVIS server—Use the **Image Upload** option from the Cisco Enterprise NFVIS portal.
- Copy the images using the USB drive—Ensure that you have plugged the USB drive that contains the required images into the server before mounting the USB drive.
- Copy using the **scp** command (scp username@external_server:/path/image.tar.gz intdatastore:image.tar.gz).

To copy an image using the USB device:

```
configure terminal
system usb-mount mount ACTIVE
system file-copy usb file name usb1/package/isrv-universalk9.16.03.01.tar.gz
commit
```

Note Use the **show system file-list disk usb** command in privileged EXEC mode to view a list of files available with the mounted USB drive. To save space, you can delete all unwanted text and TAR files from the default location using the **system file-delete** command in global configuration mode.

Verifying the Image Copied from the USB Drive

After copying the file from the USB drive to the host server, you can verify the file using the **show system file-list disk local** command:

```
nfvis# show system file-list disk local

SI NO     NAME     PATH                                                    SIZE  TYPE    DATE MODIFIED
-------------------------------------------------------------------------------------------------------------
1 lastlog-20170314.gz /data/intdatastore/logs/2017-03/14/10-00 337 Other 2017-03-14 21:55:42

2 escmanager-tagged-log.log-20170314.gz /data/intdatastore/logs/2017-03/14/10-00 167K Other
 2017-01-18 05:58:26
3 confd_audit.log-20170317.gz /data/intdatastore/logs/2017-03/17/09-30 4.6K Other 2017-03-17
 21:29:59
4 esc_postinit.log-20170317.gz /data/intdatastore/logs/2017-03/17/05-00 605K Other 2017-03-17
 16:40:19
5 error.log-20170317.gz /data/intdatastore/logs/2017-03/17/05-00 1.3K Other 2017-03-17
16:40:15
```

```
6 ovs-ctl.log-20170317.gz /data/intdatastore/logs/2017-03/17/12-00 20 Other 2017-03-16
00:00:01 4:01
!
!
!
62 ovs-ctl.log-20170323.gz /data/intdatastore/logs/2017-03/23/12-00 20 Other 2017-03-22
00:00:01
63 CentOS-7-x86_64-Everything-1511.ova /data/intdatastore/uploads 1.1G VM 2017-03-15 19:20:03
 Package
64 TinyLinux.tar.gz /data/intdatastore/uploads 17M VM 2017-03-15 18:25:00 Package
65 Cisco-KVM-vWAAS-1300-6.3.0-b98.tar.gz /data/intdatastore/uploads 979M VM 2017-03-15
19:19:11 Package
66 ubuntu_14.04.3-server-amd64-disk1.tar /data/intdatastore/uploads 527M VM 2017-03-15
19:20:17.gz Package
67 asav961.tar.gz /data/intdatastore/uploads 164M VM 2017-03-15 18:24:57 Package
68 isrv-universalk9.16.03.01.tar.gz /data/intdatastore/uploads 1.3G VM 2017-03-15 19:19:53
```

Related APIs and Commands

APIs	CLI Commands
• /api/operations/system/file-copy/usb/file • /api/config/system/usb-mount	• system file-copy usb file name • system usb-mount mount ACTIVE • system file-delete • show system file-list disk usb • show system file-list disk local

VM Bootstrap Configuration Options with a VM Deployment

You can include the bootstrap configuration (day zero configuration) of a VM in the VM deployment payload in the following ways:

- Bundle bootstrap configuration files into the VM package: In this method, the bootstrap configuration variables can be assigned tokens. Token names must be in bold text. For each variable with a token assigned, key-value pairs must be provided during deployment in the deployment payload.
- Bootstrap configuration as part of the deployment payload: The entire bootstrap configuration is copied to the payload without tokens.
- Bootstrap configuration file in the NFVIS server: In this method, the configuration file is copied or downloaded to the NFVIS server, and referenced from the deployment payload with the filename, which includes the full path.

For examples on how to use bootstrap configuration options in the deployment payload, see the API Reference for Cisco Enterprise Network Function Virtualization Infrastructure Software.

OpenStack Configuration Drive Support for Third Party VMs

To enable staging of bootstrap configuration files at the time of a third party VM deployment as per OpenStack standards, the following cloud init format is supported:

```
openstack/content
openstack/content/0000
openstack/content/0001
openstack/latest/meta_data.json
```

In the above sample, the "0000" and "0001" files are the actual bootstrap files from the deployment payload. A third party VM can use the init file to fetch its configuration files.

The following metadata file is used to provide the file path on the configuration drive and reference to the actual bootstrap configuration files.

```
{
    "files": [
        {
            "content_path": "/content/0000",
            "path": "/config/day-0.txt"
        },
        {
            "content_path": "/content/0001",
            "path": "/sample/path/iosxe_config.txt"
        }
    ]
}
```

With this implemenation, two copies of the same bootstrap configuration file will be present on the virtual CD-ROM package. The first version at the root (iosxe_config.txt) and the second inside the "openstack/content" folder.

The admin will also have to sepcify the bootstrap configuartion file in the image properties file before packaging the VM.

Example for the Bootstrap Configuartion File in the Image Properties File

```
--optimize=OPTIMIZE [REQUIRED] optimized VM: --optimize=true/false;
 --root_file_disk_bus=ROOT_FILE_DISK_BUS root disk file type:
 --root_file_disk_bus=virtio/ide; default is virtio
 --virtual_interface_model=VIRTUAL_INTERFACE_MODEL
 --virtual_interface_model=rtl8139; default is none
 --thick_disk_provisioning=THICK_DISK_PROVISIONING
 --thick_disk_provisioning=true; default is false
 --bootstrap_cloud_init_bus_type=BOOTSTRAP_CLOUD_INIT_BUS_TYPE
 --bootstrap_cloud_init_bus_type=virtio; default is ide
 --bootstrap_cloud_init_drive_type=BOOTSTRAP_CLOUD_INIT_DRIVE_TYPE
 --bootstrap_cloud_init_drive_type=disk; default is cdrom
 --bootstrap=BOOTSTRAP bootstrap file/s for VM (two parameters required in the format of
dst:src; dst filename including path has to match exactly to what the VM expects;
 upto 20 bootstrap files are accepted.)
 examples:
 --bootstrap ovf-env.xml:file1,ios-xe.txt:file2 for ISRv; both files get mounted at the
root level on the VM.
 --bootstrap day0-config:filename1 for ASAv
 --bootstrap
 /:bootstrap.xml,/license/lic.txt:license.txt
```

```
 bootstrap.xml get mounted as bootstrap.xml at root, and license.txt get mounted as
/license/lic.txt.
```



Note If any of the strings in the configuration file has wild characters, wrap the string with this #[[ ]]# so that the token/key replacement engine does not consider wild characters as key or token, and looks for key value pairs to replace during a VM deployment.

For details on the OpenStack standards, visit *http://docs.openstack.org*.

Performing Resource Verification

Given below are the APIs and commands to perform different types of resource verification:

Task	API	Command
To display CPU information for each CPU or the user specified CPU, and the VMs pinned to the CPU	• /api/operational/resources/cpu-info/cpus • /api/operational/resources/cpu-info/cpus/cpu • /api/operational/resources /cpu-info/cpus/cpu/<cpu-id>	show resources cpu-info cpus
To display information on the VMs running in all the physical CPUs or a specific physical CPU in the system	• /api/operational/resources/cpu-info/vnfs • /api/operational/resources/cpu-info/vnfs/vnf • /api/operational/resources/cpu-info/vnfs/vnf/ <deployment_name>.<vm_group_name>	show resources cpu-info vnfs
To get information on the number of CPUs allocated to VMs and the CPUs that are already used by the VMs	/api/operational/resources/cpu-info/allocation	show resources cpu-info allocation



Note To display information on all CPUs, VMs pinned to the CPUs, and VMs allocated to the CPUs, use the **show resources cpu-info** command.

CPU Over-Subscription

Cisco Enterprise NFVIS does not allow CPU over-subscription for low-latency network appliance VMs (for example, Cisco ISRv and Cisco ASAv). However, the CPU over-subscription is allowed for non low-latency VMs (for example, Linux Server VM and Windows Server VM).

Configuring Management IP Subnet

By default, all VMs with management interfaces will be provisioned with an IP address in the subnet of 10.20.0.1. To change the default subnet, the following commands needs to be executed in a sequence to first delete an exiting subnet and then add a new subnet in the network. For these commands to work, ensure there is no managed VNF's in the system before you change management network address.

To delete an existing subnet use **no vm_lifecycle networks network int-mgmt-net subnet int-mgmt-net-subnet** command.

To create a new subnet:

```
configure terminal
vm_lifecycle networks network int-mgmt-net subnet int-mgmt-net-subnet address 105.20.0.0
gateway 105.20.0.1 netmask 255.255.255.0 dhcp false
commit
```

VM States

VM States	Description
VM_UNDEF_STATE	The initial state of VM or VNF before deployment of this VM.
VM_DEPLOYING_STATE	VM or VNF is being deployed on to the NFVIS.
VM_MONITOR_UNSET_STATE	VM or VNF is deployed in the NFVIS but the monitoring rules are not applied.
VM_MONITOR_DISABLED_STATE	Due to a VM action request or recovery workflow, the monitoring or KPI rules applied on the VM or VNFs were not enabled.
VM_STOPPING_STATE	VM or VNF is being stopped.
VM_SHUTOFF_STATE	VM or VNF is in stopped or shutoff state.
VM_STARTING_STATE	VM or VNF is being started.
VM_REBOOTING_STAT	VM or VNF is being rebooted.
VM_INERT_STATE	VM or VNF is deployed but not alive. The KPI monitor is applied and waiting for the VM to become alive.
VM_ALIVE_STATE	VM or VNF is deployed and successfully booted up or alive as per the monitor or kpi metric.
VM_UNDEPLOYING_STATE	VM or VNF is being undeployed or terminated.
VM_ERROR_STATE	VM or VNF will be in error state if deployment or any other operation is failed.