SMI Cluster Manager Operations

Overview

The Subscriber Microservices Infrastructure (SMI) Kubernetes (K8s) Cluster Manager allows you to deploy the virtual machines (VMs) – provisioned through the vCenter – on your VMware environment. You can deploy the SMI K8s Cluster Manager either in All-in-one (AIO) or Multimode configurations. In the multimode configuration, the SMI K8s Cluster Manager can manage multiple clusters - with the minimum being three clusters. Also, the in-built Ingress support in K8s allows you to access the K8s cluster over HTTP.

The SMI K8s cluster includes the following VMs:

- **Control Plane** – The stateless control plane VMs.
- **ETCD** – The database for the K8s metadata. It is separated from the control plane VMs.
- **OAM** – The OAM nodes reserved for CEE. It performs monitoring and logging.
- **APP** – The SMI applications such as SMF, PCF and so on

The subsequent sections provide more information about the different components in the SMI K8s Cluster Manager.

Sync API

The Sync API is responsible for deploying the VMs, provisioning the base OS, applications, and application host OS. It also provisions the K8s and add-ons (associated with K8s) in both the VMware and OpenStack environments.

For application provisioning, the Sync API allows you to install the applications and configure it through their own APIs. All the 5G applications come equipped with its own Ops Centers. The SMI K8s Cluster Manager allows you to deploy these application Ops Center on top of it.

Note The SMI supports only the UTC time zone by default. Use this time zone for all your deployment and installation activities related to the SMI.

Offline Images

All the applications (5G and Cisco Cable) based on the SMI framework are provided as helm charts (templating for K8s) and docker images. The Helm charts provide the metadata and docker images provide the application containers in the K8s cluster.

For the offline deployment, the docker images and helm charts are provided as TAR balls for loading into the K8s.

For online deployment, anyone with access to devhub.cisco.com can pull these docker images and helm charts through the CLI. Also, you can customize specific applications that are loaded into the SMI K8s Cluster Manager using the customization (docker) image.

Command Line Interface

The SMI K8s Cluster Manager Command Line Interface (CLI) consists of two major components: Environment and Clusters.

- **Environment** – The environment defines the vCenter environment to be used. It has two options: *vCenter* or *Manual*. For OpenStack environments, select the M*anual* option in the CLI to skip deploying the VMs on to the cluster.
- **Cluster** – You can specify the SMI K8s Cluster Manager to link to the specific cluster through the CLI. In addition, you can also define cluster configurations such as Virtual-IP address, size, Kubernetes add-ons and so on

To view the current running configuration, use the *show run config* command in the CLI.

Operating the SMI Cluster Manager on Bare Metal

The SMI Cluster Manager manages the configuration and life-cycle of the Kubernetes Cluster and the VNF Nodes (UPF) deployed on Bare Metal servers. The subsequent sections describe the operations involved in deploying the remote clusters on Bare Metal (Cisco UCS) servers.

Deploying Remote Clusters

This section describes the procedure to deploy a remote Kubernetes and UPF cluster using the SMI Cluster Manager on Cisco UCS servers.

Deploying Kubernetes Cluster

You can deploy a Kubernetes Cluster when a Cluster Manager (HA or AIO or Inception) is already available. To deploy a Kubernetes Cluster:

1. Setup the cluster configuration.
   - The following is a sample UCS Configuration for deploying a Kubernetes Cluster:

```
software cnf cee
 url <repo_url>
 user <user_name>
 password <password>
 sha256 <sha256_hash>
exit

 # associating to Bare Metal environment
environments bare-metal
 ucs-server
 exit

# General cluster configuration
clusters <cluster_name>
  environment bare-metal
  addons ingress bind-ip-address <bind_ip_address>
  addons cpu-partitioner enabled
  configuration master-virtual-ip <master_vip>
  configuration master-virtual-ip-interface <master_vip_interface_name>
#For example, eno1
  configuration allow-insecure-registry true
 node-defaults initial-boot default-user <username> #For example, cloud-user

 node-defaults initial-boot default-user-ssh-public-key
  "<SSH_Public_Key>"
 node-defaults initial-boot default-user-password <password>
 node-defaults netplan template
 node-defaults initial-boot netplan ethernets eno1
  dhcp4    false
  dhcp6    false
  gateway4 <gateway_ipv4address>
  nameservers search [ <domain_name> ]
```

```
 nameservers addresses [ <ipv4address>...<ipv4address> ]
exit
node-defaults k8s ssh-username <username>
node-defaults k8s ssh-connection-private-key
"<SSH_Private_Key>"
#initial-boot section of node-defaults
 node-defaults ucs-server host initial-boot networking static-ip
netmask <ipv4_address>
 node-defaults ucs-server host initial-boot networking static-ip
gateway <ipv4_address>
 node-defaults ucs-server host initial-boot networking static-ip
dns <ipv4_address>
  node-defaults ucs-server cimc user <username>
  node-defaults ucs-server cimc password <password>
  node-defaults ucs-server cimc remote-management sol enabled
  node-defaults ucs-server cimc remote-management sol baud-rate
<baud_rate>
  node-defaults ucs-server cimc remote-management sol comport
<com_port>
  node-defaults ucs-server cimc remote-management sol ssh-port
<ssh_port>
  node-defaults ucs-server cimc networking ntp enabled
 node-defaults ucs-server cimc networking ntp servers <ntp_server_url>

  exit

node-defaults os proxy https-proxy <proxy_server>
node-defaults os proxy no-proxy <proxy_servers>
node-defaults os ntp enabled
node-defaults os ntp servers <ntp_server_url> #For exmaple,
ntp.esl.cisco.comnode-defaults os proxy https-proxy http://proxy-wsa.esl.cisco.com:80

exit

#node configuration
 ucs-server host initial-boot networking static-ip ipv4-address
<ipv4address>
 ucs-server cimc ip-address <ipv4address>
 ucs-server cimc storage-adaptor create-virtual-drive true
 exit

# control plane node configuration
nodes <control_plane_node_name> #For example, control-plane-1
 k8s node-type control-plane
 k8s ssh-ip <ipv4address>
 k8s node-labels <node_label/node_type> #For example, smi.cisco.com/oam
 exit
 ucs-server host initial-boot networking static-ip ipv4-address
<ipv4address>
 ucs-server cimc ip-address <ipv4address>
 ucs-server cimc storage-adaptor create-virtual-drive true
 exit
nodes <control_plane_node_name> #For example, control-plane-2
 k8s node-type control-plane
 k8s ssh-ip <ipv4address>
```

```
  k8s node-labels <node_label/node_type> #For example, smi.cisco.com/oam
  exit
  ucs-server host initial-boot networking static-ip ipv4-address
<ipv4address>
  ucs-server cimc ip-address <ipv4address>
  ucs-server cimc storage-adaptor create-virtual-drive true
  exit
 nodes <control_plane_node_name> #For example, control-plane-3
  k8s node-type control-plane
  k8s ssh-ip <ipv4address> \
  k8s node-labels <node_label/node_type>  #For example, smi.cisco.com/node-type
 oam
  exit
  ucs-server host initial-boot networking static-ip ipv4-address
<ipv4address>
  ucs-server cimc ip-address <ipv4address>
  ucs-server cimc storage-adaptor create-virtual-drive true
  exit
 ops-centers cee <ops_center_name> #For example, cee
  repository-local <repo_name> cee-2020-02-0-i04
 exit
exit
```

2. Login to the Cluster Manager CLI and enter the configuration mode
   - Add the Kubernetes Cluster configuration to deploy the Kubernetes Cluster.

Note A sample Kubernetes Cluster Configuration is provided here.

3. Commit the configuration.
4. Monitor the progress of the synchronization.

   ```
   monitor sync-logs cluster_name
   ```

Note The synchronization completes after 30 minutes approximately. The time taken for synchronization is based on network speed, VM power, and so on.

The node names are added to `/etc/host` as part of the sync process. You can connect to the nodes using the node name from the control plane node.

Deploying Remote UPF Clusters

The SMI Cluster Manager uses the Kernel Based Virtual Machine (KVM) to deploy the User Plane Function (UPF) on Bare Metal Servers. The subsequent sections describe the procedures involved in installing the KVM and UPF.

Installing Kernel Based Virtual Machine and User Plane VM

The SMI Cluster Manager utilizes the Kernel Based Virtualization (KVM) – a virtualization technology – to deploy the User Plane Function (UPF) VMs.

To deploy the KVM and UPF:

1. Setup the KVM and UPF configuration

   The following is a sample KVM and UPF configuration

```
software upf <version_number> #For example, v748
 url <repo_url>
 user <username>
 password <password>
 sha256 <sha256_hash> #For example,
9141df47188fb795f3805fd61abf8784d52d2916f32014f564572a6cbbf7c545
 description "<description>" #For example, "UPF software version v748"
 exit

 # associating to Bare Metal environment
environments bare-metal
 ucs-server
 exit

# General cluster configuration
clusters <cluster_name>
  environment bare-metal
  addons ingress bind-ip-address <bind_ip_address>
  addons cpu-partitioner enabled
  configuration master-virtual-ip <master_vip>
  configuration master-virtual-ip-interface <master_vip_interface_name> #For
 example, eno1
  configuration allow-insecure-registry true
  node-defaults ssh-username <username>
  node-defaults initial-boot default-user <username>
  node-defaults initial-boot default-user-ssh-public-key
  "<SSH_Public_key>"
  node-defaults ssh-connection-private-key
  "-----BEGIN OPENSSH PRIVATE KEY-----
    <SSH_Private_Key>
  -----END OPENSSH PRIVATE KEY-----\n"
   node-defaults initial-boot default-user-password <password>
   node-defaults initial-boot netplan ethernets <ethernet_interface> #For
example, eno1
    dhcp4 false
    dhcp6 false
    gateway4 <gateway_ipv4_address>
    nameservers search <<domain_name>>
    nameservers addresses <nameserver_ipv4_addresses>
    exit

 # initial-boot section of node-defaults
   node-defaults ucs-server host initial-boot networking static-ip
netmask <ipv4_address>
   node-defaults ucs-server host initial-boot networking static-ip
```

```
gateway <ipv4_address>
  node-defaults ucs-server host initial-boot networking static-ip dns
 <ipv4_address>
  node-defaults ucs-server cimc user <username>
  node-defaults ucs-server cimc password <password>
  node-defaults ucs-server cimc remote-management sol enabled
  node-defaults ucs-server cimc remote-management sol baud-rate
<baud_rate>
  node-defaults ucs-server cimc remote-management sol comport <com_port>

  node-defaults ucs-server cimc remote-management sol ssh-port <ssh_port>

  node-defaults ucs-server cimc networking ntp enabled
  node-defaults ucs-server cimc networking ntp servers <ntp_server_url>
  exit

  node-defaults os proxy https-proxy <proxy_server>
  node-defaults os proxy no-proxy <proxy_servers>
  node-defaults os ntp enabled
  node-defaults os ntp servers <ntp_server_url>
  #For monitoring the LCM's IP range (Optional)
  node-defaults kvm monitoring local-ip-address-range <ipv4address/subnet>

  exit
  node-type-defaults kvm
    os netplan-additions bridges <bridge_name> #For example, ex4000
    addresses <ipv4_address/subnet>
    exit
  exit

#node configuration
 nodes <node_name> #For example, kvm-1
  ssh-ip <ssh_ipv4address>
  type kvm
  vms <vm_name> kvm-1
    upf software <software_version> #For example, v748
    upf networking management ip <ipv4addess>
    upf networking management netmask <ipv4address>
    upf networking management gateway <ipv4address>
    upf networking management interface-type bridge
    upf networking management bridge name <bridge_name> #For example, ex4000
    type upf
    exit
  vms <vm_name> #For example, upf2
    upf software <software_version> #For example, v748
    upf networking management ip  <ipv4address>
    upf networking management netmask <ipv4address>
    upf networking management gateway <ipv4address>
    upf networking management interface-type bridge
    upf networking management bridge name <bridge_name> #For example, ex4000
    type upf
    exit

#Default VM configuration
```

```
vm-defaults upf day0 username username
  vm-defaults upf day0 password password
  ucs-server host initial-boot networking static-ip ipv4-address
<ipv4address>
  ucs-server cimc ip-address <ipv4address>
  ucs-server cimc storage-adaptor create-virtual-drive true

  initial-boot netplan ethernets <interface_name> #For example, eno1
   addresses <ipv4address/subnet>
   exit
 exit

# control-plane node configuration
  ucs-server host initial-boot networking static-ip ipv4-address
<ipv4address>
  ucs-server cimc ip-address <ipv4address>
  ucs-server cimc storage-adaptor create-virtual-drive true
    initial-boot netplan ethernets <interface_name> #For example, eno1
   addresses <ipv4address/subnet>
  #Configure secure boot (optional)
  ucs-server cimc bios configured-boot-mode Uefi
  ucs-server cimc bios uefi-secure-boot yes
  exit
 exit
   ucs-server host initial-boot networking static-ip ipv4-address
<ipv4address>
  ucs-server cimc ip-address <ipv4address>
  ucs-server cimc storage-adaptor create-virtual-drive true
  initial-boot netplan ethernets <interface_name> #For example, eno1
   addresses <ipv4address/subnet>
  exit
 exit
 ops-centers cee <ops_center_name> #For example, cee
  repository-local <repo_name> cee-2020-02-0-i04
 exit
exit
```

Note For the Clusters on Edge deployments, you must define the **ucs-server host initial-boot networking** parameter. This reduces latency in bringing up the ISO media in CIMC. The following is an example configuration for deploying UPF on remote sites:

```
ucs-server host initial-boot networking static-ip ipv4-address <IPv4address>
ucs-server host initial-boot networking static-ip netmask <IPv4address>
ucs-server host initial-boot networking static-ip gateway <IPv4address>
ucs-server host initial-boot networking static-ip dns <IPv4address>
```

If **ucs-server host initial-boot networking** parameter is not defined, the CIMC can timeout (and throw errors) while trying to download a SMI hard drive image instead of embedding it into the ISO file (30 MB versus 300 MB).

2. Login to the Cluster Manager CLI and enter configuration mode

- Add the KVM and UPF configuration to deploy the KVM and UPF clusters.

Note A sample configuration to deploy KVM and UPF clusters is provided here.

- Commit and exit the configuration
- Validate the cluster configuration in the Cluster Manager

Example:

```
clusters cndp-testbed-cm actions validate-config run log-level DEBUG vmware-checks
false k8s-node-checks false
This will run validation.  Are you sure? [no,yes] yes
message 2020-05-06 19:50:38.597 INFO __main__: Verifying ntp config ......
...
...
2020-05-06 19:50:45.723 INFO __main__: You have not run all checks together. Run
clusters cndp-testbed-cm actions validate run

valid TRUE
```

3. Configure KVM using the following configuration:

```
configure
   clusters <cluster_name>
  vm-defaults
  node-defaults kvm monitoring local-ip-address-range ip-address-and-prefix

           node-type-defaults kvm
           nodes
  type <k8s/kvm>
  vms <name>
  ssh-ip <kvm_IP>
           exit
        exit
```

4. Configure the UPF using the following configuration. The following parameters are specific to UPF configuration:
   - **day 0** – Specifies the configuration applicable to the VM when it is first created. After initial creation (day 0), changes here will not apply unless the VM is deleted and redeployed.
     - **username** – Specifies the StarOS administrator username
     - **password** – Specifies the StarOS administrator password
     - **syslog-ip** – Specifies the StarOS logging syslog IP
   - **networking** – Specifies the configuration for the management interface.

Note Other networks are expected to be provisioned as day 1 configuration by talking directly to the UPF.

- **IP_Address** - Specifies the IP address.
- **Netmask**- Specifies the Netmask
- **Gateway**- Specifies the Gateway
- **interface-type** - Only bridge is supported for now
- **bridge name** - Specifies the bridge name. For more details, see the bridge section defined in KVM configuration
- **domain-name** - Specifies the domain name
- **name-servers** - Specifies the name servers

- **ntp-servers** – Specifies the NTP (Network Time Protocol) settings for UPF
- **software** – Specifies the link to version of UPF to deploy
- **nodes [name] vms** - Currently, two UPFs are allowed per node to align with NUMA.
  - **type (upf - default upf)** - Provides the ability for additional types of VMS in the future. Currently, UPF is the only choice.
- **os enable-passthrough [true_or_false]** - To use PCI Passthrough to pass the NIC to UPF, enable-passthrough must be enabled. By default UPFs use SRIOV to configure the network interfaces.
- **os num-vfs-per-pf [vf_num]** - Specifies the number of VFs created for each PF. The default value is 16.

5. Run the synchronization

   clusters *cluster_name* **actions sync run debug true**

   - Monitor the progress of the cluster synchronization

     monitor sync-logs *cluster_name*

6. Connect to the UPF node through console in KVM after the synchronization is complete

   virsh console *<upf_name>* **serial1--force**

   or SSH to UPF if the bridge is configured locally

   ssh *username*@*management_ip*

Note For authentication, you can use the username and password configured in Day 0 configuration. Also, the SMI Cluster Manger is not required for operating the deployed VMs.

NOTES:

- **vm-defaults** - Allows configuring the necessary values across all the VMs. It is available at cluster and node level.
- **kvm monitoring local-ip-address-range** *ip-address-and-prefix* – Specifies the IP configuration for UPF LCM.

- **node-type-defaults kvm** - Allows configuring the necessary values across all the KVMs.
- **type** <*k8s/kvm*> - Allows only nodes of type kvm to be run
- **vms** <*name*> - Specifies the VM configuration.
- **ssh-ip** <*kvm_IP*> - Specifies the KVM IPv4 address

Upgrading the UPF Clusters

You can upgrade the UPF Clusters using the SMI Cluster Manager. To upgrade, use the following configurations:

1. Login to the Inception Cluster Manager CLI and enter the Global Configuration mode.
2. To upgrade, add a new software definition for the software.

   ```
   configure
     software upf <upf_software_version>
     url <repo_url>
     user <user_name>
     password <password>
     sha256 <SHA256_hash_key>
     exit
   ```

   Example:

   ```
   SMI Cluster Manager# config
   SMI Cluster Manager(config)# software upf <upf_software_version>
   SMI Cluster Manager(config)# url <repo_url>
   SMI Cluster Manager(config)#user <username>
   SMI Cluster Manager(config)#password "<password>"
   SMI Cluster Manager(config)#sha256 <sha256_hash_key>
   SMI Cluster Manager(config)#exit
   SMI Cluster Manager(config)#
   ```

3. Commit the changes.
4. Trigger the Cluster synchronization.

   ```
   configure
     clusters <cluster_name> actions sync run debug true
   ```

   Example:

   ```
   Cluster Manager# config
   Cluster Manager(config)# clusters cndp-testbed actions sync run debug true
   ```

5. Monitor the upgrade progress

   ```
   monitor syc-logs <cluster_name>
   ```

   Example:

   ```
   Cluster Manager# monitor syc-logs cndp-testbed-cm
   ```

6. Login to the CEE control plane node after the Cluster synchronization completes.

   ```
   https://cli.upf-ops-center.<ipv4_address>.<domain_name>/
   ```

7. Verify the software version using the following command.

   ```
   show system ops-center
   ```

Example:

```
SMI Cluster Manager# show system ops-center
```

NOTES:

- **software upf** <*upf_software_version*> - Specifies the UPF software package.
- **url** <*repo_url*> - Specifies the *HTTP/HTTP/file* URL of the software.
- **user** <*user_name*> - Specifies the username for *HTTP/HTTPS* authentication.
- **password** <*password*> - Specifies the password used for downloading the software package.
- **sha256** <*SHA256_hash_key*> - Specifies the SHA256 hash of the downloaded software.

Deploying the CEE Clusters

You can deploy the CEE Clusters using the SMI Cluster Manager. To deploy a CEE Cluster, use the following configurations:

Note From the SMI Cluster Manager perspective, the product refers to Common Execution Environment (CEE). The deployment procedure mentioned in the subsequent section is specific to CEE. However, you can follow the same procedure to deploy 5G Network Functions (SMF or PCF) using the SMI Cluster Manager.

1. Setup the Cluster Manager Configuration.

   The following is an sample UCS configuration for deploying CEE Clusters.

   ```
   software cnf <software_version> #For example, cee-2020-02-0-i04
    url <repo_url>
    user <username>
    password <password>
    sha256 <sha256_hash>
   exit
   environments bare-metal
    ucs-server
   exit
   clusters <cluster_name> #For example, cndp-testbed
     ops-centers cee <app_name> #For example, cee
     repository-local <repo_name> #For example, cee-2020-02-0-i04
    exit
   exit
   ```

2. Login to the Cluster Manager CLI and enter the Global Configuration mode.
3. Commit the changes.
4. Trigger the Cluster synchronization.

   ```
   configure
     clusters <cluster_name> actions sync run debug true
   ```

   Example:

```
Cluster Manager# config
Cluster Manager(config)# clusters cndp-testbed actions sync run debug true
```

5. Monitor the upgrade progress

   monitor syc-logs *<cluster_name>*

   Example:

   ```
   Cluster Manager# monitor syc-logs cndp-testbed-cm
   ```

6. Login to the CEE control plane node after the Cluster synchronization completes.

   https://cli.cee-cee-ops-center.<ipv4_address>.<domain_name>/

NOTES:

- **software cnf** <*cnf_software_version*> - Specifies the Cloud Native Function software package.
- **url** <*repo_url*> - Specifies the *HTTP/HTTP/file* URL of the software.
- **user** <*user_name*> - Specifies the username for *HTTP/HTTPS* authentication.
- **password** <*password*> - Specifies the password used for downloading the software package.
- **sha256** <*SHA256_hash_key*> - Specifies the SHA256 hash of the downloaded software.

Upgrading the CEE Clusters

You can upgrade the CEE Clusters using the SMI Cluster Manager. To upgrade, use the following configurations:

1. Login to the Inception Cluster Manager CLI and enter the Global Configuration mode.
2. To upgrade, add a new software definition for the software.

   configure
   software cnf *<cnf_software_version>*
   url *<repo_url>*
   user *<user_name>*
   password *<password>*
   sha256 *<SHA256_hash_key>*
   exit

   Example:

   ```
   SMI Cluster Manager# config
   SMI Cluster Manager(config)# software cnf <cnf_software_version>
   SMI Cluster Manager(config)# url <repo_url>
   SMI Cluster Manager(config)#user <username>
   SMI Cluster Manager(config)#password "<password>"
   SMI Cluster Manager(config)#sha256 <sha256_hash_key>
   SMI Cluster Manager(config)#exit
   SMI Cluster Manager(config)#
   ```

3. Commit the changes.
4. Trigger the Cluster synchronization.

   configure
   clusters *<cluster_name>* **actions sync run debug true**

Example:

```
Cluster Manager# config
Cluster Manager(config)# clusters cndp-testbed actions sync run debug true
```

5. Monitor the upgrade progress

   monitor syc-logs *<cluster_name>*

   Example:

   ```
   Cluster Manager# monitor syc-logs cndp-testbed-cm
   ```

6. Login to the CEE control plane node after the Cluster synchronization completes.

   https://cli.cee-ops-center.<ipv4_address>.<domain_name>/

7. Verify the software version using the following command.

   show system ops-center

   Example:

   ```
   SMI Cluster Manager# show system ops-center
   ```

NOTES:

- **software cnf** <*cnf_software_version*> - Specifies the CNF software package.
- **url** <*repo_url*> - Specifies the *HTTP/HTTP/file* URL of the software.
- **user** <*user_name*> - Specifies the username for *HTTP/HTTPS* authentication.
- **password** <*password*> - Specifies the password used for downloading the software package.
- **sha256** <*SHA256_hash_key*> - Specifies the SHA256 hash of the downloaded software.

Life Cycle Management of the Virtual Machines

The SMI Cluster Manager supports the complete Life Cycle Management (LCM) of the deployed VMs — currently UPF. The different processes in the LCM of a VM includes:

- VM Provisioning
- VM Removal
- VM Recovery
- VM Recovery Configuration
- Adding a New Node
- Removing an Existing Node
- Server Return Material Authorizations (RMAs)

The subsequent sections describe all the processes involved in LCM of the VM.

Provisioning the Virtual Machine

Provisioning a Virtual Machine involves creating and configuring a VM based on the requirements. In Bare Metal deployments, the UPF VM is provisioned using the KVM technology. For more details on provisioning the UPF VM using KVM, see Installing Kernel Based Virtual Machine and User Plane VM section.

Removing the Virtual Machine

You can remove the UPF VM through the KVM cluster from the *ConfD* CLI. To remove the UPF VM, use the following configuration:

1. Login to the *ConfD* CLI
2. Remove the VM using the following configuration

```
cluster <kvm_cm_name>
nodes <node_name>
no vms <upf_vm_name>
commit
```

3. Run the following command to trigger the cluster synchronization and remove the UPF VM from cluster

```
clusters cluster_name actions sync run debug true
```

NOTES:

- **cluster** <*kvm_cm_name*> - Specifies the KVM cluster
- **nodes** <*node_name*> - Specifies the VM node.
- **no vms** <*upf_vm_name*> - Removes the specified UPF VM

Recovering the Virtual Machine

The *staros_monitor* service is responsible for monitoring the deployed UPF VMs. You can verify the status of the *systemd* service using the following command:

```
systemctl status staros_monitor
```

You must ping the UPF VFNM interfaces repeatedly until the *staros_monitor* service recovers an unhealthy UPF VM. Also, the *staros_monitor* service restarts the UPF in the following conditions:

- When the UPF VM is stopped.
- When the UPF VM is running but not responding to the *ping* command.

Note For troubleshooting any issues, you can view the logs captured in the */var/log/staros_monitor.log* file. Also, you can configure the failure occurrence and *Ping* interval through */data/upf/monitor/staros_monitor.cfg* file. You must restart the *staros_monitor* service for the changes to take effect.

Configuring the Virtual Machine Recovery

To monitor the VMs locally, a bridge called *kvm-monitoring* is created locally. You can configure the local and VNFM IPs of the VM using the following command:

node-defaults kvm monitoring local-ip-address-range *<local_IP_address_range>*

The following are the IP assignments when you configure the local IP address range to 192.168.0.0/24:

- Local Bridge: 192.168.0.1
- VM assigned to NUMA node 0: 192.168.0.2
- VM assigned to NUMA node 1: 192.168.0.3. The default value of **local-ip-address-range** is *192.168.1.0/24*.

NOTES:

- **kvm monitoring** - Specifies the locally created bridge for montioring the VMs.
- *<local_IP_address_range>* - Specifies the local IP address range. The default value is *192.168.1.0/24*

Adding and Removing UPF Nodes

You can add a new UPF node to the existing cluster when required. To add a new UPF node, see Installing Kernel Based Virtual Machine and User Plane VM section.

Similarly, you can remove an existing UPF node when required. To remove an existing node, see Recovering the Virtual Machine section.

Server RMAs

For Return Material Authorization (RMA) servers, you can replace the old node and replace it with a new one. When the hardware is fixed, you can synchronize the cluster and deploy the VM. For more information on replacing the Bare Metal nodes, see SMI Cluster RMA section.

UCS Firmware Upgrade

The firmware upgrade process runs on the UCS server's Cisco Integrated Management Controller (CIMC). After triggering the upgrade, a synchronization process connects to the CIMC at regular (pre-configured) intervals to verify the progress of the upgrade. The upgrade process continues to run even if there is a loss of connectivity between the Cluster Manager running the synchronization process and UCS firmware upgrade.

The UCS firmware upgrade process involves:

1. The Cluster synchronization is triggered with the appropriate firmware version added to the configuration.
2. The Cluster manager (CM) authenticates with the CIMC.
3. The CM compares the current running version of the firmware with the firmware version inside the ISO image.

Note The CM proceeds with the upgrade if there is a discrepancy between the versions. Otherwise, the CM skips the upgrade process.

4. The CIMC receives an API call to begin the firmware upgrade from the CM. Also, the API call includes the link for the ISO image the CIMC server needs to download. The ISO image is located in the CM.
5. The CIMC triggers the server reboot and starts downloading the ISO image.

Note The firmware upgrade aborts if there is no access to the ISO image. Consequently, the server boots up and the sync process fails.

6. The CM starts verifying the status of the firmware (for a maximum time period of 120 minutes) with the CIMC.
7. The CIMC successfully downloads the ISO image and mounts it as media on the server. The server boots from the media and starts the firmware upgrade process.
8. The server restarts multiple times before finishing with the upgrade process. The CIMC upgrades and disconnects access to any Graphical User Interface (GUI).

Note The sync process waits for the CIMC to come online. It reconnects and re-authenticates with the CIMC.

9. The firmware upgrade process completes and the server reboots. The server boots from the media again and firmware validation beings. If successful the firmware process completes and the server boots. If a failure is identified the firmware upgrade is marked as failed and the sync process fails as well

Note If the validation is successful the firmware upgrade process ends and boots the server. If the validation fails, the failure is identified and the firmware upgrade and sync process is marked as a failure.

10. The CM sync process receives a success answer from the CIMC. The sync process continues to next step.

Updating the UCS Server Firmware

The SMI Cluster Manager allows updating the UCS server firmware remotely or locally in a data center. This section describes the procedures involved in updating the UCS Server firmware using the SMI Cluster Manager.

To update the UCS firmware, use the following configurations:

1. Setup the UCS configuration.

   The following is a sample UCS configuration:

```
software ucs software_version
 url repo_url
 sha512 file_checksum
exit
software ucs software_version
 url repo_url
 sha512 ssh_key
exit
software upf software_version
 url repo_url
 sha256 ssh_key
exit
```

```
environments ucs
 ucs-server
exit
...
 nodes node_name
  ssh-ip ipv4address
  type kvm
  vms node_name
   upf networking management ip ipv4address
  exit
  ucs-server cimc ip-address ipv4address
  ucs-server software software_version
  initial-boot netplan vlans vlan_name
   addresses ipv4address/subnet
  exit
 exit
 nodes node_name
  ssh-ip ipv4address
  type kvm
  vms node_name
   upf networking management ip ipv4address
  exit
  ucs-server cimc ip-address ipv4address
  initial-boot netplan vlans vlan_name
   addresses ipv4address/subnet
  exit
 exit
```

Note Cisco Integrated Management Controller (CIMC) manages the life cycle of all the components - including firmware updates - in UCS C Series servers. The UCS firmware is updated through the CIMC API.

2. Log in to SMI Cluster Manager CLI and enter the configuration mode.
   a. Add the UCS firmware configuration.

Note A sample UCS configuration is provided here.

   b. Commit and exit the configuration.

3. Configure the UCS firmware.
   a. To configure the firmware version for UCS servers inside a cluster, use the following configuration:

```
configure
  node-defaults
    ucs-server software software_version
    exit
```

   b. To configure the firmware version for UCS servers at the node level, use the following configuration:

```
configure
  nodes node_name
    ucs-server software software_version
    exit
```

Note All the UCS firmware configuration parameters are applicable only for UCS based environments. You can add multiple versions of the firmware software under the **software** section in the UCS configuration. It is recommended to have a minimum of two versions (new and old) provisioned. You can download the UCS firmware software from https://software.cisco.com/

4. Trigger the cluster synchronization, using the following command.

   clusters *cluster_name* **actions sync run debug true**

   Example:

   ```
   SMI-CM# clusters test actions sync run debug true
   This will run sync.  Are you sure? [no,yes] yes
   ```

5. Monitor the progress of the cluster synchronization, using the following command.

   monitor sync-logs *cluster_name*

Note

- The cluster synchronization process time depends on the number of UCS servers planned for an upgrade. The average time taken for updating a single UCS server ranges from 30-50 minutes.
- You can select the upgrade strategy manually using the following command:

  clusters *cluster_name* **actions sync run upgrade-strategy { rolling | concurrent }**

NOTES:

- **node-defaults ucs-server software** *software_version* - Specifies the default version of UCS firmware to be applied to all UCS servers inside a cluster.
- **nodes** *node_name* **ucs-server software** *software_version* - Specifies the version of UCS firmware to be applied to all UCS servers at the node level.
- **clusters** *cluster_name* - Specifies the name of the cluster.
- **actions sync run** - Triggers the cluster synchronization
- **upgrade-strategy** - Specifies the upgrade strategy. **rolling** specifies the rolling upgrade strategy. **concurrent** specifies the concurrent upgrade strategy. The default upgrade strategy is *auto*.

Failure Scenarios

The UCS firmware upgrade process may fail for multiple reasons. The following are some of the most common reasons:

1. **Hardware Issues** - Faulty components, power, and cabling issues. It most often requires Return Merchandise Authorization (RMA), component re-seating or visual inspection of the hardware.
2. **Software Failure** - This happens when certain components fail to apply the new software version. These components will skip the upgrade process and fall back to the previous or backup version.
3. **Unsupported Hardware** - The upgrade fails, if the new firmware does not support some of the components.
4. **Software Misconfiguration** - If the inappropriate firmware package is used or an attempt to upgrade to an incorrect or unsupported firmware version, the upgrade process fails.
5. **Timeout** - If the upgrade takes more than 2 hours to complete, the sync process time outs and fails.

Note The CIMC performs a list of prerequisites before proceeding with the firmware installation. When all the prerequisites are successful, the CIMC proceeds with the upgrade. There is no dry-run option available for the firmware upgrade.

Failure Impact

The impact of the firmware upgrade failure falls into the following three categories:

1. **Non-Service Impacting** - The server continues to work. This is the most common behavior. The reason for the failure are software misconfiguration, software failure or a failed prerequisites verification. In such a scenario, it's recommended to troubleshoot the failure, fix the problem and upgrade firmware again either automatically (using the same sync process) or manually. A restart of the CIMC may also reset and unblock the issue.
2. **Partially Service Impacting** – The server continues to work with some performance, connectivity or stability issues. An unsupported hardware, incorrect firmware version, or an issue introduced with the new firmware may have caused this. In such a scenario, it's recommended to perform a BUG scrub or validation to ensure that the OS version, drivers, firmware and hardware combination is officially supported by Cisco. You can perform the validation here:

   https://ucshcltool.cloudapps.cisco.com/public/
3. **Service Impacting** – The server fails to boot or becomes inaccessible. If the firmware upgrade failure affects the CIMC access, then the server either must be sent for a RMA or physical access may be required. A hardware issue or a fatal error (needs further investigation by TAC team) might have caused this.

Note When you are unable to identify the root cause, it is recommended to open a TAC case and involve a UCS expert to help further troubleshoot the issue.

Troubleshooting

If one or multiple servers fail to upgrade the firmware, you must follow these steps to isolate and identify the origin or root cause of the issue:

1. Verify the output of the sync process. It contains the high-level details of the root cause and exact step where of the firmware upgrade failed.

2. You can also use the CLI for the status of the last firmware for a node or multiple nodes in a cluster.
3. Based on the error, the root cause may have caused by either of the following reasons:
   a. An incorrect UCS CIMC password was provided
   b. An incorrect ISO was provided or the ISO was corrupt
   c. The CIMC failed to download the ISO image. A firewall, security-group, or Cluster-Manager (based on the deployment) may have caused this issue
   d. The CIMC failed to trigger, run, or validate the firmware upgrade
4. Access the CIMC server and verify the server console, SEL and alerts or issues for more details (if required).
5. Based on the sync logs and the CIMC data, identify the problem and its resolution. If required, perform a CIMC reboot
6. If you can identify the error and resolve the issue, re-run the upgrade either automatically (through sync process) or manually. You must have the ISO image to perform this operation
7. Open a TAC case and involve an UCS expert, if you are unable to find the root cause or resolve the issue.

SMI Cluster RMA

This section describes procedures involved in replacing a Bare Metal node within a SMI Cluster in the event of node failure or planned maintenance.

The NSO must subscribe to the notification stream **alert notification** to receive the *k8-node-not-ready* alert, when the node(s) become unresponsive due to unplanned node(s) outage, and requires it to be removed from the cluster before putting it in maintenance mode and continue. See the *Notifications* and *Alerts Reference* section for more details on the alert notifications and references.

Important CNDP does not support cluster upgrade and RMA in the same cluster sync.

Prerequisites for RMA Process

For GR deployment, the node-monitor pods starts automatically. During RMA procedure, the node-monitor pod automatically shutdown the rack if multi-compute failure is detected when the node is drain and deleted.

Before starting RMA process, perform the following:

1. Switch the role for both the instance to other rack using `geo switch-role role` command and make sure the target rack for RMA is in STANDBY_ERROR role for both the instances.
2. Disable the node-monitor pod.
   a. Take the backup of daemonsets.

      ```
      kubectl get daemonsets node-monitor -n cn -o yaml > node-monitor.yaml
      ```

   b. Delete node-monitor pods.

      ```
      kubectl delete daemonsets node-monitor -n cn
      ```

3. Continue with RMA procedure. For more information, see the link.
4. Once RMA procedure is complete, check if the node-monitor pods are already spawned.

   ```
   kubectl get pods -n cn -o wide | grep node-monitor
   ```

   If the node-monitor pods have not started, restart them.

   ```
   kubectl create -f node-monitor.yaml
   ```

Note `node-monitor.yaml` file is same as in Step 2.a, on page 21.

5. Correct the role for the instances accordingly.

Note For both earlier and current SMI versions:

- If you are replacing hardware components during an RMA procedure that contain firmware, such as an mLOM card, before adding the repaired or replaced node back to the cluster, you must run the HUU (Host Upgrade Utility) to ensure that the component is compatible with the system before syncing the node back into service.
- As part of RMA, if you remove a node from the cluster and before you return it to the manufacturer, you must purge all data on the device as per instructions provided by the hardware vendor.

Unified RMA Procedure for Planned and Failure Events on Bare Metal

For the SMI version 2020.02.2.41 and later, the RMA (Return Merchandise Authorization) procedure for planned maintenance and any unplanned node failure events is unified for the SMI Bare Metal stacked cluster. The same procedure is applicable for both primary and worker nodes. This feature simplifies the automation and MOP requirements for both NSO and user.

Note For information on the RMA procedures for all previous versions of SMI Bare Metal, refer to the following sections in this guide:

- Control Plane Bare Metal Node Failure - Unplanned
- Control Plane Bare Metal Node Maintenance - Planned

Unified RMA Procedure for the Control Plane and Worker Nodes

This section describes the unified RMA procedure applicable to the following scenarios:

- Replace a working control plane Bare Metal node or worker node for maintenance.
- Replace a failed Bare Metal control plane node or worker node in a stacked cluster.

Notes:

- If you are performing RMA for a control plane node, you must ensure that the majority of control plane nodes are still available during the RMA process.
- Disable **auto-sync** before you perform the RMA procedure.

Use the following steps to replace a working or failed control plane or worker node:

1. Drain and remove the node which is sent for maintenance, using the following command:

   clusters *cluster_name* **nodes***node_name* **actions sync drain remove-node true**

   Example:

   ```
   SMI Cluster Deployer# clusters kali-stacked nodes controlplane1 actions sync drain
   remove-node true
   This will run drain on the node, disrupting pods running on the node.  Are you sure?
   [no,yes] yes
   message accepted
   ```

2. For a planned maintenance scenario, shutdown the node if the node is still running.
3. Assign the node to *maintenance* mode in the cluster configuration using the following CLI commands:

   ```
   config
     clusters cluster_name
        nodes node_name
           maintenance true
        commit
     end
   ```

   Example:

   ```
   SMI Cluster Deployer(config)# clusters kali-stacked
   SMI Cluster Deployer(config-clusters-kali-stacked)# nodes controlplane1
   SMI Cluster Deployer(config-nodes-controlplane1)# maintenance true
   SMI Cluster Deployer(config-nodes-controlplane1)# commit
   Commit complete.
   SMI Cluster Deployer(config-nodes-controlplane1)# end
   SMI Cluster Deployer#
   ```

4. The node is ready for the RMA process.

Note If the remaining nodes must be upgraded or updated, run a cluster sync in this state. However, it's not a part of the RMA process.

5. Add the node back to the cluster when it is repaired or replaced and available.

Note If the remaining nodes have been upgraded to a new SMI release during the time when this node was under maintenance, then it's recommended to clear the boot drive and delete the virtual drive on the node. This step ensures that virtual drive is in a clean state without the previous state before you add it back. However, removal of the virtual drive is not required for a new replacement node.

6. Attach the new Bare Metal node and remove it from the *maintenance* mode in the cluster configuration using the following commands:

```
config
  clusters cluster_name
    nodes node_name
      maintenance false
    commit
  end
```

Example:

```
SMI Cluster Deployer(config)# clusters kali-stacked
SMI Cluster Deployer(config-clusters-kali-stacked)# nodes controlplane1
SMI Cluster Deployer(config-nodes-controlplane1)# maintenance false
SMI Cluster Deployer(config-nodes-controlplane1)# commit
Commit complete.
SMI Cluster Deployer(config-nodes-controlplane1)# end
SMI Cluster Deployer#
```

7. Run the cluster synchronization using the following command:

   clusters *cluster_name* **actions sync run debug true**

   Example:

   ```
   SMI Cluster Deployer# clusters kali-stacked actions sync run debug true
   This will run sync.  Are you sure? [no,yes] yes
   message accepted
   ```

8. Monitor the cluster synchronization using the following command:

   monitor sync-logs *cluster_name*

   Example:

   ```
   SMI Cluster Deployer# monitor sync-logs kali-stacked
   2020-09-30 01:50:02.159 DEBUG cluster_sync.kali-stacked: Cluster name: kali-stacked
   2020-09-30 01:50:02.160 DEBUG cluster_sync.kali-stacked: Force VM Redeploy: false
   2020-09-30 01:50:02.160 DEBUG cluster_sync.kali-stacked: Force partition Redeploy: false

   2020-09-30 01:50:02.160 DEBUG cluster_sync.kali-stacked: reset_k8s_nodes: false
   2020-09-30 01:50:02.160 DEBUG cluster_sync.kali-stacked: purge_data_disks: false
   2020-09-30 01:50:02.160 DEBUG cluster_sync.kali-stacked: upgrade_strategy: auto
   2020-09-30 01:50:02.160 DEBUG cluster_sync.kali-stacked: sync_phase: all
   2020-09-30 01:50:02.160 DEBUG cluster_sync.kali-stacked: debug: true
   .
   .
   .
   .
   2020-09-30 01:53:27.638 DEBUG cluster_sync.kali-stacked: Cluster sync successful
   2020-09-30 01:53:27.638 DEBUG cluster_sync.kali-stacked: Ansible sync done
   2020-09-30 01:53:27.638 INFO cluster_sync.kali-stacked: _sync finished.  Opening lock
   ```

9. To verify the status of the cluster, use the following command:

   clusters*cluster_name* **actions k8s cluster-status**

   Example:

   ```
   pods-desired-count 99
   pods-ready-count 99
   pods-desired-are-ready true
   etcd-healthy true
   all-ok true
   ```

NOTES:

- **clusters** *cluster_name* - Specifies the K8s cluster.
- **nodes** *node_name* - Specifies the control plane Bare Metal node.
- **maintenance** *true/false* - Assigns or removes the primary control plane Bare Metal mode to maintenance mode.
- **actions sync run debug true** - Synchronizes the cluster configuration.
- **actions k8s cluster-status** - Displays the status of the cluster.

SMI Bare Metal Stacked Cluster

In SMI Bare Metal deployment, a stacked cluster is defined as a cluster containing three Bare Metal control plane nodes with a primary control plane, OAM, and other applications hosted on the same three nodes. A healthy stacked cluster needs at least two working control plane nodes to perform all the functions. When the two control plane nodes fail, the entire cluster fails and you must redeploy the entire cluster. You can run the K8s cluster synchronization from the current Active node of the Cluster Manager HA.

The subsequent sections provide information about handling the stacked cluster in the event of a Bare Metal node failure (unplanned) or planned maintenance.

Control Plane Baremetal Node Failure - Unplanned

This section describes the procedure to replace a failed primary Bare Metal control plane 1 node in a stacked cluster. When the primary control plane 1 Bare Metal node fails, the node status changes to *NotReady*. You can use the following command to view the status of the nodes in the cluster:

`kubectl get nodes`

In the following example, the status of the primary control plane 1 node changes to *NotReady* after it fails.

```
user1-cloud@kali-stacked-control-plane:~$ kubectl get nodes
NAME                          STATUS     ROLES           AGE    VERSION
kali-stacked-control-plane1   NotReady   control-plane   136m   v1.21.0
kali-stacked-control-plane2   Ready      control-plane   10h    v1.21.0
kali-stacked-control-plane3   Ready      control-plane   10h    v1.21.0
```

All the pods in the failed primary control plane 1 Bare Metal node remains either terminated or in pending state. You verify the status of the pods using the **kubectl get pods** command as shown in the following example:

```
user1-cloud@kali-stacked-controlplane3:~$ kubectl get pods -A
NAMESPACE           NAME                                                READY   STATUS
RESTARTS   AGE
kube-system         calico-kube-controllers-5d7fff4bc6-lxkpc            1/1     Running   0
        7h26m
kube-system         calico-node-tx7zg                                   1/1     Running   0
        10h
kube-system         calico-node-v6m7v                                   1/1     Running   0
        10h
kube-system         coredns-66d57f55d9-6dnsn                            1/1     Running   0
        136m
kube-system         coredns-66d57f55d9-rdtbd                            1/1     Running   0
        136m
kube-system         etcd-kali-stacked-controlplane2                     1/1     Running   0
        10h
kube-system         etcd-kali-stacked-controlplane3                     1/1     Running   0
        10h
kube-system         kube-apiserver-kali-stacked-controlplane2           1/1     Running   0
        10h
```

```
kube-system         kube-apiserver-kali-stacked-controlplane3        1/1     Running   0
        10h
```

To replace the failed primary control plane 1 Bare Metal node:

1. Delete the failed primary control plane 1 Bare Metal node using the following command:

   kubectl delete node *node_name*

   Example:

   ```
   user1-cloud@kali-stacked-controlplane3:~$ kubectl delete node kali-stacked-controlplane1
   node "kali-stacked-controlplane1" deleted
   ```

2. Assign the primary control plane 1 Bare Metal node to *maintenance* mode in the cluster configuration using the following commands:

   ```
   configure
     clusters cluster_name
     nodes controlplane1
     maintenance true
     commit
     end
   ```

   Example:

   ```
   SMI Cluster Deployer(config)# clusters kali-stacked
   SMI Cluster Deployer(config-clusters-kali-stacked)# nodes controlplane1
   SMI Cluster Deployer(config-nodes-controlplane1)# maintenance true
   SMI Cluster Deployer(config-nodes-controlplane1)# commit
   Commit complete.
   SMI Cluster Deployer(config-nodes-controlplane1)# end
   SMI Cluster Deployer#
   ```

3. The node is ready for the RMA process.

Note If the remaining nodes need to be upgraded or NFs need to be synchronized, run a cluster sync in this state. However, it's not a part of the RMA process.

4. Add the node back to the cluster when it is repaired or replaced and available.

Note If you add a node after it's repaired, ensure that the disks are clean by clearing the boot drive and virtual drive on the node. This step is to ensure that the virtual drive is in a clean state without the previous state before you add it back. However, removal of the virtual drive is not required for a new replacement node.

5. Attach the new primary control plane 1 Bare Metal node and remove it from the *maintenance* mode in the cluster configuration using the following commands:

   ```
   configure
     clusters cluster_name
     nodes controlplane1
     maintenance false
     commit
     end
   ```

Example:

```
SMI Cluster Deployer(config)# clusters kali-stacked
SMI Cluster Deployer(config-clusters-kali-stacked)# nodes controlplane1
SMI Cluster Deployer(config-nodes-controlplane1)# maintenance false
SMI Cluster Deployer(config-nodes-controlplane1)# commit
Commit complete.
SMI Cluster Deployer(config-nodes-controlplane1)# end
SMI Cluster Deployer#
```

6. Run the cluster synchronization using the following command:

   clusters *cluster_name* **actions sync run debug true**

   Example:

   ```
   SMI Cluster Deployer# clusters kali-stacked actions sync run debug true
   This will run sync.  Are you sure? [no,yes] yes
   message accepted
   ```

7. Verify the status of the cluster using the following command:

   clusters *cluster_name* **actions k8s cluster-status**

   Example:

   ```
   SMI Cluster Deployer# clusters kali-stacked actions k8s cluster-status
   pods-desired-count 40
   pods-ready-count 39
   pods-desired-are-ready true
   etcd-healthy true
   all-ok true
   ```

NOTES:

- **clusters** *cluster_name* - Specifies the K8s cluster.
- **nodes controlplane1** - Specifies primary control plane 1 Bare Metal node.
- **maintenance** *true/false* - Assigns or removes the primary control plane 1 Bare Metal mode to maintenance mode
- **actions sync run debug true** - Synchronizes the cluster configuration.
- **actions k8s cluster-status** - Displays the status of the cluster.

Control Plane Bare Metal Node Maintenance - Planned

This section describes the procedures involved in replacing a working primary control plane Bare Metal node for maintenance. To replace a primary control plane Bare Metal node for maintenance, use the following:

1. Drain and remove the control plane node which is sent for maintenance, using the following command:

   clusters *cluster_name* **nodes controlplane1 actions sync drain remove-node true**

   Example:

   ```
   SMI Cluster Deployer# clusters kali-stacked nodes controlplane1 actions sync drain
   remove-node true
   This will run drain on the node, disrupting pods running on the node.  Are you sure?
   [no,yes] yes
   message accepted
   ```

2. Verify the status of the control plane node using the following command:

   clusters *cluster_name* **nodes controlplane1 actions sync logs**

   Example:

```
SMI Cluster Deployer# clusters kali-stacked nodes controlplane1 actions sync logs
logs 2020-09-30 01:46:14.498 DEBUG cluster_sync.kali-stacked.controlplane1: Cluster name:
 kali-stacked
2020-09-30 01:46:14.498 DEBUG cluster_sync.kali-stacked.controlplane1: Node name:
controlplane1
2020-09-30 01:46:14.499 DEBUG cluster_sync.kali-stacked.controlplane1: debug: false
2020-09-30 01:46:14.499 DEBUG cluster_sync.kali-stacked.controlplane1: remove_node: true
.
.
.
.
2020-09-30 01:46:53.028 DEBUG cluster_sync.kali-stacked.controlplane1: Cluster sync
successful
2020-09-30 01:46:53.029 DEBUG cluster_sync.kali-stacked.controlplane1: Ansible sync done

2020-09-30 01:46:53.029 INFO cluster_sync.kali-stacked.controlplane1: _sync finished.
Opening lock
```

3. Shutdown the node.

4. Assign the primary control plane Bare Metal node to *maintenance* mode in the cluster configuration using the following commands:

   configure
   clusters *cluster_name*
   nodes controlplane1
   maintenance true
   commit
   end

   Example:

```
SMI Cluster Deployer(config)# clusters kali-stacked
SMI Cluster Deployer(config-clusters-kali-stacked)# nodes controlplane1
SMI Cluster Deployer(config-nodes-controlplane1)# maintenance true
SMI Cluster Deployer(config-nodes-controlplane1)# commit
Commit complete.
SMI Cluster Deployer(config-nodes-controlplane1)# end
SMI Cluster Deployer#
```

5. The node is ready for the RMA process.

Note If the remaining nodes need to be upgraded or NFs need to be synchronized, run a cluster sync in this state. However, it's not a part of the RMA process.

6. Add the node back to the cluster when it is repaired or replaced and available.

Note If you add a node after it's repaired, ensure that the disks are clean by clearing the boot drive and virtual drive on the node. This step is to ensure that the virtual drive is in a clean state without the previous state before you add it back. However, removal of the virtual drive is not required for a new replacement node.

7. Attach the new primary control plane Bare Metal node and remove it from the *maintenance* mode in the cluster configuration using the following commands:

```
configure
  clusters cluster_name
  nodes controlplane1
  maintenance false
  commit
  end
```

Example:

```
SMI Cluster Deployer(config)# clusters kali-stacked
SMI Cluster Deployer(config-clusters-kali-stacked)# nodes controlplane1
SMI Cluster Deployer(config-nodes-controlplane1)# maintenance false
SMI Cluster Deployer(config-nodes-controlplane1)# commit
Commit complete.
SMI Cluster Deployer(config-nodes-controlplane1)# end
SMI Cluster Deployer#
```

8. Run the cluster synchronization using the following command:

```
clusters cluster_name actions sync run debug true
```

Example:

```
SMI Cluster Deployer# clusters kali-stacked actions sync run debug true
This will run sync.  Are you sure? [no,yes] yes
message accepted
```

NOTES:

- **clusters** *cluster_name* - Specifies the K8s cluster.
- **nodes controlplane1** - Specifies primary control plane 1 Bare Metal node.
- **maintenance** *true/false* - Assigns or removes the primary control plane 1 Bare Metal mode to maintenance mode
- **actions sync run debug true** - Synchronizes the cluster configuration.
- **actions k8s cluster-status** - Displays the status of the cluster.

Stacked Cluster with Worker Nodes

For scaling the NFs, you can add a number of worker nodes along with the stacked control plane nodes. After adding the worker nodes to the stacked control plane nodes, you can distribute the NFs to the worker nodes.

This section describes procedures involved in replacing a Bare Metal worker node within a SMI Cluster in the event of node failure or planned maintenance.

Worker Bare Metal Node Failure - Unplanned

This section describes the procedure to replace a failed primary Bare Metal worker node in a stacked cluster. When the primary worker Bare Metal node fails, the node status changes to *NotReady*.

Verify the status of the nodes in the cluster using the following command:

```
kubectl get nodes
```

In the following example, the status of the primary worker node changes to *NotReady* after it fails.

```
user1-cloud@kali-stacked-controlplane1:~$ kubectl get nodes
NAME                          STATUS     ROLES           AGE    VERSION
kali-stacked-cmts-worker1     NotReady   <none>          38m    v1.21.0
kali-stacked-cmts-worker2     Ready      <none>          38m    v1.21.0
kali-stacked-cmts-worker3     Ready      <none>          38m    v1.21.0
kali-stacked-controlplane1    Ready      control-plane   125m   v1.21.0
kali-stacked-controlplane2    Ready      control-plane   13h    v1.21.0
kali-stacked-controlplane3    Ready      control-plane   13h    v1.21.0
```

To replace the failed primary Bare Metal worker node in the Cluster:

1. Delete the failed primary worker Bare Metal node using the following command:

   kubectl delete node *node_name*

   Example:

   ```
   user1-cloud@kali-stacked-controlplane3:~$ kubectl delete node kali-stacked-cmts-worker1
   node "kali-stacked-cmts-worker1" deleted
   ```

2. Assign the primary worker Bare Metal node to *maintenance* mode in the cluster configuration using the following commands:

   ```
   configure
     clusters cluster_name
     nodes worker_node
     maintenance true
     commit
     end
   ```

   Example:

   ```
   SMI Cluster Deployer(config)# clusters kali-stacked
   SMI Cluster Deployer(config-clusters-kali-stacked)# nodes cmts-worker1
   SMI Cluster Deployer(config-nodes-cmts-worker1)# maintenance true
   SMI Cluster Deployer(config-nodes-cmts-worker1)# commit
   Commit complete.
   SMI Cluster Deployer(config-nodes-cmts-worker1)# end
   ```

3. The node is ready for the RMA process.

Note If the remaining nodes need to be upgraded or NFs need to be synchronized, run a cluster sync in this state. However, it's not a part of the RMA process.

4. Add the node back to the cluster when it is repaired or replaced and available.

Note If you add a node after it's repaired, ensure that the disks are clean by clearing the boot drive and virtual drive on the node. This step is to ensure that the virtual drive is in a clean state without the previous state before you add it back. However, removal of the virtual drive is not required for a new replacement node.

5. Attach the new primary worker Bare Metal node and remove it from the *maintenance* mode in the cluster configuration using the following commands:

   ```
   configure
     clusters cluster_name
     nodes worker_node
   ```

```
maintenance false
commit
end
```

Example:

```
SMI Cluster Deployer(config)# clusters kali-stacked
SMI Cluster Deployer(config-clusters-kali-stacked)# nodes cmts-worker1
SMI Cluster Deployer(config-nodes-cmts-worker1)# maintenance false
SMI Cluster Deployer(config-nodes-cmts-worker1)# commit
Commit complete.
SMI Cluster Deployer(config-nodes-cmts-worker1)# end
SMI Cluster Deployer#
```

6. Run the cluster synchronization using the following command:

   clusters *cluster_name* **actions sync run debug true**

   Example:

   ```
   SMI Cluster Deployer# clusters kali-stacked actions sync run debug true
   This will run sync.  Are you sure? [no,yes] yes
   message accepted
   ```

7. Verify the status of the cluster using the following command:

   clusters *cluster_name* **actions k8s cluster-status**

   Example:

   ```
   SMI Cluster Deployer# clusters kali-stacked actions k8s cluster-status
   pods-desired-count 67
   pods-ready-count 67
   pods-desired-are-ready true
   etcd-healthy true
   all-ok true
   ```

8. Verify the status of the pods redeployed on the added worker node using the following command:

   clusters *cluster_name* **nodes worker_node actions k8s pod-status show-pod-details**
   Value for 'show-pod-details' [false,true]: true

   Example:

   ```
   SMI Cluster Deployer# clusters kali-stacked nodes cmts-worker1 actions k8s pod-status
   show-pod-details
   Value for 'show-pod-details' [false,true]: true
   pods {
       name calico-node-67gs6
       namespace kube-system
       owner-kind DaemonSet
       owner-name calico-node
       ready true
   }
   pods {
       name coredns-f9fd979d6-b2gsb
       namespace kube-system
       owner-kind ReplicaSet
       owner-name coredns-f9fd979d6
       ready true
   }
   pods {
       name kube-proxy-5m9qh
       namespace kube-system
   ```

```
    owner-kind DaemonSet
    owner-name kube-proxy
    ready true
}
pods {
    name maintainer-2nxlq
    namespace kube-system
    owner-kind DaemonSet
    owner-name maintainer
    ready true
}
pods {
    name charts-cee-2020-02-0-i21-4
    namespace registry
    owner-kind StatefulSet
    owner-name charts-cee-2020-02-0-i21
    ready true
}
pods {
    name charts-cluster-deployer-2020-02-0-i22-5
    namespace registry
    owner-kind StatefulSet
    owner-name charts-cluster-deployer-2020-02-0-i22
    ready true
}
pods {
    name registry-cee-2020-02-0-i21-5
    namespace registry
    owner-kind StatefulSet
    owner-name registry-cee-2020-02-0-i21
    ready true
}
pods {
    name registry-cluster-deployer-2020-02-0-i22-5
    namespace registry
    owner-kind StatefulSet
    owner-name registry-cluster-deployer-2020-02-0-i22
    ready true
}
pods {
    name software-unpacker-3
    namespace registry
    owner-kind StatefulSet
    owner-name software-unpacker
    ready true
}
pods {
    name keepalived-jrj4g
    namespace smi-vips
    owner-kind DaemonSet
    owner-name keepalived
    ready true
}
pods-count 10
pods-available-to-drain-count 6
```

Note You can follow the same procedure to replace one or more failed worker nodes in the cluster.

NOTES:

- **clusters** *cluster_name* - Specifies the K8s cluster.
- **nodes worker** - Specifies primary worker Bare Metal node.
- **maintenance** *true/false* - Assigns or removes the primary control plane 1 Bare Metal mode to maintenance mode
- **actions sync run debug true** - Synchronizes the cluster configuration.
- **actions k8s cluster-status** - Displays the status of the cluster.

Worker Bare Metal Node Maintenance - Planned

This section describes the procedures involved in replacing a working primary worker Bare Metal node for maintenance. To replace a primary worker Bare Metal node for maintenance, use the following:

1. Drain and remove the worker node which is sent for maintenance, using the following command:

   clusters *cluster_name* **nodes** *worker_node* actions sync drain remove-node true

   Example:

   ```
   [installer-controlplane] SMI Cluster Deployer# clusters kali-stacked nodes cmts-worker1
    actions sync drain remove-node true
   This will run drain on the node, disrupting pods running on the node.  Are you sure?
   [no,yes] yes
   message accepted
   ```

2. Verify the status of the worker node using the following command:

   clusters *cluster_name* **nodes** *worker_node* **actions sync logs**

   Example:

   ```
   [installer-controlplane] SMI Cluster Deployer# clusters kali-stacked nodes cmts-worker1
    actions sync logs
   logs 2020-10-06 20:01:48.023 DEBUG cluster_sync.kali-stacked.cmts-worker1: Cluster name:
    kali-stacked
   2020-10-06 20:01:48.024 DEBUG cluster_sync.kali-stacked.cmts-worker1: Node name:
   cmts-worker1
   2020-10-06 20:01:48.024 DEBUG cluster_sync.kali-stacked.cmts-worker1: debug: false
   2020-10-06 20:01:48.024 DEBUG cluster_sync.kali-stacked.cmts-worker1: remove_node: true

   .
   .
   .
   .
   2020-10-06 20:02:30.057 DEBUG cluster_sync.kali-stacked.cmts-worker1: Cluster sync
   successful
   2020-10-06 20:02:30.058 DEBUG cluster_sync.kali-stacked.cmts-worker1: Ansible sync done

   2020-10-06 20:02:30.058 INFO cluster_sync.kali-stacked.cmts-worker1: _sync finished.
   Opening lock
   ```

3. Shutdown the node.
4. Assign the primary control plane Bare Metal node to *maintenance* mode in the cluster configuration using the following commands:

   configure
   clusters *cluster_name*
   nodes *worker_node*
   maintenance true

```
commit
end
```

Example:

```
[installer-controlplane] SMI Cluster Deployer# config
Entering configuration mode terminal
[installer-controlplane] SMI Cluster Deployer(config)# clusters kali-stacked
[installer-controlplane] SMI Cluster Deployer(config-clusters-kali-stacked)# nodes
cmts-worker1
[installer-controlplane] SMI Cluster Deployer(config-nodes-cmts-worker1)# maintenance
true
[installer-controlplane] SMI Cluster Deployer(config-nodes-cmts-worker1)# commit
Commit complete.
[installer-controlplane] SMI Cluster Deployer(config-nodes-cmts-worker1)# end
```

5. The node is ready for the RMA process.

Note If the remaining nodes need to be upgraded or NFs need to be synchronized, run a cluster sync in this state. However, it's not a part of the RMA process.

6. Add the node back to the cluster when it is repaired or replaced and available.

Note If you add a node after it's repaired, ensure that the disks are clean by clearing the boot drive and virtual drive on the node. This step is to ensure that the virtual drive is in a clean state without the previous state before you add it back. However, removal of the virtual drive is not required for a new replacement node.

7. Attach the new primary worker Bare Metal node and remove it from the *maintenance* mode in the cluster configuration using the following commands:

```
configure
  clusters cluster_name
  nodes worker_node
  maintenance false
  commit
  end
```

Example:

```
SMI Cluster Deployer# config
Entering configuration mode terminal
[installer-controlplane] SMI Cluster Deployer(config)# clusters kali-stacked
[installer-controlplane] SMI Cluster Deployer(config-clusters-kali-stacked)# nodes
cmts-worker1
[installer-controlplane] SMI Cluster Deployer(config-nodes-cmts-worker1)# maintenance
false
[installer-controlplane] SMI Cluster Deployer(config-nodes-cmts-worker1)# commit
Commit complete.
[installer-controlplane] SMI Cluster Deployer(config-nodes-cmts-worker1)# end
```

8. Run the cluster synchronization using the following command:

```
clusters cluster_name actions sync run debug true
```

Example:

```
SMI Cluster Deployer# clusters kali-stacked actions sync run debug true
This will run sync.  Are you sure? [no,yes] yes
message accepted
```

NOTES:

- **clusters** *cluster_name* - Specifies the K8s cluster.
- **nodes worker** - Specifies primary worker Bare Metal node.
- **maintenance** *true/false* - Assigns or removes the primary control plane 1 Bare Metal mode to maintenance mode
- **actions sync run debug true** - Synchronizes the cluster configuration.
- **actions k8s cluster-status** - Displays the status of the cluster.

KVM Bare Metal Node Failure

For KVM Bare Metal node failures, you must replace the failed or faulty KVM Bare Metal node and run a cluster synchronization. When the cluster synchronizes successfully, the SMI Cluster Manager retains its database for Day 0 configuration. However, for Day 1 configuration, the SMI Cluster Manager retains its database from the Network Services Orchestrator (NSO).

Note The SMI does not support any other data backup or restore methods currently.

Node Failures in SMI Cluster Manager HA

Unlike a typical a Kubernetes cluster, the SMI Bare Metal HA consists of two AIO Bare Metal nodes running independently but synchronized in data using an underlying Distributed Replicated Block Device (DRBD) driver. The data synchronization happens in a specific manner with restrictions. Also, the DRBD utilizes the master keepalived for switchovers. You can run the synchronization for the SMI Cluster Manager HA from the Inception Server.

The subsequent sections provide more information about replacing failed nodes in SMI Bare Metal Cluster Manager HA mode.

Active Bare Metal Node Failure - Unplanned

When an Active node fails in a Cluster Manager HA model, the Standby node becomes the Active node and continues to upgrade the remote clusters or the NFs. To replace the failed Active node:

1. Remove the Active node.

Note Upgrade or synchronize the remote clusters using the Standby node.

For more information on upgrading the SMI Cluster Manager in HA mode, see *Upgrading SMI Cluster Manager in HA* section in *UCC SMI Cluster Manager Deployment Guide*.

2. Update the Cluster Manager HA configuration to swap the backup and control-plane node types when the Active node is up and running.

```
nodes active
  k8s node-type backup #originally this would be control-plane type in the fresh
deployment
 exit
 nodes standby
  k8s node-type control-plane #originally this would be backup type in fresh
deployment
exit
```

3. Run the cluster synchronization.

   clusters *cluster_name* **actions sync run debug true**

   Example:

   ```
   SMI Cluster Deployer# clusters trysjc-ha actions sync run debug true
   This will run sync.  Are you sure? [no,yes] yes
   message accepted
   ```

4. Verify the status of the cluster and pods when the Active and Standby nodes are up and running.

   clusters *cluster_name* **actions k8s cluster-status**

   kubectl get pods -n *namespace*

5. Log in to the Ops Center service and verify whether the cluster configuration and CEE data are retained.

6. Once the active node is back from maintenance, update the Cluster Manager HA configuration to swap the backup and control-plane node types.

   ```
   nodes active
    k8s node-type control-plane #originally this would be control-plane type in the
   fresh deployment
   exit
   nodes standby
    k8s node-type backup #originally this would be backup type in fresh deployment
   exit
   ```

NOTES:

- **clusters** *cluster_name* - Specifies the K8s cluster.
- **actions sync run debug true** - Synchronizes the cluster configuration.
- **actions k8s cluster-status** - Displays the status of the cluster.

Active Bare Metal Maintenance - Planned

This section describes the procedures involved in replacing an Active node for maintenance. To replace a an Active node for maintenance, use the following procedure:

1. Shutdown the Active node.

Note Upgrade or synchronize the remote clusters using the Standby node.

For more information on upgrading the SMI Cluster Manager in HA mode, see *Upgrading SMI Cluster Manager in HA* section in *UCC SMI Cluster Manager Deployment Guide*.

2. Update the Cluster Manager HA configuration to swap the backup and control plane node types when the Active node is up and running.

```
nodes active
  k8s node-type backup #originally this would be control-plane type in the fresh
deployment
 exit
 nodes standby
  k8s node-type control-plane #originally this would be backup type in fresh
deployment
exit
```

3. Run the cluster synchronization.

clusters *cluster_name* **actions sync run debug true**

Example:

```
SMI Cluster Deployer# clusters trysjc-ha actions sync run debug true
This will run sync.  Are you sure? [no,yes] yes
message accepted
```

4. Verify the status of the cluster and pods when the Active and Standby nodes are up and running.

clusters *cluster_name* **actions k8s cluster-status**

kubectl get pods -n *namespace*

5. Log in to the Ops Center service and verify whether the cluster configuration and CEE data are retained.

NOTES:

- **clusters** *cluster_name* - Specifies the K8s cluster.
- **actions sync run debug true** - Synchronizes the cluster configuration.
- **actions k8s cluster-status** - Displays the status of the cluster.

Standby Bare Metal Node Failure or Maintenance

This section describes the procedures involved in replacing a Standby node which has either failed (unplanned) or removed for maintenance (planned). To replace a Standby node, use the following:

1. Shutdown the Standby node.

Note Upgrade or synchronize the remote clusters using the Standby node.

For more information on upgrading the SMI Cluster Manager in HA mode, see *Upgrading SMI Cluster Manager in HA* section

in *UCC SMI Cluster Manager Deployment Guide*.

2. Add the Standby node to the same HA configuration when the Standby node is up and running.
3. Run the cluster synchronization.

clusters *cluster_name* **actions sync run debug true**

Example:

```
SMI Cluster Deployer# clusters trysjc-ha actions sync run debug true
This will run sync.  Are you sure? [no,yes] yes
message accepted
```

4. Verify the status of the cluster and pods when the Active and Standby nodes are up and running.

   clusters *cluster_name* **actions k8s cluster-status**

   kubectl get pods -n *namespace*

5. Log in to the Ops Center service and verify whether the cluster configuration and CEE data are retained.

NOTES:

- **clusters** *cluster_name* - Specifies the K8s cluster.
- **actions sync run debug true** - Synchronizes the cluster configuration.
- **actions k8s cluster-status** - Displays the status of the cluster.

SMI Cluster Manager AIO Node Failure or Planned Maintenance

When the SMI Cluster Manager is deployed in All-In-One (AIO) mode, the CEE replicates the data across all the three nodes (*control plane*, *worker* and *etcd*).

Also, the SMI Cluster Manager AIO model does not support the back up and recovery of operational data. Therefore, you must ensure the following to prevent the loss of monitoring data when a double node or triple node failure occurs:

1. Back up the CEE configuration for the Ops Center using the standard Ops Center backup and restore functionality.
2. Back up the Cluster Manager configuration for the Ops Center (remote cluster configuration) using the standard Ops Center backup and restore functionality.
3. For planned maintenance or unplanned outage in the SMI AIO mode, you can perform a cluster synchronization and restore the backed-up configuration.
4. Export the metrics (KPIs) from bulk stats regularly.
5. Gather Alerts in an external system.

Disaster Recovery

This section describes the procedure to redeploy the SMI Cluster Manager HA, when both the Active and Standby nodes fail or when the entire site or cluster is down.

Active and Standby Node Failure - Unplanned

When both the Active and Standby node fails in a SMI Cluster Manager HA model, you can redeploy the SMI Cluster Manager in HA mode using the Inception Server. To redeploy the SMI Cluster Manager HA using the Inception Server, use the following procedure:

1. Log in to the Inception Server CLI and perform a cluster synchronization.

   clusters *cluster_name* **actions sync run debug true**

   Example:

```
SMI Cluster Deployer# clusters trysjc-ha actions sync run debug true
This will run sync.  Are you sure? [no,yes] yes
message accepted
```

Note In the event of an unplanned outage, you cannot restore the data since the SMI does not backup data periodically in the background. However, in the event of planned maintenance, you can restore the data from the storage node manually. You can restore the data using the *scp* command on the storage node:

scp -i *../test_key2.pem* **-r host_name@<ipv4address>:/data**

2. Copy the backed up data from the storage node and verify whether the NFs, CM and CEE configurations are present.

Note Restore the backed up data only when the Active and Standby nodes and all the pods within these nodes are up and running.

NOTES:

- **clusters** *cluster_name* - Specifies the K8s cluster.
- **actions sync run debug true** - Synchronizes the cluster configuration.

Restoring an Entire Site or Cluster

When an entire site or cluster is down, you can redeploy the SMI Cluster Manager HA with NFs, CEE and remote cluster configurations using the Inception Server. To restore an entire site or cluster, use the following:

1. Redeploy the SMI Cluster Manager HA through Inception server. For more information, see *Active and Standby Node Failure - Unplanned* section.
2. Verify whether the NFs, CM and CEE configurations are present.

CEE Data Replication

The CEE replicates the data across all the three nodes (*control plane*, *worker,* and *etcd*). Also, the SMI Cluster Manager AIO model does not support the back up and recovery of operational data. Therefore, you must ensure the following to prevent the loss of monitoring data when a double node or triple node failure occurs:

1. Back up the CEE configuration for the Ops Center using the standard Ops Center backup and restore functionality.
2. Back up the Cluster Manager configuration for the Ops Center (remote cluster configuration) using the standard Ops Center backup and restore functionality.
3. For planned maintenance or unplanned outage in the SMI AIO mode, you can perform a cluster synchronization and restore the backed-up configuration.
4. Export the metrics (KPIs) from bulk stats regularly.
5. Gather Alerts in an external system.

Notifications

The SMI comes equipped with different notification streams to verify the event status (cluster or node synchronization) or synchronization state of the current cluster or nodes in it.

The show notification stream provides notification details for alert notifications, license status and system status.

show notification stream ?

```
Possible completions:
  alert-notification   Notifications for alerts
  license-status       Notifications for smart agent licensing will be generated here
  system-status        Notifications for system status
```

The NSO must subscribe to the **alert-notification** notification stream to receive alerts.

You can access the notification stream using the following command *show* command in the SMI Cluster Manager CLI:

show notification stream *{node-state | node-status | sync-state | sync-status}*

Sync-State Notification

You can access the synchronization state of a given cluster using the **sync-state** notification. After every cluster synchronization, a **sync-state** notification is registered as sent.

The following example displays the deployment of a stacked cluster:

```
SMI Cluster Deployer# show notification stream sync-state
notification
 eventTime 2020-10-07T18:58:27.368+00:00
 sync-state-notification
  cluster <stacked_cluster_name>
  state DEPLOYED
 !
!
notification
 eventTime 2020-10-07T19:07:56.114+00:00
 sync-state-notification
  cluster <stacked_cluster_name>
  state DEPLOYED
 !
!
notification
 eventTime 2020-10-07T19:46:54.531+00:00
 sync-state-notification
  cluster <stacked_cluster_name>
  state DEPLOYED
 !
!
notification
 eventTime 2020-10-08T15:59:56.697+00:00
 sync-state-notification
  cluster <stacked_cluster_name>
  state DEPLOYED
 !
!
The node-state-notification could be used to assess the state of the node in a cluster
deloyment sync process. Below example shows the state of control plane and worker nodes are
 joined in <stacked_cluster_name> cluster. Before reaching JOINED state if the join process
 is still running they would show up as JOINING.
[installer-controlplane] SMI Cluster Deployer# show notification stream node-state
```

```
notification
 eventTime 2020-10-07T19:46:54.531+00:00
 node-state-notification
  cluster <stacked_cluster_name>
  node cmts-worker3
  state JOINED
 !
!
notification
 eventTime 2020-10-07T19:46:54.532+00:00
 node-state-notification
  cluster <stacked_cluster_name>
  node controlplane2
  state JOINED
 !
!
notification
 eventTime 2020-10-07T19:46:54.534+00:00
 node-state-notification
  cluster <stacked_cluster_name>
  node cmts-worker2
  state JOINED
 !
!
notification
 eventTime 2020-10-07T19:46:54.535+00:00
 node-state-notification
  cluster <stacked_cluster_name>
  node controlplane3
  state JOINED
 !
!
notification
 eventTime 2020-10-07T19:46:54.535+00:00
 node-state-notification
  cluster <stacked_cluster_name>
  node controlplane1
  state JOINED
 !
!
notification
 eventTime 2020-10-07T19:46:54.54+00:00
 node-state-notification
  cluster <stacked_cluster_name>
  node cmts-worker1
  state JOINED
 !
!
```

Node-State Notification

You can view the node state in a cluster deployment synchronization process using the **node-state** notification.

The following example displays the state of control plane and worker nodes that have joined the stacked cluster. The node state is displayed as *JOINING*, if the process is running. Otherwise, the node state is displayed as *JOINED*.

```
SMI Cluster Deployer# show notification stream node-state
notification
 eventTime 2020-10-07T19:46:54.531+00:00
 node-state-notification
  cluster <stacked_cluster_name>
```

```
  node cmts-worker3
  state JOINED
 !
!
notification
 eventTime 2020-10-07T19:46:54.532+00:00
 node-state-notification
  cluster <stacked_cluster_name>
  node controlplane2
  state JOINED
 !
!
notification
 eventTime 2020-10-07T19:46:54.534+00:00
 node-state-notification
  cluster <stacked_cluster_name>
  node cmts-worker2
  state JOINED
 !
!
notification
 eventTime 2020-10-07T19:46:54.535+00:00
 node-state-notification
  cluster <stacked_cluster_name>
  node controlplane3
  state JOINED
 !
!
notification
 eventTime 2020-10-07T19:46:54.535+00:00
 node-state-notification
  cluster <stacked_cluster_name>
  node controlplane1
  state JOINED
 !
!
notification
 eventTime 2020-10-07T19:46:54.54+00:00
 node-state-notification
  cluster <stacked_cluster_name>
  node cmts-worker1
  state JOINED
 !
!
```

Sync-Status Notification

You view the progression of the cluster synchronization process using the **sync-status** notification.

The following example displays the different states (SYNC-START, ERROR and DONE) of the cluster synchronization process:

```
SMI Cluster Deployer# show notification stream sync-status
notification
 eventTime 2020-10-08T16:05:53.326+00:00
 sync-status-notification
  cluster <stacked_cluster_name>
  status ERROR
 !
!
notification
 eventTime 2020-10-08T16:12:48.042+00:00
 sync-status-notification
  cluster <stacked_cluster_name>
```

```
  status SYNC-START
 !
!
notification
 eventTime 2020-10-08T16:26:23.444+00:00
 sync-status-notification
  cluster <stacked_cluster_name>
  status ERROR
 !
!
notification
 eventTime 2020-10-08T16:26:23.446+00:00
 sync-status-notification
  cluster <stacked_cluster_name>
  status ERROR
 !
!
notification
 eventTime 2020-10-08T16:31:42.551+00:00
 sync-status-notification
  cluster <stacked_cluster_name>
  status SYNC-START
 !
!
notification
 eventTime 2020-10-08T16:43:33.498+00:00
 sync-status-notification
  cluster <stacked_cluster_name>
  status DONE
 !
!
notification
 eventTime 2020-10-08T20:04:21.386+00:00
 sync-status-notification
  cluster kali-stacked
  status SYNC-START
 !
!
notification
 eventTime 2020-10-08T20:04:21.386+00:00
 sync-status-notification
  cluster kali-stacked
  status SYNC-START
 !
!
notification
 eventTime 2020-10-08T20:27:32.155+00:00
 sync-status-notification
  cluster kali-stacked
  status DONE
 !
!
```

Node-Status Notification

You can view the status of the node synchronization using the **node-status** notification. The following example displays the status of the nodes in a stacked cluster:

```
SMI Cluster Deployer# show notification stream node-status
notification
 eventTime 2020-09-25T19:59:56.728+00:00
 node-status-notification
cluster kali-vm3
node controlplane
```

```
status DONE
 !
!
notification
 eventTime 2020-09-29T16:36:05.962+00:00
 node-status-notification
cluster kali
node etcd1
status DONE
 !
!
notification
 eventTime 2020-09-29T20:54:18.415+00:00
 node-status-notification
cluster kali-stacked
node controlplane1
status DONE
 !
!
notification
 eventTime 2020-09-30T01:28:57.355+00:00
 node-status-notification
cluster kali-stacked
node controlplane1
status DONE
 !
!
notification
 eventTime 2020-09-30T01:46:53.034+00:00
 node-status-notification
cluster kali-stacked
node controlplane1
status DONE
 !
!
notification
 eventTime 2020-09-30T04:17:25.701+00:00
 node-status-notification
cluster kali-stacked
node controlplane1
status ERROR
 !
!
notification
 eventTime 2020-10-06T20:02:30.108+00:00
 node-status-notification
cluster kali-stacked
node cmts-worker1
status DONE
 !
!
```

VM Status Alerts

The NSO needs to know the state of the NF (Network Function) that has been deployed and notifications for the following states:

- DEPLOYED
- ALIVE
- UNDEPLOYED

- ERROR
- RECOVERING
- RECOVERY_FAILED

Note The SMI supports VM status notification for UPF only.

The following parameters are introduced in the notification streams for the VM status notifications:

Table 1:

Parameter	Description
vm-state-notification stream	The **vm-state-notification** stream has the following details: • cluster_name: the cluster holds KVM nodes • node_name: KVM node name • vm_name: UPF name • state: VM state • message: Carries useful information such as error message or mgmt-ip when the state of the VM is ALIVE.
alert-notification stream	The **alert-notification stream** has the following labels: • node_name: KVM node name • vm_name: UPF name • state: VM state • message: Carries useful info such as error message or mgmt-ip when state is ALIVE.

Depending on the life cycle stage of the VM, notifications are generated from either the Cluster Manager or CEE Ops-Center and sent to the NSO. When VM gets deleted or redeployed, the UNDEPLOYED notification is sent to the Cluster Manager notification stream. All other notifications are generated by the Alert Manager and then sent to the CEE Ops-Center notification stream.

Cluster Manager Notification

The Cluster Manager sends vm-state-notification for the UNDEPLOYED VM state.

CEE Ops-Center Notification

The CEE Ops-Center alert-notification sends the following alerts for different VM states:

- vm-deployed: minor - DEPLOYED
- vm-alive: minor – ALIVE (alert lasts for a short time and disappears automatically)
- vm-error: major - ERROR
- vm-recovering: warning - RECOVERING
- vm-recovery_failed: critical - RECOVERY_FAILED

All required fields are included in alert labels for notification from alert-notification. All VM alerts are viewable on the Grafana dashboard.

VM Action Notifications

Delete Action: When delete VM action is triggered, CM sends notifications that the VM is deleted. The VM states are UNDEPLOYED and ERROR for vm delete action.

```
clusters abc-cluster-15 nodes kvm-1 vms upf1 actions delete
```

Redeploy Action: When VM is in RECOVERY_FAILED state, NSO sends a request to redeploy the VM. A redeploy action does both delete action and sync action.

```
clusters abc-cluster-15 nodes kvm-1 vms upf1 actions redeploy
```

Redeploy Action Notification: The redeploy action sends a notification to the CM. The redeploy vm action has the following 4 states: UNDEPLOYED, ERROR, REDEPLOYED, REDEPLOY_ERROR.

```
show notification stream vm-state


notification
eventTime 2021-02-23T21:27:28.692+00:00
vm-state-notification
cluster_name cndp-testbed
node_name kvm-1
vm_name upf2
state UNDEPLOYED
message
!
!
notification
eventTime 2021-02-23T21:29:18.699+00:00
vm-state-notification
cluster_name cndp-testbed
node_name kvm-1
vm_name upf2
state REDEPLOYED
message
!
!
```

Configuring the Alert Notification in CEE

The user must configure alert notifications when they deploy the UPF VMs. Log in to the CEE cli to add the following configuration:

```
config
bulk-stats prune-interval-days 3
prometheus kvm-metrics defaults private-key "-----BEGIN OPENSSH PRIVATE
KEY-----LGXtil23N4YV=\n-----END OPENSSH PRIVATE KEY-----\n"
prometheus kvm-metrics defaults user cloud-user
```

```
prometheus kvm-metrics monitor-server 10.194.62.41
hostname abc-bm-15-controlplane
exit
```

Note The user must replace the IP, hostname, private key and user details.

Sample Notification from the Alert Notification Stream

```
notification
eventTime 2021-01-08T03:28:54.501+00:00
smi-alert-notification
starts-at 2021-01-08T03:28:24.493874101Z
ends-at 0001-01-01T00:00:00Z
alert-status firing
smi-alert-notification alert-label
name alertname
value vm-recovery-failed
!
smi-alert-notification alert-label
name cluster
value test-cee-kvm_cee-voice
!
smi-alert-notification alert-label
name hostname
value test-bm-15-controlplane
!
smi-alert-notification alert-label
name instance
value metrics-proxy-test-bm-15-controlplane:9100
!
smi-alert-notification alert-label
name job
value metrics-proxy
!
smi-alert-notification alert-label
name message
value 10.1.1.3
!
smi-alert-notification alert-label
name monitor
value prometheus
!
smi-alert-notification alert-label
name node_name
value controlplane
!
smi-alert-notification alert-label
name replica
value test-cee-kvm_cee-voice
!
smi-alert-notification alert-label
name severity
value critical
!
smi-alert-notification alert-label
name state
value RECOVERY_FAILED
!
smi-alert-notification alert-label
```

```
name vm_name
value upf2
!
smi-alert-notification alert-annotation
name summary
value upf2 failed to recover.
!
smi-alert-notification alert-annotation
name type
value Equipment Alarm
!
!
!
```

Configurable Host Profile Support

This section describes the procedure to configure BIOS using host profiles on Cisco UCS servers.

Adding a Host Profile

This section describes the sequence of operation about how to add a host-profile to the software configuration configured on the CM.

1. To use host profiles, add the **.tgz** file to the software configuration on the CM by using the following CLI configuration and sync.

   config
   software
   host-profile *host-profile_name* **url** *software_url* **user** *user_name* **password** *passowrd*

   NOTES:

   - **host-profile** *host-profile_name* - Specify the name of the host profile, which is used to configure the BIOS settings for a UCS server.
   - **url** *software_url* - Specify the HTTP, HTTPS, or file URL of the software. File format must be "file:///absolute/path/to/file".

     Must be a URI in the pattern (http:|https:|file:/)//.*.
   - **user** *user_name* - Specify the user name for HTTP/HTTPS authentication.

     Must be a string.
   - **password** *passowrd* - Specify the password for downloading software package.

     Must be an aes-cfb-128-encrypted string.

   The following CLI commands form a sample configuration.

   ```
   software host-profile cndp-upf
    url             https://www.abc.com/ucs/profile/cndp-upf-profile.tgz
    user            smi-readonly.gen
    password        $8$abc
    sha256          b01dc4d47926e7a35e352c2f2d1c9f8b280fd44a89d0657281ff858387669753
   exit
   ```

2. To add the host-profile name in the cluster configuration, use the following CLI configuration.

```
config
   clusters cluster_name
      nodes node_name
         host-profile host-profile_name
         commit
         exit
```

NOTES:

- **clusters** *cluster_name* - Specify the name used to uniquely identify the cluster. Must be an alphanumeric string, and can contain the hyphen (-) character, however it must not start with a hyphen.

  Must be a string.

- **nodes** *node_name* -

  Specify the name of the node. name can be an alphanumeric string containing the hyphen (-). A host name cannot start with a hyphen (-). For example, kashaio-123.

  Must be a string.

- **host-profile** *host-profile_name* - Specify the name of the host profile, which is used to configure the BIOS settings for a UCS server.

The following CLI commands form a sample configuration.

```
config
clusters example-upf nodes kvm-1 host-profile cndp-upf
commit
top
exit
```

To return to the default settings, remove host-profile setting from the node. The following CLI commands form a sample configuration.

```
config
no clusters example-upf nodes kvm-1 host-profile
commit
exit
```

BIOS Settings

The BIOS profile allows access to all the settings on all the CIMC variants.

The sample configuration below shows the default values when a host-profile is *not* configured for the node. You can create a host profile once you have the necessary tokens.

```
profiles:
  bios:
    name: cndp_default_settings
    description: "Default CIMC BIOS settings for CNDP"
    pids:
      UCSC-C220-M5SX:
        description: "Default CIMC BIOS settings for UCSC-C220-M5SX (the original)"
        tokens:
          cpuPerformance: hpc
          cpuEnergyPerformance: balanced-performance
          eppProfile: Performance
          intelHyperThreadingTech: disabled
          packageCstateLimit: C0 C1 State
```

```
      usbPortInternal: disabled
      usbPortKvm: enabled
      usbPortRear: disabled
      usbPortSdCard: disabled

  UCSC-C220-M6S:
    description: "Default CIMC BIOS settings for UCSC-C220-M6S"
    tokens:
      cpuPerformance: hpc
      cpuEnergyPerformance: balanced-performance
      eppProfile: Performance
      intelHyperThreadingTech: disabled
      packageCstateLimit: C0 C1 State
      usbPortInternal: disabled
      usbPortRear: disabled

  UCSC-C240-M4SX:
    tokens:
      cpuPerformance: enterprise
      cpuEnergyPerformance: balanced-performance
      ## not supported eppProfile: Performance
      intelHyperThreadingTech: Disabled
      packageCstateLimit: C0/C1
      usbPortInternal: Disabled
      usbPortKvm: Enabled
      usbPortRear: Disabled
      usbPortSdCard: Disabled
```

To add a host profile, see the Adding a Host Profile, on page 48 section.

Hyper Threading Support on KVM

The Hyper Threading (HT) allows more than one thread to run on each core to enable more work in parallel.

To enabled or disabled HT, see the BIOS Settings, on page 49 section.

When HT is enabled on the UCS server, the total CPU number doubles, which impacts the following:

- Isolation CPU setting
- VM CPU allocation
- UPF CPU Worker Count setting

Isolation CPU Pinning

When HT is disabled, one CPU from each socket is reserved for the base OS. The configuration is as follows:

```
cloud-user@user-bm-15-controlplane:~$ cat/proc/cmdline

BOOT_IMAGE=/vmlinuz

root=LABEL=cloudimg-rootfs ro intel_iommu=on isolcpus=1-19,21-39 hugepagesz=1G
            hugepages=312 console=ttyS0,115200 console=tty1
```

When HT is enabled, **isolcpus** parameter is set to:

```
isolcpus=1-19,21-39,41-59,61-79
```

CPU Worker Count

The *VPP_CPU_WORKER_CNT* is set in the day 0 configuration. It depends on a number of other parameters such as the UPF flavor configuration, SR-IOV, HT setting, and the vCPU allocation.

The *VPP_CPU_WORKER_CNT* is adjusted according to the UPF flavor configuration. The UPF VM can be with different flavors, such as full VM (1 UPF), half VM (2 UPFs) and quarter VM (4 UPFs) in the UCS environment.

The SR-IOV provides capabilities to configure multiple VFs per PF.

The NIC allows the number of Rx and Tx queues to be configured per VF. The maximum Rx and Tx queues depends on the SR-IOV or PCI_PT configuration. A maximum of 16 Rx and Tx queues can be configured per VF in SR-IOV.

The *VPP_CPU_WORKER_CNT* configuration overrides the default value for PCI_PT and SR-IOV.

When HT is enabled, the CPU number is doubled and the *VPP_CPU_WORKER_CNT* is also doubled.

The tables below summarizes the Worker Count details.

Table 2: Parameters With HT Disabled

Flavor	Number of UPFs	Worker Count	Rx Queues	Tx Queues	vCPUs
Full	1	16	16	16	38
Half (default)	2	8	8	9	19
Quarter	4	4	4	5	9

Table 3: Paraemeters with HT Enabled

Flavor	Number of UPFs	Worker Count	Rx Queues	Tx Queues	vCPUs
full	1	16	16	16	76
half (default)	2	16	16	16	38
quarter	4	8	8	9	18

You can check the worker count configuration as follows:

```
isoinfo -R -i /data/upf/libvirt/xml/upf1/upf1-cfg.iso -x /staros_param.cfg
```

Emulator Pinning

You can set the emulator pinning during the VM CPU allocation:

```
<emulatorpin cpuset="0,40”/> <emulatorpin cpuset="20,60”/>
```

To verify the VM CPU allocation, run the following command:

```
virsh dumpxml upf2
```

The KVM node must be redeployed if the HT setting is changed (for example, from enabled to disabled).

CPU Isolation

Table 4: Feature History

Feature Name	Release Information	Feature Description
CPU Isolation	2023.04	SMI provides a higher level of CPU isolation for VPP workers to support cnUPF. Using the host profile, SMI defines isolcpu to isolate CPUs from the kernel scheduler.

SMI provides a higher level of CPU isolation for VPP workers. With CPU isolation, no other processes can be scheduled on "isolcpu" CPUs where VPP workers are pinned.

SMI uses the host profile to define isolcpu that isolates CPUs from the kernel scheduler. It does not prevent K8s containers from changing their affinities to run on isolcpus. Depending on the deployment, SMI also provides the flexibility to use VPP workers and session managers for CPU isolation.

How it Works

CPU isolation utilizes a containerd Node Resource Interface (NRI) plugin (v0.3.0) that subscribes for pod or container lifecycle events.

NRI is a common framework for plugging extensions into OCI-compatible container runtimes. It provides basic mechanisms for plugins to track the state of containers and to make limited changes to their configuration.

Using the NRI Plugin

The following built-in rules apply for the plugin:

- The CPU isolator ignores containers in a pod with prefix name as "vpc-"
- If the annotation smi.cisco.com/cpuset exists, CPU isolator adjusts the CPU set using its value
- Otherwise the value of environment "CPUSET_KUBEPODS" is used.

The following steps describe how to start the NRI plugin:

- Retain the CPU set for K8s to all CPUs
- During a CreateContainer event, the plugin adjusts the container's CPU set based on the following conditions:
  - if it is a VPC container, it does nothing so that VPP workers can be pinned to isolated CPUs
  - for other non-VPC containers, it creates a customized CPU set to exclude the isolated CPU

Sample Configuration

CM calculates the CPU set of kubepods and passes it as the environment value "CPUSET_KUBEPODS" to create a static pod for CPU isolator.

To define isolcpu from host profile, the following is a sample host profile configuration:

```
---
profile_version: 2
profiles:
  bios:
    name: test_bios_settings
    description: "CIMC BIOS settings"
    pids:
      UCSC-C220-M5SX:
        description: "Copy of Default CIMC BIOS settings for UCSC-C220-M5SX"
        tokens:
          cpuPerformance: hpc
          cpuEnergyPerformance: balanced-performance
          eppProfile: Performance
          packageCstateLimit: C0 C1 State
          usbPortInternal: disabled
          usbPortKvm: enabled
          usbPortRear: disabled
          usbPortSdCard: disabled
  linux:
    cisco_c220_m5_dual_numa:
      # name: no longer used
      description: "Cisco M5 Dual Numa Test Profle"
      grub:
        dedicate-cpus:
          numa-0: '0%'
          numa-1: '26-39'
        hugepages:
          numa-0:
            2m: 0%
            1g: 0%
          numa-1:
            2m: 20%
            1g: 50
       addition-grub-cmd-line: "intel_iommu=on intel_pstate=disable intel_idle.max_cstate=1
 pcie_aspm.policy=performance idle=poll clocksource=tsc tsc=reliable skew_tick=1 nosoftlockup"

      sysctl:
        net.ipv4.neigh.default.gc_thresh1: 8192
        net.ipv4.neigh.default.gc_thresh2: 32768
        net.ipv4.neigh.default.gc_thresh3: 65536
        net.ipv6.neigh.default.gc_thresh1: 8192
        net.ipv6.neigh.default.gc_thresh2: 32768
        net.ipv6.neigh.default.gc_thresh3: 65536
      sysfs:
        /sys/module/nvme_core/parameters/io_timeout: 4294967295
      sriov:
      - match:
          pf-name:  enp94s*
        vfs-per-pf: 4
        dpdk-vfs-per-pf: 2
        dpdk-bind: vfio-pci
      - match:
          pf-name:  enp216s*
        vfs-per-pf: 16

    cisco_c220_m5_single_numa:
      description: "Cisco M5 Single Numa Test Profle"
      grub:
        dedicate-cpus:
          numa-0: '26-39'
        hugepages:
          numa-0:
            2m: 20%
            1g: 0%
```

```
        # single core needs intel_iommu=off
       addition-grub-cmd-line: "intel_iommu=off intel_pstate=disable intel_idle.max_cstate=1
 pcie_aspm.policy=performance idle=poll clocksource=tsc tsc=reliable skew_tick=1 nosoftlockup"

      sysctl:
        net.ipv4.neigh.default.gc_thresh1: 8192
        net.ipv4.neigh.default.gc_thresh2: 32768
        net.ipv4.neigh.default.gc_thresh3: 65536
        net.ipv6.neigh.default.gc_thresh1: 8192
        net.ipv6.neigh.default.gc_thresh2: 32768
        net.ipv6.neigh.default.gc_thresh3: 65536
      sysfs:
        /sys/module/nvme_core/parameters/io_timeout: 4294967295
      sriov:
      - match:
          pf-name:  enp94s*
        vfs-per-pf: 4
        dpdk-vfs-per-pf: 2
        dpdk-bind: vfio-pci
      - match:
          pf-name:  enp216s*
        vfs-per-pf: 16
```

Single-Root Input/Output Virtualization

Overview

The Single Root I/O Virtualization (SR-IOV) specification is a Peripheral Component Interconnect (PCI) passthrough hardware standard that regulates the device assignment. The PCI passthrough natively shares a single resource with multiple guests. PCI passthrough enables the virtual machines to bypass the hypervisor and virtual switch layer, and communicate directly with the PCI devices residing on the host. SR-IOV is an extension of the PCI passthrough specification. In the PCI configuration space, a single physical device that has SR-IOV capability enabled is virtualized as multiple devices. Each device is a secured and distinct unit with isolation at the resource level, which means it has its own set of storage and configurations. This mechanism provides data transfer at the near wire-speed along with low latency.

The SMI on Bare Metal platform supports both, PCF passthrough and the SR-IOV standards in the form of plugins. In the CNDP-KVM-UPF implementation, the VM and network devices are by default configured to support SR-IOV Virtual Functions (VFs). If you want to pass the network resources to network function (UPF) through the PCI passthrough standard, then enable the PCI passthrough for the network device.

How it Works

The SMI on Bare Metal platform lets the NF (UPF) select the PCI passthrough or SR-IOV as a network resource management plugin. The PCI passthrough allows the NF to have an exclusive access to the the physical PCI devices such as network card through a VM. The PCI devices may not be physically attached to the guest host however, they provide an efficient and seamless interaction to the NF.

The SR-IOV enables the virtual machines to have parallel access to the physical network cards installed on the hypervisor. Traditionally, the physical devices had one-to-one mapping with the VMs, which required several hardware resources and memory capacity. SR-IOV has introduced Virtual Functions (VFs) that are lightweight Peripheral Component Interconnect Express (PCIs) function. The PCI is an interface standard that is responsible for connecting high-speed components. The VFs are the self-sufficient components required for processing input/output requests between VMs. Each VF is derived from the Physical Function (PF). The number of VFs that a VM supports is based on the underlying hardware device's capacity. For instance, the hypervisor can map one or more VFs to a virtualized guest. The VF's configuration space is mapped to the

space that the virtualized guest has presented to the VF. This is essentially the space that the hypervisor allocates to the virtualized guest.

Following is the sample XML configuration of PCI passthrough and SR-IOV standards:

PCI passthrough:

```
<hostdev mode='subsystem' type='pci' managed='yes'>
  <driver name='vfio'/>
  <source>
    <address domain='0x0000' bus='0x5e' slot='0x00' function='0x0'/>
  </source>
</hostdev>
```

SR-IOV:

```
<interface type='hostdev' managed='yes'>
  <mac address='02:11:11:22:22:04'/>
  <driver name='vfio'/>
  <source>
    <address type='pci' domain='0x0000' bus='0x5e' slot='0x0a' function='0x1'/>
  </source>
</interface>
```

Enabling Support for PCI Passthrough

This section describes how to enable a VM to support PCI Passthrough.

If the VM uses the SR-IOV mechanism, then ensure that the VF number for the associated PF and MAC address is available.

Use the following configuration to configure the PCF passthrough option on the VM:

```
cluster cluster_name
   nodes node_name
      type kvm
         os enable-passthrough  {true|false}
         os num-vfs-per-pf number_of_vf_per_pf
   exit
exit
```

NOTES:

- **cluster** *cluster_name* – Specify the name of the cluster.
- **nodes** *node_name* – Specify the name of the node.
- **os enable-passthrough {true|false}** – Configures the PCI passthrough capability for the VM. Default value is "false".
- **os num-vfs-per-pf** *number_of_vf_per_pf* – Specifies the number of the VF for each physical function. Default value is 16.

IPSec Support for SMF N4 Interfaces

Feature Summary and Revision History

Summary Data

Applicable Product (s) or Functional Area	KVM-based application deployment support
Applicable Platforms	Bare Metal, OpenStack, VMware
Feature Default Setting	Disabled – Configuration Required
Related Changes in this Release	Not Applicable
Related Documentation	*UCC SMI Operations Guide*

Revision History

Revision Details	Release
Added support for the following functionality: • IPSec Monitoring • Configuring IPSec certificates under strongSwan configuration	2023.01.0
First introduced.	2020.02.2.47

Feature Description

This feature introduces strongSwan, a keying daemon, which uses the Internet Key Exchange (IKE) protocols, IKEv1 and IKEv2, to establish **security associations** (SA) between two peers in a network. Such an IKE session is denoted as **IKE_SA** in this chapter. The IKE provides strong authentication for both peers and derives unique cryptographic session keys. Besides authentication and key material, IKE also provides the means to exchange configuration information and to negotiate IPsec SAs, which are often called as **CHILD_SAs**. IPsec SAs define which network traffic is to be secured and how it has to be encrypted and authenticated.

The strongSwan feature is available as an add-on from the Cluster Manager (CM). Use the CM Ops-Center to configure this add-on. In the current release, the SMI uses strongSwan version 5.9.3.

SMI allows monitoring of IPSec certificates—sends certificate expiry alerts and updates certificate through strongSwan configuration.

Configuration Parameters

In this section, see the description for different configuration parameters available for the strongSwan add-on feature. Use the CM Ops-Center to configure these parameters.

- **`name`**: Specifies the name of the connection, which can be used for connection specific operations, for example, up or down.

- **auto { ignore |add |route |start }**: Specifies the operation, if any, that should be automatically performed at IPsec startup. The **add** option loads a connection without starting it, whereas **route** loads a connection and installs kernel traps. If traffic is detected between the leftsubnet and rightsubnet, a connection is established. The **start** option loads a connection and brings it up immediately. The **ignore** option ignores the connection and is the same as deleting a connection from the config file.

  The default value is **ignore**.

- **keyexchange { ikev1 | ikev2 }** : Specifies the method of key exchange and the protocol to use to initialize the connection.

- **type { tunnel | transport | transport_proxy | passthrough | drop }**: Specifies the type of the connection. Currently, the accepted values are **tunnel**, signifying a host-to-host, host-to-subnet, or subnet-to-subnet tunnel. The **transport** option signifies a host-to-host transport mode, whereas the **transport_proxy** option signifies the special Mobile IPv6 transport proxy mode. The **passthrough** option signifies that no IPsec processing should be done at all and **drop** signifies that packets must be discarded.

- **left** or **right { ip address** *ip_address* **| fqdn** *fqdn* **| %any | %any4 | %any6 | range | subnet }**: Specifies the IP address or FQDN of the participant public-network interface. The value **%any** for the local endpoint signifies an address to be filled in (by automatic keying) during negotiation. If the local peer initiates the connection setup, then the routing table is queried to determine the correct local IP address. If the local peer is responding to a connection setup, then any IP address that is assigned to a local interface is accepted. The value **%any4** restricts address selection to IPv4 addresses and **%any6** restricts address selection to IPv6 addresses.

- **leftsubnet** or **rightsubnet** *ip subnet*: Specifies the private subnet behind the left participant, expressed as either network or netmask.

- **leftid** or **rightid** *id*: Specifies how the left or right participant must be identified for authentication. The default values are left or right or the subject of the certificate configured. It must match the full subject DN or one of the subjectAltName extensions contained in the certificate.

- **leftsendcert { never | no | ifasked | always | yes }**: Defines whether a peer must send a certificate request (CR) payload in order to get a certificate in return.

- **leftauth** or **rightauth{ pubkey | psk | eap | xauth }**: Specifies the authentication method to use locally (left) or require from the remote (right) side. The acceptable values are **pubkey** for public key encryption (RSA/ECDSA), **psk** for pre-shared key authentication, **eap** to use the Extensible Authentication Protocol, and **xauth** for IKEv1 eXtended Authentication.

  Pubkey is the default option.

- **psk** *pre-shared key*: Specifies the required setting if leftauth or rightauth is configured as **psk**.

- **esp { cipher suites | aes128-sha256 }**: A comma-separated list of ESP encryption or authentication algorithms is used for the connection, for example, **aes128-sha256**. The notation is encryption-integrity[-dhgroup][-esnmode]. For IKEv2, multiple algorithms (separated by -) of the same type can be included in a single proposal. IKEv1 only includes the first algorithm in a proposal.

  aes128-sha256 is the default option.

- **ike { cipher suites | aes128-sha256-modp3072 }**: A comma-separated list of IKE/ISAKMP SA encryption or authentication algorithms is used, for example, **aes128-sha256-modp3072**. The notation is encryption-integrity[-prf]-dhgroup. In IKEv2, multiple algorithms and proposals might be included, such as aes128-aes256-sha1-modp3072-modp2048 or 3des-sha1-md5-modp1024.

- **ikelifetime { time** *time* **| 3h }**: Specifies how long the keying channel of a connection (ISAKMP or IKE SA) must last before being renegotiated.
- **lifetime { time** *time* **| 1h }**: Specifies how long a particular instance of a connection should last, from successful negotiation to expiry.
- **dpdaction { none | clear | hold | restart }**: Specifies the action to be taken when dead peer is detected.

  none is the default value.
- **dpddelay { time** *time* **| 30s }**: Defines the period time interval with which INFORMATIONAL exchanges are sent to the peer. These are only sent if no other traffic is received.
- **dpdtimeout { time** *time* **| 150s }**: Defines the timeout interval after which, all the connections to a peer are deleted in case of inactivity.
- **inactivity time** *time*: Defines the timeout interval after which, a CHILD_SA is closed if it did not send or receive any traffic.
- **closeaction { none | clear | hold | restart }**: Defines the action to take if the remote peer unexpectedly closes a CHILD_SA (see **dpdaction** for the description of different options). If the peer uses reauthentication or uniqueids checking, **closeaction** must not be used, these events might trigger the defined action when it's not desired.
- **nodes** *list_of_node_names*: Specifies the node names on which IPSec connection must be established.
- **serverCert** *server_certificate*: Specifies the content of Server certificate in the **pem** format to be used for this connection.

Note This keyword is not supported under strongSwan configuration.

- **serverPrivKey** *server_private_key*: Specifies the content of server private key in the **pem** format to be used for this connection.

Note This keyword is not supported under strongSwan configuration.

- **serverPrivKeyPassphrase** *passphrase*: Specifies the passphrase used to encrypt the **server-priv-key** value.
- **server-secret**: Pass an existing TLS secret for this connection.

Installing strongSwan

This section describes how to install the strongSwan feature.

Install strongSwan as an Add-on from the CM

Use the following steps to install strongSwan as an add-on from the CM Ops-Center:

1. Use the following CLI commands to enable the strongSwan add-on:

   clusters *cluster_name* **addons strongswan enabled**

2. Set all the strongSwan parameters for **connection** (refer to the *Configuration Parameters* section for more details on available parameters).
3. Trigger the cluster sync operation.

Note The strongSwan pods run on all the nodes, however traffic is accepted only on those nodes, which are configured by using the "nodes" parameter in the CM Ops-Center. strongSwan does not accept or send any traffic on non-configured nodes.

Configuring IPSec Certificates

To configure IPSec certificates under strongSwan configuration, use the following procedure:

1. Create TLS associated secret for server and CA certificate.

   Note: Create strongSwan-related secrets inside the smi-strongswan namespace.

   Example:

```
[test-cm-controlplane] SMI Cluster Deployer# show running-config clusters secrets ca-cert
clusters test-aio
 secrets ca-cert smi-strongswan 134-ca
  certificate "-----BEGIN
CERTIFICATE-----\nMIIDqzCzQubm..................1Ac1L+s4M3ug==\n-----END
CERTIFICATE-----\n"
 exit
 secrets ca-cert smi-strongswan 135-ca
  certificate "-----BEGIN
CERTIFICATE-----\nMIIFqzCCA5Og..................9XdMDiQANHgf7w\n-----END
CERTIFICATE-----\n"
 exit
 secrets ca-cert smi-strongswan ca-1
  certificate "-----BEGIN
CERTIFICATE-----\nMIID0TCCArmg..................UNvF0nAmIX0qxg4\n-----END
CERTIFICATE-----\n"
 exit
 secrets ca-cert smi-strongswan ca-2
  certificate "-----BEGIN PRIVATE
KEY-----\nMIIEvQIBADAN..................tbNDzGAnF29nus=\n-----END PRIVATE KEY-----\n"
 exit
exit
```

2. Refer the secrets in strongSwan configuration. The strongSwan configuration shows the available TLS and certificates.

   Example:

```
[test-cm-controlplane] SMI Cluster Deployer# show running-config clusters karan-aio
strongswan connections server-secret
clusters test-aio
 strongswan connections a-to-b
  server-secret a-to-b
 exit
exit

[test-cm-controlplane] SMI Cluster Deployer# show running-config clusters karan-aio
strongswan ca-certs
clusters test-aio
```

```
strongswan ca-certs [ 134-ca 135-ca ]
exit
```

Operating the SMI Cluster Manager on vCenter VMware

This section describes how to instantiate the K8s Cluster on the SMI Cluster Manager in a VMware vSphere environment. The SMI Cluster Manager is the de-facto orchestrator for the K8s Cluster. It requires add-ons like *Ngnix-Ingress*, *Istio*, and so on. Also, the Ops Center is required for all the Cisco Cloud products using the SMI.

SMI supports datacenters at the root level and within folders. If the datacenter is within a folder, then the entire path from the root until the datacenter is mentioned in the *datacenter-path* field. If the vSphere cluster is organised within folders, SMI can auto-detect the cluster as long as the name is unique with the datacenter.

The SMI Cluster Manager currently provisions the following:

1. Base OS, creation of nodes and virtual machines in VMware vSphere virtualization environment.
2. Deployment of K8s multi-control plane cluster with Calico pod networking.
3. Deployment of K8s addon applications.
4. Application provisioning. You can either deploy the Ops Center alone or the entire product.
5. Product based Host OS customization for nodes with specific labels.

Supported Configurations

The SMI Cluster Manager supports the following three VM configurations:

Table 5: Functional Test HA

	CPU	Cores Per Socket	RAM	Data Disk	Home Disk	Root Disk
Control Plane	2 CPU	1	16 GB	20 GB	5 GB	100 GB
ETCD	1 CPU	1	8 GB	20 GB	5 GB	100 GB
Worker	8 CPU	1	32 GB	200 GB	5 GB	100 GB

Important These configurations are applicable only for testing in the Lab environment. It is not supported in the production environment.

Table 6: Functional Test AIO

	CPU	Cores Per Socket	RAM	Data Disk	Home Disk	Root Disk
Control Plane	6 CPU	1	32 GB	200 GB	5 GB	100 GB

Important These configurations are applicable only for testing in the Lab environment. It is not supported in the production environment.

Note Individual NFs are deployed as K8s workers through SMI. They each have their own VM recommendations. Refer to the NF documentation for details.

Table 7: Production

	CPU	Cores Per Socket	RAM	Data Disk	Home Disk	Root Disk
Control Plane	2 CPU	2	16 GB	20 GB	5 GB	100 GB
ETCD	2 CPU	2	16 GB	20 GB	5 GB	100 GB
Worker	36 CPU	36	164 GB	200 GB	5 GB	100 GB

Prerequisites

The prerequisites for instantiating the K8s Cluster on SMI Cluster Manager are:

1. SMI Cluster Manager AIO.
2. Local web server that hosts the products offline TAR ball version(s).

Requirements

SMI Cluster Manager (all versions are supported).

Components Used

The following component is used for deploying and upgrading the product in the offline environment:

- SMI Cluster Manager.

Configuring the vCenter Environment

To configure the vCenter environment, use the following configuration:

1. Configure the vCenter environment with the required configuration parameters through the SMI Cluster Manager CLI. A sample configuration is shown below:

```
environments laas
   vcenter server vcenter_server_ipv4_address
   vcenter port vcenter_port
   vcenter allow-self-signed-cert true (to allow self signed certs)
   vcenter user vcenter_username
```

```
    vcenter password vcenter_password
   vcenter datastore vcenter_host_datastore (the corresponding vcenter host datastore)

    vcenter cluster vcenter_cluster (the vcenter cluster containing the above host)
    vcenter datacenter-path datacenter_path
    vcenter datacenter vcenter_datacenter
    vcenter host vcenter_host_ipv4_address
    vcenter nics network_ID
    exit
exit
```

Important You can add each vCenter environment to one or more K8s Cluster configuration. For VMs managed in the OpenStack environment, you can use the following configuration:

```
environments openstack
   manual
exit
```

2. Configure the cluster essentials like node defaults which includes, initial boot, K8s, operating system NTP, and node configurations. In a multi-mode environment, a minimum of 3 control planes, 3 etcd, and 3 OAM (worker or product) nodes are required. The number of worker nodes and its type depends on the product that is being installed. For more information about the worker nodes and labels, see the relevant product documentation. The following example shows the cluster configuration which is not specific to any products.

Note Based on the customer requirements, you can choose to either include or exclude the following cluster configurations:

- Volume provisioning – Configure volume provisioning while using persistent volumes.
- Network Proxy – Configure network proxies based on the requirements.
- Local NTP with Authentication – For configuring local NTP server with authentication, see Configuring the Local NTP Server with Authentication and Tracking section.
- Virtual IPs (VIPs) – Configure virtual IP addresses based on the requirements.
- Product registry secrets – Set up secrets to protect product registries.
- Node labels – If required, assign specific labels to nodes.

```
clusters <cluster_name>

        # associating an existing vcenter environment
        environment <vcenter_environment> #Example:laas

        # General cluster configuration
        configuration master-virtual-ip <keepalived_ipv4_address>
        configuration master-virtual-ip-cidr
<netmask_of_additional_master_virtual_ip> #Default is 32
        configuration master-virtual-ip-interface <interface_name>
```

```
        configuration additional-master-virtual-ip <ipv4_address>
        configuration additional-master-virtual-ip-cidr
<netmask_of_additional_master_virtual_ip> #Default is 32
        configuration additional-master-virtual-ip-interface
<interface_name>
       configuration virtual-ip-vrrp-router-id <virtual_router_id> #To support
 multiple instances of VRRP in the same subnet
        configuration pod-subnet <pod_subnet> #To avoid conflict with already
existing subnets
        configuration size <functional_test_ha/functional_test_aio/production>
        configuration allow-insecure-registry <true> #To allow insecure
registries

       # istio and nginx ingress addons
        addons ingress bind-ip-address <keepalived_ipv4_address>
        addons istio enabled

        # vsphere volume provider configuration
        addons vsphere-volume-provider server <vcenter_server_ipv4_address>
        addons vsphere-volume-provider server-port <vcenter_port>
       addons vsphere-volume-provider allow-insecure <true> #To allow self
 signed certs
        addons vsphere-volume-provider user <vcenter_username>
        addons vsphere-volume-provider password <vcenter_password>
        addons vsphere-volume-provider datacenter <vcenter_datacenter>
        addons vsphere-volume-provider datastore <vcenter_nfs_storage>
#Corresponding vcenter nfs storage
        addons vsphere-volume-provider network <network_id>
        addons vsphere-volume-provider folder
<cluster_folder_containing_the_VMs>

        # Openstack volume provider configuration
        addons openstack-volume-provider username <username>
        addons openstack-volume-provider password <password>
        addons openstack-volume-provider auth-url <auth_url>
        addons openstack-volume-provider tenant-id <tenant_id>
        addons openstack-volume-provider domain-id <domain_id>

        # initial-boot section of node-defaults for vmware
        node-defaults initial-boot default-user <default_username>
        node-defaults initial-boot default-user-ssh-public-key
<public_ssh_key>
        node-defaults initial-boot netplan template


        # initial-boot section of node-defaults for VMs managed in Openstack
        node-defaults initial-boot default-user <default_user>
        node-defaults netplan template
          #jinja2:variable_start_string:'__DO_NOT_ESCAPE__' ,
variable_end_string:'__DO_NOT_ESCAPE__'
          #

        #k8s related config of node-defaults
        node-defaults k8s ssh-username <default_k8s_ssh_username>
        node-defaults k8s ssh-connection-private-key
               -----BEGIN RSA PRIVATE KEY-----
               <SSH_Private_Key>
```

```
        -----END RSA PRIVATE KEY-----

      # os related config of node-defaults
      node-defaults os proxy https-proxy <https_proxy>
      node-defaults os proxy no-proxy <no_proxy_info>
      node-defaults os ntp servers <local_ntp_server>
      exit

      # node configuration of multinode cluster. vmware related info overrides the
defaults provided in the environment 'laas' associated with the cluster

    nodes node_name #For example, etcd1
      k8s node-type etcd
      k8s ssh-ip ipv4address
      k8s node-ip ipv4address
      vmware datastore datastore_name
      vmware host host_name
      vmware performance latency-sensitivity normal
      vmware performance memory-reservation false
      vmware performance cpu-reservation false
      vmware sizing ram-mb ram_size_in_mb
      vmware sizing cpus cpu_size
      vmware sizing disk-root-gb disk_root_size_in_gb
      vmware nics network_ID
    exit
  exit
  nodes node_name #For example, etcd2
     k8s node-type etcd
     k8s ssh-ip ipv4address
     k8s node-ip ipv4address
     vmware datastore datastore_name
     vmware host host_name
     vmware performance latency-sensitivity normal
     vmware performance memory-reservation false
     vmware performance cpu-reservation false
     vmware sizing ram-mb ram_size_in_mb
     vmware sizing cpus cpu_size
     vmware sizing disk-root-gb disk_root_size_in_gb
     vmware nics network_ID
   exit
  exit
  nodes node_name #For example, etcd3
     k8s node-type etcd
     k8s ssh-ip ipv4address
     k8s node-ip ipv4address
     vmware datastore datastore_name
     vmware host host_name
     vmware performance latency-sensitivity normal
     vmware performance memory-reservation false
     vmware performance cpu-reservation false
     vmware sizing ram-mb ram_size_in_mb
     vmware sizing cpus cpu_size
     vmware sizing disk-root-gb disk_root_size_in_gb
```

```
    vmware nics network_ID
  exit
exit
nodes node_name #For example, controlplane1
    k8s node-type control-plane
    k8s ssh-ip ipv4address
    k8s node-ip ipv4address
    vmware datastore datastore_name
    vmware host host_name
    vmware performance latency-sensitivity normal
    vmware performance memory-reservation false
    vmware performance cpu-reservation false
    vmware sizing ram-mb ram_size_in_mb
    vmware sizing cpus cpu_size
    vmware sizing disk-root-gb disk_root_size_in_gb
    vmware nics network_ID
   exit
exit
nodes node_name #For example, controlplane2
   k8s node-type control-plane
   k8s ssh-ip ipv4address
   k8s node-ip ipv4address
   vmware datastore datastore_name
   vmware host host_name
   vmware performance latency-sensitivity normal
   vmware performance memory-reservation false
   vmware performance cpu-reservation false
   vmware sizing ram-mb ram_size_in_mb
   vmware sizing cpus cpu_size
   vmware sizing disk-root-gb disk_root_size_in_gb
   vmware nics network_ID
   exit
 exit
nodes node_name #For example, controlplane3
   k8s node-type control-plane
   k8s ssh-ip ipv4address
   k8s node-ip ipv4address
   vmware datastore datastore_name
   vmware host host_name
   vmware performance latency-sensitivity normal
   vmware performance memory-reservation false
   vmware performance cpu-reservation false
   vmware sizing ram-mb ram_size_in_mb
   vmware sizing cpus cpu_size
   vmware sizing disk-root-gb disk_root_size_in_gb
   vmware nics network_ID
   exit
exit
nodes node_name #For example, oam1
   k8s node-type worker
   k8s ssh-ip ipv4address
```

```
    k8s node-ip ipv4address
    k8s node-labels node_labels
    exit
    vmware datastore datastore_name
    vmware host host_name
    vmware performance latency-sensitivity normal
    vmware performance memory-reservation false
    vmware performance cpu-reservation false
    vmware sizing ram-mb ram_size_in_mb
    vmware sizing cpus cpu_size
    vmware sizing disk-root-gb disk_root_size_in_gb
    vmware nics network_ID
    exit
exit
nodes node_name #For example, oam2
    k8s node-type worker
    k8s ssh-ip ipv4address
    k8s node-ip ipv4address
    k8s node-labels node_labels
    exit
    vmware datastore datastore_name
    vmware host host_name
    vmware performance latency-sensitivity normal
    vmware performance memory-reservation false
    vmware performance cpu-reservation false
    vmware sizing ram-mb ram_size_in_mb
    vmware sizing cpus cpu_size
    vmware sizing disk-root-gb disk_root_size_in_gb
    vmware nics network_ID
    exit
exit
nodes node_name #For example, oam3
    k8s node-type worker
    k8s ssh-ip ipv4address
    k8s node-ip ipv4address
    k8s node-labels node_labels
    exit
    vmware datastore datastore_name
    vmware host host_name
    vmware performance latency-sensitivity normal
    vmware performance memory-reservation false
    vmware performance cpu-reservation false
    vmware sizing ram-mb ram_size_in_mb
    vmware sizing cpus cpu_size
    vmware sizing disk-root-gb disk_root_size_in_gb
    vmware nics network_ID
    exit
exit
nodes node_name #For example, session-data1
    k8s node-type worker
    k8s ssh-ip ipv4address
```

```
 k8s node-ip ipv4address
 k8s node-labels node_labels #For example, smi.cisco.com/cdl-ep true
 exit
 k8s node-labelsnode_labels #For example, smi.cisco.com/cdl-index-1 true
 exit
 k8s node-labelsnode_labels #For example, smi.cisco.com/cdl-index-2 true
 exit
 k8s node-labelsnode_labels #For example, smi.cisco.com/cdl-slot-1 true
 exit
 k8s node-labelsnode_labels #For example, smi.cisco.com/cdl-slot-2 true
 exit
 k8s node-labelsnode_labels #For example, smi.cisco.com/cdl-slot-3 true
 exit
 k8s node-labelsnode_labels #For example, smi.cisco.com/cdl-slot-4 true
 exit
 k8s node-labelsnode_labels/node_type #For example, smi.cisco.com/node-type db

 exit
 k8s node-labelsnode_labels/vm_type #For example, smi.cisco.com/vm-type session

 exit
 vmware datastore datastore_name
 vmware host host_name
 vmware performance latency-sensitivity normal
 vmware performance memory-reservation false
 vmware performance cpu-reservation false
 vmware sizing ram-mb ram_size_in_mb
 vmware sizing cpus cpu_size
 vmware sizing disk-root-gb disk_root_size_in_gb
 vmware nics network_ID
 exit
exit
nodes node_name #For example, session-data2
 k8s node-type worker
 k8s ssh-ip ipv4address
 k8s node-ip ipv4address
 k8s node-labels node_labels #For example, smi.cisco.com/cdl-ep true
 exit
 k8s node-labelsnode_labels #For example, smi.cisco.com/cdl-index-1 true
 exit
 k8s node-labelsnode_labels #For example, smi.cisco.com/cdl-index-2 true
 exit
 k8s node-labelsnode_labels #For example, smi.cisco.com/cdl-slot-1 true
 exit
 k8s node-labelsnode_labels #For example, smi.cisco.com/cdl-slot-2 true
 exit
 k8s node-labelsnode_labels #For example, smi.cisco.com/cdl-slot-3 true
 exit
 k8s node-labelsnode_labels #For example, smi.cisco.com/cdl-slot-4 true
 exit
 k8s node-labelsnode_labels/node_type #For example, smi.cisco.com/node-type db
```

```
    exit
   k8s node-labelsnode_labels/vm_type #For example, smi.cisco.com/vm-type session

    exit
    vmware datastore datastore_name
    vmware host host_name
    vmware performance latency-sensitivity normal
    vmware performance memory-reservation false
    vmware performance cpu-reservation false
    vmware sizing ram-mb ram_size_in_mb
    vmware sizing cpus cpu_size
    vmware sizing disk-root-gb disk_root_size_in_gb
    vmware nics network_ID
    exit
exit
nodes node_name #For example, session-data3
    k8s node-type worker
    k8s ssh-ip ipv4address
    k8s node-ip ipv4address
    k8s node-labels node_labels #For example, smi.cisco.com/cdl-ep true
    exit
    k8s node-labelsnode_labels #For example, smi.cisco.com/cdl-index-3 true
    exit
    k8s node-labelsnode_labels #For example, smi.cisco.com/cdl-index-4 true
    exit
    k8s node-labelsnode_labels #For example, smi.cisco.com/cdl-slot-5 true
    exit
    k8s node-labelsnode_labels #For example, smi.cisco.com/cdl-slot-6 true
    exit
    k8s node-labelsnode_labels #For example, smi.cisco.com/cdl-slot-7 true
    exit
    k8s node-labelsnode_labels #For example, smi.cisco.com/cdl-slot-8 true
    exit
    k8s node-labelsnode_labels/node_type #For example, smi.cisco.com/node-type db

    exit
   k8s node-labelsnode_labels/vm_type #For example, smi.cisco.com/vm-type session

    exit
    vmware datastore datastore_name
    vmware host host_name
    vmware performance latency-sensitivity normal
    vmware performance memory-reservation false
    vmware performance cpu-reservation false
    vmware sizing ram-mb ram_size_in_mb
    vmware sizing cpus cpu_size
    vmware sizing disk-root-gb disk_root_size_in_gb
    vmware nics network_ID
    exit
 exit
 nodes node_name #For example, session-data4
    k8s node-type worker
```

```
      k8s ssh-ip ipv4address
      k8s node-ip ipv4address
      k8s node-labels node_labels #For example, smi.cisco.com/cdl-ep true
      exit
      k8s node-labelsnode_labels #For example, smi.cisco.com/cdl-index-3 true
      exit
      k8s node-labelsnode_labels #For example, smi.cisco.com/cdl-index-4 true
      exit
      k8s node-labelsnode_labels #For example, smi.cisco.com/cdl-slot-5 true
      exit
      k8s node-labelsnode_labels #For example, smi.cisco.com/cdl-slot-6 true
      exit
      k8s node-labelsnode_labels #For example, smi.cisco.com/cdl-slot-7 true
      exit
      k8s node-labelsnode_labels #For example, smi.cisco.com/cdl-slot-8 true
      exit
      k8s node-labelsnode_labels/node_type #For example, smi.cisco.com/node-type db

      exit
     k8s node-labelsnode_labels/vm_type #For example, smi.cisco.com/vm-type session

      exit
      vmware datastore datastore_name
      vmware host host_name
      vmware performance latency-sensitivity normal
      vmware performance memory-reservation false
      vmware performance cpu-reservation false
      vmware sizing ram-mb ram_size_in_mb
      vmware sizing cpus cpu_size
      vmware sizing disk-root-gb disk_root_size_in_gb
      vmware nics network_ID
      exit
   exit
exit
         # Virtual IPs
        virtual-ips <name> #Example: rxdiam

          vrrp-interface <interface_name>
          vrrp-router-id <router_id>

          ipv4-addresses <ipv4_address>
            mask <netmassk>
            broadcast <broadcast_ipv4_address>
            device <interface_name>
          exit
          # nodes associated with the virtual-ip
          hosts <node_name> #Example: smi-cluster-core-protocol1
            priority <priority_value>
          exit
          hosts <node_name> #Example: smi-cluster-core-protocol2
            priority <priority_value>
          exit
        exit
```

```
 # Secrets for product registry
secrets docker-registry <secret_name>
  docker-server <server_name or docker_registry>
  docker-username <username>
  docker-password <password>
  docker-email <email>
  namespace <k8s_namespace> #Example: cee-voice
exit
ops-centers <app_name> <instance_name> #Example: cee data
  repository <artifactory_url>



  username <username>
  password <password>

  initial-boot-parameters use-volume-claims <true/false> #True to
use persistent volumes and vice versa
  initial-boot-parameters first-boot-password <password> #First
boot password for product opscenter
  initial-boot-parameters auto-deploy <true/false> #Auto deploys all
 the services of the product else deploys the opscenter only
  initial-boot-parameters single-node <true/false> #True for single
node and false for multi node deployments
  initial-boot-parameters image-pull-secrets
<docker_registry_secrets_name>
  exit
exit
```

Important For clusters managed within the OpenStack environment, you can exclude the *initial-boot* section of *node-defaults* configuration parameters. Also, replace the K8s **vSphere-volume-provider** configuration with K8s **openstack-volume-provider** configuration.

Triggering the Cluster Synchronization

You can trigger the cluster synchronization to complete the vCenter configuration. To synchronize the cluster, use the following configurations:

1. Trigger the cluster synchronization.

```
configure
    clusters cluster_name actions sync run
    clusters cluster_name actions sync run debug true
    clusters cluster_name actions sync logs
    monitor sync-logs cluster_name
    clusters cluster_name actions sync status
    exit
```

Example:

```
SMI Cluster Manager# clusters test1 actions sync run
SMI Cluster Manager# clusters test1 actions sync run debug true
```

```
SMI Cluster Manager# clusters test1 actions sync logs
SMI Cluster Manager# monitor sync-logs test1
SMI Cluster Manager# clusters test1 actions sync status
```

2. Upgrade the cluster using the following configuration.

```
configure
    clusters cluster_name actions sync run upgrade-strategy concurrent
    clusters cluster_name actions sync run upgrade-startegy concurrent
debug true reset-k8s-nodes true
    clusters cluster_name actions sync run upgrade-strategy rolling
   clusters cluster_name actions sync run upgrade-strategy rolling debug
 true reset-k8s-nodes true
    exit
```

Example:

```
SMI Cluster Manager# clusters test1 actions sync run upgrade-strategy concurrent
SMI Cluster Manager# clusters test1 actions sync run upgrade-startegy concurrent debug
true reset-k8s-nodes true
SMI Cluster Manager# clusters test1 actions sync run upgrade-strategy rolling
SMI Cluster Manager# clusters test1 actions sync run upgrade-strategy rolling debug true
 reset-k8s-nodes true
```

3. Redeploy the nodes using the following configuration.

```
configure
    clusters cluster_name actions sync run upgrade-strategy rolling
force-vm-redeploy true debug true
    clusters cluster_name actions sync run force-vm-redeploy true
purge-data-disks true
    exit
```

Example:

```
SMI Cluster Manager# clusters test1 actions sync run upgrade-strategy rolling
force-vm-redeploy true debug true
SMI Cluster Manager# clusters test1 actions sync run force-vm-redeploy true
purge-data-disks true
```

Notes:

- **clusters** *cluster_name* – Specifies the information about the nodes to be deployed. *cluster_name* is the name of the cluster.
- **actions** – Specifies the actions performed on the cluster.
- **sync run** – Triggers the cluster synchronization.
- **sync logs** – Shows the current cluster synchronization logs.
- **sync status** – Shows the current status of the cluster synchronization.
- **debug true** – Enters the debug mode.
- **monitor sync logs** – Monitors the cluster synchronization process.
- **upgrade-strategy concurrent** – This strategy is similar to the existing cluster synchronization where everything is upgraded at once.

- **reset-k8s-nodes** – Resets the K8s on the node instead of deleting and redeploying them all at once.

- **upgrade-strategy rolling** – The rolling upgrade is a new upgrade strategy where upgrades are performed node-by-node.

Note You can use the rolling upgrade strategy to upgrade only the K8s cluster and product. If there are no changes in the product charts, the upgrade fails. For upgrading one node at a time, see Upgrading Node-by-Node (OpenStack) section.

  - **reset-k8s-nodes** – Resets the K8s on that specific node instead of redeploying it.

- **force-vm-redeploy true** – Traverses through each node (one at a time) to delete and upgrade the nodes. The redeploying process is similar to a fresh installation of nodes except for the retention of the data directory, which holds information about the previous installation. Redeploying the node involves:
  - Making API calls for draining and replacing the VMs.
  - Synchronizing (through the Sync API) the node.
  - Verifying the cluster and pod status before proceeding to the next node.

- **purge-data-disks true** – Removes the data disks and makes it as new installation. You can use this option for corrupted *etcds*. For instance, when you have replaced two *etcds* and ended up with the one having the old data, you can purge the disk and reset the cluster completely.

Upgrading Node-by-Node (OpenStack)

You can upgrade nodes within a K8s cluster one at a time using the node-by-node upgrade process. But this upgrade process is limited only to the node level. You cannot run the cluster synchronization and node-by-node synchronization in tandem. It is possible to run the synchronization on two independent nodes, which has been replaced and are ready to go back to the cluster, simultaneously. Also, you can run parts of the synchronization specific to that node (*control plane*, *etcd*, and *worker*) without running a cluster wide synchronization. You need to drain and synchronize a combination of *etcd*, *control plane* and *worker* nodes to begin with the upgrade process.

To upgrade node-by-node, use the following configurations:

1. Drain the node.

   clusters *cluster_name* **nodes** *node_name* **actions sync drain**

   Example:

   ```
   SMI Cluster Manager# clusters test1 nodes etcd1 actions sync drain
   ```

2. Verify the status of the drained node.

   clusters *cluster_name* **nodes** *node_name* **actions sync status**

   Example:

   ```
   SMI Cluster Manager# clusters test1 nodes etcd1 actions sync status
   ```

3. Verify the pod status of the drained node.

clusters *cluster_name* **nodes** *node_name* **actions k8s pod-status**

Example:

```
SMI Cluster Manager# clusters test1 nodes etcd1 actions k8s pod-status
```

4. Run a synchronization on the node.

   clusters *cluster_name* **nodes** *node_name* **actions sync run**

   Example:

   ```
   SMI Cluster Manager# clusters test1 nodes etcd1 actions sync run
   ```

Note The host OS is upgraded during cluster synchronization automatically on Bare Metal deployments.

5. Verify the cluster status. Ensure that the parameter **all-ok** is **true** before proceeding to the next node.

   clusters *cluster_name* **actions k8s cluster-status**

   Example:

   ```
   SMI Cluster Manager# clusters test1 actions k8s cluster-status
    pods-desired-count 26
    pods-ready-count 26
    pods-desired-are-ready true
    etcd-healthy true
    all-ok true
   ```

6. Verify the status of the pods on a specific node before proceeding to the next node.

   clusters *cluster_name* **nodes** *node_name* **actions k8s pod-status show-pod-details**

   Example:

   ```
   SMI Cluster Manager# clusters test1 nodes cmts-worker1 actions k8s pod-status
   show-pod-details
    Value for 'show-pod-details' [false,true]: true
    pods {
     name calico-node-c65zf
     namespace kube-system
     owner-kind DaemonSet
     owner-name calico-node
     ready true
    }
    pods {
     name coredns-6db4464669-k6pqz
     namespace kube-system
     owner-kind ReplicaSet
     owner-name coredns-6db4464669
     ready true
    }
    pods {
     name kube-proxy-tfxcq
     namespace kube-system
     owner-kind DaemonSet
     owner-name kube-proxy
     ready true
    }
    pods {
     name nginx-ingress-controller-6f8f8c4cc7-q5b7c
     namespace nginx-ingress
     owner-kind ReplicaSet
   ```

```
   owner-name nginx-ingress-controller-6f8f8c4cc7
   ready true
  }
  pods {
   name keepalived-cmbnf
   namespace smi-vips
   owner-kind DaemonSet
   owner-name keepalived
   ready true
  }
 pods-count 5
 pods-available-to-drain-count 2
```

Notes:

- **nodes** *node_name* – Specifies the nodes present in the cluster. *node_name* is the name of the node.
- **sync drain** – Drains or cordons the selected node in preparation for an upgrade.
- **sync status** – Shows the status of the drained node.
- **k8s pod-status** – Shows the status of the k8s pods scheduled on the node.
- **sync run** – Upgrades or synchronizes the node.
- **k8s cluster-status** – Shows an overall status of the cluster including pod and *etcd* based statistics.
- **show-pod-details** *true* – Shows the list of pods in addition to the counts.

Deploying and Upgrading the Products in Offline Environments

Using the SMI Cluster Manager, you can download the product's offline TAR ball and the host its charts and corresponding images in the local registries. The SMI Cluster Manager supports the deployment of the product's Ops Center and all the applications and services associated with it. This section describes the procedures involved in deploying and upgrading the products in offline environment using the SMI Cluster Manager.

Perquisites

The prerequisites for deploying and upgrading the products in offline environments are:

1. SMI Cluster Manager AIO.
2. Local repositories that hosts the product offline TAR ball version(s).

Requirements

SMI Cluster Manager (all versions are supported).

Components Used

The following component is used to orchestrate the K8s Cluster and load the products:

- The SMI Cluster Manager.

Deploying the Products in Offline Environments

This section describes the procedures involved in deploying the products in offline environment using the SMI Cluster Manager.

Note From the SMI Cluster Manager perspective, the product refers to Common Execution Environment (CEE). The deployment procedure mentioned in the subsequent section is specific to CEE. However, you can follow the same procedure to deploy 5G Network Functions (SMF or PCF) using the SMI Cluster Manager.

Deploying the Product

To deploy the product, perform the following:

1. Use the following configuration to install the product

```
configure
  software cnf software_name
    url HTTP_HTTPS_File_URL
    user username
    password password
    sha256 sha256_hash
    exit
```

2. Link the CEE into the desired cluster in the **ops-centers** section.

```
configure
  clusters cluster_name ops-center app_name instance_name
  repository-local cnf_repo
  exit
```

3. Download the TAR ball from the URL.

```
software-packages download URL
```

Example:

```
SMI Cluster Manager# software-packages download
http://<ipv4address>:<port_number>/packages/cee-2019-08-21.tar
```

4. Verify whether the TAR balls are loaded.

```
software-packages list
```

Example:

```
SMI Cluster Manager# software-packages list
[ cee-2019-08-21 ]
[ sample ]
```

5. Configure the necessary Ops Center parameters in the required cluster to deploy the product.

```
configure
   cluster cluster_name
      ops-centers app_name instance_name
         repository url
         netconf-ip ipv4_address
```

```
            netconf-port port
            ssh-ip ipv4_address
            ssh-port port
            ingress-hostname <ipv4_address>.<customer_specific_domain_name>
            initial-boot-parameters use-volume-claims true/false
            initial-boot-parameters first-boot-password password
            initial-boot-parameters auto-deploy true/false
            initial-boot-parameters single-node true/false
            initial-boot-parameters image-pull-secrets
            exit
    exit
```

Example:

```
SMI Cluster Manager# config
SMI Cluster Manager(config)# clusters test2
SMI Cluster Manager(config-clusters-test2)# ops-centers cee data
SMI Cluster Manager(config-ops-centers-cee/data)# repository
http://charts.<ipv4address>.<domain_name>/cee-2019-08-21/
SMI Cluster Manager(config-ops-centers-cee/data)# initial-boot-parameters
use-volume-claims false
SMI Cluster Manager(config-ops-centers-cee/data)# initial-boot-parameters
first-boot-password Cisco@123
SMI Cluster Manager(config-ops-centers-cee/data)# initial-boot-parameters auto-deploy
true
SMI Cluster Manager(config-ops-centers-cee/data)# initial-boot-parameters single-node
false
SMI Cluster Manager(config-ops-centers-cee/data)# exit
SMI Cluster Manager(config-clusters-test2)# exit
SMI Cluster Manager(config)#
```

6. Configure the secrets, if your local registry contains secrets.

```
configure
   cluster cluster_name
      secrets docker-registry secret_name
         docker-server server_name
         docker-username username
         docker-password password
         docker-email email
         namespace k8s namespace
         commit
         exit
      exit
```

Example:

```
SMI Cluster Manager# config
SMI Cluster Manager(config)# clusters test2
SMI Cluster Manager(config-clusters-test2)# secrets docker-registry sec1
SMI Cluster Manager(config-docker-registry-sec1)# docker-server serv1
SMI Cluster Manager(config-docker-registry-sec1)# docker-username user1
SMI Cluster Manager(config-docker-registry-sec1)# docker-password Cisco@123
SMI Cluster Manager(config-docker-registry-sec1)# docker-email reg@cisco.com
SMI Cluster Manager(config-docker-registry-sec1)# namespace ns1
SMI Cluster Manager(config-docker-registry-sec1)# exit
SMI Cluster Manager(config-clusters-test2)# exit
SMI Cluster Manager(config)#
```

7. Run the cluster synchronization.

clusters *cluster_name* **actions sync run**

Example:

```
SMI Cluster Manager# clusters test2 actions sync run
```

Notes:

- **software-packages download** *url* – Specifies the software packages to be downloaded through HTTP/HTTPS.
- **software-packages list** – Specifies the list of available software packages.
- **ops-centers** *app_name instance_name* – Specifies the product Ops Center and instance. *app_name* is the application name. *instance_name* is the name of the instance.
- **repository** *url* - Specifies the local registry URL for downloading the charts.
- **netconf-ip** *ipv4_address* – Specifies the Ops Center netconf IPv4 address.
- **netconf-port** *port* – Specifies the Ops Center netconf port number.
- **ssh-ip** *ipv4_address* – Specifies the SSH IPv4 address for the Ops Center.
- **ssh-port** *port* - Specifies the SSH port number for the Ops Center.
- **ingress-hostname** <*ipv4_address*>.<*customer_specific_domain_name*> – Specifies the ingress hostname to be set to the Ops Center. <*customer_specific_domain_name*> specifies the customer's domain name.
- **initial-boot-parameters** – Specifies the initial boot parameters for deploying the helm charts.
  - **use-volume-claims** *true/false* – Specifies the usage of persistent volumes. Set this option to True to use persistent volumes. The default value is true.
  - **first-boot-password** *password* – Specifies the first boot password for the product's Ops Center.
  - **auto-deploy** *true/false* – Auto deploys all the services of the product. Set this option to false to deploy only the product's Ops Center.
  - **single-node** *true/false* – Specifies the product deployment on a single node. Set this option to false for multi node deployments.
  - **image-pull-secrets** – Specifies the docker registry secret name to be used.
- **secrets docker-registry** *secret_name* – Specifies the secret name for your docker registry.
  - **docker-server** *server_name* – Specifies the docker server name.
  - **docker-username** *username* – Specifies the docker registry user name.
  - **docker-password** *password* – Specifies the docker registry password.
  - **docker-email** *email* – Specifies the docker registry email.
  - **namespace** *namespace* – Specifies the docker registry namespace.

NOTES:

- **software cnf** *software_name* - Specifies the Cisco's Cloud Native software. *software_name* is the name of the Cloud Native software.

- **url** *HTTP_HTTPS_File_URL* - Specifies the repository URL.
- **user** *username* - Specifies the username for HTTP/HTTPS authentication.
- **password** *password* - Specifies the password used for downloading the software package.
- **sha256** *sha256_hash* - Specifies the SHA256 hash of the software download.

To deploy the product, perform the following:

1. Login to the SMI Cluster Manager CLI using the ingress URL.

   https://cli.smi-cluster-manager.<*IP_address*>.<*customer_specific_domain_name*>

   NOTES:

   - *customer_specific_domain_name* - Specifies the customer's domain name.

2. Use the following configuration to install the CEE

   ```
   configure
     software cnf software_name
       url HTTP_HTTPS_File_URL
       user username
       password password
       sha256 sha256_hash
       exit
   ```

   NOTES:

   - **software cnf** *software_name* - Specifies the Cisco's Cloud Native software. *software_name* is the name of the Cloud Native software.
     - **url** *HTTP_HTTPS_File_URL* - Specifies the repository URL.
     - **user** *username* - Specifies the username for HTTP/HTTPS authentication.
     - **password** *password* - Specifies the password used for downloading the software package.
     - **sha256** *sha256_hash* - Specifies the SHA256 hash of the software download.

3. Link the product (CEE or Network Functions) into the desired cluster in the **ops-centers**.

   ```
   configure
     clusters cluster_name ops-center app_name instance_name
     repository-local cnf_repo
     exit
   ```

   NOTES:

   - **repository-local** *cnf_repo* - Specifies the CNF repository.

4. Run the cluster synchronization to deploy the CEE Ops Center and wait for the synchronization to complete.

   ```
   clusters cluster_name actions sync run
   ```

   NOTES:

- **clusters** *cluster_name* **actions sync run** - Synchronizes the committed changes to the cluster.

5. Verify the cluster synchronization through cluster sync status or log commands.

   clusters *cluster_name* **actions sync status**
   clusters *cluster_name* **actions sync logs**

   NOTES:

   - **clusters** *cluster_name* **actions sync status** - Displays the status of the cluster synchronization.
   - **clusters** *cluster_name* **actions sync logs**- Displays the logs generated during the cluster synchronization process.

Verifying the Product Deployment

You can verify the status of the product deployment through the product CLI. To verify, use the following commands:

1. Log in to the SMI product CLI. For example, CEE.
2. Verify whether the charts are loaded in the specific instance (verify the namespace).

   show helm charts

   Example:

```
cee# show helm charts
CHART         INSTANCE                              STATUS    VERSION     REVISION RELEASE
                                  NAMESPACE
-------------------------------------------------------------------------------------------
cee-ops-center         cee-global-ops-center             deployed  2023.02.1.d249  1
0.7.0-2023-02-1-0513-230331051211-dec612f  cee-global
cnat-monitoring        cee-global-cnat-monitoring        deployed  2023.02.1.d249  1
0.7.0-2023-02-1-0031-230331183330-58ec41c  cee-global
product-documentation  cee-global-product-documentation  deployed  2023.02.1.d249  1
0.8.0-2023-02-1-0131-230321085503-2699cb5  cee-global
pv-manager             cee-global-pv-manager             deployed  2023.02.1.d249  1
0.3.0-2023-02-1-0029-230320155437-e484272  cee-global
smi-autoheal           cee-global-smi-autoheal           deployed  2023.02.1.d249  1
0.2.0-2023-02-1-0030-230330084451-99684bf  cee-global
smi-show-tac           cee-global-smi-show-tac           deployed  2023.02.1.d249  1
0.4.0-2023-02-1-0189-230331050005-81130f1  cee-global
storage-provisioner    cee-global-storage-provisioner    deployed  2023.02.1.d249  1
0.3.0-2023-02-1-0120-230320160505-1597fdb  cee-global
telegraf-monitoring    cee-global-telegraf-monitoring    deployed  2023.02.1.d249  1
0.1.0-2023-02-1-0048-230330084426-9b02da0  cee-global
```

3. Verify the status of the system.

   show system status

   Example:

```
cee# show system status
system status deployed true
system status percent-ready 100.0
```

Notes:

- **show helm charts** – Displays the helm release details.

- **show system status** – Displays the status of the system.

Upgrading the Products in Offline Environments

Using the SMI Cluster Manager, you can upgrade the product's Ops Center and all the applications and services associated with it. This section describes the procedures involved in upgrading the products in offline environment using the SMI Cluster Manager.

Upgrading the Product

To upgrade the product, perform the following:

1. Download the latest TAR ball from the URL.

   software-packages download *URL*

   Example:

   ```
   SMI Cluster Manager# software-packages download
   http://<ipv4address>:<port_number>/packages/cee-2019-08-21.tar
   ```

2. Verify whether the TAR balls are loaded.

   software-packages list

   Example:

   ```
   SMI Cluster Manager# software-packages list
   [ cee-2019-08-21 ]
   [ sample ]
   ```

3. Update the repository URL to point the correct product chart release to upgrade the product.

   ```
   configure
      cluster cluster_name
         ops-centers app_name instance_name
            repository url
            exit
         exit
   ```

   Example:

   ```
   SMI Cluster Manager# config
   SMI Cluster Manager(config)# clusters test2
   SMI Cluster Manager(config-clusters-test2)# ops-centers cee data
   SMI Cluster Manager(config-ops-centers-cee/data)# repository
   http://charts.<ipv4address>.<domain_name>/cee-2019-08-21/
   SMI Cluster Manager(config-ops-centers-cee/data)# exit
   SMI Cluster Manager(config-clusters-test2)# exit
   ```

4. Configure the secrets, if your local registry contains secrets.

   ```
   configure
      cluster cluster_name
         secrets docker-registry secret_name
            docker-server server_name
            docker-username username
            docker-password password
            docker-email email
   ```

```
        namespace k8s namespace
        commit
        exit
    exit
```

Example:

```
SMI Cluster Manager# config
SMI Cluster Manager(config)# clusters test2
SMI Cluster Manager(config-clusters-test2)# secrets docker-registry sec1
SMI Cluster Manager(config-docker-registry-sec1)# docker-server serv1
SMI Cluster Manager(config-docker-registry-sec1)# docker-username user1
SMI Cluster Manager(config-docker-registry-sec1)# docker-password Cisco@123
SMI Cluster Manager(config-docker-registry-sec1)# docker-email reg@cisco.com
SMI Cluster Manager(config-docker-registry-sec1)# namespace ns1
SMI Cluster Manager(config-docker-registry-sec1)# exit
SMI Cluster Manager(config-clusters-test2)# exit
SMI Cluster Manager(config)#
```

5. Run the cluster synchronization.

   clusters *cluster_name* **actions sync run**

   Example:

   ```
   SMI Cluster Manager# clusters test2 actions sync run
   ```

Verifying the Product Upgrade

You can verify the status of the product upgrade through the product CLI. To verify, use the following commands:

1. Log in to the SMI product CLI. For example, CEE.
2. Verify whether the charts are loaded in the specific instance (verify the namespace).

   show helm charts

   Example:

   ```
   cee# show helm charts
   CHART         INSTANCE                              STATUS    VERSION     REVISION RELEASE
                                    NAMESPACE
   ------------------------------------------------------------------------------------------
   cee-ops-center         cee-global-ops-center             deployed  2023.02.1.d249  1
   0.7.0-2023-02-1-0513-230331051211-dec612f  cee-global
   cnat-monitoring        cee-global-cnat-monitoring        deployed  2023.02.1.d249  1
   0.7.0-2023-02-1-0031-230331183330-58ec41c  cee-global
   product-documentation  cee-global-product-documentation  deployed  2023.02.1.d249  1
   0.8.0-2023-02-1-0131-230321085503-2699cb5  cee-global
   pv-manager             cee-global-pv-manager             deployed  2023.02.1.d249  1
   0.3.0-2023-02-1-0029-230320155437-e484272  cee-global
   smi-autoheal           cee-global-smi-autoheal           deployed  2023.02.1.d249  1
   0.2.0-2023-02-1-0030-230330084451-99684bf  cee-global
   smi-show-tac           cee-global-smi-show-tac           deployed  2023.02.1.d249  1
   0.4.0-2023-02-1-0189-230331050005-81130f1  cee-global
   storage-provisioner    cee-global-storage-provisioner    deployed  2023.02.1.d249  1
   0.3.0-2023-02-1-0120-230320160505-1597fdb  cee-global
   telegraf-monitoring    cee-global-telegraf-monitoring    deployed  2023.02.1.d249  1
   0.1.0-2023-02-1-0048-230330084426-9b02da0  cee-global
   ```

3. Verify the status of the system.

show system status

Example:

```
cee# show system status
system status deployed true
system status percent-ready 100.0
```

Notes:

- **show helm charts** – Displays the helm release details.
- **show system status** – Displays the status of the system.

Rollback to an Earlier Version

To rollback to an earlier version of the product, perform the following:

1. Update the repository URL to point an earlier version of the product chart release to rollback the product.

   configure
   cluster *cluster_name*
   ops-centers *app_name instance*
   repository *url*
   exit
   exit

   Example:

   ```
   SMI Cluster Manager# config
   SMI Cluster Manager(config)# clusters test2
   SMI Cluster Manager(config)# ops-centers cee data
   SMI Cluster Manager(config-clusters-test2)# ops-centers cee data
   SMI Cluster Manager(config-ops-centers-cee/data)# repository
   http://charts.<ipv4address>.<domain_name>/cee-2019-07-22/
   SMI Cluster Manager(config-ops-centers-cee/data)# exit
   SMI Cluster Manager(config-clusters-test2)# exit
   ```

2. Configure the secrets, if your local registry contains secrets.

   configure
   cluster *cluster_name*
   secrets docker-registry *secret_name*
   docker-server *server_name*
   docker-username *username*
   docker-password *password*
   docker-email *email*
   namespace k8s *namespace*
   commit
   exit
   exit

   Example:

   ```
   SMI Cluster Manager# config
   SMI Cluster Manager(config)# clusters test2
   SMI Cluster Manager(config-clusters-test2)# secrets docker-registry sec1
   SMI Cluster Manager(config-docker-registry-sec1)# docker-server serv1
   SMI Cluster Manager(config-docker-registry-sec1)# docker-username user1
   ```

```
SMI Cluster Manager(config-docker-registry-sec1)# docker-password Cisco@123
SMI Cluster Manager(config-docker-registry-sec1)# docker-email reg@cisco.com
SMI Cluster Manager(config-docker-registry-sec1)# namespace ns1
SMI Cluster Manager(config-docker-registry-sec1)# exit
SMI Cluster Manager(config-clusters-test2)# exit
SMI Cluster Manager(config)#
```

3. Run the cluster synchronization.

   clusters *cluster_name* **actions sync run**

   Example:

   ```
   SMI Cluster Manager# clusters test2 actions sync run
   ```

Viewing Pod Details

You can view the details of the current pods through CEE Ops Center. To view the pod details, use the following command (in CEE Ops Center CLI):

cluster pods *instance_name pod_name* **detail**

Note

- **cluster** – Specifies the K8s cluster.
- **pods** – Specifies the current pods in the cluster.
- *instance_name* – Specifies the name of the instance.
- *pod_name* – Specifies the name of the pod.
- **detail** – Displays the details of the specified pod.

The following example displays the details of the pod named *alertmanager-0* in the *cee-data* instance.

Example:

```
cee# cluster pods cee-data alertmanager-0 detail
details apiVersion: "v1"
kind: "Pod"
metadata:
  annotations:
    alermanager.io/scrape: "true"
    cni.projectcalico.org/podIP: "192.168.1.137/32"
    config-hash: "5532425ef5fd02add051cb759730047390b1bce51da862d13597dbb38dfbde86"
  creationTimestamp: "2020-02-26T06:09:13Z"
  generateName: "alertmanager-"
  labels:
    component: "alertmanager"
    controller-revision-hash: "alertmanager-67cdb95f8b"
    statefulset.kubernetes.io/pod-name: "alertmanager-0"
  name: "alertmanager-0"
  namespace: "cee"
  ownerReferences:
  - apiVersion: "apps/v1"
    kind: "StatefulSet"
    blockOwnerDeletion: true
    controller: true
    name: "alertmanager"
    uid: "82a11da4-585e-11ea-bc06-0050569ca70e"
```

```
  resourceVersion: "1654031"
  selfLink: "/api/v1/namespaces/cee/pods/alertmanager-0"
  uid: "82aee5d0-585e-11ea-bc06-0050569ca70e"
spec:
  containers:
  - args:
    - "/alertmanager/alertmanager"
    - "--config.file=/etc/alertmanager/alertmanager.yml"
    - "--storage.path=/alertmanager/data"
    - "--cluster.advertise-address=$(POD_IP):6783"
    env:
    - name: "POD_IP"
      valueFrom:
        fieldRef:
          apiVersion: "v1"
          fieldPath: "status.podIP"
    image: "<path_to_alert_manager_image>"
    imagePullPolicy: "IfNotPresent"
    name: "alertmanager"
    ports:
    - containerPort: 9093
      name: "web"
      protocol: "TCP"
    resources: {}
    terminationMessagePath: "/dev/termination-log"
    terminationMessagePolicy: "File"
    volumeMounts:
    - mountPath: "/etc/alertmanager/"
      name: "alertmanager-config"
    - mountPath: "/alertmanager/data/"
      name: "alertmanager-store"
    - mountPath: "/var/run/secrets/kubernetes.io/serviceaccount"
      name: "default-token-kbjnx"
      readOnly: true
  dnsPolicy: "ClusterFirst"
  enableServiceLinks: true
  hostname: "alertmanager-0"
  nodeName: "for-smi-cdl-1b-worker94d84de255"
  priority: 0
  restartPolicy: "Always"
  schedulerName: "default-scheduler"
  securityContext:
    fsGroup: 0
    runAsUser: 0
  serviceAccount: "default"
  serviceAccountName: "default"
  subdomain: "alertmanager-service"
  terminationGracePeriodSeconds: 30
  tolerations:
  - effect: "NoExecute"
    key: "node-role.kubernetes.io/oam"
    operator: "Equal"
    value: "true"
  - effect: "NoExecute"
    key: "node.kubernetes.io/not-ready"
    operator: "Exists"
    tolerationSeconds: 300
  - effect: "NoExecute"
    key: "node.kubernetes.io/unreachable"
    operator: "Exists"
    tolerationSeconds: 300
  volumes:
  - configMap:
      defaultMode: 420
```

```
      name: "alertmanager"
    name: "alertmanager-config"
  - emptyDir: {}
    name: "alertmanager-store"
  - name: "default-token-kbjnx"
    secret:
      defaultMode: 420
      secretName: "default-token-kbjnx"
status:
  conditions:
  - lastTransitionTime: "2020-02-26T06:09:02Z"
    status: "True"
    type: "Initialized"
  - lastTransitionTime: "2020-02-26T06:09:06Z"
    status: "True"
    type: "Ready"
  - lastTransitionTime: "2020-02-26T06:09:06Z"
    status: "True"
    type: "ContainersReady"
  - lastTransitionTime: "2020-02-26T06:09:13Z"
    status: "True"
    type: "PodScheduled"
  containerStatuses:
  - containerID: "docker://821ed1a272d37e3b4c4c9c1ec69b671a3c3fe6eb4b42108edf44709b9c698ccd"

    image: "<path_to_alert_manager_image>"
    imageID:
"docker-pullable:/<path_to_alert_manager_directory>@sha256:c4bf05aa677a050fba9d86586b04383ca089bd784d2cb9e544b0d6b7ea899d9b"

    lastState: {}
    name: "alertmanager"
    ready: true
    restartCount: 0
    state:
      running:
        startedAt: "2020-02-26T06:09:05Z"
  hostIP: "<host_ipv4address>"
  phase: "Running"
  podIP: "<pod_ipv4address>"
  qosClass: "BestEffort"
  startTime: "2020-02-26T06:09:02Z"
cee#
```

Configuring the Local NTP Server with Authentication and Tracking

This section describes how to configure the local NTP server with authentication and tracking.

Configuring the Local NTP Server

On a network with multiple systems, it's always recommended to set up a single system as the NTP server for all the other local systems. The cloud providers follow the same model to run their own NTP pools within their data centers. The benefits of following this model include, reduced load on external connections and remote NTP servers, proper synchronization of the local systems with each other even when the external connection or servers goes down.

You can enable a local server in the configuration file.

1. Specify the network and subnet from where the connections arrive to enable the local server. In addition, you can create an access list and test it on the server using the following command:

   accheck *address*

   For instance, you can use the following configuration to allow connections from 192.168.2.0/24 and all of the 10.0.0.0/8 subnet:

   ```
   allow 192.168.2.0/24
   allow 10.0.0.0/8
   ```

2. Restart the *Chrony* service for the configuration to take effect as shown in the following sample configuration file.

   ```
   user1@cluster-manager:~$ vi /etc/chrony/chrony.conf
   user1@cluster-manager:~$ sudo vi /etc/chrony/chrony.conf
   user1@cluster-manager:~$ sudo systemctl daemon-reload
   user1@cluster-manager:~$ sudo systemctl restart chrony
   user1@cluster-manager:~$ sudo systemctl status chrony
    chrony.service - chrony, an NTP client/server
      Loaded: loaded (/lib/systemd/system/chrony.service; enabled; vendor preset: enabled)

      Active: active (running) since Tue 2019-11-19 17:54:26 UTC; 10s ago
        Docs: man:chronyd(8)
              man:chronyc(1)
              man:chrony.conf(5)
     Process: 14237 ExecStartPost=/usr/lib/chrony/chrony-helper update-daemon (code=exited,
    status=0/SUCCESS)
     Process: 14196 ExecStart=/usr/lib/systemd/scripts/chronyd-starter.sh $DAEMON_OPTS
   (code=exited, status=0/SUCCESS)
    Main PID: 14233 (chronyd)
       Tasks: 1 (limit: 4915)
      CGroup: /system.slice/chrony.service
              └─14233 /usr/sbin/chronyd

   Nov 19 17:54:26 cluster-manager systemd[1]: Starting chrony, an NTP client/server...
   Nov 19 17:54:26 cluster-manager chronyd[14233]: chronyd version 3.2 starting (+CMDMON
   +NTP +REFCLOCK +RTC +PRIVDROP +SCFILTER +SECHASH +SIGND +ASYNCDNS +IPV6 -DEBUG)
   Nov 19 17:54:26 cluster-manager chronyd[14233]: Frequency -12.134 +/- 0.024 ppm read
   from /var/lib/chrony/chrony.drift
   Nov 19 17:54:26 cluster-manager systemd[1]: Started chrony, an NTP client/server.
   Nov 19 17:54:31 cluster-manager chronyd[14233]: Selected source 171.68.38.65
   ```

3. Verify the connection to the server using the **chronyc activity** command. Also, the **chronyc clients** command allows you to view the list of clients connected to the server. In the following example, you can verify the server connection and clients connected to it.

   ```
   user1@cluster-manager:~$ sudo chronyc activity
   200 OK
   2 sources online
   0 sources offline
   0 sources doing burst (return to online)
   0 sources doing burst (return to offline)
   0 sources with unknown address
   user1@cluster-manager:~$ sudo chronyc clients
   Hostname                      NTP   Drop Int IntL Last     Cmd   Drop Int  Last
   ===============================================================================
   ```

   Note There are no clients displayed in the clients list since none of them are configured currently.

4. Enter the local server in Cluster Manger configuration.

5. Run the cluster synchronization as follows.

```
configure
   node-defaults os ntp servers clock.cisco.com
   exit
```

6. Verify the clients on the local server after all the nodes synchronize successfully.

```
user1cluster-manager:~$ sudo chronyc clients
Hostname                      NTP   Drop Int IntL Last     Cmd   Drop Int  Last
===============================================================================
192.168.2.109                  11      0   6   -     7       0      0   -     -
192.168.2.110                   9      0   6   -    49       0      0   -     -
192.168.2.111                   8      0   6   -    52       0      0   -     -
192.168.2.107                   4      0   1   -    59       0      0   -     -
192.168.2.108                   4      0   1   -    59       0      0   -     -
192.168.2.106                   4      0   1   -    58       0      0   -     -
192.168.2.51                    4      0   1   -    58       0      0   -     -
192.168.2.53                    4      0   1   -    58       0      0   -     -
192.168.2.52                    4      0   1   -    58       0      0   -     -
```

7. Alternatively, you can verify the sources on any of the nodes in the cluster and track the status of the synchronization using the **chronyc sources** and **chronyc tracking** commands.

```
user1@kali-worker3:~$ sudo chronyc sources
210 Number of sources = 1
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================

192.168.2.56                 2   6   377    25    -13us[  -18us] +/- 1106us

user1-cloud@kali-worker3:~$ sudo chronyc tracking
Reference ID : AC161238 (192.168.2.56)
Stratum : 3
Ref time (UTC) : Tue Nov 19 19:07:07 2019
System time : 0.000000037 seconds slow of NTP time
Last offset : +0.000035999 seconds
RMS offset : 0.000020778 seconds
Frequency : 13.682 ppm slow
Residual freq : +0.084 ppm
Skew : 0.228 ppm
Root delay : 0.001795322 seconds
Root dispersion : 0.000163470 seconds
Update interval : 64.2 seconds
Leap status : Normal
```

Configuring the Local NTP Server with Authentication

Generally, all cloud providers set up authentication for their local servers. To verify the authentication status of the local servers, use the **chronyc** command as follows:

```
user1@kali-worker3:~$ sudo chronyc ntpdata 192.168.2.56 | grep Authenticated
Authenticated   : No
```

To secure your local servers:

1. Select the key-id and n-bit Secure Hash Algorithm (SHA) key.
   - The following example shows the default key generation.

```
user1@cluster-manager:~$ sudo chronyc keygen
1 SHA1 HEX:959623F106595B9E75BE328C265CA9C86560D88E
```

- The following example shows the key generation with key-id 27 and 512 bit SHA key.

```
user1@cluster-manager:~$ sudo chronyc keygen 27 SHA512 512
27 SHA512
HEX:80E68E6AEB1B994217282568AF2A0EA8E4731F6CDC5CC5635C799676864BD68B4317FA897B54F10DCFE8F5F36
7E03626ACD0A5048BAA8E1A615A44C4FCF731B3
```

2. Add the keys to the **/etc/chrony/chrony.keys** file to configure the authentication.
3. Restart the *Chrony* as follows.

```
user1@cluster-manager:~$ sudo systemctl daemon-reload
user1-cloud@cluster-manager:~$ sudo systemctl restart chrony
user1-cloud@cluster-manager:~$ sudo systemctl status chrony
● chrony.service - chrony, an NTP client/server
   Loaded: loaded (/lib/systemd/system/chrony.service; enabled; vendor preset: enabled)

   Active: active (running) since Tue 2019-11-19 19:29:08 UTC; 8s ago
     Docs: man:chronyd(8)
           man:chronyc(1)
           man:chrony.conf(5)
  Process: 20452 ExecStartPost=/usr/lib/chrony/chrony-helper update-daemon (code=exited,
 status=0/SUCCESS)
  Process: 20406 ExecStart=/usr/lib/systemd/scripts/chronyd-starter.sh $DAEMON_OPTS
(code=exited, status=0/SUCCESS)
 Main PID: 20445 (chronyd)
    Tasks: 1 (limit: 4915)
   CGroup: /system.slice/chrony.service
           └─20445 /usr/sbin/chronyd

Nov 19 19:29:08 cluster-manager systemd[1]: Starting chrony, an NTP client/server...
Nov 19 19:29:08 cluster-manager chronyd[20445]: chronyd version 3.2 starting (+CMDMON
+NTP +REFCLOCK +RTC +PRIVDROP +SCFILTER +SECHASH +SIGND +ASYNCDNS +IPV6 -DEBUG)
Nov 19 19:29:08 cluster-manager chronyd[20445]: Frequency -12.095 +/- 0.044 ppm read
from /var/lib/chrony/chrony.drift
Nov 19 19:29:08 cluster-manager systemd[1]: Started chrony, an NTP client/server.
Nov 19 19:29:13 cluster-manager chronyd[20445]: Selected source 171.68.38.65
```

4. Update the NTP configuration in the cluster manager.
5. Run the cluster synchronization as follows.

```
node-defaults os ntp servers 192.168.2.56
  key-id   27
  sha-type SHA512
  sha-key
80E68E6AEB1B994217282568AF2A0EA8E4731F6CDC5CC5635C799676864BD68B4317FA897B54F10DCFE8F5F36
7E03626ACD0A5048BAA8E1A615A44C4FCF731B3
 exit
```

6. Verify the authentication status on all the nodes connected to the local server after synchronization.

```
user1@kali-worker3:~$ sudo chronyc ntpdata 192.168.2.56 | grep Authenticated
Authenticated   : Yes
```

K8s Certificates Auto-Renewal

Certificate Management with Kubeadm

In kubeadm v1.21.0, client certificates generated by kubeadm expire after 1 year. The root certificates expires in 10 years. This feature enables monitoring and automatic renewal of kubeadm certificates before the expiry date from the CM or CEE. The CEE triggers an alert to notify the user of any certificate that is going to expire in 30 days.

The smi-cluster-maintainer pod monitors the k8s certificates and automate the renewal process, regardless of the cluster sync.

How it Works

This section describes the sequence of operation for the feature.

1. The certificates in CM managed K8s clusters, control planes, workers, and external ETCD nodes is checked every 12 hours.
2. If any certificate is expiring in 60 days on the nodes, then the auto-renew process is triggered.
   - If the renewal is successful, then the following checks shows all the certificates as valid.
   - If the renewal is unsuccessful, then the auto-renew process is re-initiated for the next cycle or iteration of validating the certificates.
3. If any certificate is expiring in 30 days on the nodes, then the auto-renew process is triggered along with sending an alert to the user.

   In such cases, a manual intervention might be required to renew the certificates, which are nearing their expiry date.

   The kubernetes certificate expiry alert is show below.

   Rules:

   - **Alert:** kube_certificate_expiring
     - **Annotations:**
       - **Type:** Kubernetes Certificate Expiring Alarm
       - **Summary:** "Kubernetes certificate {{ $labels.cert_path }} on host: {{ $labels.node_name }} is expiring in {{ $labels.days_to_expiry }} days."
     - **Expression:**

       ```
       |
            kube_certificate_expiring != 0
       ```

     - **Labels:**
       - **Severity:** critical

Note The certificate auto-renewal process must restart the api-server. You might experience a temporary k8s API downtime during the certificate auto-renewal process.

Ops Center Converged Core Naming Convention Support

Feature Description

The SMF Ops Center name has changed to 'cn' from 'SMF' to accommodate the Ops Centers for SMF, cnSGW, or a combination of both. This name change breaks backward compatibility for the pods with earlier versions in relation to namespaces and IMS nodes.

This feature provides a new optional CLI configuration 'app-name-override' which ensures backward compatibility. The package name is derived by using this new field and is used in the URL to download the correct package. The previous method to derive namespaces is used to ensure that the existing namespaces remain unchanged.

Configuring the Ops Center Converged Core Naming Convention Support

This section describes a sample CLI command configuration for configuring the feature.

```
ops-centers smf data
  app-name-override       cn
  repository-local        smfi25
  sync-default-repository true
  netconf-ip              209.165.200.224
  netconf-port            2026
  ssh-ip                  209.165.200.224
  ssh-port                2028
  ingress-hostname        209.165.200.224.nip.io
  initial-boot-parameters use-volume-claims false
  initial-boot-parameters first-boot-password $8$5WnH/gUgKtPAPfXdU8CaURcKnaQOAc9imkIBHnjZLM=

  initial-boot-parameters auto-deploy true
  initial-boot-parameters single-node true
exit
```

NOTES:

- <*app-name*>: Mention the application name. In the above code sample, it is **smf**.
- <*instance*>: Mention the instance name. In the above code sample, it is **data**.
- **app-name-override** <*override_value*>: Mention the override value. In the above code sample, it is **cn**.

To download the ops center packages, the default behavior is to derive the ops center package name as **smf-ops-center**. In this release, if the app-name-override value is set to 'cn', the new behaviour is to derive the package name as **cn-ops-center**.

Note This feature ensures that you keep the existing namespaces, yet use the converged ops center name 'cn' from the override field to ensure backward compatibility for the NF pods, which were deployed in previous releases.

Docker Subnet Override Support

Feature Description

By default, Docker uses the subnet range, 172.17.0.0/16 for container networking. If the same subnet range or an IP address from the range is already being used by some other resource in the same cluster environment, it might lead to a conflict.

This feature enables the user to configure and override the default value for the Docker subnet used by the SMI Cluster Manager (CM) or Inception VM. For the CM, this configuration is set by using the CM Ops-Center, whereas the Inception VM uses the deploy.yaml file to achieve the same configuration.

The deploy.yaml is enhanced to contain additional parameter, **configuration** with a sub-parameter, **docker-address-pools**. This YAML file contains a **base** for the CIDR range to use and a **size** for the size of the subnet to reserve for the new network.

Configuring the Docker Subnet Override

This section describes the configuration details for the Docker subnet override feature.

Use the following command to configure the Docker subnet override feature.

configuration docker-address-pools pool-name *docker_bridge_address_pool_name* **[base** *docker_bridge_subnet* **| size** *size* **]**

base ***docker_bridge_subnet***

Specify the docker bridge subnet.

Must be a string in the ipv4-address-and-prefix-length pattern.

-Or-

Must be a string in the ipv6-address-and-prefix-length pattern.

Default Value: 172.17.0.0/16.

pool-name *docker_bridge_address_pool_name*

Specify the pool name of the docker bridge address pool.

Must be a string.

size *size*

Specify the size. For example, 16, 24, etc.

Must be an integer in the range of 8-24.

Default Value: 24.

IPSec Support for SMF N4 Interfaces

Feature Summary and Revision History

Summary Data

Applicable Product (s) or Functional Area	KVM-based application deployment support
Applicable Platforms	Bare Metal, OpenStack, VMware
Feature Default Setting	Disabled – Configuration Required
Related Changes in this Release	Not Applicable
Related Documentation	*UCC SMI Operations Guide*

Revision History

Revision Details	Release
Added support for the following functionality: • IPSec Monitoring • Configuring IPSec certificates under strongSwan configuration	2023.01.0
First introduced.	2020.02.2.47

Feature Description

This feature introduces strongSwan, a keying daemon, which uses the Internet Key Exchange (IKE) protocols, IKEv1 and IKEv2, to establish **security associations** (SA) between two peers in a network. Such an IKE session is denoted as **IKE_SA** in this chapter. The IKE provides strong authentication for both peers and derives unique cryptographic session keys. Besides authentication and key material, IKE also provides the means to exchange configuration information and to negotiate IPsec SAs, which are often called as **CHILD_SAs**. IPsec SAs define which network traffic is to be secured and how it has to be encrypted and authenticated.

The strongSwan feature is available as an add-on from the Cluster Manager (CM). Use the CM Ops-Center to configure this add-on. In the current release, the SMI uses strongSwan version 5.9.3.

SMI allows monitoring of IPSec certificates—sends certificate expiry alerts and updates certificate through strongSwan configuration.

Configuration Parameters

In this section, see the description for different configuration parameters available for the strongSwan add-on feature. Use the CM Ops-Center to configure these parameters.

- **name**: Specifies the name of the connection, which can be used for connection specific operations, for example, up or down.
- **auto { ignore |add |route |start }**: Specifies the operation, if any, that should be automatically performed at IPsec startup. The **add** option loads a connection without starting it, whereas **route** loads a connection and installs kernel traps. If traffic is detected between the leftsubnet and rightsubnet, a connection is established. The **start** option loads a connection and brings it up immediately. The **ignore** option ignores the connection and is the same as deleting a connection from the config file.

  The default value is **ignore**.
- **keyexchange { ikev1 | ikev2 }** : Specifies the method of key exchange and the protocol to use to initialize the connection.
- **type { tunnel | transport | transport_proxy | passthrough | drop }**: Specifies the type of the connection. Currently, the accepted values are **tunnel**, signifying a host-to-host, host-to-subnet, or subnet-to-subnet tunnel. The **transport** option signifies a host-to-host transport mode, whereas the **transport_proxy** option signifies the special Mobile IPv6 transport proxy mode. The **passthrough** option signifies that no IPsec processing should be done at all and **drop** signifies that packets must be discarded.
- **left** or **right { ip address** *ip_address* **| fqdn** *fqdn* **| %any | %any4 | %any6 | range | subnet }**: Specifies the IP address or FQDN of the participant public-network interface. The value **%any** for the local endpoint signifies an address to be filled in (by automatic keying) during negotiation. If the local peer initiates the connection setup, then the routing table is queried to determine the correct local IP address. If the local peer is responding to a connection setup, then any IP address that is assigned to a local interface is accepted. The value **%any4** restricts address selection to IPv4 addresses and **%any6** restricts address selection to IPv6 addresses.
- **leftsubnet** or **rightsubnet** *ip subnet*: Specifies the private subnet behind the left participant, expressed as either network or netmask.
- **leftid** or **rightid** *id*: Specifies how the left or right participant must be identified for authentication. The default values are left or right or the subject of the certificate configured. It must match the full subject DN or one of the subjectAltName extensions contained in the certificate.
- **leftsendcert { never | no | ifasked | always | yes }**: Defines whether a peer must send a certificate request (CR) payload in order to get a certificate in return.
- **leftauth** or **rightauth{ pubkey | psk | eap | xauth }**: Specifies the authentication method to use locally (left) or require from the remote (right) side. The acceptable values are **pubkey** for public key encryption (RSA/ECDSA), **psk** for pre-shared key authentication, **eap** to use the Extensible Authentication Protocol, and **xauth** for IKEv1 eXtended Authentication.

  Pubkey is the default option.
- **psk** *pre-shared key*: Specifies the required setting if leftauth or rightauth is configured as **psk**.
- **esp { cipher suites | aes128-sha256 }**: A comma-separated list of ESP encryption or authentication algorithms is used for the connection, for example, **aes128-sha256**. The notation is encryption-integrity[-dhgroup][-esnmode]. For IKEv2, multiple algorithms (separated by -) of the same type can be included in a single proposal. IKEv1 only includes the first algorithm in a proposal.

  aes128-sha256 is the default option.
- **ike { cipher suites | aes128-sha256-modp3072 }**: A comma-separated list of IKE/ISAKMP SA encryption or authentication algorithms is used, for example, **aes128-sha256-modp3072**.

The notation is encryption-integrity[-prf]-dhgroup. In IKEv2, multiple algorithms and proposals might be included, such as aes128-aes256-sha1-modp3072-modp2048 or 3des-sha1-md5-modp1024.

- **ikelifetime { time** *time* **| 3h }**: Specifies how long the keying channel of a connection (ISAKMP or IKE SA) must last before being renegotiated.
- **lifetime { time** *time* **| 1h }**: Specifies how long a particular instance of a connection should last, from successful negotiation to expiry.
- **dpdaction { none | clear | hold | restart }**: Specifies the action to be taken when dead peer is detected.

  none is the default value.
- **dpddelay { time** *time* **| 30s }**: Defines the period time interval with which INFORMATIONAL exchanges are sent to the peer. These are only sent if no other traffic is received.
- **dpdtimeout { time** *time* **| 150s }**: Defines the timeout interval after which, all the connections to a peer are deleted in case of inactivity.
- **inactivity time** *time*: Defines the timeout interval after which, a CHILD_SA is closed if it did not send or receive any traffic.
- **closeaction { none | clear | hold | restart }**: Defines the action to take if the remote peer unexpectedly closes a CHILD_SA (see **dpdaction** for the description of different options). If the peer uses reauthentication or uniqueids checking, **closeaction** must not be used, these events might trigger the defined action when it's not desired.
- **nodes** *list_of_node_names*: Specifies the node names on which IPSec connection must be established.
- **serverCert** *server_certificate*: Specifies the content of Server certificate in the **pem** format to be used for this connection.

Note This keyword is not supported under strongSwan configuration.

- **serverPrivKey** *server_private_key*: Specifies the content of server private key in the **pem** format to be used for this connection.

Note This keyword is not supported under strongSwan configuration.

- **serverPrivKeyPassphrase** *passphrase*: Specifies the passphrase used to encrypt the **server-priv-key** value.
- **server-secret**: Pass an existing TLS secret for this connection.

Installing strongSwan

This section describes how to install the strongSwan feature.

Install strongSwan as an Add-on from the CM

Use the following steps to install strongSwan as an add-on from the CM Ops-Center:

1. Use the following CLI commands to enable the strongSwan add-on:

   clusters *cluster_name* **addons strongswan enabled**

2. Set all the strongSwan parameters for **connection** (refer to the *Configuration Parameters* section for more details on available parameters).

3. Trigger the cluster sync operation.

Note The strongSwan pods run on all the nodes, however traffic is accepted only on those nodes, which are configured by using the "nodes" parameter in the CM Ops-Center. strongSwan does not accept or send any traffic on non-configured nodes.

Configuring IPSec Certificates

To configure IPSec certificates under strongSwan configuration, use the following procedure:

1. Create TLS associated secret for server and CA certificate.

   Note: Create strongSwan-related secrets inside the smi-strongswan namespace.

   Example:

```
[test-cm-controlplane] SMI Cluster Deployer# show running-config clusters secrets ca-cert
clusters test-aio
 secrets ca-cert smi-strongswan 134-ca
  certificate "-----BEGIN
CERTIFICATE-----\nMIIDqzCzQubm..................1Ac1L+s4M3ug==\n-----END
CERTIFICATE-----\n"
 exit
 secrets ca-cert smi-strongswan 135-ca
  certificate "-----BEGIN
CERTIFICATE-----\nMIIFqzCCA5Og..................9XdMDiQANHgf7w\n-----END
CERTIFICATE-----\n"
 exit
 secrets ca-cert smi-strongswan ca-1
  certificate "-----BEGIN
CERTIFICATE-----\nMIID0TCCArmg..................UNvF0nAmIX0qxg4\n-----END
CERTIFICATE-----\n"
 exit
 secrets ca-cert smi-strongswan ca-2
  certificate "-----BEGIN PRIVATE
KEY-----\nMIIEvQIBADAN..................tbNDzGAnF29nus=\n-----END PRIVATE KEY-----\n"
 exit
exit
```

2. Refer the secrets in strongSwan configuration. The strongSwan configuration shows the available TLS and certificates.

   Example:

```
[test-cm-controlplane] SMI Cluster Deployer# show running-config clusters karan-aio
strongswan connections server-secret
clusters test-aio
 strongswan connections a-to-b
  server-secret a-to-b
 exit
exit
```

```
[test-cm-controlplane] SMI Cluster Deployer# show running-config clusters karan-aio
strongswan ca-certs
clusters test-aio
strongswan ca-certs [ 134-ca 135-ca ]
exit
```

Parallel Node Upgrade with Deployment Zone Strategy

Feature Summary and Revision History

Summary Data

Applicable Product (s) or Functional Area	KVM-based application deployment support K8s-based application deployment support
Applicable Platforms	Bare Metal, OpenStack, VMware
Feature Default Setting	Disabled – Configuration Required
Related Changes in this Release	Not Applicable
Related Documentation	*UCC SMI Operations Guide*

Revision History

Revision Details	**Release**
First introduced.	2020.02.3.10

Feature Description

The current in-service upgrade strategy only supports upgrading one node at a time. For bigger clusters, more than six nodes, this upgrade strategy leads to longer upgrade periods, which mostly exceed the maintenance window (MW) limits.

This feature enables you to perform parallel upgrades for multiple nodes concurrently for faster in-service upgrades without impacting the availability and replication for any NF.

Architecture

This following images show the high-level design of the group upgrade flow for K8s and KVMs.

Figure 1: Upgrade Flow for K8s Clusters

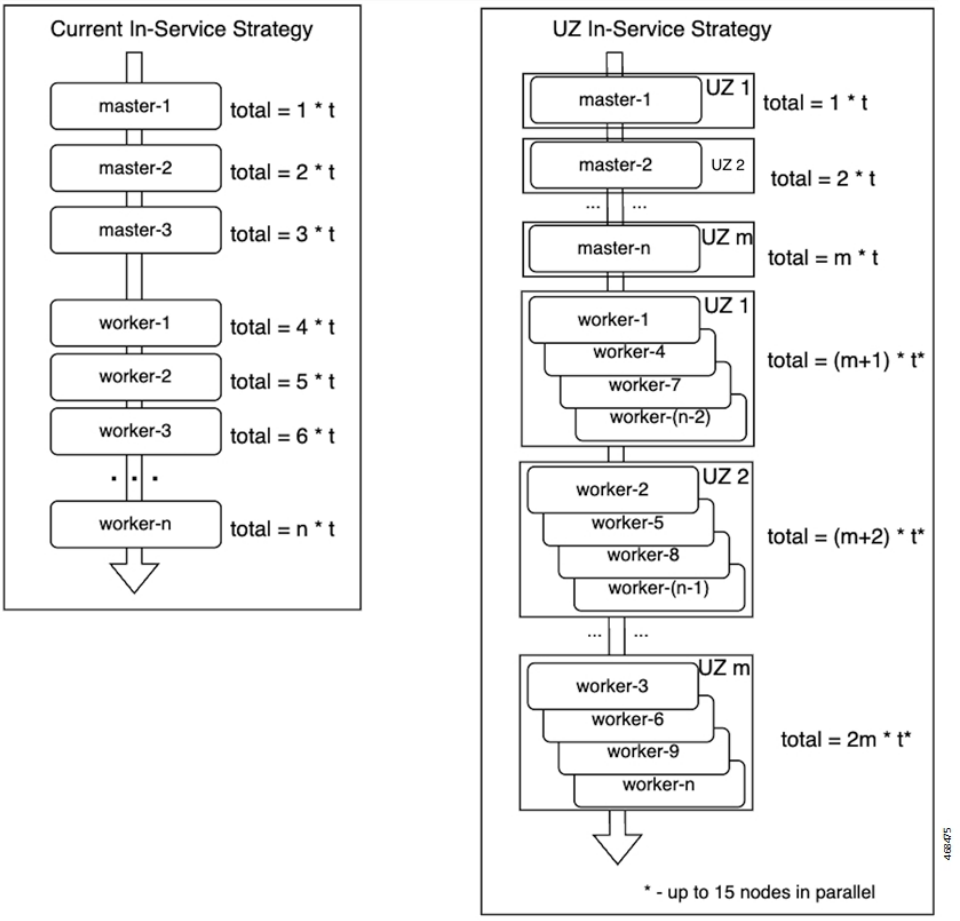
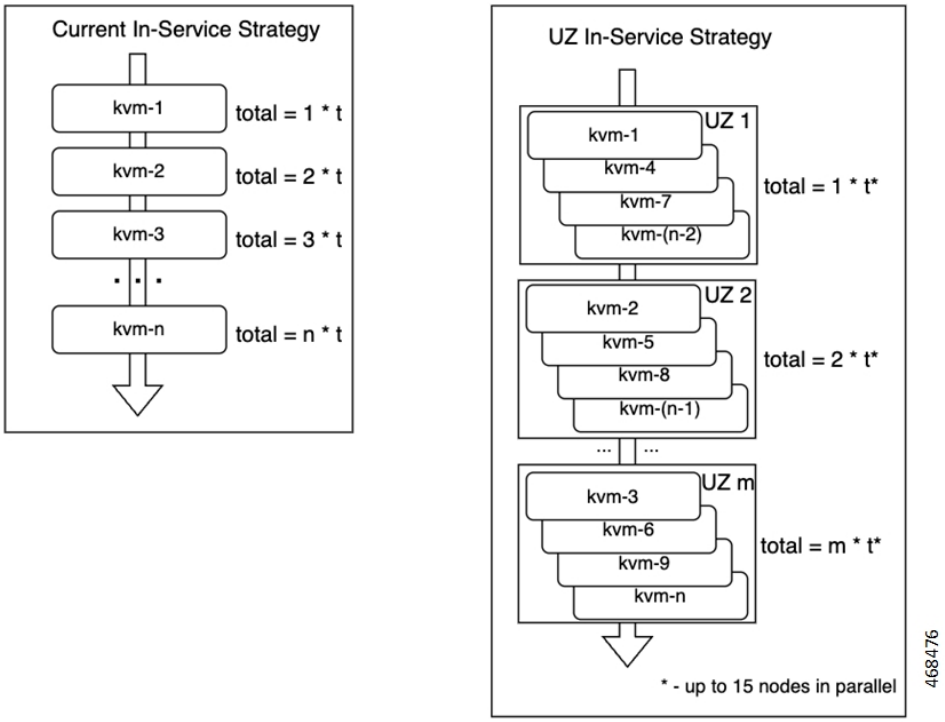


Figure 2: Upgrade Flow for KVM Clusters



How it Works

This section describes how the feature works.

This feature enables users to group servers into upgrade groups, which are similar to availability zones. The nodes in each upgrade group are upgraded in parallel (the maximum number of parallel nodes supported is 15).

The upgrade groups are upgraded in a sequential manner. For example, the control plane groups are not upgraded concurrently with the worker groups, but one at a time.

Requirements and Limitations

Some requirements and limitations associated with this feature are as follows.

- If the feature is disabled, the SMI reverts to the previous method of performing consecutive upgrade for the nodes.
- If the feature is enabled, the upgrade group configuration becomes mandatory for all nodes.

- There must be a majority of control plane or etcd nodes running at all time (for example, two out of three control planes should be always running).
- All the worker or KVM nodes should be distributed among the upgrade zone in a manner that ensures the majority of nodes never gets upgraded at the same time.
- The upgrade groups feature applies to the control plane, worker, KVM, and etcd node types,but doesn't apply to the CM-HA nodes.
- For the K8s clusters, the nodes include the upgrade group name as a new label. This label enables the NFs to use the affinity and anti-affinity rules to achieve proper HA and replication. The NFs can use the upgrade-zone provisioned node label or use custom defined ones to enable the application to align with the affinity rule.

Configuring the Deployment Zone Strategy

This section describes how to configure the upgrade groups for different nodes.

Use the following command to configure the upgrade groups for different nodes.

```
configuration enable-upgrade-zones true
   upgrade-zones zone_name
   exit
   nodes node_name
   upgrade-zone zone_name
   exit
```

Note In this release, the zone upgrade strategy is applicable for only the **auto** option for cluster **upgrade-strategy**. See the following example configuration:

```
clusters foo actions sync run upgrade-strategy
Possible completions:
auto concurrent rolling
```

When **upgrade-strategy** is set to **auto** and calculated as **rolling**, Cluster Manager evaluates the upgrade zone configuration and performs a zone-based upgrade. If the **upgrade-strategy** is **auto** and calculated as **concurrent**, then it performs a concurrent upgrade regardless of the initial configuration.

Configuration Example:

```
clusters ott-bm1-c1

configuration enable-upgrade-zones true

upgrade-zones zone1
exit
upgrade-zones zone2
exit
upgrade-zones zone3
exit

nodes mm1-controlplane1
upgrade-zone zone1
exit
nodes mm1-controlplane2
upgrade-zone zone2
```

```
exit
nodes mm1-controlplane3
upgrade-zone zone3
exit
nodes mm1-etcd1
upgrade-zone zone1
exit
nodes mm1-etcd2
upgrade-zone zone2
exit
nodes mm1-etcd3
upgrade-zone zone3
exit
nodes mm1-worker1
upgrade-zone zone1
exit
nodes mm1-worker2
upgrade-zone zone2
exit
nodes mm1-worker3
upgrade-zone zone3
exit
nodes mm1-worker4
upgrade-zone zone1
exit
nodes mm1-worker5
upgrade-zone zone2
exit
nodes mm1-worker6
upgrade-zone zone3
exit
nodes mm1-worker7
upgrade-zone zone1
exit
commit
end
```

Path Based Routing for Inception Server

Feature Summary and Revision History

Summary Data

Applicable Product (s) or Functional Area	KVM-based application deployment support K8s-based application deployment support
Applicable Platforms	Bare Metal, OpenStack, VMware
Feature Default Setting	Disabled – Configuration Required
Related Changes in this Release	Not Applicable
Related Documentation	*UCC SMI Operations Guide*

Revision History

Revision Details	Release
First introduced.	2022.02.1

Feature Description

This feature enables the SMI to support path based URL routing for its nginx routing (external traffic) in Inception VM from the traditional host based approach for the following ingress.

- cli.smi-deployer.deployer.example.com
- restconf.smi-deployer.deployer.example.com

Configuring the Path Based Routing for Inception Server

This section describes how to enable the path based routing for Inception server.

Use the following argument in the deploy script to enable path based ingress for RESTCONF and CLI:

```
-i or --path-based-ingress
```

Configuration Example:

```
./deploy -p 209.165.200.224 -f Passwd@123 -i
```

After you enable the path based ingress, SSH and RESTCONF of inception server are accessible using the following URLs:

```
API: https://209.165.200.224/smi-deployer/restconf
```

If you provide the hostname in "--external-zone-name" along with the path based ingress argument, then the entire hostname is replaced with the provided host name.

Configuration Example:

```
./deploy -p 209.165.200.224 -f Csco@123 -i --external-zone-name abc.com
```

After you enable the path based ingress, SSH and RESTCONF of inception server are accessible using the following URLs:

```
SSH (cli): ssh admin@127.0.0.1 -p 2022

API: https://abc.com/smi-deployer/restconf
```

CA Signed Certificate for Path-based Ingress

Feature Summary and Revision History

Summary Data

Applicable Product (s) or Functional Area	KVM-based application deployment support K8s-based application deployment support
Applicable Platforms	Bare Metal, OpenStack, VMware
Feature Default Setting	Disabled – Configuration Required
Related Changes in this Release	Not Applicable
Related Documentation	*UCC SMI Operations Guide*

Revision History

Revision Details	Release
Added support for provisioning CA certificates.	2023.01.0
First introduced.	2022.02.1

Feature Description

This feature enables you to configure certificates signed by your own CA or external CA for path-based ingress URLs.

You can provision the certificates used for both REST APIs and K8s APIs through cluster manager and Ops-center. The recommended method to configure a certificate and its corresponding private key is to provision the certificate as a TLS secret using the existing yang container.

Certificate Expiry Check

The provisioned certificates must be monitored for expiry. The kube-certificate-expiring alert is automatically raised in advance to renew and update the certificate and key.

The alerts have the following severity levels:

- 30 days before expiry—Raise alert with Info severity
- 20 days before expiry—Raise alert with Major severity
- 15 days before expiry—Raise alert with Critical severity

Configuring Certificate for Path-based Ingress

This section describes how to configure TLS and CA certificates for path-based ingress.

Configuring TLS Certificate

Use the following procedure from cluster deployer to configure the certificates for path-based ingress.

1. Create a secret.

   Use the following sample configuration to populate a certificate and its corresponding private key. The provided certificate and private key is stored as K8s TLS secret on the cluster under the mentioned namespace.

   ```
   cluster cluster_name
      secrets tls namespace secret_name
         private-key private_key_content
         certificate certificate_content
         exit
      exit
   ```

   Example:

   ```
   clusters sample-cluster
     secrets tls cee-global sample-secret
       private-key "$8$9n3U7OLEclVQoDpp/4VqkSLkeSmFbjx/
   Mt6eEGN4EWoKPY1r9nqSWSZ40advmhDFsPFQZWfM\nhq/wpRzHXBZGp/
   dNtNO+wpaQuxsT3CmkmRKFIHviUn4bEwBKfTCCsw7a5+66q3rm5vX4/nSw\
   nNy4DrgTu4iFDzVYVKAYzoxWGzCqhKIaSqELjsW7gchEowC\n
       certificate "-----BEGIN CERTIFICATE-----\nMIID0zCCArugAw
   IBAgIUPHTzpMTVUNVDQzJ/FM9tfCsAG2AwDQYJKoZIhvcNAQEL
   \nBQAwaDELMAkGA1UEBhMCVVMxCzAJBgNVBAgMAkNBMQsw
   \n-----END CERTIFICATE-----\n"
       exit
     exit
   exit
   ```

2. Configure path-based ingress secret.

   Use the following sample configuration to add the secret name for path-based ingresses.

   ```
   clusters <cluster_name>
      ops-centers <opscenter_name> <instance_name>
         initial-boot-parameters path-based-ingress true
         initial-boot-parameters path-based-ingress-secret <secret_name>
         exit
      exit
   exit
   ```

Note You must set **path-based-ingress** to **true** for getting the option to configure **path-based-ingress-secret**.

   Example:

   ```
   clusters sample-cluster
    ops-centers cee global
     initial-boot-parameters path-based-ingress true
     initial-boot-parameters path-based-ingress-secret sample-secret
   ```

```
    exit
   exit
  exit
```

3. Run cluster sync to create and configure the secret as well as configure ingress to use the secret.

Verifying the Certificate for Path-based Ingress Configuration

This section describes how to verify the certificate for path-based ingress configuration.

Use the following CLI command to get the ingress in YAML and verify the configured secret name:

```
kubectl get ing -n <namespace> <ingress-name> -o yaml
```

Command Output Example:

```
cloud-user@sample-aio-controlplane:~$ kubectl get ing -n cee-global
cli-ingress-cee-global-ops-center -o yaml

apiVersion: networking.k8s.io/v1
kind: Ingress
...
spec:
  rules:
  - host: 10.x.x.x
    http:
      paths:
      - backend:
          service:
            name: ops-center-cee-global-ops-center
            port:
              number: 7681
        path: /cee-global/cli
        pathType: ImplementationSpecific
  tls:
  - hosts:
    - 10.x.x.x.
    secretName: sample-secret
```

Run the **curl** command to verify the section "Server certificate:" to check whether the certificate is used properly.

```
cloud-user@satya-aio-controlplane:~$ curl -k -v https://10.x.x.x.nip.io/cee-global/cli
*   Trying 10.x.x.x...
* TCP_NODELAY set
* Connected to 10.x.x.x.nip.io (10.x.x.x) port 443 (#0)
* ALPN, offering h2
* ALPN, offering http/1.1
* successfully set certificate verify locations:
*   CAfile: /etc/ssl/certs/ca-certificates.crt
  CApath: /etc/ssl/certs
* TLSv1.3 (OUT), TLS handshake, Client hello (1):
* TLSv1.3 (IN), TLS handshake, Server hello (2):
* TLSv1.3 (IN), TLS Unknown, Certificate Status (22):
* TLSv1.3 (IN), TLS handshake, Unknown (8):
* TLSv1.3 (IN), TLS Unknown, Certificate Status (22):
* TLSv1.3 (IN), TLS handshake, Certificate (11):
* TLSv1.3 (IN), TLS Unknown, Certificate Status (22):
* TLSv1.3 (IN), TLS handshake, CERT verify (15):
* TLSv1.3 (IN), TLS Unknown, Certificate Status (22):
* TLSv1.3 (IN), TLS handshake, Finished (20):
* TLSv1.3 (OUT), TLS change cipher, Client hello (1):
* TLSv1.3 (OUT), TLS Unknown, Certificate Status (22):
* TLSv1.3 (OUT), TLS handshake, Finished (20):
```

```
* SSL connection using TLSv1.3 / TLS_AES_256_GCM_SHA384
* ALPN, server accepted to use h2
* Server certificate:
*  subject: C=US; ST=CA; L=SF; O=sample-signed.cisco.com; CN=10.x.x.x
*  start date: Jul 12 04:19:56 2022 GMT
*  expire date: Jul 11 04:19:56 2024 GMT
*  issuer: C=US; ST=CA; L=SF; O=sample-signed.cisco.com; CN=10.x.x.x
*  SSL certificate verify result: self signed certificate (18), continuing anyway.
* Using HTTP2, server supports multi-use
* Connection state changed (HTTP/2 confirmed)
* Copying HTTP/2 data in stream buffer to connection buffer after upgrade: len=0
* TLSv1.3 (OUT), TLS Unknown, Unknown (23):
* TLSv1.3 (OUT), TLS Unknown, Unknown (23):
* TLSv1.3 (OUT), TLS Unknown, Unknown (23):
* Using Stream ID: 1 (easy handle 0x56498909f550)
* TLSv1.3 (OUT), TLS Unknown, Unknown (23):
> GET /cee-global/cli HTTP/2
> Host: 10.x.x.x
> User-Agent: curl/7.58.0
> Accept: */*
```

Configuring CA Certificate

To configure the CA certificate, use the following configuration in Ops-center:

```
secrets ca-cert secret_name
   certificate certificate_content
   exit
```

To configure the CA certificate, use the following configuration in cluster-manager:

```
cluster cluster_name
   secrets ca-cert namespace secret_name
      private-key private_key_content
      certificate certificate_content
      exit
   exit
```

NOTES:

- If you add invalid certificate content and expired certificate, you will be prompted to correct the configuration.
- CA certificate is stored in generic (Opaque) secret type.
- The secrets are monitored and auto-healed if the user deletes the data by mistake.

OnDemand LDAP Connectivity Check

Feature Summary and Revision History

Summary Data

Applicable Product (s) or Functional Area	KVM-based application deployment support

	K8s-based application deployment support
Applicable Platforms	Bare Metal, OpenStack, VMware
Feature Default Setting	Disabled – Configuration Required
Related Changes in this Release	Not Applicable
Related Documentation	*UCC CEE Configuration and Administration Guide* *UCC SMI Operations Guide*

Revision History

Revision Details	Release
First introduced.	2022.02.1

Feature Description

The SMI Ops Center provides an external authentication using LDAP support. The LDAP configuration can be configured in the SMI Ops Center using CLI or the RESTCONF APIs.

This feature enables you to validate a new LDAP configuration before adding it to the system or an existing LDAP configuration.

How it Works

This section describes how the feature works.

How to Validate a New Configuration

The steps to validate a new LDAP configuration are as follows.

1. Login to the SMI Ops Center.
2. Provide the LDAP new configuration inputs to validate (see the following example).

```
[pv/global] cee# smildap validate-security-config validate-new-security-config { ?
Possible completions:
  base-dn                LDAP Base DN
  bind-dn                LDAP Bind DN
  group-attr             Group attribute
  group-mapping          LDAP group to application security mapping
  ldap-filter            LDAP Filter - use %s to sub username
  ldap-server-url        LDAP Server URL (https://tools.ietf.org/html/rfc2255)
  ldap-username-domain   LDAP Username Domain
  password               Password
  username               Existing User name in LDAP server
```

3. Validate the LDAP new configuration (see the following example configuration).

```
cee(config)# smildap validate-security-config validate-new-security-config
{ base-dn dc=smi-lab,dc=com bind-dn cn=%s,ou=people,dc=smi-lab,dc=com group-attr
memberOf group-mapping { group admin ldap-group group1 } username user5 password
Passwd@123 ldap-filter cn=%s ldap-server-url ldap://209.165.200.224 }
```

```
Mon Jun  20 05:02:24.635 UTC+00:00
message accept "admin" external-user-group 1117 1117 /tmp
```

How to Validate an Existing LDAP Configuration

Use the following example configuration to validate an existing LDAP configuration.

```
cee# smildap validate-security-config validate-current-security-config

Mon Jun  20 05:07:41.765 UTC+00:00

Value for 'username' (<string>): user5

Value for 'password' (<string>): ********

message accept "admin" external-user-group 1117 1117 /tmp
```

Provisioning Local Users

A new YANG model is introduced in SMI to support user management in compliance with Cisco Secure Development Life-cycle (CSDL) requirements.

Note This new YANG model is applicable to SMI Cluster Manager and all other product Ops Centers.

User Management

This chapter describes how to create and manage local users using the Ops Center CLI (for both the products and SMI Cluster Manager Ops Center).

Important Users with administrator privileges can add, modify, and delete other users and groups. All the other users only have privileges to change their own password.

Adding a User

To add a new user, use the following configurations:

```
configure
  smiuser add-user username username password password
  exit
```

Notes:

- **smiuser add-user** - Adds a new local user.
- **username** *username* - Specifies the name of the user.

  username must be alphanumeric string.
- **password** *password* - Specifies the password. The password must meet the following criteria:
  - Minimum 8 characters in length.

- Contain at least one lowercase character.
- Contain at least one uppercase character.
- Contain at least one numeric character.
- Contain at least one special character, which includes the following:
  - ['~', '@', '#', '%', '^', '&', '*', '(', ')', '_', '+', '`', '-', '=', '{', '}', '[', ']', ':', "", ';', '\', '|', '<', '>', '?', ',', '.', '/', '$']
  - Password must not start with '$'.
- Password must not be too simplistic or based on dictionary word.
- Do not re-use passwords.

  Use the following command to configure the number of passwords to keep in history:

  ```
  password requisite pam_pwhistory.so debug enforce_for_root remember=12
  ```
- Minimum number of days that are allowed between password changes is seven.

The following example adds a new user called 'user1' and assigns the password for the new user.

```
cee# configure terminal
  smiuser add-user username user1 password Cisco@123
message User added
```

The following example adds a new user called 'user2' and assigns the password for the new user.

```
cee# configure terminal
  smiuser add-user username user2 password Cisco@12345
message User added
```

In the following example, when an existing user name (user2) is added as a new user, the Ops Center displays an error message.

```
cee# configure terminal
  smiuser add-user username user2 password Cisco@12345
message User already exists
```

Creating Unprivileged Users with SSH Key

The SMI Cluster Manager allows creating unprivileged users on cluster nodes with SSH key access. These users will remain even after the SMI Cluster Manager is upgraded. Also, the SMI Cluster Manager considers the users created with the comment *smi.user* to be managed by the Cluster Manager. If an existing user, who is not an *smi.user*, is added to the configuration, the SMI Cluster Manager throws an error during cluster synchronization to prevent damaging or blocking communication to the system.

To add a SSH key and password to an user on all the nodes, use the following configuration:

```
configure
  node-defaults os users username
    password password
    authorized-keys key_name
    algorithm ssh_algorithm
    key-data  key_data
    exit
  authorized-keys key_name
```

```
   algorithm ssh_algorithm
   key-data key_data
   exit
 exit
```

To add a SSH key and password to an user on a specific node, use the following configuration:

```
configure
  node node_name os users username
    password password
    authorized-keys key_name
    algorithm ssh_algorithm
    key-data  key_data
    exit
   authorized-keys key_name
    algorithm ssh_algorithm
    key-data key_data
    exit
  exit
```

NOTES:

- **node-defaults os users** *username* - Specifies the default value applicable to all the nodes for the selected user. *username* is the name of the user to be created.
- **node** *node_name* **os users** *username* - Specifies the default value applicable to the specific node for the selected user. *node_name* is the name of the specific node. *username* is the name of the user to be created.
- **password** *password* - Specifies the password used for authentication.
- **authorized-keys** *key_name* - Specifies the name of the SSH key.
- **algorithm** *ssh_algorithm* - Specifies the SSH algorithm used for generating the SSH key. For example, SSH-RSA or SSH-Ed25519 algorithm.
- **key-data** *key_data* - Specifies the generated SSH key.

Deleting a User

To delete a user, use the following configuration:

```
configure
  smiuser delete-user username username
  exit
```

Note

- **smiuser delete-user** - Deletes a local user.
- **username** *username* - Specifies the name of the user.

  username must be alphanumeric string.

The following example deletes a user called 'user2'.

```
cee# configure terminal
  smiuser delete-user username user2
message User deleted
```

In the following example, when a non-existing user is deleted, the Ops Center displays an error message.

```
cee# configure terminal
  smiuser delete-user username user2
message User does not exist
```

Modifying the Password

To modify the password (for self), use the following configuration:

```
configure
 smiuser change-self-password current_password current password new_password
new_password
 confirm_password new_password password_expire_days number_of_days
 exit
```

Note

- **smiuser change-password** - Modifies the password for an user.
- **current_password** *current_password* - Specifies the current password for an user.
- **new_password** *new_password* - Assign a new password for the user. For information on password policy, see Adding a User section.
- **confirm_password** *new_password* - Enter the newly assigned password one more time.
- **password_expire_days** *number_of_days* - (Optional) Specifies the expiry date of the password. The default value is 180 days.

The following example updates the password for the current user.

```
cee# configure terminal
  smiuser change-self-password current_password Cisco@123 new_password Cisco@345
  confirm_password Cisco@345 password_expire_days 180
message Password updated successfully
```

The following example updates the password for the user called 'user1' without assigning the password expiry date.

```
cee# configure terminal
  smiuser change-self-password current_password Cisco@123 new_password Cisco@345
confirm_password Cisco@345
message Password updated successfully
```

Reset the Administrator Password

You can reset the administrator password if you have access to the K8s Cluster through **kubectl** command-line utility.

To reset the administrator password:

1. Enter the Ops Center Pod's EXEC mode.
2. Use the following command to reset the administrator password.

   kubectl exec -it *<pod_name>* **-n** *<pod_namespace>* **/usr/local/bin/reset-admin**

3. Enter the new password when prompted.

NOTES:

- **kubectl exec -it** - Executes a command inside a container. **-it** passes the standard input stream to the container or TTY.
- <*pod_name*> **-n** - Specifies the name of the Pod. **-n** specifies the namespace scope for this CLI request.
- <*pod_namespace*> - Specifies the namespace of the Pod.
- **/usr/local/bin/reset-admin** - Resets the administrator password.

Modifying the Password for Other Users

You can modify the password for other users using the following configuration:

```
configure
  smiuser change-password username username current_password current_password
new_password new_password
  confirm_password new_password password_expire_days number_of_days
  exit
```

Note

- **smiuser change-password** - Modifies the password for an user.
- **username** *username* - Specifies the name of the user.

  username must be alphanumeric string.
- **current_password** *current_password* - Specifies the current password for an user.
- **new_password** *new_password* - Assign a new password for the user. For information on password policy, see Adding a User section.
- **confirm_password** *new_password* - Enter the newly assigned password one more time.
- **password_expire_days** *number_of_days* - (Optional) Specifies the expiry date of the password. The default value is 180 days.

The following example updates the password for the user called 'user1'.

```
cee# configure terminal
  smiuser change-password username user1 current_password Cisco@123 new_password Cisco@345
 confirm_password Cisco@345 password_expire_days 180
message Password updated successfully
```

The following example updates the password for the user called 'user1' without assigning the optional password expiry date.

```
cee# configure terminal
  smiuser change-password username user1 current_password Cisco@123 new_password Cisco@345
 confirm_password Cisco@345
message Password updated successfully
```

The following example updates the password for the user called 'user1' without assigning the password expiry date.

```
cee# configure terminal
  smiuser change-password username user1 current_password Cisco@123 new_password Cisco@345
 confirm_password Cisco@345
message Password updated successfully
```

The following example updates the password for the user called 'user1' with an existing password.

```
cee# configure terminal
  smiuser change-password username user1 current_password Cisco@345 new_password Cisco@345
 confirm_password Cisco@345
message Password has been already used
```

The following example updates the password for the user called 'user1' with different values for new password and confirm password parameters.

```
cee# configure terminal
  smiuser change-password username user1 current_password Cisco@345 new_password Cisco@345
 confirm_password Cisco@567
message Passwords do not match
```

Updating the Password Length

To update the length of the password, use the following configuration:

```
configure
smiuser update-password-length length number_of_characters
 exit
```

Note

- **smiuser update-password-length** - Updates the length of the password.
- **length** *number_of_characters* - Specifies the length of the password. *number_of_characters* must be a numeric value.

The following example updates the minimum length of the password to 10 characters.

```
cee# configure terminal
  smiuser update-password-length length 10
message Password updated successfully
```

Group Management

This chapter describes how to create and manage user groups using the Ops Center CLI (of both the products and SMI Cluster Manager).

Adding a User Group

To add a user group, use the following configuration:

```
configure
smiuser add-group groupname group_name
 exit
```

Note

- **smiuser add-group** - Adds a new user group.
- **groupname** *group_name* - Specifies the name of the user group. *group_name* must be a alphanumeric value.

The following example adds a new user group called 'group1'.

```
cee# configure terminal
  smiuser add-group groupname group1
message Group added
```

In the following example, when a user group that already exists is added, the Ops Center displays an error message.

```
cee# configure terminal
  smiuser add-group groupname group1
message Group already exists
```

Deleting a User Group

To delete a user group, use the following configuration:

```
configure
smiuser delete-group groupname group_name
 exit
```

Note

- **smiuser delete-group** - Deletes a user group.
- **groupname** *group_name* - Specifies the name of the user group. *group_name* must be a alphanumeric value.

The following example deletes a new user group called 'group2'.

```
cee# configure terminal
  smiuser delete-group groupname group2
message Group deleted
```

In the following example, when a user group that does not exist is deleted, the Ops Center displays an error message.

```
cee# configure terminal
  smiuser delete-group groupname group2
message Group does not exist
```

Assigning an User to an User Group

To assign an user to an user group, use the following configuration:

```
configure
  smiuser assign-user-group username username group group_name
   exit
```

Note

- **smiuser assign-user-group** - Assigns an user to a user group.
- **username** *username* - Specifies the name of the user. *username* must be alphanumeric value.
- **groupname** *group_name* - Specifies the name of the user group. *group_name* must be a alphanumeric value.

The following example assigns an user called 'user1' to a group called 'group1'.

```
cee# configure terminal
  smiuser assign-user-group username user1 group group1
message User assigned to group successfully
```

The following example assigns a non-existing user to an existing group.

```
cee# configure terminal
  smiuser assign-user-group username user20 group group1
message User does not exist
```

The following example assigns a non-existing group to an existing user.

```
cee# configure terminal
  smiuser assign-user-group username user1 group group10
message Group does not exist
```

Unassigning a User from a User Group

To unassign a user from a user group, use the following configuration:

```
configure
  smiuser unassign-user-group username username group group_name
   exit
```

Note

- **smiuser unassign-user-group** - Removes an user from a user group.
- **username** *username* - Specifies the name of the user. *username* must be alphanumeric value.
- **groupname** *group_name* - Specifies the name of the user group. *group_name* must be a alphanumeric value.

The following example removes an user from a group.

```
cee# configure terminal
  smiuser unassign-user-group username user1 group group1
message User un-assigned from group successfully
```

The following example removes a non-existing user from a group.

```
cee# configure terminal
  smiuser unassign-user-group username user10 group group1
message User is not a member of this group
```

The following example removes an user from an non-existing group.

```
cee# configure terminal
  smiuser unassign-user-group username user1 group group10
message Group does not exist
```

Resiliency and Redundancy

For resiliency and redundancy, SMI utilizes Kubernetes version 1.16. For more information on the various Kubernetes components, see https://v1-16.docs.kubernetes.io/docs/concepts/overview/components/

Note SMI implements Kubernetes with a highly available three ETCD and three control plane nodes setup.

SMI User and Audit Tracking Commands

This section provides the list of user and audit tracking commands used in SMI.

Note

- All these commands are executed on the node's terminal. Users with **sudo** access can execute these commands.
- You need the **wtmp** and **btmp** files in the **/var/log** directory for the system to store log information.

User Tracking Commands

The following commands are used for user tracking in SMI.

Table 8: User Tracking Commands

Command	Description
last -F	Displays the list of users logged in the last session along with information such as date and time of last log in. **Note** • All the user information (log in and log out details) are stored in the **/var/log/wtmp** file. The command fetches all the user information stored in this file from the time the **wtmp** file was created. • The list of users in the current session are displayed as *still logged in*.
last reboot	Displays the number of times the system was rebooted.
lastb	Displays the list of bad log in attempts recorded in the **/var/log/btmp** file.

Command	Description
lastlog	Displays each user's last logged information - login name, port, and last login time - recorded in **var/log/lastlog** file.

Audit Tracking Commands

The following commands are used for audit tracking in SMI.

Table 9: Audit Tracking Commands

Command	Description
sudo aureport -au -i | more	Displays a summary of audit daemon logs report.
sudo cat /var/log/auth.log | grep "Failed password"	Displays a list of failed SSH log in attempts.
sudo journalctl _SYSTEMD_UNIT=ssh.service | egrep "Failed|Failure"	Displays a list of all the failed log in attempts.
sudo journalctl -q _TRANSPORT=audit	Displays audit logs for SSH, SFTP and SCP.

TCP and UDP Open Ports

This section lists the TCP and UDP services and the corresponding open ports in the Kubernetes cluster nodes (*control plane*, *worker* and *etcd*).

The following table lists the TCP and UDP services and the corresponding open ports for the Primary *control plane* node.

Table 10: Primary Control Plane Node - Open Ports

Pod	Description	Port
kubelet	kubelet is the lowest level component in Kubernetes. It's responsible for what's running on an individual machine. You can think of it as a process watcher like supervisord but focused on running containers. It has one job: given a set of containers to run, make sure they are all running.	10248, 10250
kube-proxy	kube-proxy is a network proxy that runs on each node in your cluster, implementing part of the Kubernetes Service. concept. kube-proxy maintains network rules on nodes. These network rules allow network communication to your Pods from network sessions inside or outside of your cluster.	10249, 443, 10256
calico-node	A node resource representing a node running Calico. When adding a host to a Calico cluster, a node resource needs to be created which contains the configuration for the calico/node instance running on the host.	9099

Pod	Description	Port
kube-controller	The Kubernetes controller manager is a daemon that embeds the core control loops shipped with Kubernetes. The controller is a control loop that watches the shared state of the cluster through the apiserver and makes changes attempting to move the current state towards the desired state.	10257
kube-scheduler	kube-scheduler is the default scheduler for Kubernetes and runs as part of the control plane. A scheduler watches for newly created Pods that have no Node assigned. For every Pod that the scheduler discovers, the scheduler becomes responsible for finding the best Node for that Pod to run on	10259
bird	BIRD is an open source BGP client that is used to exchange routing information between hosts. The routes that Felix programs into the kernel for endpoints are picked up by BIRD and distributed to BGP peers on the network, which provides inter-host routing. If configured, there will be two BIRD processes running in the calico/node container. One for IPv4 (bird) and one for IPv6 (bird6).	3179
systemd-resolv	systemd-resolved is a system service that provides network name resolution to local applications.	53
sshd	sshd (SSH Daemon) is the daemon program for ssh. Together these programs replace rlogin and rsh and provide secure encrypted communications between two untrusted hosts over an insecure network.	22
kube-apiserver	The kubelet takes a set of PodSpecs and ensures that the described containers are running and healthy. kube-apiserver - REST API that validates and configures data for API objects such as pods, services, replication controllers.	6443
node_exporter	Node Exporter is a Prometheus exporter for hardware and OS metrics with pluggable metric collectors. It allows you to measure various machine resources such as memory, disk and CPU utilization.	9100
chronyd	chronyd provides support to work out the gain or loss rate of the 'real-time clock', i.e. the clock that maintains the time when the computer is turned off. It can use this data when the system boots to set the system time from a corrected version of the real-time clock.	323, 123

The following table lists the TCP and UDP services and the corresponding open ports for Secondary *control plane* node.

Table 11: Secondary Control Plane Node - Open Ports

Pod	Description	Port
kubelet	kubelet is the lowest level component in Kubernetes. It's responsible for what's running on an individual machine. You can think of it as a process watcher like supervisord but focused on running containers. It has one job: given a set of containers to run, make sure they are all running.	10248, 36189, 10250
kube-proxy	kube-proxy is a network proxy that runs on each node in your cluster, implementing part of the Kubernetes Service. concept. kube-proxy maintains network rules on nodes. These network rules allow network communication to your Pods from network sessions inside or outside of your cluster.	10249, 10256
calico-node	A node resource representing a node running Calico. When adding a host to a Calico cluster, a node resource needs to be created which contains the configuration for the calico/node instance running on the host.	9099
kube-controller	The Kubernetes controller manager is a daemon that embeds the core control loops shipped with Kubernetes. The controller is a control loop that watches the shared state of the cluster through the apiserver and makes changes attempting to move the current state towards the desired state.	10257
kube-scheduler	kube-scheduler is the default scheduler for Kubernetes and runs as part of the control plane. A scheduler watches for newly created Pods that have no Node assigned. For every Pod that the scheduler discovers, the scheduler becomes responsible for finding the best Node for that Pod to run on	10259
bird	BIRD is an open source BGP client that is used to exchange routing information between hosts. The routes that Felix programs into the kernel for endpoints are picked up by BIRD and distributed to BGP peers on the network, which provides inter-host routing. If configured, there will be two BIRD processes running in the calico/node container. One for IPv4 (bird) and one for IPv6 (bird6).	3179
systemd-resolv	systemd-resolved is a system service that provides network name resolution to local applications.	53

Pod	Description	Port
sshd	sshd (SSH Daemon) is the daemon program for ssh. Together these programs replace rlogin and rsh, and provide secure encrypted communications between two untrusted hosts over an insecure network.	22
kube-apiserver	The kubelet takes a set of PodSpecs and ensures that the described containers are running and healthy. kube-apiserver - REST API that validates and configures data for API objects such as pods, services, replication controllers.	6443
node_exporter	Node Exporter is a Prometheus exporter for hardware and OS metrics with pluggable metric collectors. It allows you to measure various machine resources such as memory, disk and CPU utilization.	9100
chronyd	chronyd provides support to work out the gain or loss rate of the 'real-time clock', i.e. the clock that maintains the time when the computer is turned off. It can use this data when the system boots to set the system time from a corrected version of the real-time clock.	323, 123

The following table lists the TCP and UDP services and the corresponding open ports for *etcd* node.

Table 12: ETCD Node - Open Ports

Pod	Description	Port
kubelet	kubelet is the lowest level component in Kubernetes. It's responsible for what's running on an individual machine. You can think of it as a process watcher like supervisord but focused on running containers. It has one job: given a set of containers to run, make sure they are all running.	10248, 10250, 10255
etcd	etcd is a distributed key-value store, which accepts TLS traffic, non-TLS traffic or both TLS and non-TLS traffic.	2379, 2380, 2381
systemd-resolv	systemd-resolved is a system service that provides network name resolution to local applications.	53
sshd	sshd (SSH Daemon) is the daemon program for ssh. Together these programs replace rlogin and rsh, and provide secure encrypted communications between two untrusted hosts over an insecure network.	22

Pod	Description	Port
chronyd	chronyd provides support to work out the gain or loss rate of the 'real-time clock', i.e. the clock that maintains the time when the computer is turned off. It can use this data when the system boots to set the system time from a corrected version of the real-time clock.	323, 123
node-exporter	Exports the node metrics to Prometheus and to be viewable on the Grafana dashboard in Host details and summary dashboards.	9100

The following table lists the TCP and UDP services and the corresponding open ports for *worker* node.

Table 13: Worker Node - Open Ports

Pod	Description	Port
kubelet	kubelet is the lowest level component in Kubernetes. It's responsible for what's running on an individual machine. You can think of it as a process watcher like supervisord but focused on running containers. It has one job: given a set of containers to run, make sure they are all running.	10248, 10250
kube-proxy	kube-proxy is a network proxy that runs on each node in your cluster, implementing part of the Kubernetes Service. concept. kube-proxy maintains network rules on nodes. These network rules allow network communication to your Pods from network sessions inside or outside of your cluster.	10249, 10256
calico-node	A node resource representing a node running Calico. When adding a host to a Calico cluster, a node resource needs to be created which contains the configuration for the calico/node instance running on the host.	9099
bird	BIRD is an open source BGP client that is used to exchange routing information between hosts. The routes that Felix programs into the kernel for endpoints are picked up by BIRD and distributed to BGP peers on the network, which provides inter-host routing. If configured, there will be two BIRD processes running in the calico/node container. One for IPv4 (bird) and one for IPv6 (bird6).	3179
systemd-resolv	systemd-resolved is a system service that provides network name resolution to local applications.	53

Pod	Description	Port
sshd	sshd (SSH Daemon) is the daemon program for ssh. Together these programs replace rlogin and rsh, and provide secure encrypted communications between two untrusted hosts over an insecure network.	22
node_exporter	Node Exporter is a Prometheus exporter for hardware and OS metrics with pluggable metric collectors. It allows you to measure various machine resources such as memory, disk and CPU utilization.	9100
chronyd	chronyd provides support to work out the gain or loss rate of the 'real-time clock', i.e. the clock that maintains the time when the computer is turned off. It can use this data when the system boots to set the system time from a corrected version of the real-time clock.	323, 123

Configurable Option to Control Ping Properties

SMI monitors applications such as the UPF that are deployed in KVM-based clusters. Monitoring, in part, is performed through pings sent at regular intervals to verify that the applications are alive. If an application is unresponsive for a certain number of times, then the monitoring service attempts to restart it.

The ping interval and the number of failure occurrences are now configurable using the following commands:

node-defaults kvm monitoring ping-interval *<#_seconds>*

node-defaults kvm monitoring failure-occurrence *<#_instances>*

<#_seconds> is any integer value. The minimum value is 3 and the default value is 10 seconds.

<#_instances> is any integer value. The minimum value is 3 and the default value is 10 times.

Upon changing these values and running a sync, the monitoring service is restarted and the new configuration is applied.

IFTASK Forwarder type

In past releases, SMI provided support for the Vector Packet Processing (VPP) data plane development kit (DPDK) forwarder for use with StarOS-based applications such as the 5G User Plane Function.

SMI now supports the use of the DPDK Internal Forwarder (IFTASK) for use with StarOS-based applications.

IFTASK support is enabled by the forwarder-type parameter as part of the day 0 configuration:

nodes control-plane

[no] vm-defaults upf day0 forwarder-type { IFTASK | VPP }

If the forwarder type command is not issued, then VPP will be used as the forwarder type by default.

Once IFTASK has been set, the no variant of this command can be used to re-enable the VPP forwarder type.

When the forwarder type is set to IFTASK, the following additional parameters are used as part of the UPF day 0 configuration:

- IFTASK_SERVICE_TYPE=0
- IFTASK_CORES=44
- IFTASK_MCDMA_CORES=50

These parameters are hard-coded and set automatically by SMI during the deployment process.

Customer Data Recovery and Backup

Feature Description

Since the server-less Amazon Aurora database (DB) service is not available across regions, it's necessary to have a disaster recovery process for customer data when the DB in an active region fails. The disaster recovery procedure mainly involves generating the same DB snapshot in a different backup region and making all the other clusters connect to the restored DB.

Data Recovery and Backup Procedure

This section describes the customer data recovery and backup procedure.

Prerequisite

You must enable continuous automatic backup of ConfigDB Aurora DB on an active cluster.

AWS Backup Overview

Amazon Web Services (AWS) Backup is a fully-managed service that makes it easy to centralize and automate data protection across AWS services, in the cloud, and on premises. Using this service, one can configure backup policies and monitor activity for your AWS resources in one place. It automates and consolidate backup tasks that were previously performed service-by-service and removes the need to create custom scripts and manual processes.

Note To use the AWS Backup service, you must opt in to have the AWS Backup service to back up the assigned resources.

Supported Resources

The supported resources include Amazon Elastic Compute Cloud (Amazon EC2) instances, Amazon Elastic Block Store (Amazon EBS) volumes, Amazon Relational Database Service (Amazon RDS) databases (including Amazon Aurora clusters), Amazon DynamoDB tables, Amazon Neptune databases, Amazon DocumentDB (with MongoDB compatibility) databases, Amazon Elastic File System (Amazon EFS) file systems, Amazon FSx for Lustre file systems, Amazon FSx for Windows File Server file systems, and AWS Storage Gateway volumes.

Backup Rules

Define a backup rule to specify the backup schedule, backup window, destination regions, and lifecycle rules.

- **Frequency** - **hourly**, daily, weekly, monthly
- **Backup window** - **default** or custom
- **Retention period** - **days**, weeks, months, years
- **Destination regions** - List of available regions

Note The recommended values are for every 1 hour with a retention period of 7 days with at least 3 destination regions.

Resource Assignment

After you create the rule, assign the desired services to perform the backup. In this case, the Aurora DB is assigned to the rule.

DB Recovery Procedure

This section describes the steps required to recover data.

1. Create an AWS SMI substrate on the specified region provided to you with the AWS account , for example, us-west-2, so you can deploy the SMI cluster.
2. Deploy the SMI cluster and initiate all the required SMI cluster components except the ConfigDB database, these are ConfigFE, ConfigBE, and the monitoring clusters on the specified region provided with your AWS account.
3. After a resource (ConfigDB) is backed up at least once in the active region, for example, us-east-2, it is considered protected and is available to be restored using the AWS Backup dashboard under **protected resources** in each region. Each resource has specific steps to restore from the backup. Initiate the ConfigDB in the backup region, for example, us-west-2.
4. Configure and initiate the backup cluster active state. After the Aurora DB is restored, configure the ConfigDB to point to the newly restored resource ARN and secret ARN to enable the ConfigDB to access the DB.

   After the DB is connected, ensure that all the other clusters connect to ConfigDB and are fully active.
5. Configure the vault KMS and storage using information about the newly restored DB.
6. Verify the vault encryption and decryption. For example, the encrypted fields on the us-east-2 are decrypted on the us-west-2 region.

CIMC Certificate Renewal

The Cisco® Integrated Management Controller (IMC) is a baseboard management controller that provides embedded server management for Cisco UCS® C-Series Rack Servers and Cisco UCS S-Series Storage Servers. The Cisco IMC enables system management in the data center and across distributed locations.

The CIMC certificates are valid only for 3 years. If the certificate expires in less than 90 days, it must be renewed.

To renew the CIMC certificate, use the following configuration:

```
config
  clusters cluster_name
    node-defaults ucs-server cimc certificate rehydrate { true | false}
    exit
```

NOTES:

- When the certificate is renewed, the CIMC drops connections for 15 to 60 seconds while the host key is updated.
- The default setting is **false**. When set to **true**, it renews the certificate that expires in less than 90 days.
- Every cluster synchronization log displays the expiry date of the certificate.

XFS File System

Table 14: Feature History

Feature Name	Release Information	Feature Description
Configurable XFS File System	2023.04	SMI supports a new XFS filesystem to install the */data* partition using Mongo DB for new deployments.

SMI utilizes the XFS filesystem to install */data* partition using Mongo DB.

By default, all partitions are formatted using ext4.

Note XFS works only with new deployments.

How it Works

The following steps describe how to enable data partition using XFS:

1. When **partition** is defined under **os**, run cluster-sync to create a file named **smi-fs** that contains the customized file system type information.

   smi-fs is a valid shell file containing variable assignments (all instances of **-** are converted to **_**).

   The file is created under:

   - */scripts* in cloud-init iso file for VMware
   - */smi/cloud-init/* in smi-install-iso file for bare metal

   For example:

   ```
   # cat /smi/cloud-init/smi-fs
   smi_fs_smi_state=FS_TYPE=xfs
   ```

The above variable assignment indicates that smi-state partition must be formatted using XFS as the file system type.

smi_fs_ is added as prefix to the variable name to avoid any name conflict.

2. During a fresh installation, the **init**ialize script (shared-iso-files/templates/**init**ialize) in the bootable ISO creates the file system using the information from **smi-fs**.
3. After reboot, another **init**ialize script (shared-iso-files/templates/**base**-image-initialize) which is installed under */etc/initramfs-tools/smi-init/initialize* updates */etc/fstab* with the correct entries.

Note If a separate disk is used for */data*, the initialization of data partition is done in **base**-image-initialize.

Configuring XFS

To format OS partition with the XFS option, use the following sample configuration:

Cluster Level:

```
config
  clusters cluster_name
    node-defaults os partition smi-data
      fs-type { ext4 | xfs }
      exit
```

Node Level:

```
config
  clusters cluster_name
    nodes node_name
      os partition smi-data
      fs-type { ext4 | xfs }
      exit
```

NOTES:

- **smi-data**—Use when there is a separate disk for the */data* directory (VMware, OpenStack, and so on).
- **fs-type { ext4 | xfs }**—Specify the ext4 or XFS file system.

Verifying the Configuration

To verify the configuration, use the following commands:

- **smi-state** is formatted with **xfs** using the following configuration:

```
os partition smi-state
   fs-type xfs
   exit
```

- **fstab**:

```
$ cat /etc/fstab | grep xfs
LABEL=smi-state /mnt/stateful_partition xfs defaults 0 0
```

- **blkid**:

```
$ blkid | grep xfs
/dev/sda5: LABEL="smi-state" UUID="9e717cb9-c1fb-4190-9283-aa20afe24d3a" TYPE="xfs"
PARTLABEL="smi-state" PARTUUID="7251ecd2-9623-4a37-88f8-b969b836634d"
```

Cluster Access for OS Users

SMI supports the access of OS users to SMI cluster upon login using the configurable **addons secure-access { enabled | disabled }** CLI command in the Cluster Configuration mode. By default, this command is disabled to reduce the resource usage in the K8s cluster.

The helm chart is created to deploy Daemonsets onto master nodes. Only the access controller pod on the active master will run and manage user access.

To enable or disable the access of OS users to the SMI cluster, use the following configuration:

```
config
  clusters cluster_name
    addons secure-access { enabled | disabled }
    end
```