Cisco Virtual Managed Services (VMS) 3.1.1 Release Notes

Introduction

Cisco Virtual Managed Services (VMS) is a software solution platform that enables a set of secure end-to-end cloud services overlay solution to deliver virtual services seamlessly, cost-effectively and on-demand to the remote sites, users and businesses.

Cisco VMS provides a complete self-service user experience that allows your users to select, create, customize and activate services on-demand in minutes with the click of a mouse from a simple self-service portal. These services are instantiated through VMS Network Services Orchestrator (NSO).

Cisco VMS is available with multiple infrastructure solutions such as Cisco NFVI or Cisco MetaCloud – a Cisco managed Openstack infrastructure solution.

The Cisco VMS solution is a combination of platform and service packs. The following service packs are available with this release:

- Software-Defined WAN (SDWAN)
- Cloud VPN/VCE
- vBranch
- IWAN

For a list of new features in 3.1.1, see **Cisco VMS 3.1.1 Features and Enhancements** section.

Cisco Virtual Managed Services 3.1.1 Features and Enhancements

Cisco Virtual Managed Services (VMS) 3.1.1 includes a new SDWAN service pack. For more information about the VMS SDWAN service pack, see *Cisco Virtual Managed Services (VMS) 3.1.1 SDWAN Service Pack Guide*.

Installation Notes

A complete end-to-end VMS installation includes:

- **Installation of the VMS Platform**: This includes installation of the VMS Base infrastructure services, microservices, the service interface, and various applications and services needed by VMS.
- **Deployment of the Service Packs**: Use the automated service pack deployer on the installed VMS platform to deploy Function Packs, Service Integration Framework (SIF), and a Data platform. The NSO

instances are also installed with each Service Pack. You can install CVPN, vBranch, and IWAN service packs in any order. However, vBranch and SDWAN service packs must be deployed together.

For more information on the prerequisites and the installation process of the platform and service packs, refer to the Cisco Virtual Managed Services (VMS) 3.1 Installation Guide. You will need Cisco Customer or Cisco Employee privileges to access the 3.1 documentation.

Upgrading VMS 3.1.0 to 3.1.1

If you are on 3.0.x releases and want to upgrade to 3.1.0, please contact Cisco Technical Assistance Center (TAC) to guide you through the process. However, to upgrade from 3.1.0 to 3.1.1, use the procedure detailed in this section. Upgrading to 3.1.1 includes the following steps:

1. Validating the VMS 3.1.0 Platform Status
2. Backing up the 3.1.0 VMS data.
3. Preparing a new container for 3.1.1.
4. Upgrading steps from 3.1.0 to 3.1.1.

Validating the VMS 3.1.0 Platform Status

DETAILED STEPS

	Command or Action	Purpose
Step 1	Verify the Kubernetes Pod status.	`ansible kube-master -m command -a "kubectl get pod -n vms -o wide"`
Step 2	Verify the health status of the VMS 3.1.0 Platform.	`ansible-playbook --vault-password-file vault checks/platform-health.yml`
Step 3	Verify the overall health of VMS 3.1.0.	`ansible-playbook --vault-password-file vault checks/check-vms.yml`

Backing Up the VMS Data

Step 1 Back up Cassandra. To back up, do the following:

a) Retrieve `cassandra_pass` from 3.1.0 container.

```
ansible-vault --vault-password-file vault view group_vars/all/passwords.yml | grep cassandra_pass
```

b) Login to a Kubernetes Master Node and determine the IP address of a kubernetes-master using the following command:

```
nova list | grep kubernetes-master
```

c) ssh to the kubernetese-master from the VMS 3.1.0 container with this command:

```
ssh -F ssh.cfg <kubernetes-master_IP>
```

d) From the kubernetes-master node, connect to the cassandra-0 with this command:

```
kubectl -n vms exec -it cassandra-0 bash
```

e) Execute the following commands inside the cassandra-0 container. Replace the <cassandra_pass> in the following command with the one that was retrieved in the step **a** above.

```
for i in $(ls /var/lib/cassandra/data |grep skyfall); do echo -e "describe $i;\r\n" |cqlsh -u
cassandra -p <cassandra_pass> > $i.sql;
done;

for i in $(ls /var/lib/cassandra/data |grep skyfall); do nodetool -u cassandra -pw <cassandra_pass>
 snapshot $i;
done;

mkdir cassandra_backup

mv *.sql cassandra_backup/
```

f) Exit from the cassandra-0 container and execute the following commands one by one from the master. Run the following commands as a root user:

```
cd /data/vms

mkdir cassandra_backup

kubectl -n vms cp cassandra-0:/cassandra_backup/ ./cassandra_backup/
```

Note This backup will be available on all the masters at `/data/vms/cassandra_backup`.

Step 2 Back up NSO by running the following playbook from the VMS 3.1.0 container:

```
ansible-playbook nso-backup.yml --extra-vars "servicepack_name=cloudvpn"
```

This playbook creates a folder called 'ncs-data-vol.backup' with two tar files under the '/vms-3.1.0/ansible' folder in the container. You can save these files to be used while restoring the VMS data files.

Step 3 Back up Service Metrics Engine (SME) using the following playbook:

```
ansible-playbook sme-backup.yml
```

This playbook creates two different tar files in the '/vms-3.1.0/ansible/backup' folder. You can save these files to be used while restoring the VMS data files.

Step 4 Back up Elastic Search using the following playbook:

```
ansible-playbook elasticsearch-backup.yml
```

This playbook creates a tar file and six different json files on the kubernetes-master nodes under '/data/vms/elasticserach_backup' folder. You can save these files to be used while restoring the VMS data files.

Step 5 Back up the main.yml file from the VMS 3.1.0 container by copying it to the directory that is shared with the host the container is running on.

```
cp /vms-3.1.0/ansible/group_vars/all/main.yml /VMS3config/
```

Step 6 (Only for CVPN) Back up the cloudvpn_variables file from the VMS 3.1.0 container by copying it to the directory that is shared with the host the container is running on.

```
cp /vms-3.1.0/ansible/group_vars/all/cloudvpn_variables.yml /VMS3config/
```

Step 7 Copy the following files from 3.1.0 container to the shared directory (VMS3config).

- Keys folder
- Inventory file
- ssh.cfg file
- passwords.yml file

```
cp /vms-3.1.0/ansible/keys/id_rsa /VMS3config/
cp /vms-3.1.0/ansible/keys/id_rsa.pub /VMS3config/
cp /vms-3.1.0/ansible/inventory/inventory /VMS3config/
cp /vms-3.1.0/ansible/ssh.cfg /VMS3config/
cp /vms-3.1.0/ansible/group_vars/all/passwords.yml /VMS3config/
```

Step 8 Copy the OpenStackRC file to the shared directory (for example: VMS3config).

```
cp /root/OpenStackRC VMS3config/
```

Preparing a New Container for 3.1.1

To prepare for a new 3.1.1 container, do the following:

Before You Begin

Download the VMS 3.1.1 image file that contains all VMS binaries from the Download Software page on www.cisco.com. Copy the VMS 3.1.1 image, vms-installer-3.1.1-Bundle.tar to the server from where the VMS 3.1.1 Installation Container will be run.

Step 1 Set up the Docker and load the 3.1.1 container image into the docker. To do so, perform the following steps:

a) Install Docker and start the docker after installation.

```
yum install docker -y
service docker start
```

b) Load the container image into docker.

```
docker load < vms-installer-3.1.1-10002-Bundle.tar
docker images
```

Docker images command will list the <docker_image_ID>.

c) Launch the docker image just loaded. Also, name the container for future use. If you do not provide a name, Docker will generate a random one.

```
docker run -v /FullPathOnLocalMachine:/PathOnContainer:z --name <Name_of_Container> -it
<docker_image_ID> /bin/bash
```

Step 2 Copy the files we backed up to the shared directory (VMS3config) into the proper locations in the new container.

Step 3 Source the OpenStackRC file from the shared directory (VMS3config). Execute this command from 3.1.1 container.

```
source /VMS3config/ OpenStackRC
```

Step 4 Create a vault password file and export the path of the file to `ANSIBLE_VAULT_PASSWORD_FILE` variable.

```
export ANSIBLE_VAULT_PASSWORD_FILE=/vms-3.1.1/ansible/vault
```

If the vault file was created in 3.1.0, export the vault file from /vms-3.1.0/ansible/vault.

Step 5 To reuse the same floating IPs, do the following:

a) Edit the `/vms-3.1.1/ansible/group_vars/all/external_addresses.yml` file and do the following:

- Add the values for the inception, two edge nodes, and csrhub (if used) floating IPs that were used in VMS 3.1.0.
- Remove the # from the following entries: `inception_ext_ip`, `edge_ext_ips`, and `csrhub_ext_ip`

b)

c) Edit the main.yml file to change the use_existing_ips flag.

```
sed -ri 's/use_existing_ips: no/use_existing_ips: yes/' /vms-3.1.1/ansible/group_vars/all/main.yml
```

Step 6 Configure the `main.yml` file. Modify the parameters in the main.yml file, such as the proxy, external network, DNS and NTP settings, VMS domain, and subdomain details. You must change the default setting to settings that match your environment.

Note
- In 3.1.1, the value of `use_route53` in `main.yml` file is by default set to 'yes.' Change this to 'no,' if you do not wish to use route53 for your domain registration. Also, ensure to change the DNS and NTP default setting to the settings that match your environment.
- Set the `auto_generate_password` as 'false' if you wish to use the 3.1.0 passwords.

To configure `main.yml` file, do the following:

1. Access the `main.yml` file and update the main.yml file based on your 3.1.0 main.yml file.

   ```
   /vms-3.1.1/ansible/group_vars/all/main.yml
   ```

2. Change the status of all the micro services schema_mode to 'Upgrade'. Use the following command to change the status of all the micro services in main.yml:

   ```
   sed -ri "/(schema_mode:)/s/\s+.*/ Upgrade/" ./group_vars/all/main.yml
   ```

3. (Only for AWS) To change the variables in `main.yml`, use the following commands:

   ```
   sed -i -e 's/cloud: openstack/#cloud: openstack/' ./group_vars/all/main.yml
   sed -i -e 's/#cloud: aws/cloud: aws/' ./group_vars/all/main.yml
   ```

Step 7 Configure the `servicepack_vars_<servicepack_name>.yml` file on the VMS 3.1.1 container. In this variable file, change the microservice mode to "Upgrade" for the service packs that were installed in 3.1.0. To change the `servicepack_vars_.yml` files in the `/vms-3.1.1/ansible/` directory use these commands:

```
sed -ri 's/schema_mode: "NewInstall"/schema_mode: "Upgrade"/'
/vms-3.1.1/ansible/servicepack_vars_cloudvpn.yml
sed -ri 's/schema_mode: "NewInstall"/schema_mode: "Upgrade"/'
/vms-3.1.1/ansible/servicepack_vars_iwan.yml
sed -ri 's/schema_mode: "NewInstall"/schema_mode: "Upgrade"/'
/vms-3.1.1/ansible/servicepack_vars_vbranch.yml
```

Upgrade Steps

To upgrade VMS 3.1.0 to 3.1.1, do the following:

Step 1 Invoke the following playbooks from the 3.1.1 container:

```
ansible-playbook upload-isolated-binaries.yml
ansible-playbook upgrade-cassandra.yml
ansible-playbook check_schema_mode.yml
ansible-playbook -i inventory/inventory upgrade-vms-microservices.yml
```

Note
- All the infra-services and micro services should be in the running state on one of the masters.
- Change schema mode to "Upgrade" in all the 'microservices' pod.yml' files. These files are available at `/vms-3.1.1/ansible/roles/vms-ms-pod/templates`.

Step 2 Upgrade CVPN. To upgrade, invoke the following playbook:

```
ansible-playbook -i inventory/inventory deploy-servicepack.yml --extra-vars
'{"microservices_list":[cloudvpn,firewall,dhcp,vce],"fp_list":[cloudvpn],"servicepack_name":cloudvpn,
"schema_mode":Upgrade}'
```

Step 3 Ugrade vBranch. To upgrade, invoke the following playbook:

```
ansible-playbook -i inventory/inventory deploy-servicepack.yml --extra-vars '{"microservices_list":
[vbranch,statemachine],"fp_list":[vbranch], "vnfimages_list":[],
"servicepack_name":vbranch,"schema_mode":Upgrade}'
```

Installation of Software-Defined WAN (SDWAN)

SDWAN service pack must be installed along with the vBranch service pack. To install vBranch, see *Cisco Virtual Managed Services 3.1 Install and Upgrade Guide*.

To deploy SDWAN service pack, you need to invoke the following playbook from the installation container after deploying vBranch service pack. For more information on installing the vBranch service pack, see *Cisco Virtual Managed Services 3.1 Installation Guide*.

```
ansible-playbook --vault-password-file vault deploy-servicepack.yml --extra-vars
'{"microservices_list":
[sdwan],"fp_list":[sdwan],"servicepack_name":sdwan,"schema_mode":NewInstall,"populate":
database}' --skip-tags no-nso
```

VMS High Availability

In VMS 3.1 release, almost all components are deployed in High Availability (HA) Mode. The following components are deployed in the HA mode in this release:

- Microservices (with the exception of statemachine ms for vBranch microservice) and UI

- Platform Infra services (Cassandra, Kafka, Kong, Redis etc.)
- NSO
- HAproxy
- iPNP service

Note The only component that is not deployed in HA mode is SME.

Expected Behavior:

- HA is not supported for vBranch service pack. The statemachine microservice runs only one replica (or) instance of the service. When the edge node that runs the iPNP and haproxy service goes down, the ENCS devices pointing to the proxy using the floating IP address of the edge node would needs to be manually updated to point to the other IP address of the edge node.
- In a scenario where IWAN microservice restarts or container crashes and an Operator is trying to view the portal at around the same time, there may be some inconsistency in the data displayed for a provisioned service.
- During a NSO failover, the switch over to the next node can take up to 16 to 20 seconds. In case of a reboot, both nodes will take master role briefly and it takes about 30 to 45 seconds for the final master election to happen. Commits will go to the node that was first elected the master.
- In case of a Kong service or a container crash, the switch over will take 3 to 4 seconds without any interruption to the services.
- Platform Infra services such as, Elastic Search, Cassandra, Kafka, Zookeeper, and Redis, failover takes a few seconds without any disruption to the functionality.
- Microservices failover takes up to 15 to 20 seconds. The UI displays an error message until the failover is complete. The Operator has to refresh the UI multiple times and redo the operation that was in progress when the failover happens.

Graceful Shutdown and Restart

During scheduled network outages such as system upgrades or hardware maintenance, you may have to close or disable some services or instances. A power outage or randomly shutting down an instance may result in the data loss. In such cases, you can use the procedure detailed in this section to gracefully shut down and restart the instances after a power outage without any data corruption.

Step 1 Shut down the instances. To shut down these instances, access these instances from the the installer container and shut down these instances one by one in the following order:

a) All edge-instances

```
nova stop $(nova list 2>&1 | grep edge | awk '{print $4}')
nova list 2>&1 | grep edge | awk '{print $6}'
```

Note Run the list instances command until 'Shutdown' is shown on the screen.

b) SME-instance

```
nova stop $(nova list 2>&1 | grep sme | awk '{print $4}')
nova list 2>&1 | grep  sme | awk '{print $6}'
```

Note Run the list instances command until 'Shutdown' is shown on the screen.

c) All esc instances

```
nova stop $(nova list 2>&1 | grep esc | awk '{print $4}')
nova list 2>&1 | grep  esc | awk '{print $6}'
```

Note Run the list instances command until 'Shutdown' is shown on the screen.

d) All nso-instances

```
nova stop $(nova list 2>&1 | grep nso | awk '{print $4}')
nova list 2>&1 | grep  nso | awk '{print $6}'
```

Note Run the list instances command until 'Shutdown' is shown on the screen.

e) All kubernetes-master nodes

```
nova stop $(nova list 2>&1 | grep master | awk '{print $4}')
nova list 2>&1 | grep  master | awk '{print $6}'
```

Note Run the list instances command until 'Shutdown' is shown on the screen.

f) All kubernetes-node, including 8 kube-nodes and 3 es-log-nodes

```
nova stop $(nova list 2>&1 | grep node | awk '{print $4}')
nova list 2>&1 | grep  node | awk '{print $6}'
```

Note Run the list instances command until 'Shutdown' is shown on the screen.

g) Inception VM

```
nova stop $(nova list 2>&1 | grep inception | awk '{print $4}')
nova list 2>&1 | grep  inception | awk '{print $6}'
```

Note Run the list instances command until 'Shutdown' is shown on the screen.

Step 2 Start the instances. To start these instances, you will need to access these instances from the installer container and start these one by one in the following order:

a) Inception VM

```
nova start $(nova list 2>&1 | grep inception | awk '{print $4}')
nova list 2>&1 | grep  inception | awk '{print $6}'
```

Note Run the list instances command until 'Running' is shown on the screen.

b) All kubernetes-nodes, including 8 kube-nodes and 3 es-log-nodes

```
nova start $(nova list 2>&1 | grep node | awk '{print $4}')
nova list 2>&1 | grep  node | awk '{print $6}'
```

Note Run the list instances command until 'Running' is shown on the screen.

c) All kubernetes-master nodes

```
nova start $(nova list 2>&1 | grep master | awk '{print $4}')
nova list 2>&1 | grep master | awk '{print $6}'
```

Note Run the list instances command until 'Running' is shown on the screen.

Access the kubernetes-master-1 node by using the following command:

```
ssh -F ssh.cfg $(nova list 2>&1 | grep 'kubernetes-master' | awk  '{print $12}' | sed -n 1p |
grep -o [[:digit:]].*)
sudo docker ps
```

Verify the health status by using the following command:

```
etcdctl cluster-health
```

Verify the expiry status of the vault client token. To do so, use the following commands:

```
curl -s -H "X-Vault-Token: $(sudo docker exec consul sh -c "consul kv get
userviceconfiguration/defaultapplication/spring.cloud.vault.token")"
```

If the instances have been offline for more than 24 hours or if the microservices keeps failing, execute the following command:

```
curl -s -X POST -H "X-Vault-Token: $(sudo cat /etc/vault/certs/vault.token | grep -o
[[:digit:]].*)" -d '{"policies":["skyfall_policy"],"period": 86400}'
http://vault.service.consul:8200/v1/auth/token/create |python -m json.tool
```

Use the client token output as an input to the root cron table and consul.

```
sudo docker exec consul sh -c "consul kv put userviceconfiguration/defaultapplication/
spring.cloud.vault.token $client_token"
sudo crontab -e
```

Replace the vault token string with the client token and save.

Exit from the kuberenetes master and re-enter the installer container.

d) All nso-instances

```
nova start $(nova list 2>&1 | grep nso | awk '{print $4}')
nova list 2>&1 | grep nso | awk '{print $6}'
```

Note Run the list instances command until 'Running' is shown on the screen.

e) All esc instances

```
nova start $(nova list 2>&1 | grep esc | awk '{print $4}')
nova list 2>&1 | grep esc | awk '{print $6}'
```

Note Run the list instances command until 'Running' is shown on the screen.

f) SME-instance

```
nova start $(nova list 2>&1 | grep sme | awk '{print $4}')
nova list 2>&1 | grep sme | awk '{print $6}'
```

Note Run the list instances command until 'Running' is shown on the screen.

g) All edge-instances

```
nova start $(nova list 2>&1 | grep edge | awk '{print $4}')
nova list 2>&1 | grep edge | awk '{print $6}'
```

Note Run the list instances command until 'Running' is shown on the screen.

After starting up all the virtual machines, kubernetes and all the pods will come up in about 5 to 10 minutes. To verify the status of various pods and the microservices after starting the instances, see the section 'Verifying the Microservices Status After the Graceful Shut Down and Restart'.

Verifying the Microservices Status After the Graceful Shut Down and Restart

Step 1 Verify the status of the **Kubernetes master node pods** (microservices). To do so, access the kubernetes-master-1 node and check the status of all the pods using the following commands

```
ssh -F ssh.cfg $(nova list 2>&1 | grep 'kubernetes-master' | awk  '{print $12}' | sed -n 1p |
grep -o [[:digit:]].*)

kubectl get po -n vms | awk '$3 !~ /Running/'
```

Note If the pods are running, nothing is displayed under the NAME tag. In that case, you can skip the below commands and move to step **2** to verify the status of the web interface pod.

(Optional Steps) Restart the pods if the pods are not in a running state. To do so, run the following commands:

```
kubectl delete -f /etc/kube-manifests/{microservice_name-rc.yml}

kubectl create -f /etc/kube-manifests/{microservice_name-rc.yml}
```

Note In the above commands, the pod name is the microservice_name.

Step 2 Verify the status of the **Web Interface pods** (microservices). To do so, use the following command:

```
curl $(kubectl describe pod -n vms skyfallui 2>&1 | grep IP | grep -o [[:digit:]].*):80 2>&1 |grep
'Index of'
```

Note If the pods are running, nothing is displayed after executing the above command. In that case, you can skip the below commands and move to step **3** to verify the status of the Orchestration pods.

(Optional Steps) Restart the web interface pods if the pods are not in a running state. To restart, run the following commands:

```
kubectl delete -f /etc/kube-manifests/skyfallui-rc.yml

kubectl create -f /etc/kube-manifests/skyfallui-rc.yml
```

After restarting the web interface pods, the pods will come up in about 2 minutes. Then restart step 2, and verify the status of the web interface pod.

Step 3 Verify the status of the **Orchestration pods** (microservices). To do so, use the following commands:

```
kubectl get po -n vms | grep orchestrationservice | awk '{print $1}
```

The above command will display names of the orchestration pods.

```
kubectl describe pod -n vms {orchestration_pod_name} | grep 'unhealth'
```

If the above command displays the keyword `unhealth`, it is recommended to restart the Orchestration pods. (Optional Steps)To restart the Orchestration pods, run the following commands:

```
kubectl delete -f /etc/kube-manifests/orchestrationservice-rc.yml

kubectl create -f /etc/kube-manifests/orchestrationservice-rc.yml
```

Step 4 Verify if the free marker templates (cloudvpn.ftl, dhcp.ftl, firewall.ftl) are available under /opt/cisco/templates/. To verify, use the following commands:

```
kubectl exec {orchestration_pod_name} -n vms -- cat /opt/cisco/templates/cloudvpn.ftl | grep
'<payload>'
```

If one of them shows the output as "no such file or directory", use the below command. If the outputs have a key word "<payload>", skip the below commands. If nothing shows up, restart the orchestration pods.

```
kubectl delete -f /etc/kube-manifests/orchestrationservice-rc.yml
kubectl create -f /etc/kube-manifests/orchestrationservice-rc.yml
```

After restarting the orchestration pods, the pods will come up in about 5 minutes. Then restart step 3, and verify the status of the orchestration pods.

Step 5 (Only for CVPN) If a service chain is stuck in the provisioned state, synchronize NSO-CVPN data to the NSO instance. To synchronize, do the following:

1. Exit from the kuberenetes master and execute the following commands from the installer container:

   ```
   ssh -F ssh.cfg $(nova list 2>&1 | grep 'nso.*cloudvpn' | awk '{print $12}' | grep -o [[:digit:]].*
    |
   sed -n 1p) sudo docker logs vms-ha-nso 2>&1 | grep  "still Master" | sed -n 1p
   ```

   If the output shows Master, execute the following command:

   ```
   ssh -F ssh.cfg $(nova list 2>&1 | grep 'nso.*cloudvpn' | awk '{print $12}' | grep -o [[:digit:]].*
    | sed -n 1p)
   ```

   If the output does not show Master, execute the following command:

   ```
   ssh -F ssh.cfg $(nova list 2>&1 | grep 'nso.*cloudvpn' | awk '{print $12}' | grep -o [[:digit:]].*
    | sed -n 2p)
   ```

2. Access the NSO container:

   ```
   sudo docker exec -it nso /bin/bash
   ```

3. Execute the following commands to synchronize the data:

   ```
   ncs_cli -u vmsnso
   request devices device nso-shim ssh fetch-host-keys
   request devices device nso-shim sync-to
   ```

Important Notes

In VMS 3.1 vBranch Service Pack, on the NSO server, you need to specify the location of the webserver on which the software image and the day 0 configuration available for NSO to deploy the VNFs.

To configure the location of the images, you need to do the following:

Step 1 Login in to NSO.

Step 2 Switch to the configuration mode.

```
vmsnso@ncs> configure
Entering configuration mode private
[ok][2017-12-01 15:38:02]

[edit]
vmsnso@ncs%
```

Step 3 Execute the following commands:

```
vmsnso@ncs% set nfvo vnfd vBranch-ISRv-1.0 vdu ISRv-small software-image-descriptor
image http://<IP Address>/vbranch/images/isrv-universalk9.16.06.01.tar.gz
[ok][2017-12-01 14:27:11]

[edit]
vmsnso@ncs% set nfvo vnfd vBranch-ASAv-1.0 vdu ASAv10 software-image-descriptor
image http://<IP Address>/vbranch/images/asav101-2-1-7.tar.gz
[ok][2017-12-01 14:27:11]

[edit]
vmsnso@ncs% set catalog vBranch deployment ISR day0-url iosxe_config.txt url http://<IP
Address>/day0/ISRv-day0.txt
[ok][2017-12-01 14:27:11]

[edit]
vmsnso@ncs% set catalog vBranch deployment ASA day0-url asav_config.txt url http://<IP
Address>/day0/ASAv-day0.txt
[ok][2017-12-01 14:27:11]

[edit]
vmsnso@ncs% set catalog vBranch deployment ISR-Security day0-url iosxe_config.txt url
http://<IP Address>/day0/ISRv-day0.txt [ok][2017-12-01 14:27:13]
```

Step 4 Commit the change.

```
[edit]
vmsnso@ncs% commit
Commit complete.
[ok][2017-12-01 14:27:19]
```

Limitations

- When more than one service pack is deployed on VMS, ensure that the PnP management address pool is unique. For example, if both the IWAN and CVPN service packs are running on a single VMS, the configuration should be as shown below
  - For IWAN
    ```
    set pnp mgmt-address-start 192.0.2.1
    ```
  - For CVPN
    ```
    set pnp mgmt-address-start 192.0.2.2
    ```
- Self-signed certificate is used during the installation for the PnP proxy server. Hence, the revocation check needs to be off on the CPE devices to be able to talk to the PnP proxy server

  Login into CPE and execute the following command to turn off the revocation check:
  ```
  crypto pki trustpoint ipnp
  enrollment terminal
  revocation-check none
  ```

Cisco Virtual Managed Services Bugs

For a complete list of open bugs for this release, use the Cisco Bug Search Tool.

- **Bug Search Tool Requirements**: Register for a Cisco account if you do not have one. Go to https://tools.cisco.com/RPF/register/register.do.
- You can filter for Known and Fixed Issues in the Bug Search Tool.

To filter: Select **Product** as *Cisco Virtual Managed Services* and enter *3.1.1* version for the **Releases** field.

See Bug Search Tools & Resources on Cisco.com. For more details on the tool overview and functionality, check out the help page, located at http://www.cisco.com/web/applicat/cbsshelp/help.html .

Accessibility Features

For a list of accessibility features in Cisco VMS, see Voluntary Product Accessibility Template (VPAT) on the Cisco website, or contact accessibility@cisco.com.

All product documents are accessible except for images, graphics, and some charts. If you would like to receive the product documentation in audio format, braille, or large print, contact accessibility@cisco.com.

Related Documentation

You can access the VMS documentation at http://www.cisco.com/c/en/us/support/cloud-systems-management/virtual-managed-services/tsd-products-support-series-home.html.

For additional reference, you can access the following 3.1 documents. You will need Cisco Customer or Cisco Employee privileges to access the 3.1 documentation.

Document	Description
Cisco Virtual Managed Services (VMS) 3.1 Installation Guide	This guide covers the installation of VMS solution including Cisco Network Service Orchestrator (NSO), Cisco Elastic Services Controller (ESC), and the VMS Service/Function Packs.
Cisco Virtual Managed Services (VMS) 3.1 IWAN Service Pack Guide	This guide includes details related to subscribing for IWAN service pack, configuring the service and troubleshooting service errors.
Cisco Virtual Managed Services (VMS) 3.1 Cloud VPN and Cloud VCE Service Pack Guide	This guide includes details related to subscribing for Cloud VPN and VCE service pack, configuring the service and troubleshooting Cloud VPN and VCE services
Cisco Virtual Managed Services(VMS) 3.1 vBranch Service Pack Guide	This guide includes details related to subscribing for vBranch service pack, configuring the service and troubleshooting service errors.
Cisco Virtual Managed Services(VMS) 3.1.1 SDWAN Service Pack Guide	This guide includes details related to deploy and manage SDWAN service, configuring the service and troubleshooting service errors.
Cisco Virtual Managed Services (VMS) 3.1 Solution Overview Guide	This guide provides a comprehensive explanation and guide to the design of the Cisco Virtual Managed Services solution that enables service providers to offer flexible and extensible services to their business customers.
Cisco Virtual Managed Services (VMS) 3.1 Platform Troubleshooting Guide	This guide covers troubleshooting information for Service Provisioning, Portal, including Microservices, Identification Management (IDM), Service Metrics Engine (SME), and so on.
Open Source	This guide contains licenses and notices for Open Source software used in this product.