Virtual Infrastructure Manager, Release 12.2.2

Table of Contents

New and Changed Information

The following table provides an overview of the significant changes up to this current release. The table does not provide an exhaustive list of all changes nor of the new features up to this release.

Release Version	Feature	Description
There were no major changes from the previous release.		

Virtual Infrastructure Manager

UI Path: **Virtual Management** > **Virtual Infrastructure Manager**

Ensure that you have enabled Network visualization of Virtual Machines feature for Cisco Nexus Dashboard Fabric Controller.

1. Choose **Settings** > **Feature Management**, choose the following check boxes:
   - Kubernetes Visualizer
   - VMM Visualizer
   - Openstack Visualizer
2. Click **Apply**.

The following table describes the fields that appear on Virtual Infrastructure Manager window:

Field	Description
Server	Specifies the Server IP Address.
Type	Specifies the type of instance that can be one of the following: ▪ vCenter ▪ Kubernetes Cluster ▪ OpenStack Cluster
Managed	Specifies the status of the cluster either Managed or Unmanaged.
Status	Specifies the status of the added cluster.
User	Specifies the user created the cluster.
LastUpdated Time	Specifies the last updated time for the cluster.

Click **Refresh** icon to refresh the Virtual Infrastructure Manager table.

The following table describes the action items, in the Actions menu drop-down list, that appear on Virtual Infrastructure Manager window:

Action Item	Description
Add Instance	From the **Actions** drop-down list, choose **Add Instance**. For more instructions, see Adding an Instance. NOTE: Ensure that you have configured same IP address on Routes. Refer to Configuring Routes IP Address.
Edit Instance	Choose an instance to edit. From the **Actions** drop-down list, choose **Edit Instance**. Make the necessary changes and click **Save**. Click **Cancel** to discard the changes.

Action Item	Description
Delete Instance(s)	Choose one or more required instance to delete. From the **Actions** drop-down list, choose **Delete Instance(s)**. Click **Confirm** to delete the instance. Click **Cancel** to discard the delete.
Rediscover Instance(s)	Choose one or more required instance to rediscover. From the **Actions** drop-down list, choose **Rediscover Instance(s)**. A confirmation message appears.

Support for Cisco UCS B-Series Blade Servers

NDFC supports hosts running on UCS type B (chassis UCS) that are behind the Fabric interconnect. You must enable CDP of the vNIC on Cisco UCSM to use this feature.

By default, CDP is disabled on Cisco UCSM.

Consider two VMMs, VMM-A and VMM-B, for reference. After the discovery of Cisco UCS UCS B-Series Blade Servers, the Topology displays the blue colored VMM-A and VMM-B are fabric interconnect nodes. A sample topology is as shown in the figure below.

To enable CDP on UCSM, you must create a new Network Control policy using the following steps:

1. On the USCM, choose **LAN** and expand the policies.
2. Right-click on the **Network Control Policies** to create a new policy.
3. In the Name field, enter the policy name as **EnableCDP**.
4. Choose **enabled** option for CDP.
5. Click **OK** to create the policy.

To apply the new policy to the ESX NICs, perform the following steps:

- If you are using updated vNIC templates, choose each vNIC template for your ESXi vNICs, and apply the EnableCDP policy from the Network Control Policy drop-down list.
- If you are not using any vNIC templates, use the updated Service Profile Template. Apply EnableCDP policy on each of the service profile template.
- If you are using one-off Service Profiles (i.e., if each server using its own service profile), then you must go to every Service Profile and enable EnableCDP policy on every vNIC.

For more information about Cisco UCSM, refer to Cisco UCSM Network Management Guide.

Configuring Routes IP Address

Before you add IP address to vCenter, you must configure same IP address on Cisco Nexus Dashboard.

To configure Routes on Cisco Nexus Dashboard, perform the following steps:

1. Choose **Infrastructure** > **Cluster Configuration**.
2. On **General** tab, in **Routes** card, click **Edit** icon.

   The **Routes** window appears.

3. To configure IP addresses, click **Add Management Network Routes**, enter required IP addresses, and click **check** icon.
4. Click **Save**.

   The route configuration is governed by following two scenarios:

   - For vCenter, which is an application server is typically reachable over mgmt network.
   - The ESXi servers that are managed by vCenters and the baremetal servers hosting the K8s instances and/or OpenStack instances would be connected to the fabric network directly. Hence, they will be reachable over data networks.

Adding vCenter Visualization

You can perform various actions in the **Actions** menu drop-down list, that appear on **Virtual Management > Virtual Infrastructure Manager.**

1. Choose **Actions** > **Add Instance**.

   The **Add Instance** window appears.

2. Choose **vCenter** from Select Type drop-down list.

   Enter required IP address or Domain name and password in the respective fields.

3. Click **Add**.

   You can view added vCenter cluster in the *Virtual Infrastructure Manager*window.

4. To edit an instance, choose required vCenter, choose **Actions** > **Edit Instance** and click **Save** changes.

   You can update password for the selected vCenter cluster and change the admin status to Managed or Unmanaged and vice-versa.

   

   For the vCenter cluster in Unmanaged status, you cannot view the topology and vCenter cluster details on dashboard.

5. To delete one or more vCenter cluster, choose the required vCenter, choose **Actions** > **Delete Instance(s)** and click **Confirm** changes.

   

   All the data will be deleted if you delete the Cluster. The Cluster will be removed from the Topology view also.

6. To rediscover one or more vCenter cluster, choose the required vCenter, choose **Actions** > **Rediscover Instance(s)**.

   A confirmation message appears.

Kubernetes Cluster

Ensure that you have enabled Network Visualization of K8s clusters feature for Cisco Nexus Dashboard Fabric Controller .

Choose **Admin > System Settings > Feature Management** choose check box **Kubernetes Visualizer** and click **Apply**.

You can view the added Kubernetes Visualizer details on dashboard. Navigate **Dashboard > Kubernetes Pods**.

To enable LLDP on NDFC, choose **Settings > Server > Settings > Discovery**. Choose check box **enable / disable neighbor link discovery using LLDP**.

LLDP is applicable for Bare-metal Kubernetes clusters only.

- Ensure that the LLDP feature is enabled on all fabric switches for which the cluster node is connected. (Switches may be spine or leaf switches).
- On the Kubernetes cluster, ensure that LLDP and SNMP services are enabled on all Bare-metal nodes.
- If the Cisco UCS is using an Intel NIC, LLDP neighborship fails to establish due to FW-LLDP.

  Workaround - For selected devices based on the Intel® Ethernet Controller (for example, 800 and 700 Series), disable the LLDP agent that runs in the firmware. Use the following command to disable LLDP:

  echo 'lldp stop' > /sys/kernel/debug/i40e/<bus.dev.fn>/command

  To find the bus.dev.fn for a given interface, run the following command and select the ID associated with the interface. The ID is highlighted in the below sample output.

  [ucs1-lnx1]# dmesg | grep enp6s0 [12.609679] IPv6: ADDRCONF(NETDEV_UP): enp6s0: link is not ready [12.612287] enic 0000:06:00.0 enp6s0: Link UP [12.612646] IPv6: ADDRCONF(NETDEV_UP): enp6s0: link is not ready [12.612665] IPv6: ADDRCONF(NETDEV_CHANGE): enp6s0: link becomes ready[ucs1-lnx1]#

LLDP feature is enabled on those fabric switches, to which the bare-metal cluster nodes are connected. They can also be connected to the border gateway switches. If the Fabric, to which the Kubernetes cluster is connected to, is discovered after the Cluster was discovered, you must rediscover the cluster to display the topology correctly.

If the Bare-metal-based Kubernetes cluster is discovered after configuring LLDP, you must rediscover the Baremetal cluster to display the topology correctly.

To find the bus.dev.fn for a given interface, run the following command and select the ID associated with the interface. The ID is highlighted in the below sample output.

When discovering or visualizing VM-based Kubernetes cluster, it must first onboard the vCenter cluster which is managing the VMs hosting the Kubernetes cluster being

discovered. Without this, Kubernetes cluster discovery would result in failure.

Configuring Routes IP Address

Before you add IP address to Kubernetes cluster, you must configure same IP address on Cisco Nexus Dashboard.

To configure Routes on Cisco Nexus Dashboard, perform the following steps:

1. Choose **Infrastructure** > **Cluster Configuration**.
2. On **General** tab, in **Routes** card, click **Edit** icon.

   The **Routes** window appears.
3. To configure IP addresses, click **Add Management Network Routes**, enter required IP addresses, and click **check** icon.
4. Click **Save**.

Adding Kubernetes Cluster

You can perform various actions in the **Actions** menu drop-down list, that appear on **Virtual Management > Virtual Infrastructure Manager**.

Ensure that you have configured same IP address on Routes. Refer to Configuring Routes IP Address.

1. Choose **Actions > Add Instance**

   The **Add Instance** window appears.
2. Choose **Kubernetes Cluster** from Select Type drop-down list.
3. Enter the **Cluster IP address** and the **Username** in appropriate fields.
4. Click **Fetch CSR** to obtain a Certificate Signing Request (CSR) from the Kubernetes Visualizer application.

   

   This option is disabled until you enter a valid Cluster IP address and username.

   Use the **Fetch CSR** only if you haven't obtained the SSL certificate. If you already have a valid certificate, you need not fetch the CSR.

   Click **Download CSR**. The certificate details are saved in the **<username>.csr** in your directory. Paste the contents of the CSR to a file **kubereader.csr**, where kubereader is the username of the API Client to connect to Kubernetes.

   The CSR file name must adhere to naming convention *[username].csr*.

   

   As the certificates are generated on the Kubernetes cluster, you need Kubernetes admin privileges to generate certificates.
   Refer to Annexure to generate the certificate **genk8clientcert.sh**.

5. Login to the Kubernetes cluster controller node.

   You need admin privileges to generate the certificates.

6. Copy the genk8clientcert.sh and kubereader.csr from the NDFC server location to the Kubernetes Cluster controller node.

   Perform a "vnc cut and paste" operation to ensure that all the characters are copied correctly.

7. Generate the CSR for the user name, by using the **genk8sclientcert.sh** script.

   (k8s-root)# *./genk8sclientcert.sh kubereader 10.x.x.x*where,

   - kubereader is the username of the API Client to connect to Kubernetes. (as defined in Step 3).
   - 10.x.x.x is the IP address of the NDFC server.

     There are two new certificates generated in the same location:

   - k8s_cluster_ca.crt
   - *username_dcnm-IP.crt*

     For example: kubereader_10.x.x.x.crt (where, kubereader is the username, and 10.x.x.x is the NDFC IP address)

   ` dcnm(root)# cat k8s_cluster_ca.crt`

8. Use the cat command to extract the certificate from these 2 files.

   ```
   dcnm(root)# cat kubereader_10.x.x.x.crt
   dcnm(root)# cat k8s_cluster_ca.crt
   ```

   Provide these two certificates to the user, who is adding the Kubernetes cluster on Cisco NDFC.

9. Copy the content in the kubereader_10.x.x.x.crt to **Client Certificate** field.

   

   Perform a "vnc cut and paste" operation to ensure that all the characters are copied correctly.

10. Copy the content in the k8s_cluster_ca.crt to **Cluster Certificate** field.

    

    Perform a "vnc cut and paste" operation to ensure that all the characters are copied correctly.

11. Click **Add**.

    You can view added Kubernetes cluster in the **VirtualInfrastructure Manager** window.

    

    You can view details of the added Kubernetes cluster details on the dashboard and topology window. Navigate **Dashboard** > **Kubernetes Pods** and topology window.

12. To edit Kubernetes cluster, choose required cluster, choose **Actions > Edit Instance**, click Edit to modify the values appropriately. You can update the Cluster and the Client certificates. You can also update the Managed status of the Kubernetes cluster. If you choose to update the Managed status, certificates are not required.

    

    For the kubernetes cluster in Unmanaged status, you cannot view the topology and Kubernetes cluster details on dashboard.

13. Click **Save** to save the changes or click **Cancel** to discard changes.
14. To delete one or more Kubernetes Cluster, choose the required cluster, choose **Actions** > **Delete Instance(s)** to delete the cluster.

    

    All the data will be deleted if you delete the Cluster. The Cluster will be removed from the Topology view also.

15. Click **Confirm** to delete the cluster.
16. To rediscover one or more Kubernetes cluster, choose required Kubernetes cluster, choose **Actions** > **Rediscover Instance(s)**.

    A confirmation message appears.

Annexure

The following message is displayed, after the certificates are generated successfully:

```
#!/usr/bin/bash
#########################################################################
# Title: Script to provision the client CSR and generat the #
#        the client SSL certificate.                    #
#########################################################################

# Create CSR resource template.
function create_csr_resource() {
    K8SUSER=$1
    DCNM=$2
    FILE=${K8SUSER}_${DCNM}_csr_res.yaml
    echo "
apiVersion: certificates.k8s.io/v1
kind: CertificateSigningRequest
metadata:
  name: ${K8SUSER}_${DCNM}csr
spec:
  groups:
  - system:authenticated
  request: ${BASE64_CSR}
  signerName: kubernetes.io/kube-apiserver-client
  usages:
  - digital signature
  - key encipherment
  - client auth"  > $FILE
}

# Create CLUSTER ROLE resource template
function create_cluster_role() {
    K8SUSER=$1
    DCNM=$2
    FILE=${K8SUSER}_${DCNM}_cluster_role_res.yaml
    echo "
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: clustrole_${K8SUSER}_${DCNM}
rules:
- apiGroups: [\" \" ]
  resources: [" nodes" , " namespaces" , " pods" , " services" ]
  verbs: [" get" , " list" , " watch" ]"  > $FILE
```

```
}

# Create CLUSTER ROLE BINDING template
function create_cluster_role_binding() {
    K8SUSER=$1
    DCNM=$2
    FILE=${K8SUSER}_${DCNM}_cluster_rolebinding_res.yaml
    echo "
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: clustrolebind_${K8SUSER}_${DCNM}
roleRef:
  kind: ClusterRole
  name: clustrole_${K8SUSER}_${DCNM}
  apiGroup: rbac.authorization.k8s.io
subjects:
- kind: User
  name: ${K8SUSER}
  apiGroup: rbac.authorization.k8s.io"  > $FILE
}

function valid_ip() {
    local  ip=$1
    local  stat=1

    if [[ $ip =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]]; then
        OIFS=$IFS
        IFS='.'
        ip=($ip)
        IFS=$OIFS
        [[ ${ip[0]} -le 255 && ${ip[1]} -le 255 \
            && ${ip[2]} -le 255 && ${ip[3]} -le 255 ]]
        stat=$?
    fi
    return $stat
}

# Start of the script
if [ "$#"  -ne 2 ]; then
    echo "Please provide the username and IP of the DCNM"
    echo
    exit 1
else
```

```
    # Check if user have required K8s privileges
    LINUX_USER=$(whoami)
    K8S_CONF_PATH=" "
    echo
    echo " Hello ${LINUX_USER}! I am going to help you generate K8s cluster CA and K8s
client certificate."

    if [ ${LINUX_USER} == " root"  ] ; then
        # You are root
        if [ ! -d " /root/.kube"  ] ; then
            echo
            echo " Directory /root/.kube does not exists."
            echo " User ${LINUX_USER} does not have required K8s privileges"
            echo " Please make sure you are logged into K8s cluster's master node"
            echo
            exit 1
        else
            K8S_CONF_PATH=/${LINUX_USER}/.kube/config
        fi
    else
        # You are not root
        if [ ! -d " /home/${LINUX_USER}/.kube"  ] ; then
            echo
            echo " Directory /home/${LINUX_USER}/.kube does not exists."
            echo " User ${LINUX_USER} does not have required K8s privileges"
            echo " Please make sure you are logged into K8s cluster's master node"
            echo
            exit 1
        else
            K8S_CONF_PATH=/home/${LINUX_USER}/.kube/config
        fi
    fi

    # Check if K8s config file exist
    if [ ! -f ${K8S_CONF_PATH} ]; then
        echo
        echo " ${K8S_CONF_PATH} file does not exist"
        echo " K8s CA certificate can not be exported"
        echo " Please make sure you are logged into K8s cluster's master node"
        echo
        exit 1
    fi

    K8SUSER=$1
    DCNM=$2
```

```
K8S_CA_CRT=" k8s_cluster_ca.crt"

# Validate the IP address
if valid_ip $DCNM; then
    echo -e
else
    echo " ${2} is not a valid IP address"
    echo
    exit 1
fi

# Validate the CSR file format
if [ ${K8SUSER: -4} == " .csr"  ]; then
    K8SUSER=${K8SUSER%.csr}
fi

if [ ! -f " ./${K8SUSER}.csr"  ]; then
    echo
    echo " ./${K8SUSER}.csr does not exist"
    echo " CSR file is required for creation of client certificate"
    echo
    exit 1
fi

echo " Generating certificate for ${K8SUSER} for DCNM ${DCNM}"
echo

# Encoding the .csr file in base64
export BASE64_CSR=$(cat ./${K8SUSER}.csr | tr -d '\n')

# Create the CSR resource in K8s cluster
create_csr_resource $K8SUSER $DCNM

# Delete if the CSR resource already exist. We need a fresh one.
kubectl delete csr ${K8SUSER}_${DCNM}csr &> /dev/null
status=$?
if test $status -eq 0
then
    echo " ./${K8SUSER}_${DCNM}csr CSR resource already exist, removing it"
else
    echo " ./${K8SUSER}_${DCNM}csr CSR resource does not exist, creating it"
fi

# Create the CertificateSigninRequest resource
kubectl apply -f ${K8SUSER}_${DCNM}_csr_res.yaml
```

```
    # Check the status of the newly created CSR
    kubectl get csr

    # Approve this CSR
    echo " Approving the CSR"
    kubectl certificate approve ${K8SUSER}_${DCNM}csr

    # Check the status of the newly created CSR
    kubectl get csr

    # Create role resource definition
    kubectl delete clusterrole clustrole_${K8SUSER}_${DCNM} &> /dev/null
    create_cluster_role $K8SUSER $DCNM
    kubectl apply -f ${K8SUSER}_${DCNM}_cluster_role_res.yaml

    # Create role binding definition
    kubectl delete clusterrolebinding clustrolebind_${K8SUSER}_${DCNM} &> /dev/null
    create_cluster_role_binding $K8SUSER $DCNM
    kubectl apply -f ${K8SUSER}_${DCNM}_cluster_rolebinding_res.yaml

    # Extract the client certificate
    echo " Extracting the user SSL certificate"
    kubectl get csr ${K8SUSER}_${DCNM}csr -o jsonpath='{.status.certificate}' >
${K8SUSER}_${DCNM}.crt
    echo " "  >> ${K8SUSER}_${DCNM}.crt

    # Export the K8s cluster CA cert
    if [ -f ${K8S_CONF_PATH} ]; then
        echo " Exporting K8s CA certificate"
        cat ${K8S_CONF_PATH} | grep certificate-authority-data | awk -F ' ' '{print $2}' >
${K8S_CA_CRT}
    fi
    echo
    echo " ------------------------------------------------------------"
    echo " Notes: "
    echo " 1. The K8s CA certificate is copied into ${K8S_CA_CRT} file."
    echo "       This to be copied into \" Cluster CA\"  field."
    echo " 2. The client certificate is copied into ${K8SUSER}_${DCNM}.crt file."
    echo "       This to be copied into \" Client Certificate\"  field."
    echo " ------------------------------------------------------------"
    echo
fi
```

Copyright