Install Cisco Optical Network Controller Using OpenStack

To deploy the Cisco Optical Network Controller using OpenStack, follow the instructions in this task. The deployment leverages a Heat Orchestration Template to automate the creation of necessary components and configurations.

Heat Orchestration Template

A Heat orchestration template will be provided to create the required components for the instance. The template includes configurations for block storage, security groups, and network settings.

Components Created by Heat Template

- **Block Storage**: Image and data volumes are created and attached to the instance.
- **Security Groups**: Security groups for network ports are established.
- **Network Configuration**: A control plane network and subnet are created as a private network, and a northbound port will be created.
- **Join Token**: Random text is generated to be used as a join token.
- **Cloud-Init Configuration**: The cloud-init is prefilled based on the parameters that are obtained during stack launch.

Before you begin

- **OpenStack Version**: 2024.1

  See OpenStack Documentation for release 2024.1 for details on how to use OpenStack.

- **Upload Image**: Upload the Cisco Optical Network Controller (qcow2) image to the server..

  Use the following CLI command to upload the image to the OpenStack project.

  ```
  openstack image create --disk-format=qcow2 --file <path-to-image>.qcow2 \
     --shared \
     --property hw_firmware_type='uefi' \
     --property hw_machine_type='q35' \
     --property architecture='x86_64' \
     --progress \
     "Image Name"
  ```

  After you perform these commands, the qcow2 image is available for deployment in OpenStack.

Note Install the OpenStack Command Line Interface (CLI) and source the OpenStack Cloud RC file or clouds.yaml before running the command. For installation instructions, see Install the OpenStack command-line clients.

- **Configure Network**: Use the northbound network for Cisco Optical Network Controller to expose the UI and REST APIs. Cisco Optical Network Controller uses this northbound network to connect to the devices.
- **Create Flavors:** It is optional to have physical disks/ephemeral storage. While creating a flavor both physical disks/ephemeral storage cane be set to 0GB as block storage volumes handle both the image and data volumes.

  To create a flavor, in the OpenStack Dashboard, select the admin project from the drop-down list, select **Admin** > **Compute** > **Flavors** > **Create Flavor** and enter the parameters for the flavor.

  Note You need administrative access to OpenStack to create flavors.

  The minimum requirement for Cisco Optical Network Controller 24.3.1 installation are in the following table.

  Table 1: Minimum Requirement

Sizing	CPU	Memory	Disk
XS	16 vCPU	64 GB	800 GB
S	32 vCPU	128 GB	1.5 TB

- **Create Key Pair**: Create a key pair using the ed25519 algorithm. Upload Public SSH Key to OpenStack by going to **Project** > **Compute** > **Key Pairs** and select Import Public Key.

  Run the following command in a UNIX-based environment to create an SSH key pair:

```
ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/Users/xyz/.ssh/id_ed25519):
./<file-name-of-your-key>.pem
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in ./<file-name-of-your-key>.pem
Your public key has been saved in ./<file-name-of-your-key>.pem.pub
The key fingerprint is:
SHA256:zGW6aGn8rxvEq82sA/97jOaHrl9rnoTaYi+TqU3MeRU xyz@abc
The key's randomart image is:
+--[ED25519 256]--+
|                 |
|                 |
|          E      |
|       + + .     |
|        S .      |
|    .+ = =       |
|     o@o*+o      |
|     =XX++=o     |
|    .o*#/X=      |
+----[SHA256]-----+

#Once created you can cat the file with .pub extension for the public key. ( ex:
<file-name-of-your-key>.pem.pub )

cat <file-name-of-your-key>.pem.pub
#The above key has to be used in the deployment template ( SSH Public Key ) in the
Deployment process
```

Follow the prompts to save the key. The key pair will be used to access Cisco Optical Network Controller after the installation.

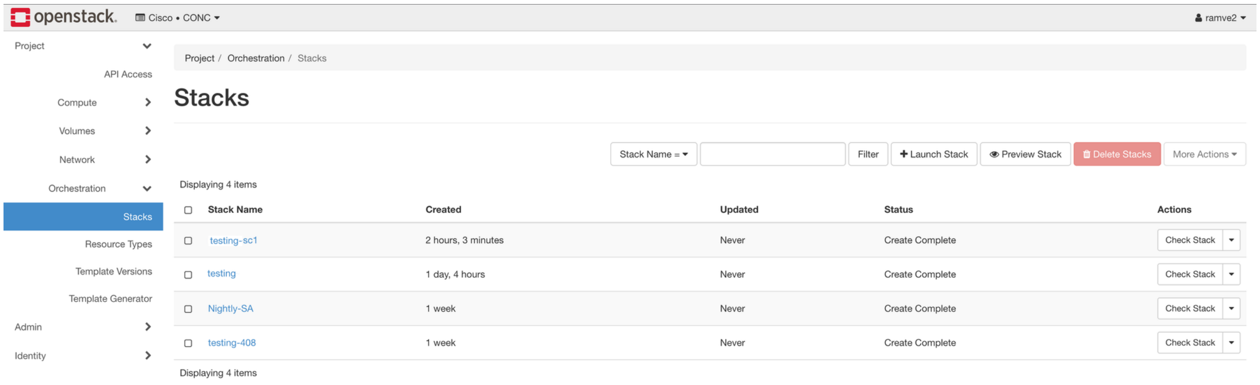
- You must have an NTP server or NTP Pool for time synchronization.
- You must have a DNS server. The DNS server can be an internal DNS server if the Cisco Optical Network Controller instance is not exposed to the internet.

Perform the following steps to install Cisco Optical Network Controller using OpenStack.

Step 1 Log in to OpenStack.

Step 2 Select **Project** > **Orchestration** > **Stacks** from the sidebar.

Figure 1: OpenStack Stacks Screen



Stacks

Displaying 4 items

□	Stack Name	Created	Updated	Status	Actions
□	testing-sc1	2 hours, 3 minutes	Never	Create Complete	Check Stack
□	testing	1 day, 4 hours	Never	Create Complete	Check Stack
□	Nightly-SA	1 week	Never	Create Complete	Check Stack
□	testing-408	1 week	Never	Create Complete	Check Stack

Displaying 4 items

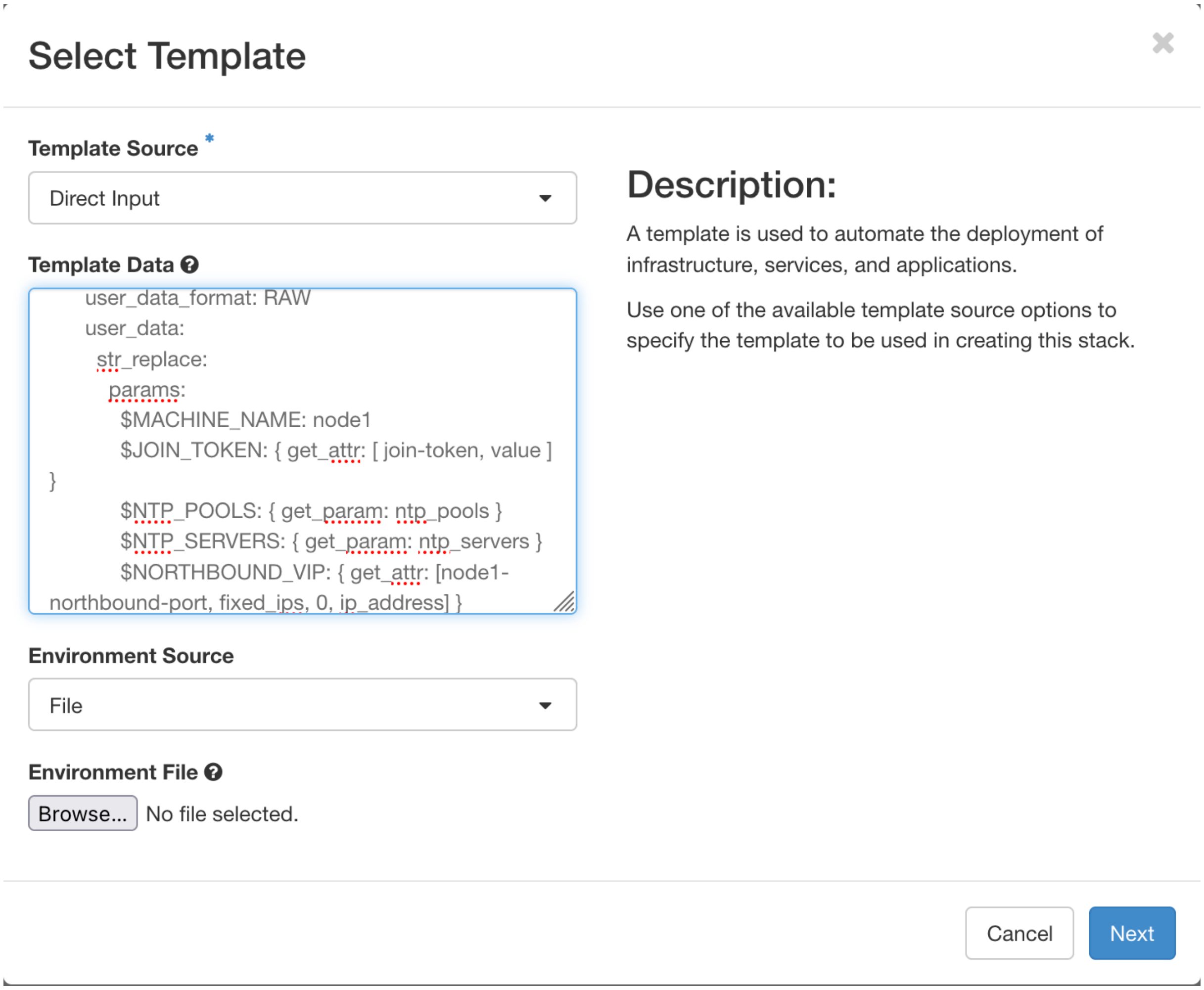
Step 3 Launch Stack.

a) Click **Launch Stack**

b) Choose **Template as File** and Upload the Heat orchestration template file or choose **Direct Input** and paste the contents of the file.

Note Incorrect indentation causes parsing errors. Validate the file with a YAML validator.

Figure 2: Select Template



Select Template

Template Source *

Direct Input

Template Data

```
     user_data_format: RAW
     user_data:
       str_replace:
         params:
           $MACHINE_NAME: node1
           $JOIN_TOKEN: { get_attr: [ join-token, value ]
}
           $NTP_POOLS: { get_param: ntp_pools }
           $NTP_SERVERS: { get_param: ntp_servers }
           $NORTHBOUND_VIP: { get_attr: [node1-
northbound-port, fixed_ips, 0, ip_address] }
```

Environment Source

File

Environment File

Browse... No file selected.

Description:

A template is used to automate the deployment of infrastructure, services, and applications.

Use one of the available template source options to specify the template to be used in creating this stack.

The following sample is a Heat Orchestration Template file for Cisco Optical Network Controller.

```
heat_template_version: "2021-04-16"
description: "NxFOS Heat Template"
parameters:
  instance_flavor:
    type: string
    label: Instance Flavor
    constraints:
    - custom_constraint: nova.flavor
  image_name:
    type: string
    label: CONC Image Name
    constraints:
    - custom_constraint: glance.image
  northbound_network:
    type: string
    label: Northbound Network
    constraints:
    - custom_constraint: neutron.network
  northbound_subnet:
    type: string
    label: Northbound Subnet
  northbound_vip:
    type: string
    label: Northbound VIP address
    default: "10.1.1.1"
  control_key_pair:
    type: string
    label: Control plane SSH key-pair
    constraints:
    - custom_constraint: nova.keypair
  data_volume_size_gb:
    type: number
    label: Data volume size in GB
    default: 200
  ntp_pools:
    type: comma_delimited_list
    description: List of NTP pools
    default: "0.pool.ntp.org,1.pool.ntp.org"
  ntp_servers:
    type: comma_delimited_list
    description: List of NTP servers
    default: ""

resources:
  # Security Groups
  control-sec-group:
    type: OS::Neutron::SecurityGroup
    properties:
      rules:
      # K8s
      - { protocol: tcp, remote_ip_prefix: 10.1.0.0/24, port_range_min: 443, port_range_max:
443 }
      - { protocol: tcp, remote_ip_prefix: 10.1.0.0/24, port_range_min: 6443, port_range_max:
6443 }
      - { protocol: tcp, remote_ip_prefix: 10.1.0.0/24, port_range_min: 10250, port_range_max:
 10250 }

      # Etcd (Port 2379 + 2380)
      - { protocol: tcp, remote_ip_prefix: 10.1.0.0/24, port_range_min: 2379, port_range_max:
2380 }

      # Flannel CNI
      - { protocol: udp, remote_ip_prefix: 10.1.0.0/24, port_range_min: 8472, port_range_max:
8472 }

      # Ping between nodes
      - { protocol: icmp, remote_ip_prefix: 10.1.0.0/24 }

  northbound-sec-group:
    type: OS::Neutron::SecurityGroup
    properties:
      rules:
      # SSH (Debug purposes only)
      - { protocol: tcp, remote_ip_prefix: 0.0.0.0/0, port_range_min: 22, port_range_max: 22 }

      # Northbound ingress-proxy
     - { protocol: tcp, remote_ip_prefix: 0.0.0.0/0, port_range_min: 8443, port_range_max: 8443
 }

  # Networks
  control-plane-network:
    type: OS::Neutron::Net
    properties:
      admin_state_up: true
  control-plane-subnet:
    type: OS::Neutron::Subnet
    properties:
      network_id: { get_resource: control-plane-network }
      gateway_ip: null
      cidr: "10.1.0.0/24"
      ip_version: 4

  # Control Ports
  node1-control-port:
    type: OS::Neutron::Port
    properties:
      security_groups: [ { get_resource: control-sec-group } ]
      network: { get_resource: control-plane-network }
      fixed_ips:
      - subnet_id: { get_resource: control-plane-subnet }
        ip_address: "10.1.0.10"

  # Northbound Ports
  node1-northbound-port:
    type: OS::Neutron::Port
    properties:
      security_groups: [ { get_resource: northbound-sec-group } ]
      network: { get_param: northbound_network }
      fixed_ips:
      - subnet_id: { get_param: northbound_subnet }
        ip_address: { get_param: northbound_vip }

  # Join Token
  join-token-id:
    type: OS::Heat::RandomString
    properties:
      character_classes:
      - class: lowercase
      - class: digits
      length: 6

  join-token-secret:
    type: OS::Heat::RandomString
    properties:
      character_classes:
      - class: lowercase
      - class: digits
      length: 16

  join-token:
    type: OS::Heat::Value
    properties:
      type: string
      value:
        list_join: [ '.', [ { get_resource: join-token-id }, { get_resource: join-token-secret
 } ] ]

  # Data Volumes
  node1-data-volume:
    type: OS::Cinder::Volume
    properties:
      size: { get_param: data_volume_size_gb }

  # Instances
  node1:
    type: OS::Nova::Server
    properties:
      networks:
      - port: { get_resource: node1-control-port }
      - port: { get_resource: node1-northbound-port }
      flavor: { get_param: instance_flavor }
      key_name: { get_param: control_key_pair }
      block_device_mapping_v2:
      - device_name: vda
        image: { get_param: image_name }
        volume_size: 50
        delete_on_termination: true
      - device_name: vdb
        volume_id: { get_resource: node1-data-volume }
        boot_index: -1
        delete_on_termination: true
      user_data_format: RAW
      user_data:
        str_replace:
          params:
            $MACHINE_NAME: node1
            $JOIN_TOKEN: { get_attr: [ join-token, value ] }
            $NTP_POOLS: { get_param: ntp_pools }
            $NTP_SERVERS: { get_param: ntp_servers }
            $NORTHBOUND_VIP: { get_attr: [node1-northbound-port, fixed_ips, 0, ip_address] }
            $POSTGRES_CONFIG: '{"config": {"max_connections": "1000","idle_session_timeout":
"900000"},"resources": {"requests": {"memory": "3.22%","cpu": "3.33%"},"limits": {"memory":
"9.66%","cpu": "11%"}}}'
            $KAFKA_CONFIG:
'{"enabled":true,"resources":{"requests":{"memory":"7.52%","cpu":"3.33%"},"limits":{"memory":"10.74%","cpu":"5.4%"}},"config":{"message.max.bytes":15000012}}'

          template: |
            #cloud-config
            fs_setup:
            - label: data
              device: /dev/vdb
              filesystem: ext4

            mounts:
            - [ "/dev/vdb", "/data" ]

            ntp:
              enabled: true
              ntp_client: chrony
              pools: $NTP_POOLS
              servers: $NTP_SERVERS

            nxf:
              minControlPlaneCount: 1
              node:
                name: $MACHINE_NAME
                controlPlaneInterface: enp3s0
                vip:
                  northbound:
                    interface: enp4s0

              initiator:
                vip:
                  northbound:
                    ip: $NORTHBOUND_VIP
                postgres: $POSTGRES_CONFIG
                kafka: $KAFKA_CONFIG
                minio:
                  resources:
                    limits:
                      memory: "5.37%"
```

```
joinToken: $JOIN_TOKEN
security:
  localUsers:
  - username: admin
    displayName: NxF Admin
    description: NextFusion Default Administrator
    locked: true
    mustChangePassword: false
    expiresInDays: 0
    access:
    - permission/admin
```

Step 4 In the Launch Stack dialog box, enter the Stack Parameters.

Table 2: Stack Parameters

Key	Value
Stack Name	Name of the stack, which will be used as part of the Node name.
Creation Timeout (minutes)	Can be left to default. Value can be changed to support the respective environment.
Password for the user	Enter the password of the OpenStack account used to log in.
Control Plane SSH Key Pair	Select the key pair (Should be an ed25519 SSH key).
Data Volume Size in GB	Enter the size of the data volume size based on the Cisco Optical Network Controller profiles.
CONC Image Name	Select the Cisco Optical Network Controller Image (qcow2).
Instance Flavor	Select the respective Cisco Optical Network Controller flavor based on the profiles.
Northbound Network	Select the Northbound Network.
Northbound Subnet	Enter the name in the text field of the Northbound Subnet.
Northbound VIP Address	Public IP, which will be used for both management and Northbound communications.
NTP Pools	Enter the NTP Pools. Leave empty if you are using an NTP Server.
NTP Server	Enter the NTP Server. Leave empty if you are using an NTP Pool.

Step 5 Click **Launch**.
This creates the stack. Use the PEM key to SSH into the node.

Note Wait for the stack creation status to change to **Create Complete** before you try to SSH into the node. Stack creation can take up to 10 minutes.

Step 6 **SSH to the node** and execute the following CLI command.

```
ssh -i [ed25519 Private key] nxf@<northbound-vip>
Enter passphrase for key '<file-name-of-your-key>.pem':
```

Note Private key is created as part of the key generation with just the **.pem** extension, and it must be set with the least permission level before using it.

Step 7 After you SSH into the node, use the sedo system status command to check the status of all the pods.

```
sedo system status
```

```
┌──────────────────────────────────────────────────────────────────────────────────┐
│ System Status (Fri, 20 Sep 2024 08:21:27 UTC)                                    │
├────────┬──────────────────────────────┬───────┬─────────┬──────────┬─────────────┤
│ OWNER  │ NAME                         │ NODE  │ STATUS  │ RESTARTS │ STARTED     │
├────────┼──────────────────────────────┼───────┼─────────┼──────────┼─────────────┤
│ onc    │ monitoring                   │ node1 │ Running │ 0        │ 3 hours ago │
│ onc    │ onc-alarm-service            │ node1 │ Running │ 0        │ 3 hours ago │
│ onc    │ onc-apps-ui-service          │ node1 │ Running │ 0        │ 3 hours ago │
│ onc    │ onc-circuit-service          │ node1 │ Running │ 0        │ 3 hours ago │
│ onc    │ onc-collector-service        │ node1 │ Running │ 0        │ 3 hours ago │
│ onc    │ onc-config-service           │ node1 │ Running │ 0        │ 3 hours ago │
│ onc    │ onc-devicemanager-service    │ node1 │ Running │ 0        │ 3 hours ago │
│ onc    │ onc-inventory-service        │ node1 │ Running │ 0        │ 3 hours ago │
│ onc    │ onc-nbi-service              │ node1 │ Running │ 0        │ 3 hours ago │
│ onc    │ onc-netconfcollector-service │ node1 │ Running │ 0        │ 3 hours ago │
│ onc    │ onc-osapi-gw-service         │ node1 │ Running │ 0        │ 3 hours ago │
│ onc    │ onc-pce-service              │ node1 │ Running │ 0        │ 3 hours ago │
│ onc    │ onc-pm-service               │ node1 │ Running │ 0        │ 3 hours ago │
│ onc    │ onc-pmcollector-service      │ node1 │ Running │ 0        │ 3 hours ago │
│ onc    │ onc-topology-service         │ node1 │ Running │ 0        │ 3 hours ago │
│ onc    │ onc-torch-service            │ node1 │ Running │ 0        │ 3 hours ago │
│ system │ authenticator                │ node1 │ Running │ 0        │ 12 hours ago│
│ system │ controller                   │ node1 │ Running │ 0        │ 12 hours ago│
│ system │ flannel                      │ node1 │ Running │ 0        │ 12 hours ago│
│ system │ ingress-proxy                │ node1 │ Running │ 0        │ 12 hours ago│
│ system │ kafka                        │ node1 │ Running │ 0        │ 12 hours ago│
│ system │ loki                         │ node1 │ Running │ 0        │ 12 hours ago│
│ system │ metrics                      │ node1 │ Running │ 0        │ 12 hours ago│
│ system │ minio                        │ node1 │ Running │ 0        │ 12 hours ago│
│ system │ postgres                     │ node1 │ Running │ 0        │ 12 hours ago│
│ system │ promtail-cltmk               │ node1 │ Running │ 0        │ 12 hours ago│
│ system │ vip-add                      │ node1 │ Running │ 0        │ 12 hours ago│
└────────┴──────────────────────────────┴───────┴─────────┴──────────┴─────────────┘
```

Note

- All the services with owner *onc* must display the status as *Running.* After stack creation, it can take up to 20 minutes for all services to reach the *Running* state.

Step 8 SSH to the node and set the initial UI password for the admin user.

```
sedo security user set admin --password
```

Step 9 You can check the current version using the **sedo version** command.

```
sedo version
```

```
┌─────────────────────────────────────────────────────────────────────────────────────┐
│ Installer: CONC 24.3.1                                                              │
├───────────┬───────────────────────────────────────────────────────┬─────────────────┤
│ NODE NAME │ OS VERSION                                            │ KERNEL VERSION  │
├───────────┼───────────────────────────────────────────────────────┼─────────────────┤
│ node1     │ NxFOS 3.0-408 (f2beddad9abeb84896cc13efcd9a87c48ccb5d0c) │ 6.1.0-23-amd64 │
└───────────┴───────────────────────────────────────────────────────┴─────────────────┘

┌────────────────────────────────────────────────────────────────────┬──────────┬───┐
│ IMAGE NAME                                                         │ VERSION
│ NODES │
├──────────────────────────────────────────────────────────────────────┼───────────────────┼───────┤
│ docker.io/library/alpine                                                   │ 3.20.0
│ node1 │
│ docker.io/rancher/local-path-provisioner                                   │ v0.0.27
│ node1 │
│ quay.io/coreos/etcd                                                        │ v3.5.12
│ node1 │
│ registry.k8s.io/coredns/coredns                                            │ v1.11.1
│ node1 │
│ registry.k8s.io/kube-apiserver                                             │ v1.30.2
│ node1 │
│ registry.k8s.io/kube-controller-manager                                    │ v1.30.2
│ node1 │
│ registry.k8s.io/kube-proxy                                                 │ v1.30.2
│ node1 │
│ registry.k8s.io/kube-scheduler                                             │ v1.30.2
│ node1 │
│ registry.k8s.io/pause                                                      │ 3.9
│ node1 │
│ registry.nxf-system.svc:8443/cisco-onc-docker/dev/alarmservice             │ 24.3.1-5
│ node1 │
│ registry.nxf-system.svc:8443/cisco-onc-docker/dev/circuit-service          │ 24.3.1-5
│ node1 │
│ registry.nxf-system.svc:8443/cisco-onc-docker/dev/collector-service        │ 24.3.1-5
│ node1 │
│ registry.nxf-system.svc:8443/cisco-onc-docker/dev/config-service           │ 24.3.1-5
│ node1 │
│ registry.nxf-system.svc:8443/cisco-onc-docker/dev/devicemanager-service    │ 24.3.1-5
│ node1 │
│ registry.nxf-system.svc:8443/cisco-onc-docker/dev/inventory-service        │ 24.3.1-5
│ node1 │
│ registry.nxf-system.svc:8443/cisco-onc-docker/dev/monitoring               │ release2431_latest
│ node1 │
│ registry.nxf-system.svc:8443/cisco-onc-docker/dev/nbi-service              │ 24.3.1-5
│ node1 │
│ registry.nxf-system.svc:8443/cisco-onc-docker/dev/netconfcollector-service │ 24.3.1-5
│ node1 │
│ registry.nxf-system.svc:8443/cisco-onc-docker/dev/onc-apps-ui-service      │ 24.3.1-5
│ node1 │
│ registry.nxf-system.svc:8443/cisco-onc-docker/dev/osapi-gw-service         │ 24.3.1-5
│ node1 │
│ registry.nxf-system.svc:8443/cisco-onc-docker/dev/pce_service              │ 24.3.1-5
│ node1 │
│ registry.nxf-system.svc:8443/cisco-onc-docker/dev/pm-service               │ 24.3.1-5
│ node1 │
│ registry.nxf-system.svc:8443/cisco-onc-docker/dev/pmcollector-service      │ 24.3.1-5
│ node1 │
│ registry.nxf-system.svc:8443/cisco-onc-docker/dev/topology-service         │ 24.3.1-5
│ node1 │
│ registry.nxf-system.svc:8443/cisco-onc-docker/dev/torch                    │ 24.3.1-5
│ node1 │
│ registry.sedona.ciscolabs.com/nxf/authenticator                            │ 3.0-348
│ node1 │
│ registry.sedona.ciscolabs.com/nxf/bgp                                      │ 3.0-365
│ node1 │
│ registry.sedona.ciscolabs.com/nxf/controller                               │ 3.0-384
│ node1 │
│ registry.sedona.ciscolabs.com/nxf/firewalld                                │ 3.0-365
│ node1 │
│ registry.sedona.ciscolabs.com/nxf/flannel                                  │ 3.0-365
│ node1 │
│ registry.sedona.ciscolabs.com/nxf/ingress-proxy                            │ 3.0-370
│ node1 │
│ registry.sedona.ciscolabs.com/nxf/iptables                                 │ 3.0-370
│ node1 │
│ registry.sedona.ciscolabs.com/nxf/kafka                                    │ 3.0-365
│ node1 │
│ registry.sedona.ciscolabs.com/nxf/loki                                     │ 3.0-365
│ node1 │
│ registry.sedona.ciscolabs.com/nxf/metrics-exporter                         │ 3.0-365
│ node1 │
│ registry.sedona.ciscolabs.com/nxf/minio                                    │ 3.0-365
│ node1 │
│ registry.sedona.ciscolabs.com/nxf/service-proxy                            │ 3.0-370
│ node1 │
│ registry.sedona.ciscolabs.com/nxf/syslog-forwarder                         │ 3.0-340
│ node1 │ registry.sedona.ciscolabs.com/nxf/timescale                                │ 3.0-359
        │ node1 │
└─────────────────────────────────────────────────────────────────────┴──────────────────┴───────┘
```

Step 10 To check the default admin user ID, use the command **sedo security user list**.

Step 11 Use a web browser to access *https://<virtual ip>:8443/* to access the Cisco Optical Network Controller Web UI. Use the admin id and the password that you set to log in to Cisco Optical Network Controller.

Note Access the web UI only after all the `onc` services are running. Use the **sedo system status** command to verify that all services are running.