Installing OpenShift 4.6 on OpenStack 13

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 or of the new features up to this release.

| Cisco ACI CNI plug-in Release Version	Feature
5.1(3)	Cisco Application Centric Infrastructure (ACI) supports Red Hat OpenShift 4.6 nested in Red Hat OpenStack Platform (OSP).

OpenShift 4.6 on OpenStack

Cisco Application Centric Infrastructure (ACI) supports Red Hat OpenShift 4.6 nested in Red Hat OpenStack Platform (OSP) 13. To enable this support, Cisco ACI provides customized Ansible modules to complement the upstream OpenShift installer. This document provides instructions and guidance that follows the recommended OpenShift on OpenStack User-Provisioned Infrastructure (UPI) installation process as outlined in the following documents:

- *Installing a cluster on OpenStack with customizations* for OpenShift 4.6 on the Red Hat OpenShift website

- *Installing OpenShift on OpenStack User-Provisioned Infrastructure* on GitHub

Network Design and the Cisco ACI CNI Plug-in

This section provides information about the network design that takes advantage of the Cisco ACI Container Network Interface (CNI) plug-in.

The design separates OpenShift node traffic from the pod traffic on different Neutron networks. The separation results in the bootstrap, control, and compute virtual machines (VMs) having two network interfaces, as shown in the following illustration:
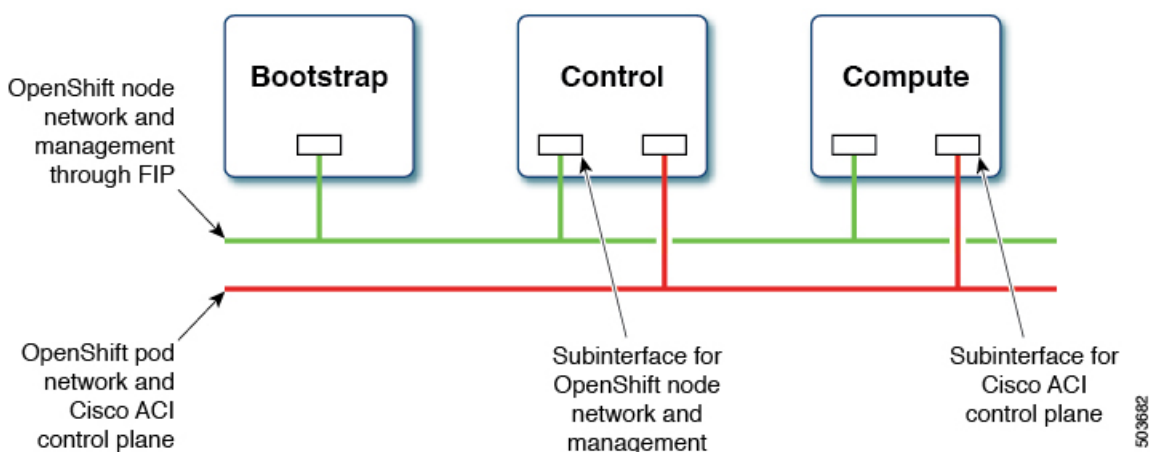
One interface is for the node network and the second is for the pod network. The second interface also carries Cisco ACI control plane traffic. A VLAN tagged subinterface is configured on the second interface to carry the pod traffic and the Cisco ACI control plane traffic.

This network design requires some changes to the Red Hat OpenShift Installer UPI Ansible modules. These changes are implemented in the Cisco-provided OpenShift Installer UPI Ansible modules, which are packaged in the OpenShift installer tar file (openshift_installer-5.1.3.<z>.src.tar.gz) that is made available along with the other Cisco ACI CNI 5.1(3) release artifacts. More specifically, the changes are to:

- Create a second Neutron network in a separate playbook.

- Modify the existing playbooks that launch the control, and compute virtual machines (VMs) to:

    - Create a second port on the second Neutron network and add it as a second interface to the VM configuration.

    - Add an extra attribute (nat_destination) to the Neutron floating IP address.

- Update the playbook that creates the first Neutron network to:

    1. Create the Neutron address-scope to map to a predefined Cisco ACI virtual routing and forwarding (VRF) context.

    2. Create a Neutron subnet-pool for the address-scope in the previous step.

    3. Change the subnet creation to pick a subnet from the subnet-pool in the previous step.

    4. Set the maximum transmission unit (MTU) for the neutron Network (which is picked up from the configuration file described later).

- In addition to creating a second network interface (and subinterfaces on that interface), the stock ignition files created by the "openshift-install create ignition-configs" step need to be updated. This is being done by additional playbooks, which are also provided.

Note The configuration required to drive some of the customization in this section done through new parameters in the inventory file.

Prerequisites for Installing OpenShift 4.6

To successfully install OpenShift Container Platform (OCP) 4.6 on OpenStack 13, you must meet the following requirements:

Cisco ACI

1. Ensure that the border leaf switch is a dedicated one and not connected to the compute nodes.

2. Configure a Cisco ACI Layer 3 outside connection (L3Out) in an independent Cisco ACI VRF and "common" Cisco ACI tenant so that endpoints can do the following:

    - Reach outside to fetch packages and images.

    - Reach the Cisco Application Policy Infrastructure Controller (APIC).

3. Configure a separate L3Out in an independent VRF that is used by the OpenShift cluster (configured in the acc-provision input file) so that the endpoints can do the following:

    - Reach API endpoints outside the OpenShift cluster.

    - Reach the OpenStack API server.

The OpenShift pod network uses this L3Out.

4. Identify the Cisco ACI infra VLAN.

5. Identify another unused VLAN that you can use for OpenShift cluster service traffic.

   The service is configured in the `service_vlan` field in the `acc_provision` input file for the OpenShift cluster.

OpenStack

1. Install Red Hat OpenStack Platform (OSP) 13 with Cisco ACI Neutron plug-in (release 5.1(3)) in nested mode by setting the following parameters in the Cisco ACI `.yaml` Modular Layer 2 (ML2) configuration file:

   • `ACIOpflexInterfaceType: ovs`

   • `ACIOpflexInterfaceMTU: 8000`

   Refer to *Cisco ACI Installation Guide for Red Hat OpenStack Using the OpenStack Platform 13 Director* on Cisco.com.

2. Create an OpenStack project and the required quotas to host the OpenShift cluster and perform other required configuration.

   Follow the procedure *Installing a cluster on OpenStack on your own infrastructure* for OpenStack 4.6 on the Red Hat OpenStack website.

3. Create an OpenStack Neutron external network, using the relevant Cisco ACI extensions and mapping to the OpenStack L3Out to include the following:

   • A subnet configured for Secure Network Address Translation (SNAT).

   • A subnet that is configured for floating IP addresses.

   Refer to the chapter "OpenStack External Network" in *Cisco ACI Installation Guide for Red Hat OpenStack Using the OpenStack Platform 13 Director* on Cisco.com.

   > ✎
   > **Note**   All OpenStack projects can share the OpenStack L3Out and Neutron external network.

4. If direct access to the OpenShift node network is required (i.e by not using the Neutron Floating IPs) from endpoints that are not managed by the Cisco ACI fabric, identify every IP subnet from where this direct access is anticipated. These IP subnets will later be used to create Neutron subnet pools during the installation process.

5. Follow the instructions in the section "Red Hat Enterprise Linux CoreOS (RHCOS)" of *Installing OpenShift on OpenStack User-Provisioned Infrastructure* to obtain the RHCOS and create an OpenStack image:

```
$ openstack image create --container-format=bare --disk-format=qcow2 --file
rhcos-4.6.1-x86_64-openstack.x86_64.qcow2 rhcos-4.6
```

OpenShift

Identify the SNAT IP address that will be used by the Cisco ACI Container Network Interface (CNI) for source NATing the traffic from all the pods during installation. You will use the SNAT IP addresses in the `cluster_snat_policy_ip` configuration in the `aci_cni` section of the `inventory.yaml` file.

Installer Host

You need access to a Linux host to run installation scripts with access to node network and OpenStack Director API. It should have the following installed:

- Install Ansible 2.8 or later.

  Refer to *Installing Ansible* on the Ansible website.

- Python 3

- jq – JSON linting

- yq – YAML linting: **sudo pip install yq**

- python-openstackclient 3.19 or later: **sudo pip install python-openstackclient==3.19.0**

- openstacksdk 0.17 or later: **sudo pip install openstacksdk==0.17.0**

- python-swiftclient 3.9.0: **sudo pip install python-swiftclient==3.9.0**

- Kubernetes module for Ansible: **sudo pip install --upgrade --user openshift**

This document uses the name `openupi` for the OpenShift cluster and the directory structure: `~/openupi/openshift-env/upi`.

```
$ cd ~/
$ mkdir -p openupi/openshift-env/upi
$ cd openupi/
$ tar xfz <path>/openshift_installer-5.1.3.<z>.src.tar.gz
$ cp openshift_installer/upi/openstack/* openshift-env/upi/
```

Installing OpenShift 4.6 on OpenStack 13

You initiate installation from the installer host that you prepared earlier.

Before you begin

Complete the tasks in the Prerequisites for Installing OpenShift 4.6 section .

Procedure

Step 1   Download and untar the `oc` client and `openshift-install` binary file:

```
$ cd ~/openupi/openshift-env/
$ wget https://mirror.openshift.com/pub/openshift-v4/clients/oc/4.6/linux/oc.tar.gz
$ tar xfz oc.tar.gz
$ mv oc /usr/local/bin/
$ wget
https://mirror.openshift.com/pub/openshift-v4/clients/ocp/latest-4.6/openshift-install-linux.tar.gz
$ tar xfz openshift-install-linux.tar.gz
```

Note   The links in the preceding text refer to the OpenShift 4.6 release, which Cisco has validated. However, subsequent minor releases are also expected to work.

Step 2   Install the `acc-provision` package present in the Cisco ACI Container Network Interface (CNI) 5.1(3) release artifacts.

Note   Due to Python 3 dependencies that are currently available only on RHEL8, acc-provision tool is supported to only run on RHEL8 operating system.

Step 3   Run the acc-provision tool to configure the Cisco APIC for the OpenShift cluster, which will also generate the manifests for installing the Cisco ACI CNI plug-in.

Example:

```
$ cd ~/openupi
$ acc-provision -a -c acc-provision-input.yaml -u <user> -p <password> -o aci_deployment.yaml -f
openshift-4.6-openstack
```

This step generates the `aci_deployment.yaml` file and also a `.tar.gz` file containing the Cisco ACI CNI manifests with the name `aci_deployment.yaml.tar.gz`. Note the location of the `aci_deployment.yaml.tar.gz` file; you must specify it later in the `install-config.yaml` file.

The following is an example of an acc-provision input file: (Note that the acc-provision flavor used here is `openshift-4.6-openstack`.)

```
#
# Configuration for ACI Fabric
#
aci_config:
  system_id: <cluster-name>              # Every opflex cluster on the same fabric must have a
distinct ID
  tenant:
    name: <openstack-tenant-name>
  apic_hosts:                            # List of APIC hosts to connect to for APIC API access
    - <apic-ip>
  apic_login:
    username: <username>
    password: <password>
  vmm_domain:                            # Kubernetes VMM domain configuration
    encap_type: vxlan                    # Encap mode: vxlan or vlan
    mcast_range:                         # Every vxlan VMM on the same fabric must use a distinct
range
        start: 225.125.1.1
        end: 225.125.255.255
  # The following resources must already exist on the APIC,
  # this is a reference to use them
  aep: sauto-fab3-aep       # The attachment profile for ports/VPCs connected to this cluster
  vrf:                      # VRF used to create all subnets used by this Kubernetes cluster
    name: l3out_2_vrf       # This should exist, the provisioning tool does not create it
    tenant: common          # This can be tenant for this cluster (system-id) or common
  l3out:                    # L3out to use for this kubernetes cluster (in the VRF above)
    name: l3out-2           # This is used to provision external service IPs/LB
    external_networks:
      - l3out_2_net         # This should also exist, the provisioning tool does not create it
#
# Networks used by Kubernetes
#
net_config:
  node_subnet: 10.11.0.1/27          # Subnet to use for nodes
  pod_subnet: 10.128.0.1/16          # Subnet to use for Kubernetes Pods
  extern_dynamic: 150.3.1.1/24       # Subnet to use for dynamically allocated ext svcs
  extern_static: 150.4.1.1/21        # Optional: Subnet for statically allocated external services
  node_svc_subnet: 10.5.168.1/21     # Subnet to use for service graph
  service_vlan: 1022                 # The VLAN used for external LoadBalancer services
  infra_vlan: 4093
  interface_mtu: 1400
```

Step 4   Run the `install/create/wait-for` commands from the `openshift-env` directory.

Ensure that the `clouds.yaml` file is either present in the current working directory or in `~/.config/openstack/clouds.yaml` with the environment OS_CLOUD set to the correct cloud name.

See *Configuration* for python-openstackclient3.12.3.dev2 on the OpenStack website.

Step 5 Untar the `aci_deployment.yaml.tar.gz` file that the acc-provision tool generated earlier.

```
$ cd ~/openupi
$ tar xfz aci_deployment.yaml.tar.gz
```

Step 6 Create the `install-config.yaml` as described in the "Install Config" section of *Installing OpenShift on OpenStack User-Provisioned Infrastructure* for release 4.6 on GitHub.

```
$ cd ~/openupi/openshift-env
$ ./openshift-install create install-config --dir=upi --log-level=debug
```

The following is an example of an `install-config.yaml` file that sets Cisco ACI Container Network Interface (CNI) as the `networkType`:

```
apiVersion: v1
baseDomain: noiro.local
compute:
- architecture: amd64
  hyperthreading: Enabled
  name: worker
  platform: {}
  replicas: 0
controlPlane:
  architecture: amd64
  hyperthreading: Enabled
  name: master
  platform: {}
  replicas: 3
metadata:
  creationTimestamp: null
  name: openupi
networking:
  clusterNetwork:
  - cidr: 15.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 15.11.0.0/27
  networkType: CiscoACI
  serviceNetwork:
  - 172.30.0.0/16
platform:
  openstack:
    cloud: openstack
    computeFlavor: aci_rhel_huge
    externalDNS: ["<ip>"]
    externalNetwork: sauto_l3out-2
lbFloatingIP: 60.60.60.199
    octaviaSupport: "0"
    region: ""
    trunkSupport: "1"
    clusterOSImage: rhcos-4.6
publish: External
proxy:
  httpsProxy: <proxy-ip>
  httpProxy: <proxy-ip>
  noProxy: "localhost,127.0.0.1,<add-more-as-relevant>,172.30.0.1,172.30.0.10,oauth-
      openshift.apps.openupi.noiro.local,console-openshift-
      console.apps.openupi.noiro.local,downloads-openshift-
      console.apps.openupi.noiro.local,downloads-openshift-
```

```
                console.apps.openupi.noiro.local,alertmanager-main-openshift-
                monitoring.apps.openupi.noiro.local"
pullSecret:
sshKey:
```

Step 7   Edit the file generated in the previous step to match your environment.

As noted in the example, the edits must include changing the `networkType` as described in the "Fix the Node Subnet" and "Empty Compute Pools" sections of *Installing OpenShift on OpenStack User-Provisioned Infrastructure* for Release 4.6 on GitHub.

Step 8   Edit the `inventory.yaml` file to match the relevant fields in the `install-config.yaml` and `acc-provision-input.yaml` files, as shown in the following example:

```
all:
  hosts:
    localhost:
      aci_cni:
        acc_provision_tar: <path>/aci_deployment.yaml.tar.gz
        kubeconfig: <path>/kubeconfig
      ansible_connection: local
      ansible_python_interpreter: "{{ansible_playbook_python}}"

      # User-provided values
      os_subnet_range: '15.11.0.0/27'
      os_flavor_master: 'aci_rhel_huge'
      os_flavor_worker: 'aci_rhel_huge'
      os_image_rhcos: 'rhcos-4.6.'
      os_external_network: 'l3out-2'
      # OpenShift API floating IP address
      os_api_fip: '60.60.60.6'
      # OpenShift Ingress floating IP address
      os_ingress_fip: '60.60.60.8'
      # Service subnet cidr
      svc_subnet_range: '172.30.0.0/16'
      os_svc_network_range: '172.30.0.0/15'
      # Subnet pool prefixes
      cluster_network_cidrs: '15.128.0.0/14'
      # Subnet pool prefix length
      host_prefix: B
      # Name of the SDN.
      os_networking_type: 'CiscoACI'

      # Number of provisioned Control Plane nodes
      # 3 is the minimum number for a fully-functional cluster.
      os_cp_nodes_number: 3
      # Number of provisioned Compute nodes.
      # 3 is the minimum number for a fully-functional cluster.
      os_compute_nodes_number: 3
```

Note   • The `inventory.yaml` file is updated after you run the `update_ign.py` script later in this procedure. We recommend that you make a copy of the `inventory.yaml` file at this stage so you can reuse it to install the same cluster again.

• The Cisco ACI CNI-specific configuration is added to the `aci_cni` section of the `inventory.yaml` file. The example in this step captures the required fields; however, more optional configurations are available. For a list of the options see the *Optional Configurations* section in this guide.

Note that after you run `update_ign.py` as described in Step 11, some default and derived values are added to the inventory file. For example, to see the configuration with all optional and derived values that are populated, see `openshift_installer/upi/openstack/inventory.yaml` on GitHub.

Step 9　　Generate the OpenShift manifests and copy the Cisco ACI CNI manifests:

> **Note**　　Remove the control-plane Machines, as described in the "Machines and MachineSets" section of *Installing OpenShift on OpenStack User-Provisioned Infrastructure* for Release 4.6 on GitHub.

```
$ cd ~/openupi/openshift-env
$ ./openshift-install create manifests  --log-level debug --dir=upi
# Copy the ACI CNI manifests obtained earlier in Step 5
$ cp ../cluster-network-* upi/manifests/
$ rm -f upi/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

Step 10　　Update the Machineset for the compute notes.

```
$ cd ~/openupi/openshift-env/upi
$ python update_machineset.py
Running the above script updates the generated MachineSet for the compute nodes:
99_openshift-cluster-api_worker-machineset-0.yaml
```

Step 11　　Make control-plane nodes unschedulable.

Follow the instructions in the "Make control-plane nodes unschedulable" section of *Installing OpenShift on OpenStack User-Provisioned Infrastructure* for Release 4.6 on GitHub.

Step 12　　Update the ignition files:

```
$ cd ~/openupi/openshift-env
$ ./openshift-install create ignition-configs --log-level debug --dir=upi
$ cd upi
$ export INFRA_ID=$(jq -r .infraID metadata.json)
# Run the update_ign.py from the Cisco OpenShift installer package
$ sudo -E python update_ign.py # This assumes that the inventory file is already configured
$ source ~/openupi/overcloudrc

$ swift upload bootstrap bootstrap.ign
(To be executed in undercloud after copying the ignition file or host having connectivity to openstack
 controller with overcloudrc)

$ swift post bootstrap --read-acl ".r:*,.rlistings"

(To be executed in undercloud after copying the ignition file host having connectivity to openstack
 controller with overcloudrc)
```

The commands in this step create the ignition files and update them according to Cisco ACI CNI and upload the `bootstrap.ign` file to swift storage. It also generates the `bootstrap-ignition-shim` as described in the "Bootstrap Ignition Shim" section of *Installing OpenShift on OpenStack User-Provisioned Infrastructure* for Release 4.6 on GitHub.

Step 13　　Complete the following tasks by running Ansible playbooks obtained from the Cisco OpenShift installer package:

a) Create security groups and networks:

```
ansible-playbook -i inventory.yaml security-groups.yaml
ansible-playbook -i inventory.yaml network.yaml
ansible-playbook -i inventory.yaml 021_network.yaml
```

b) For direct access to the OpenShift node network from endpoints that are not managed by the Cisco ACI fabric, create a Neutron subnet pool for every IP subnet from where this direct access is anticipated, as shown in the following example:

```
$ neutron subnetpool-create --pool-prefix <direct_access_src_subnet> --address-scope
node_network_address_scope <subnetpool_name>
```

In the preceding example, `node_network_address_scope` is the name of the Neutron address-scope that is created by the `network.yaml` file.

   c) Install the control plane:

```
ansible-playbook -i inventory.yaml bootstrap.yaml
ansible-playbook -i inventory.yaml control-plane.yaml
```

   d) Check that the bootstrap/control plane installation is complete:

```
./openshift-install wait-for bootstrap-complete --dir=upi --log-level=debug
```

   e) After the control plane is installed, remove the bootstrap node:

```
ansible-playbook -i inventory.yaml down-bootstrap.yaml
```

   f) (Optional) After the control plane is up, configure cluster Source IP Network Address Translation (SNAT) policy:

```
ansible-playbook -i inventory.yaml cluster_snat_policy.yaml
```

   g) Launch the compute nodes using one of the following methods:

      • Through an Ansible playbook:

```
ansible-playbook -i inventory.yaml compute-nodes.yaml
```

      • The worker machinesets can be scaled as described in the Optional MachineSet and MachineConfigPool Configurations section of this document, by using the `99_openshift-cluster-api_worker-machineset-0.yaml` updated in Step 10.

Step 14   If you created the compute nodes through Ansible playbooks, approve the pending Certificate Signing Requests.

```
oc get csr -ojson | jq -r '.items[] | select(.status == {} ) | .metadata.name' | xargs oc adm
certificate approve
```

Step 15   Update the default IngressController publish strategy to use the LoadBalancerService:

```
ansible-playbook -i inventory.yaml post-install.yaml
```

Step 16   Check the status of the installation:

```
./openshift-install wait-for install-complete --dir=upi --log-level=debug
```

Step 17   Destroy the cluster:

```
ansible-playbook -i inventory.yaml down-compute-nodes.yaml
ansible-playbook -i inventory.yaml down-control-plane.yaml
ansible-playbook -i inventory.yaml down-network.yaml
ansible-playbook -i inventory.yaml down-security-groups.yaml
```

After your run the playbooks in this step, the Cisco ACI BridgeDomain corresponding to the node network will also be deleted. To reinstall the cluster, run `acc-provision` again with the `-a` as described earlier in this document.

Optional Configurations

This section provides instructions for making several optional configurations.

Optional Inventory Configurations

You add Cisco ACI Container Network Interface (CNI) configuration to the `aci_cni` section of the `inventory.yaml` file. The Installing OpenShift 4.6 on OpenStack 13 section provides the required fields. This section provides optional configurations and the default values.

Option			Description and Default Values
`cluster_snat_policy_ip`			By default, this value is not set.
			The Source IP Network Address Translation (SNAT) IP address is used to create a Cisco ACI-CNI SNAT policy that applies to the whole cluster. This SNAT policy is created by running the `cluster_snat_policy.yaml` Ansible playbook as described in Installing OpenShift 4.6 on OpenStack 13. (If this value is not set, do not run this playbook.)
`dns_ip`			By default, this value is not set.
			Set this field if you do not follow the procedure that is described in the section "Subnet DNS (optional)" in *Installing OpenShift on OpenStack User-Provisioned Infrastructure* on GitHub. The procedure controls the default resolvers that your Nova servers use.
			Use the value to set the `dns_nameservers` field of the subnet associated with the `*-primaryClusterNetwork` network. You can specify one or more DNS server IPs.
`network_interfaces`	`node`	`name`	The name of the node network interface as set by the RHCOS image.
			The default value is "`ens3`".
		`mtu`	The MTU set for the `*-primaryClusterNetwork` Neutron network.
			The default value is 1500.
	`opflex`	`name`	The name of the node network interface as set by the RHCOS image.
			The fault value is "`ens4`".
		`mtu`	The MTU set for the `*-secondaryClusterAciNetwork` Neutron network.
			The default value is 1500.
		`subnet`	The default value is `192.168.208.0/20`.
			This is the CIDR used for the subnet that is associated with the `*-secondaryClusterAciNetwork` Neutron network. The size of this subnet should at least be as large as that of the one used for the `*-primaryClusterNetwork` Neutron network. It should also not overlap any other CIDR in the OpenShift project's address scope.

Optional MachineSet and MachineConfigPool Configurations

Scale the Existing Worker MachineSet

Scale the replicas as shown in the following example:

```
$ oc get machineset -A
NAMESPACE               NAME                     DESIRED   CURRENT   READY   AVAILABLE   AGE
openshift-machine-api   openupi-vkkn6-worker     0         0                             5h10m
$ oc scale machineset -n openshift-machine-api   openupi-vkkn6-worker --replicas=1
```

Create a New MachineSet with Two Networks and a MachineConfigPool

The following example is for MachineConfigPool:

```
$ cat machineconfigpool.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfigPool
metadata:
  name: logging
  labels:
    machine.openshift.io/cluster-api-cluster: openupi-8zq9j
    machine.openshift.io/cluster-api-machine-role: logging
    machine.openshift.io/cluster-api-machine-type: logging
spec:
  machineConfigSelector:
    matchExpressions:
      - {key: machineconfiguration.openshift.io/role, operator: In, values: [worker,logging]}
  maxUnavailable: 0
  nodeSelector:
    matchLabels:
      node-role.kubernetes.io/logging: ""
  paused: false
```

The following example is for MachineSet:

```
$ cat logging_machineset.yaml
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
machine.openshift.io/cluster-api-cluster: openupi-8zq9j
    machine.openshift.io/cluster-api-machine-role: logging
    machine.openshift.io/cluster-api-machine-type: logging
  name: openupi-8zq9j-logging
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: openupi-8zq9j
      machine.openshift.io/cluster-api-machineset: openupi-8zq9j-logging
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: openupi-8zq9j
        machine.openshift.io/cluster-api-machine-role: logging
        machine.openshift.io/cluster-api-machine-type: logging
        machine.openshift.io/cluster-api-machineset: openupi-8zq9j-logging
    spec:
      metadata:
        labels:
          node-role.kubernetes.io/logging: ""
```

```
providerSpec:
  value:
    apiVersion: openstackproviderconfig.openshift.io/v1alpha1
    cloudName: openstack
    cloudsSecret:
      name: openstack-cloud-credentials
      namespace: openshift-machine-api
    flavor: aci_rhel_huge
    image: rhcos-4.6
    kind: OpenstackProviderSpec
    networks:
    - filter: {}
      subnets:
      - filter:
          name: openupi-8zq9j-nodes
          tags: openshiftClusterID=openupi-8zq9j
    - filter: {}
      subnets:
      - filter:
          name: openupi-8zq9j-acicontainers-nodes
          tags: openshiftClusterID=openupi-8zq9j
    securityGroups:
    - filter: {}
      name: openupi-8zq9j-worker
    serverMetadata:
      Name: openupi-8zq9j-logging
      openshiftClusterID: openupi-8zq9j
    trunk: true
    tags:
    - openshiftClusterID=openupi-8zq9j
```

Upgrading from OpenShift 4.5 to OpenShift 4.6

Use this procedure for an in-cluster upgrade from OpenShift 4.5 to OpenShift 4.6.

Before you begin

Ensure the cluster version is 4.5.z and is in a healthy state. Use the `$ oc get clusterversion` command.

Procedure

Step 1   Get the latest version of `openshift_installer-5.1.1.<z>.src.tar.gz` installer package.

Step 2   Update the 02-worker-network machine-config using the following commands:

```
$ cd openshift-env/upi
$ ls *.ign
bootstrap.ign  master.ign  worker.ign
# Take backup of these 3 files
# Run updated update_ign.py again
$ python update_ign.py
# Open one of the update <ifra-id>-worker-<id>-ignition.json
$ cat openupi-4d74k-worker-0-ignition.json

# Extract the base64 data from above file for ifcfg-ens3, ifcfg-ens4, ifcfg-opflex-conn,
route-opflex-conn  and replace it with machine-config 02-worker-network
$ oc edit mc 02-worker-network
```

Step 3   After updating 02-worker-network MachineConfig, wait for the MachineConfigPool to come to the updated state.

Monitor the MachineConfigPool (mcp) for the worker using the `oc get mcp worker` command.

Step 4   Delete the MachineConfig using the `oc delete mc 02-worker-mco-check-disable` command, and wait for the MachineConfigPool to come to the updated state.

Step 5   You can now upgrade the cluster via the OpenShift Web Console or CLI commands.