



## Installing OpenShift 4.6 on VMware vSphere

<b>New and Changed Information</b>	<b>2</b>
Openshift 4.6 on VMware vSphere	2
Prerequisites for Installing OpenShift 4.6 on VMware vSphere	3
Installing OpenShift 4.6 on VMware vSphere	3
Updating the Default Ingress Controller	7
Configuring MachineSets with ACI CNI	7
Sample Files for Installing OpenShift 4.6 on VMware vSphere	11
Decommissioning OpenShift	15

Revised: August 11, 2023

## 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
5.2(1)	Support for Red Hat OpenShift 4.6 on VMware vSphere 7 User-Provisioned Infrastructure (UPI).
5.2(3)	Added the Decommissioning OpenShift section.

## Openshift 4.6 on VMware vSphere

Cisco ACI supports Red Hat OpenShift 4.6 on VMware vSphere 7 User-Provisioned Infrastructure (UPI). This document provides the instructions on using Ansible playbooks to provision OpenShift 4.6 on VMware vSphere with the Container Network Interface (CNI) plug-in.

The Ansible playbooks provision virtual machines (VMs) with the needed interface configuration and generate the ignition configuration files. You must deploy your own DHCP, DNS, and load-balancing infrastructure following high-availability best practices.

The Ansible playbooks are available on [Github](#).

The following are the Ansible playbooks:

- `asserts.yml`: This playbook performs basic validations of variable declarations in the `all.yml` file.
- `setup.yml`: This playbook performs the following tasks:
  - Configures the orchestrator node:
    - Installs Terraform, the OpenShift client, and the OpenShift installer. It creates the following: Terraform variables for the bootstrap, master, and worker nodes; the master and worker machine-config operator; the OpenShift install config file.
    - Configures load balancer node: It disables Security-Enhanced Linux (SELinux), configures HAProxy, sets up DHCP and DNS if selected.

This optional step configures these three components only if you set the `provision_dhcp`, `provision_dns`, and `provision_lb` variables to `true`.

- `oshift_prep.yml`:
  - Sets up the install and bootstrap directories.
  - Generates manifests using `openshift-install`.
  - Adds the additional machine-config operator manifests.
  - Adds the Cisco ACI-CNI manifests.
  - Creates a backup of the manifests.

- Sets up the bootstrap, master, and worker nodes ignition files.
- Copies the bootstrap ignition file to the loadbalancer node.
- `create_nodes.yml`:
  - Provisions the bootstrap, master, and worker nodes, using Terraform.
  - Sets up a cron job to approve Cisco Certificate Signing Requests (CSRs), if selected.
- `delete_nodes.yml`: Deletes all master and worker nodes.

## Prerequisites for Installing OpenShift 4.6 on VMware vSphere

To install OpenShift Container Platform (OCP) 4.6 on VMware vSphere, fulfill the following prerequisites:

### Cisco ACI

- Download the acc-provision tool version 5.2.3.3 or later.  
Specify the “--flavor” option value as “openshift-4.6-esx,” and use the “-z” option. The tool creates a .tar archive file as specified by the “-z” option value. You need this archive file during installation.  
Make sure that the Cisco ACI container images that are specified as input to the acc-provision tool are version 5.2.3.3 or later.

### VMware vSphere

Obtain user credentials with privileges to create virtual machines (VMs).

### OpenShift

Obtain the following from the Red Hat website:

- The OCP4 Open Virtualization Appliance (OVA)- ensure to download the relevant release of the OVA image. Navigate to the *mirror* page on the OpenShift website where all the RHCOS versions are listed, and select the required version. Download the `rhcos-vmware.x86_64.ova` file.
- OCP4 client tools - navigate to the *mirror* page on the OpenShift website where the installation and client tool versions are listed, and select the required version. Download the `openshift-client-linux.tar.gz` and `openshift-install-linux.tar.gz` files.
- Pull Secret

## Installing OpenShift 4.6 on VMware vSphere




---

**Note** See [Sample Ansible group\\_vars/all.yml File](#) section.

---

### Before you begin

Complete the tasks in the [Prerequisites for Installing OpenShift 4.6 on VMware vSphere](#) section.

It is recommended to see the *RedHat OpenShift documentation* for prerequisites and other details about Installing a Cluster on vSphere.

## Procedure

---

**Step 1** Provision the Cisco ACI fabric using the acc-provision utility:

```
acc-provision -a -c acc_provision_input.yaml -u admin -p ### -f openshift-4.6-esx -z manifests.tar.gz
```

**Note** See [Sample acc-provision-input File](#) section.

**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 2** After the Cisco ACI fabric is provisioned, verify that a port group with the name *system\_id\_vlan\_kubeapi\_vlan* is created under distributed switch.

This document refers to this port group as *api-vlan-portgroup*.

**Note** *api-vlan-progroup* port-group in VMware Distributed Virtual Switch is created using custom VLAN ID provided in the *acc\_provision\_input* file as *kubeapi\_vlan*.

Figure 1: VMM VMware domain association with aci-containers-node EPG

## Edit VMM Domain Association - VMware/hypflex-vswitch

Delimiter:

Enhanced Lag Policy:

Allow Micro-Segmentation:

Untagged VLAN Access:

VLAN Mode:  Dynamic  Static

Primary VLAN:   
For example, vlan-1

Port Encap:   
For example, vlan-1

Port Binding:  Dynamic Binding  Ephemeral  Default  Static Binding

Netflow:  Disable  Enable

Allow Promiscuous:

Forged Transmits:

MAC Changes:

Active Uplinks Order:   
Enter IDs of uplinks separated by comma

Standby Uplinks:   
Enter IDs of uplinks separated by comma

Custom EPG Name:

Kube\_api VLAN is added to the dynamic VLAN pool associated with the VMware VMM Domain. Allocation mode will be set to Static.

Figure 2: VLAN Pool used for VMM VMware domain

Allocation Mode: Dynamic Allocation

Encap Blocks:

VLAN Range	Description	Allocation Mode	Role
[20-25]		Inherit allocMode from parent	External or On the wire encapsulations
[35]		Static Allocation	External or On the wire encapsulations

**Step 3** In VMware vSphere, import the OpenShift Container Platform 4 (OCP4) Open Virtual Appliance (OVA) image. Specify `api-vlan-portgroup` as the port group for the network interface.

**Step 4** Provision a Red Hat Enterprise load balancer virtual machine (VM) with the network interface that is connected to the `api-vlan-portgroup`.

The Ansible playbooks optionally configure this VM as a load balancer, DNS server, and DHCP server for the OpenShift cluster.

**Step 5** Provision a Red Hat Enterprise orchestrator VM with the network interface that is connected to the `api-vlan-portgroup`.

The Ansible playbooks play from the orchestrator VM.

**Step 6** Perform the following tasks on the orchestrator VM:

- a) Clone the Git repository on GitHub: [https://github.com/noironetworks/openshift\\_vsphere\\_upi](https://github.com/noironetworks/openshift_vsphere_upi).
- b) Checkout the "ocp46" branch.
- c) Generate Secure Shell (SSH) keys and copy them to the load balancer VM.
- d) Enable the `ansible-2.9-for-rhel-8-x86_64-rpms` repository.
- e) Update the Ansible package to the latest version.
- f) Change the directory to the Git-cloned directory.
- g) Install the Ansible module requirements:

```
ansible-galaxy install -r requirements.yaml
```

- h) Edit the `groups/all.yml` and `hosts.ini` files and set the values that your site requires.

**Note** See [Sample hosts.ini file](#) section. See the *Sample Ansible File* section.

- i) Perform basic validation of variable values using the `asserts.yml` playbook:

```
ansible-playbook asserts.yml
```

- j) Copy the archive file that was created by the `acc-provision` utility to the `files` directory, giving it the name `aci_manifests.tar.gz`.

- k) Run the Ansible `setup` playbook:

```
ansible-playbook setup.yml
```

Running the `setup` playbook configures the orchestrator and load balancer VMs.

- l) Run the Ansible `oshift_prep` playbook:

```
ansible-playbook oshift_prep.yml
```

Running the `oshift_prep` generates the OpenShift manifests and ignition files.

- m) Run the Ansible `create_nodes` playbook:

```
ansible-playbook create_nodes.yml
```

The `create_nodes` playbook creates the VMs. After the VMs are created, the OCP4 installation process starts in the background. At this stage, you should be able to access the cluster APIs by using the `kubeconfig` files created by the installer.

The `kubeconfig` files are located at `base_dir/bootstrap/auth` directory. The `base_dir` is set in `group_vars/all.yml` file, and the default value is `/root/ocpinstall`.

---

## What to do next

You can use the commands `openshift-install wait-for bootstrap-complete` and `openshift-install wait-for install-complete` to check the progress of the installation. Execute the commands from the `bootstrap` directory.

## Updating the Default Ingress Controller

For updating the default Ingress Controller publish strategy to use the ACI Loadbalancer, log in as a user with cluster-admin privileges and run the following:

```
oc replace --force --wait --filename - <<EOF
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  namespace: openshift-ingress-operator
  name: default
spec:
  endpointPublishingStrategy:
    type: LoadBalancerService
    loadBalancer:
      scope: External
EOF
```

For more details, see the *Configuring the default Ingress Controller for your cluster to be internal* section in the *Ingress Operator in OpenShift Container Platform* Red Hat guide.

## Configuring MachineSets with ACI CNI

The Machine API is a combination of primary resources that are based on the upstream Cluster API project and custom OpenShift Container Platform resources.

For OpenShift Container Platform 4.6 clusters, the Machine API performs all node host provisioning management actions after the cluster installation is completed. OpenShift Container Platform 4.6 offers an elastic, dynamic provisioning method on top of public or private cloud infrastructure because of the Machine API.

The two primary resources are:

- **Machines**-A fundamental unit that describes the host for a node. A machine has a `providerSpec` specification, which describes the types of compute nodes that are offered for different cloud platforms. For example, a machine type for a worker node on Amazon Web Services (AWS) might define a specific machine type and required metadata.
- **MachineSet**-MachineSet resources are groups of machines. Machine sets are to machines as replica sets are to pods. If you need more machines or must scale them down, you change the `replicas` field on the machineset to meet your compute need.

Changes to operating systems on OpenShift Container Platform nodes can be done by creating `MachineConfig` objects that are managed by the Machine Config Operator.

## Creating a MachineConfiguration File

Use this procedure to create a `MachineConfig` file that will configure network interfaces for the new nodes with:

- ACI Infra interface (ens224)
- ACI Infra SubInterface (ens224. {InfraVlanID})
- Opflex-route for BUM traffic replication (224.0.0.0/4)

The required configurations (sample) are shown in the procedure below, however, they need to be customized to match your specific environment. In general the following changes are required:

- Replace every occurrence of {InfraVLAN} with the ACI Infra VLAN for your fabric.
- Replace every occurrence of {MTU} with the MTU you selected for your cluster.



---

**Note** Ensure that your network interface is *ens224* (not a different name).

---

## Procedure

---

### Step 1 Create an 80-opflex-route.

```
#!/bin/bash
if [ "$1" == "ens224.{InfraVLAN}" ] && [ "$2" == "up" ]; then
route add -net 224.0.0.0 netmask 240.0.0.0 dev ens224.{InfraVLAN}
fi
```

### Step 2 Create an **ens224.nmconnection**.

```
[connection]
id=ens224
type=ethernet
interface-name=ens224
multi-connect=1
permissions=

[ethernet]
mac-address-blacklist=
mtu={MTU}

[ipv4]
dns-search=
method=disabled

[ipv6]
addr-gen-mode=eui64
dns-search=
method=disabled

[proxy]
```

### Step 3 Create an **ens224.{InfraVLAN}.nmconnection**.

```
[connection]
id=ens224.{InfraVLAN}
type=vlan
interface-name=ens224.{InfraVLAN}
multi-connect=1
permissions=

[ethernet]
mac-address-blacklist=

[vlan]
egress-priority-map=
flags=1
```



```
id={InfraVLAN}
ingress-priority-map=
parent=ens224
```

```
[ipv4]
dns-search=
method=auto
```

```
[ipv6]
addr-gen-mode=eui64
dns-search=
method=auto
```

```
[proxy]
```

**Step 4** Convert the above three configurations (steps 1,2,3) into *base64* encoded strings, and use it in the machineconfig (after *base64*) template, as shown below.

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 00-worker-cni
spec:
  config:
    ignition:
      version: 3.1.0
    storage:
      files:
        - contents:
            source: data:text/plain;charset=utf-8;base64,
              IyEvYmluL2Jhc2gKIgImlIFsgIiQxIiA9PSAiZW5zMjI0LjM0NTYiIF0gJiYgWyAi
              JDIIID09ICJlcCIgXTsgdGh1bGogcm91dGUgYWRkIC1uZXQgMjI0LjAuMC4wIG5ld
              G1hc2sgMjQwLjAuMC4wIGRldiBlbnMyMjQuMzQ1NgogZmk=
            mode: 0755
            overwrite: true
            path: /etc/NetworkManager/dispatcher.d/80-opflex-route
        - contents:
            source: data:text/plain;charset=utf-8;base64,
              W2Nvbm5lY3Rpb25dCmlkPWVuczIyNAp0eXB1PWV0aGVybmV0CmludGVyZmFjZS1uYW1lPWVuczIy
              NAptdWx0aS1jb25uZW50PTEKcGVybWlzc2l2bnM9CgpbZXRoZXJ1ZXJlcm1hYy1hZGRyZXNzLWJs
              YWNrbGlzdD0KbXR1PTkwMDAKCltpcHY0XQpkbnMtc2VhcmNoPQptZXRob2Q9ZGlzYWJsZWQKCltpc
              HY2XQphZGRyLWdlbi1tb2RlPWV1aTY0CmRucy1zZWYyZg9CmlldGhvZD1kaXNhYm1lZAoKW3Byb3h5XQ==
            mode: 0600
            overwrite: true
            path: /etc/NetworkManager/system-connections/ens224.nmconnection
        - contents:
            source: data:text/plain;charset=utf-8;base64,
              W2Nvbm5lY3Rpb25dCmlkPWVuczIyNC4zNDU2CnR5cGU9dmxhbGppbnRlcmZhY2UtbnFtZT1lbnMyMjQuMz
              Q1NgptdWx0aS1jb25uZW50PTEKcGVybWlzc2l2bnM9CgpbZXRoZXJ1ZXJlcm1hYy1hZGRyZXNzLWJsYWNrb
              GlzdD0KClt2bGFuXQplZ3Jlc3MtcHJpb3JpdHktbWFwPQpmcGFncz0xCmlkPTM0NTYKaW5ncmVzcy1wcm1v
              cm10eS1tYXA9CnBhcmVudD1lbnMyMjQKCltpcHY0XQpkbnMtc2VhcmNoPQptZXRob2Q9YXV0bwoKW21wdjZd
              CmFkZHItd2VuLW1vZGU9ZXVpNjQKZG5zLXNlYXJjaD0KbWV0aG9kPWF1dG8KCltwcm94eV0=
            mode: 0600
            overwrite: true
            path: /etc/NetworkManager/system-connections/ens224.{InfraVLAN}.nmconnection
```

**Note** Replace the {InfraVLAN} with your ACI InfraVLAN ID in path:  
/etc/NetworkManager/system-connections/ens224.{InfraVLAN}.nmconnection (last line in the sample above). You may also customize the name of the MachineConfig. In the above example, the name is 00-worker-cni.

**Step 5** Use the **oc create -f** command to create the MachineConfig for your cluster.

The machineconfig is applied to the pre-existing workers nodes (two) and replaces the three files already existing with identical copies. When you include the md5 configuration in the machine config, three files are created on the node. The pre-existing worker nodes will reboot one at a time.

**Step 6** Get you cluster id using the **oc get -o jsonpath='{.status.infrastructureName}{"\n"}' infrastructure cluster** command.

**Step 7** Create the machineset.

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  name: ocp4aci-jlff9-worker
  namespace: openshift-machine-api
  labels:
    machine.openshift.io/cluster-api-cluster: ocp4aci-jlff9
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: ocp4aci-jlff9
      machine.openshift.io/cluster-api-machineset: ocp4aci-jlff9-worker
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: ocp4aci-jlff9
        machine.openshift.io/cluster-api-machine-role: worker
        machine.openshift.io/cluster-api-machine-type: worker
        machine.openshift.io/cluster-api-machineset: ocp4aci-jlff9-worker
    spec:
      metadata: {}
      providerSpec:
        value:
          numCoresPerSocket: 1
          diskGiB: 120
          snapshot: ''
          userDataSecret:
            name: worker-user-data
          memoryMiB: 8192
          credentialsSecret:
            name: vsphere-cloud-credentials
          network:
            devices:
              - networkName: ocp4aci_vlan_35
              - networkName: ocp4aci
      metadata:
        creationTimestamp: null
      numCPUs: 2
      kind: VSphereMachineProviderSpec
      workspace:
        datacenter: my-dc
        datastore: mydatastore
        folder: /mydatastore/vm/ocp4aci
        server: myvsphere.local.lab
      template: RHCOS46
      apiVersion: vsphereprovider.openshift.io/v1beta1
```

The above is a sample configuration, you may need to modify as below:

- Cluster ID is `ocp4aci-j1ff9`. Replace it with the relevant cluster ID.
- Replace `ocp4aci_vlan_35` and `ocp4aci` with your port-groups names (the order is important, ensure not to swap it).
- Set the `replicas` value to how many new workers you want on top of the one that were created during cluster bring up.
- Edit all the `workspace` parameters (such as, `datacenter`, `datastore`), to match your vSphere settings.
- Edit the VM specifications, if required (memoryMiB, numCPUs and diskGiB).

**Note** The original workers are not managed by the MachineSet and you can choose to delete them. If you delete a worker, wait for the OCP routers to be started on the new node.

---

## Scaling of Compute Nodes Using MachineSet

Use this procedure to scale compute nodes using MachineSet.

### Procedure

---

**Step 1** Create a VM folder - `/DC name/vm/cluster name` in vCenter; the `cluster name` is the OpenShift cluster ID. In the example, the folder has been named: `/mydatastore/vm/ocp4aci`.

**Step 2** Create a Template in vCenter.

Ensure that the `template name` matches the `cluster name` in the RHCOS46 template.

**Step 3** Create a TAG category called `id` in vCenter.

**Step 4** Configure the DHCP server to allocate IP addresses to the nodes.

---

## Sample Files for Installing OpenShift 4.6 on VMware vSphere

This section contains sample files that you need for installing OpenShift 4.6 on VMware vSphere.

### Sample `acc-provision-input` File

The following is a sample `acc-provision-input.yaml`. The highlighted or bold values are those that you must modify to meet your site requirements.

```
#
# Configuration for ACI Fabric
#
aci_config:
  system_id: ocp4aci
  #apic-refresh-time: 1200
  apic_hosts:
    - 1.1.1.1
```

```

vmm_domain:
  encap_type: vxlan
  mcast_range:          # Every opflex VMM must use a distinct range
    start: 225.28.1.1
    end: 225.28.255.255
  nested_inside:
    type: vmware
    name: my-vswitch
elag_name: <eLAG_name> # Beginning Cisco APIC 5.0(1), you can configure VMware teaming policy
                    # when link aggregation groups (LAGs) are used.
  installer_provisioned_lb_ip: 10.213.0.201

# The following resources must already exist on the APIC.
# They are used, but not created, by the provisioning tool.
aep: my-aep
vrf:          # This VRF used to create all kubernetes EPs
  name: myl3out_vrf
  tenant: common
l3out:
  name: myl3out
  external_networks:
    - myl3out_net

#
# Networks used by ACI containers
#
net_config:
  node_subnet: 192.168.18.1/24
  pod_subnet: 10.128.0.1/16 # Subnet to use for Kubernetes
                          # Pods/CloudFoundry containers

  extern_dynamic: 10.3.0.1/24 # Subnet to use for dynamic external IPs
  extern_static: 10.4.0.1/24 # Subnet to use for static external IPs
  node_svc_subnet: 10.5.0.1/24 # Subnet to use for service graph
  kubeapi_vlan: 35
  service_vlan: 36
  infra_vlan: 3901
  #interface_mtu: 1600
  #service_monitor_interval: 5 # IPSLA interval probe time for PBR tracking
                              # default is 0, set to > 0 to enable, max: 65535
  #pbr_tracking_non_snat: true # Default is false, set to true for IPSLA to
                              # be effective with non-snat services

#
# Configuration for container registry
# Update if a custom container registry has been setup
#
kube-config:
  image_pull_policy: Always
  ovs_memory_limit: 1Gi

registry:
  image_prefix: quay.io/noiro

```

## Sample Ansible `group_vars/all.yml` File

The following is a sample `group_vars/all.yml`. The highlighted or bold values are those that you must modify to meet your site requirements.

```

#domainname
# type: string, base dns domain name, cluster metadata name is added as subdomain to this

```

```

# required: yes
domainname: ocplab.local

#provision_dns
# type: boolean, True or False
# required: yes
# notes: If set to true, load balancer is configured as dns server.
#       If false, it is assumed that the dns server pre-exists.
provision_dns: True

#dns_forwarder:
# type: ip address
# required: yes
# notes: This value is used when setting up a dhcp service and also for 'forwarders' value in dns configuration.
dns_forwarder: 172.28.184.18

#loadbalancer_ip:
# type: ip address or resolvable hostname
# required: yes
# notes: This host is configured as load balancer for cluster and also as dhcp and dns server if required .
#       This IP address is the same as the one that you configure in installer_provisioned_lb_ip in the acc-provision
#       config.
loadbalancer_ip: 192.168.18.201. This IP address is the same as the one that you configure in
installer_provisioned_lb_ip in the acc-provision config.

#auto_approve_csr:
# type: boolean
# required: yes
# notes: when set to true, sets up a cron job to auto approve openshift csr
auto_approve_csr: True

#proxy_env
#
proxy_env:
#donot remove dummy field, irrespective of whether setup needs a proxy or not.
dummy: dummy
#set the http/https proxy server, if setup does not need proxy, comment the below values.
#these values are used for ansible tasks and also passed on to openshift installer
http_proxy: http://1.1.1.1:80
https_proxy: http://1.1.1.1:80
no_proxy: 1.2.1.1,1.2.1.2

#packages
# defines the urls to download terraform, openshift client and openshift-install tools from.
packages:
  validate_certs: False
  terraform_url: https://releases.hashicorp.com/terraform/0.13.6/terraform_0.13.6_linux_amd64.zip
  openshift_client_linux_url:
https://mirror.openshift.com/pub/openshift-v4/clients/ocp/4.6.21/openshift-client-linux-4.6.21.tar.gz
  openshift_install_linux_url:
https://mirror.openshift.com/pub/openshift-v4/clients/ocp/4.6.21/openshift-install-linux-4.6.21.tar.gz

#default_aci_manifests_archive:
# default filename that is searched under files directory.
# this can be overridden by passing extra parameter aci_manifests_archive on ansible command line
default_aci_manifests_archive: aci_manifests.tar.gz

#opflex_interface_mtu:
# required: yes
# MTU size for interface connected to fabric, must be greater than 1564
opflex_interface_mtu: 1800

#vsphere

```

```

vsphere:
  server: myvsphere.local.lab
  user: administrator@vsphere.local
  passwd: xxxx
  allow_unverified_ssl: true
  datacenter_name: my-dc
  cluster_name: my-cluster
  datastore_name: mydatastore
  RHCOS_template_name: RHCOS46

#base_dir
# type: directory path
# required: yes
# notes: All install files and directories are created under this directory
base_dir: /root/ocpinstall

#node network details. This is common for bootstrap, master and worker nodes.
node_network_cidr: 192.168.53.0/24
node_network_gateway: 192.168.53.1
node_network_netmask: 255.255.255.0

#bootstrap node variables
bootstrap_vars:
  node_ip: 192.168.18.210      #required
  cpu_count: 8                #optional: defaults to 4
  memory_KB: 16384            #optional: defaults to 8192
  disk_size_MB: 40            #optional: defaults to 40

masters_vars:
  cpu_count: 8                #optional: defaults to 4
  memory_KB: 16384            #optional: defaults to 16384
  disk_size_MB: 40            #optional: defaults to 40
nodes:
  #mac address and ip address for each node is required
  - master-1:
    ip: 192.168.18.211
  - master-2:
    ip: 192.168.18.212
  - master-3:
    ip: 192.168.18.213

workers_vars:
  cpu_count: 8                #optional: defaults to 4
  memory_KB: 16384            #optional: defaults to 16384
  disk_size_MB: 40            #optional: defaults to 40
nodes:
  #mac address and ip address for each node is required
  - worker-1:
    ip: 192.168.18.214
  - worker-2:
    ip: 192.168.18.215

#user_ssh_key:
# required: no
# notes: if specified this key is setup on nodes, else ssh key of current
# user is used.
user_ssh_key: ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQD...

#additional_trust_bundle:
# required: no
# notes: use this field to add a certificate for private repository
#
# example:

```

```
#additional_trust_bundle: |
# -----BEGIN CERTIFICATE-----
# MIIDDDCCAfQCCQDuOnV7XBjpODANBgkqhkiG9w0BAQsFADBIMQswCQYDVQQGEwJV
# UzELMAkGALUECAwCQ0ExDDAKBgNVBACMA1NKQzEOMAwGALUECgwFQ21zY28xDjAM
# -----END CERTIFICATE-----

#openshift_pullsecret:
# required: yes
# notes: refer to https://cloud.redhat.com/openshift/install/pull-secret
# example:
#   openshift_pullsecret: {"auths":{"cloud.openshift.com":{"auth":.....}}
openshift_pullsecret: xxx
```

## Sample `hosts.ini` file

The following is a sample `hosts.ini`. The highlighted or bold values are those that you must modify to meet your site requirements.

```
[orchestrator]
192.168.18.200

[lb]
192.168.18.201
```

## Decommissioning OpenShift

Use this procedure to decommission OpenShift and remove the ACI-provisioned configuration from ACI.




---

**Note** Starting with Cisco APIC release 5.2, VMM domains for OpenShift cannot be removed from the APIC GUI. It is only possible using REST API, therefore, it is convenient to use the `acc-provision` tool to remove the VMM domain, and other related objects used by the decommissioned OpenShift cluster. Ensure you have the `acc-input-config.yaml` file and certificates used by the `acc-provision` tool to access APIC.

---

### Before you begin

In case of decommissioning or removing OpenShift cluster, ACI configuration provisioned for that cluster should be removed from ACI. The `acc-provision` tool can be used to remove that configuration.

### Procedure

Use the following command from the machine and folder which was used to provision the ACI infrastructure, to delete the pre-provisioned configurations and the VMM domain.

```
acc-provision -d -f openshift-4.6-esx -c acc-input-file -u user -p password
```

Example:

```
acc-provision -d -f openshift-4.6-esx -c acc-input-config.yaml -u admin -p password
```







**Americas Headquarters**  
Cisco Systems, Inc.  
San Jose, CA 95134-1706  
USA

**Asia Pacific Headquarters**  
CiscoSystems(USA)Pte.Ltd.  
Singapore

**Europe Headquarters**  
CiscoSystemsInternationalBV  
Amsterdam,TheNetherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at [www.cisco.com/go/offices](http://www.cisco.com/go/offices).