Red Hat OpenShift Container Platform 4.19
On Oracle Database Appliance
Deployment Guide

11 December 2025, Version 1.43

ORACLE

RED HAT OPENSHIFT Container Platform
ORACLE

Contents

1. Introduction

Red Hat OpenShift Container Platform is an enterprise-grade Kubernetes distribution designed to automate the deployment, scaling, and lifecycle management of containerized applications across hybrid and multi-cloud environments. Its built-in operational tooling and developer productivity enhancements make it a preferred platform for modern application workloads.

The **Red Hat OpenShift Assisted Installer**—delivered through the Red Hat Hybrid Cloud Console—provides a simplified, guided workflow for provisioning clusters. It supports Single Node OpenShift (SNO) as well as Highly Available (HA) multi-node clusters. It eliminates the need for a dedicated bootstrap node by leveraging the control plane nodes for cluster initialization.

The **Oracle Database Appliance (ODA)** is an engineered system optimized to deliver superior Oracle Database performance, availability, and simplicity. With built-in virtualization (KVM) capabilities, ODA can also host application VMs—including Red Hat OpenShift control plane and worker nodes—alongside database workloads.

This guide details the end-to-end process of deploying **OpenShift Container Platform 4.19** on **Oracle Database Appliance X10/X11**.

1.1. Recommended Oracle Database Appliance Configuration for Red Hat OpenShift

1.1.1. Oracle Database Appliance (ODA) Requirements

- **Model**: X10 or X11 (Single Node)
- **Memory**: Minimum 512 GB
- **CPU**: All cores enabled (64 cores)
- **ODA Release Version**: 19.28 or later

1.1.2. Storage Requirements

- **VM Storage:** Minimum 1 TB for vmstorage to host Application KVMs (control plane and worker nodes)

1.1.3. OpenShift Cluster Deployment

Node Type	Count	vCPU	Memory	Disk
Control Plane	3	20	24 GB	200 GB
Worker Node	3	16	24 GB	200 GB

1.1.4. Additional Recommendations:

- Use **dedicated CPU pools** for each VM to ensure predictable performance.

1.2. Additional Prerequisites

- A Red Hat account with an active OpenShift subscription
- Internet connectivity for downloading images and accessing Red Hat services
- Installation of DHCP, DNS, and HAProxy if using ODA-local infrastructure

Note: Before ODA patching, remove any manually installed system packages (DHCP, DNS, HAProxy) if flagged by the pre-patch report. Removing these services will disrupt OpenShift cluster functionality—plan maintenance windows accordingly. After patching is complete, re-add the removed packages to restore required services.

1.3. Modifying resources for Control Plane and Worker nodes

OpenShift nodes run as KVM guests inside ODA. Resources such as CPU, memory, virtual disks, and network interfaces must be configured through:

- odacli modify-vm commands
- ODA Browser User Interface (BUI)

Note: Refer to the ODA documentation for detailed usage.

2. Pre-deployment tasks

Note: Perform the following steps on bare metal layer.

2.1. Add all Control Plane and Worker nodes to /etc/hosts file

```
# cat /etc/hosts

127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6
127.0.0.1 oak0
10.10.60.77 oda.example.com
192.168.16.24 oda-priv.example.comoda-priv dcs0-priv

10.10.60.123 oda-os01.example.com api.ocp1.example.com api-int.ocp1.example.com *.apps.ocp1.example.com

# Master Nodes (Control Plane)
10.10.60.125    oda-os03.example.com
10.10.60.126    oda-os04.example.com
10.10.60.127    oda-os05.example.com

# Worker Nodes
10.10.60.128     oda-os06.example.com
10.10.60.129     oda-os07.example.com
10.10.60.130     oda-os08.example.com
```

2.2. HAProxy Configuration for Load Balancer

The example below demonstrates a load balancer configuration running on the ODA.

Install haproxy package if there is no external load balancer.

```
# dnf install -y haproxy

# cat /etc/haproxy/haproxy.cfg

global
   log 127.0.0.1:514 local0
   maxconn 2000
defaults
   log global
   mode tcp
   timeout connect 10s
   timeout client 1m
   timeout server 1m
   retries 3
```



```
# API Server Load Balancer (Port 6443)
frontend api_frontend
    bind 10.10.60.123:6443
    default_backend api_backend
backend api_backend
    balance roundrobin
    option tcp-check
    default-server check inter 10s fall 3 rise 2
    server oda-os03 10.10.60.125:6443 check
    server oda-os04 10.10.60.126:6443 check
    server oda-os05 10.10.60.127:6443 check

# Machine Config Server (Port 22623)
frontend mcs_frontend
    bind 10.10.60.123:22623
    default_backend mcs_backend
backend mcs_backend
    balance roundrobin
    option tcp-check
    default-server check inter 10s fall 3 rise 2
    server oda-os03 10.10.60.125:22623 check
    server oda-os04 10.10.60.126:22623 check
    server oda-os05 10.10.60.127:22623 check
# Ingress Load Balancer (80)
frontend ingress_http_frontend
    bind 10.10.60.123:80
    mode tcp
    default_backend ingress_http_backend
backend ingress_http_backend
    mode tcp
    balance roundrobin
    default-server check inter 10s fall 3 rise 2
    server oda-os06 10.10.60.128:80 check
    server oda-os07 10.10.60.129:80 check
    server oda-os08 10.10.60.130:80 check
# Ingress Load Balancer (443)
frontend ingress_https_frontend
    bind 10.10.60.123:443
    mode tcp
    default_backend ingress_https_backend

backend ingress_https_backend
    mode tcp
    balance roundrobin
    default-server check inter 10s fall 3 rise 2
    server oda-os06 10.10.60.128:443 check
    server oda-os07 10.10.60.129:443 check
```



```
   server oda-os08 10.10.60.130:443 check
```

Enable and start haproxy
systemctl enable haproxy
systemctl start haproxy
systemctl status haproxy

2.3. DHCP Configuration

The example below demonstrates a DHCP configuration running on the ODA.

Install DHCP server package
dnf install -y dhcp-server

Configure DHCP server with static reservations for all VMs.

```
# cat /etc/dhcp/dhcpd.conf
authoritative;
ddns-update-style interim;
allow booting;
allow bootp;
allow unknown-clients;
ignore client-updates;
default-lease-time 14400;
max-lease-time 14400;
subnet 10.10.60.0 netmask 255.255.252.0 {
 option routers                  10.10.60.1; # lan
 option subnet-mask              255.255.252.0;
 option domain-name              "example.com";
 option domain-name-servers      10.10.60.77;
 range 10.10.60.123 10.10.60.138;
}
host oda-os03 {
 hardware ethernet 52:54:00:ad:2b:57;
 fixed-address 10.10.60.125;
}
host oda-os04 {
 hardware ethernet 52:54:00:f6:89:5a;
 fixed-address 10.10.60.126;
}
host oda-os05 {
 hardware ethernet 52:54:00:9d:a2:d4;
 fixed-address 10.10.60.127;
}

host oda-os06 {
 hardware ethernet 52:54:00:b5:9f:20;
 fixed-address 10.10.60.128;
```



```
}
host oda-os07 {
 hardware ethernet 52:54:00:33:c5:d2;
 fixed-address 10.10.60.129;
}
host oda-os08 {
 hardware ethernet 52:54:00:1a:6e:e2;
 fixed-address 10.10.60.130;
}
```

```
# systemctl stop dhcpd
```

Important: Do not restart DHCP until all VMs have been created and MAC addresses updated.

2.4. Configure DNS server

Install BIND packages:
dnf install -y bind bind-utils

Configure:

- Forward lookup zone
- Reverse lookup zone
- API and ingress endpoints

Ensure:

- All A and PTR records match the OpenShift Assisted Installer configuration.
- DNS resolves both forward and reverse for every node.

```
# cat /etc/named.conf
//
// named.conf
//
// Provided by Red Hat bind package to configure the ISC BIND named(8) DNS
// server as a caching only nameserver (as a localhost DNS resolver only).
//
// See /usr/share/doc/bind*/sample/ for example named configuration files.
//
// See the BIND Administrator's Reference Manual (ARM) for details about the
// configuration located in /usr/share/doc/bind-{version}/Bv9ARM.html

options {
        listen-on port 53 { 127.0.0.1; 10.10.60.77; };
#       listen-on-v6 port 53 { ::1; };
        directory       "/var/named";
        dump-file       "/var/named/data/cache_dump.db";
        statistics-file "/var/named/data/named_stats.txt";
        memstatistics-file "/var/named/data/named_mem_stats.txt";
        recursing-file  "/var/named/data/named.recursing";
        secroots-file   "/var/named/data/named.secroots";
        allow-query     { localhost; 10.10.60.0/22; };

        /*
         - If you are building an AUTHORITATIVE DNS server, do NOT enable recursion.
         - If you are building a RECURSIVE (caching) DNS server, you need to enable
           recursion.
         - If your recursive DNS server has a public IP address, you MUST enable access
           control to limit queries to your legitimate users. Failing to do so will
           cause your server to become part of large scale DNS amplification
           attacks. Implementing BCP38 within your network would greatly
           reduce such attack surface
        */
```



```
        recursion yes;

        dnssec-enable yes;
        dnssec-validation no;

        # Using Google DNS
        forwarders {
         10.10.0.4;
        };

        /* Path to ISC DLV key */
        bindkeys-file "/etc/named.root.key";

        managed-keys-directory "/var/named/dynamic";

        pid-file "/run/named/named.pid";
        session-keyfile "/run/named/session.key";
};

logging {
      channel default_debug {
            file "data/named.run";
            severity dynamic;
      };
};

zone "." IN {
        type hint;
        file "named.ca";
};

# Include ocp zones

zone "example.com" {
   type master;
   file "/etc/named/zones/db.example.com"; # zone file path
};

#zone "211-212.91.10.in-addr.arpa" IN {
#    type master;
#    file "/etc/named/zones/10.91.211-212.rev";
#    allow-update { none; };
#};

zone "60.10.10.in-addr.arpa" {
   type master;
   file "/etc/named/zones/db.reverse";
```



```
};
include "/etc/named.rfc1912.zones";
include "/etc/named.root.key";
```

2.4.1. Set up reverse DNS resolution

```
# cat /etc/named/zones/db.reverse

$TTL 1W
@       IN      SOA     ns1.example.com.        root (
                        2019070701      ; serial
                        3H              ; refresh (3 hours)
                        30M             ; retry (30 minutes)
                        2W              ; expiry (2 weeks)
                        1W )            ; minimum (1 week)
        IN      NS      ns1.example.com.
;
; ------------------------------------------------
; Load Balancer & API Endpoints  (10.91.211.249)
; ------------------------------------------------
123     IN      PTR     lb.ocp1.example.com.
123     IN      PTR     api.ocp1.example.com.
123     IN      PTR     api-int.ocp1.example.com.
124     IN      PTR     ingress.ocp1.example.com.
; ------------------------------------------------
; Control Plane Nodes
; ------------------------------------------------
125     IN      PTR     oda-os03.example.com.
126     IN      PTR     oda-os04.example.com.
127     IN      PTR     oda-os05.example.com.
; ------------------------------------------------
; Worker Nodes
; ------------------------------------------------
128     IN      PTR     oda-os06.example.com.
129     IN      PTR     oda-os07.example.com.
130     IN      PTR     oda-os08.example.com.
```

2.4.2. Set up DNS resolution

```
# cat db.example.com

$TTL    604800
@       IN      SOA     lb.ocp1.example.com. contact.ocp1.example.com. (
                  3     ; Serial (Increment every change)
             604800     ; Refresh
              86400     ; Retry
```



```
          2419200     ; Expire
           604800     ; Minimum
)
      IN     NS     lb.ocp1.example.com.
ingress.ocp1.example.com.    IN     A      10.10.60.124
lb.ocp1.example.com.         IN     A      10.10.60.123
; Control Plane Nodes
oda-os03.example.com.        IN     A      10.10.60.125
oda-os04.example.com.        IN     A      10.10.60.126
oda-os05.example.com.        IN     A      10.10.60.127
; Worker Nodes
oda-os06.example.com.        IN     A      10.10.60.128
oda-os07.example.com.        IN     A      10.10.60.129
oda-os08.example.com.        IN     A      10.10.60.130
; OpenShift Internal - Load balancer
api.ocp1.example.com.        IN    A    10.10.60.123
api-int.ocp1.example.com.    IN    A    10.10.60.123
*.apps.ocp1.example.com.     IN    A    10.10.60.123
oauth-openshift.apps.ocp1.example.com.              IN     A     10.10.60.123
console-openshift-console.apps.ocp1.example.com.    IN     A     10.10.60.123
```

2.5. Configure resolv.conf for name resolution

This step is necessary only if a local DNS server is being used for the OpenShift deployment. f not, the DNS server specified in resolv.conf was already configured during the initial ODA deployment.

```
# cat /etc/resolv.conf

search example.com
nameserver 10.10.60.77
```


3. Deploying OpenShift on ODA

3.1. Create vmstorage for the VM guests

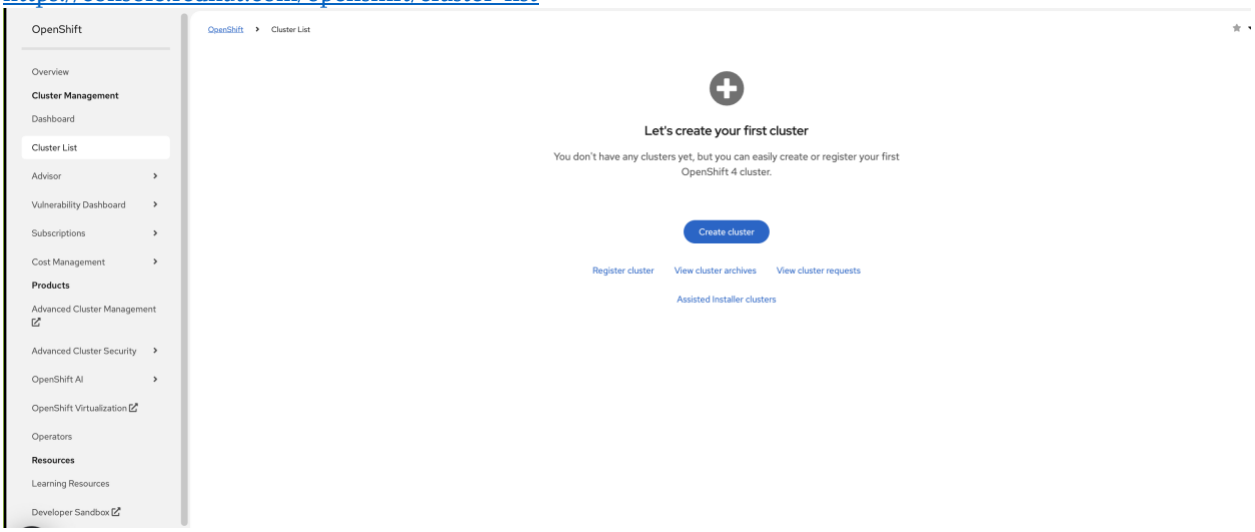
```
# odacli create-vmstorage -n kvmvmstore -s 2T -dg DATA
```

3.2. Create dedicated CPU pools for all nodes

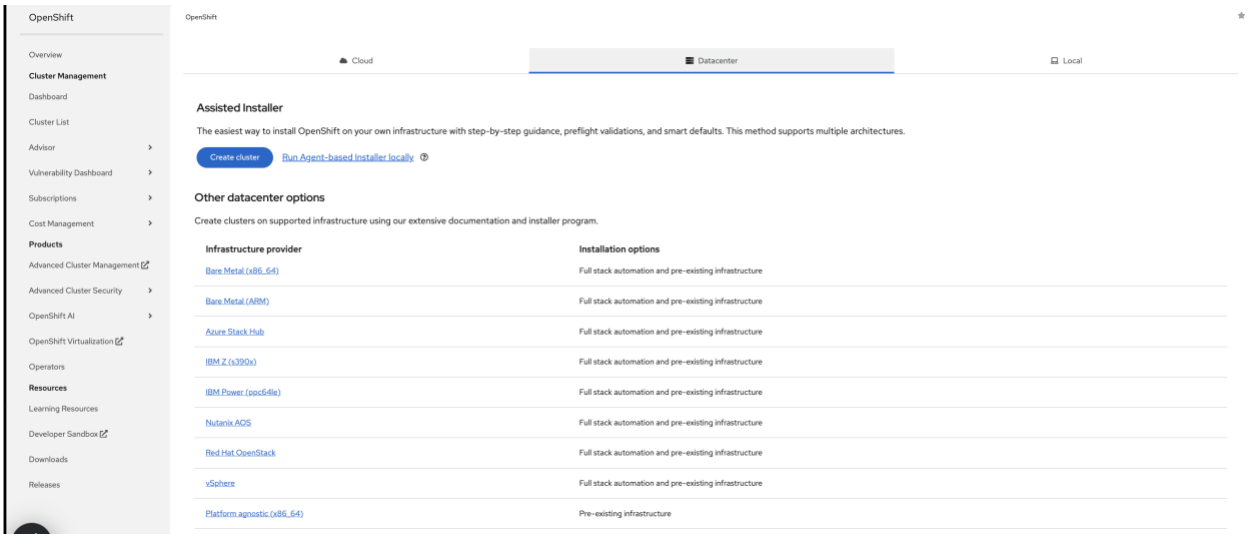
```
# odacli create-cpupool -c 10 -vm -n controlplane1
# odacli create-cpupool -c 10 -vm -n controlplane2
# odacli create-cpupool -c 10 -vm -n controlplane3

# odacli create-cpupool -c 8 -vm -n worker1
# odacli create-cpupool -c 8 -vm -n worker2
# odacli create-cpupool -c 8 -vm -n worker3
```

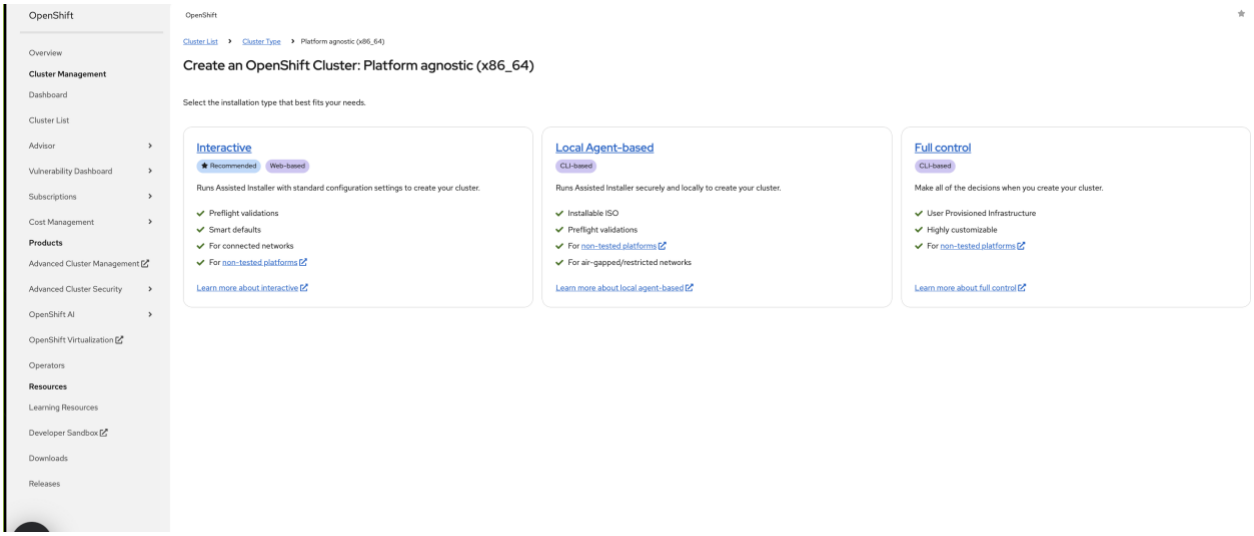

3.3. Setup OpenShift Cluster using OpenShift Console

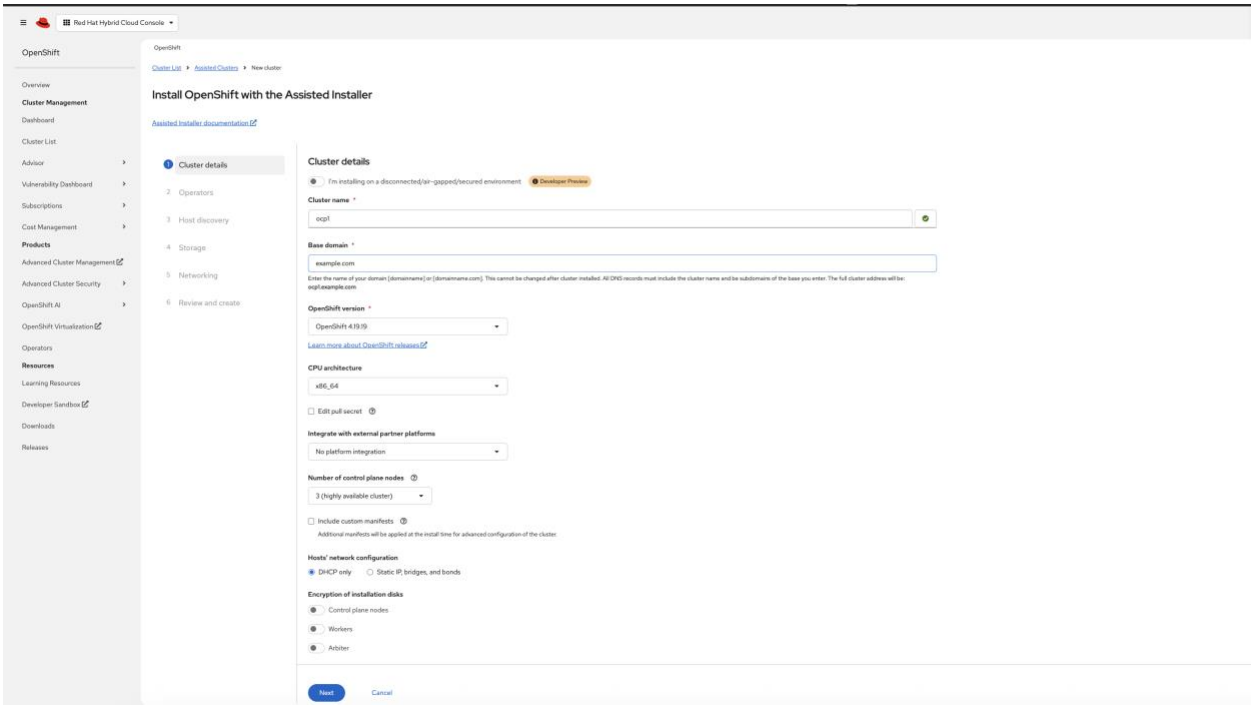
Log in to the Red Hat Hybrid Cloud Console.
https://console.redhat.com/openshift/cluster-list

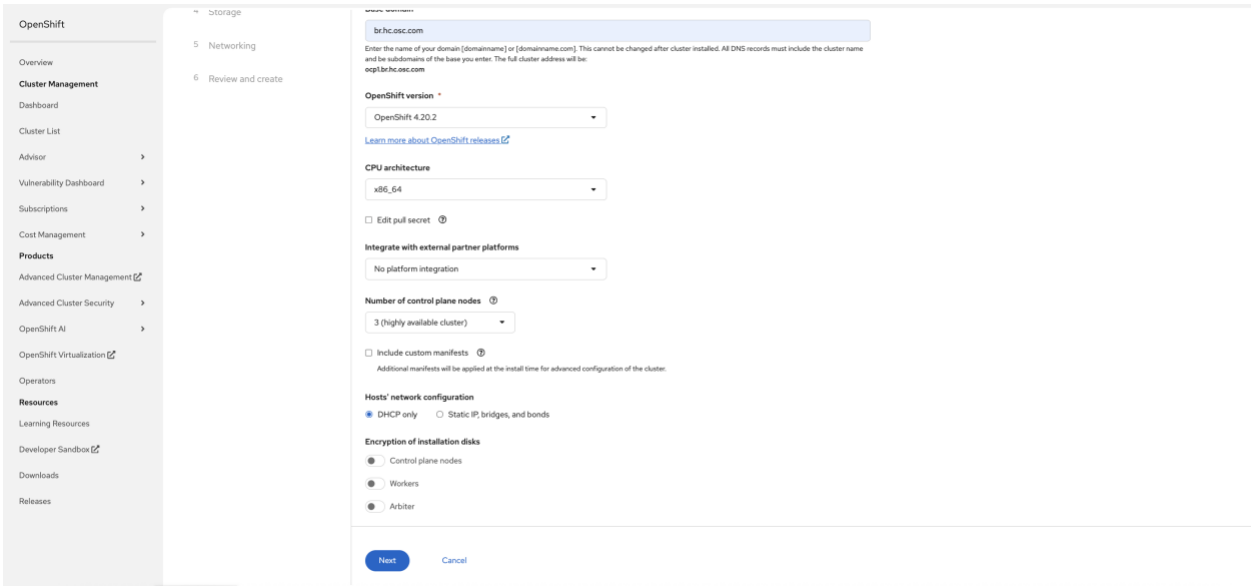


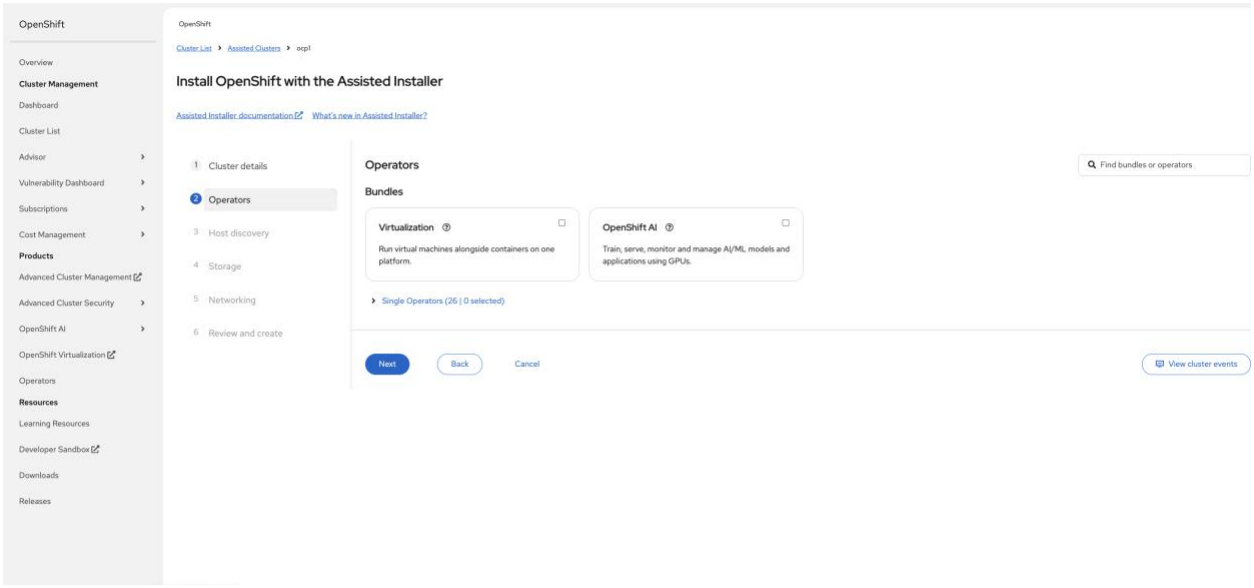
Navigate to Assisted Installer → Create Cluster.

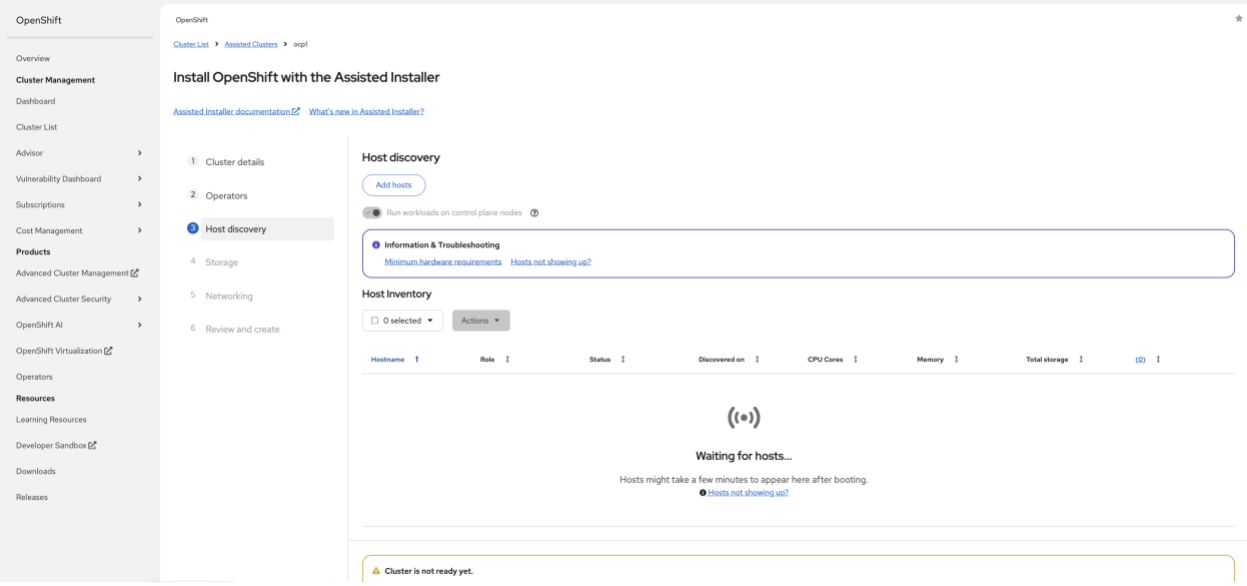








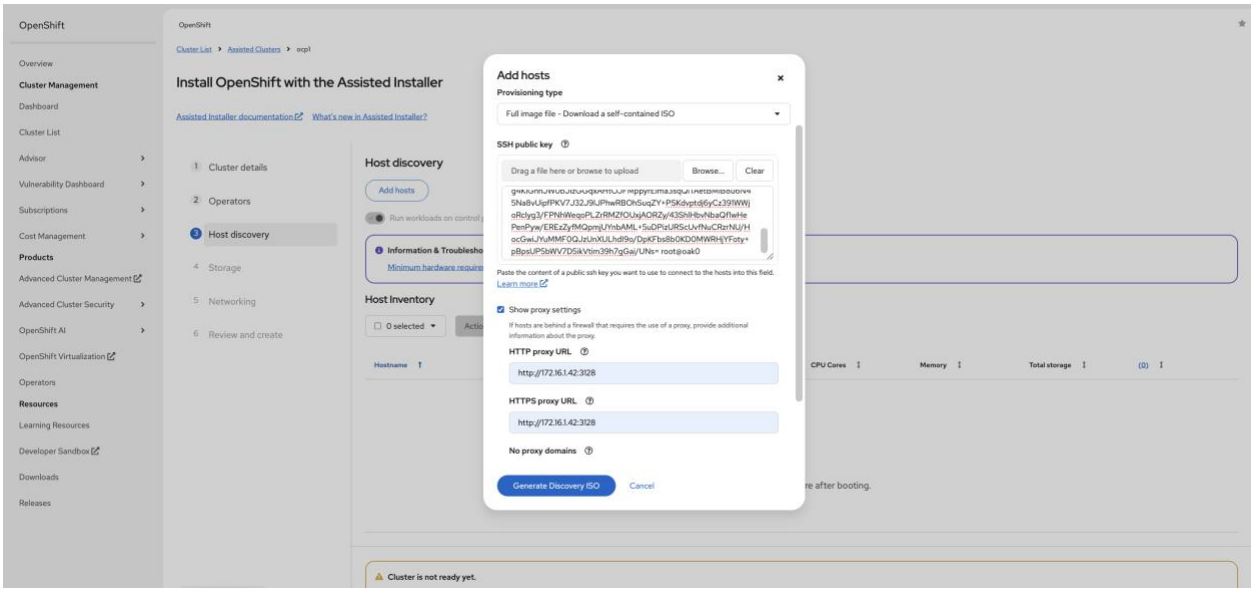


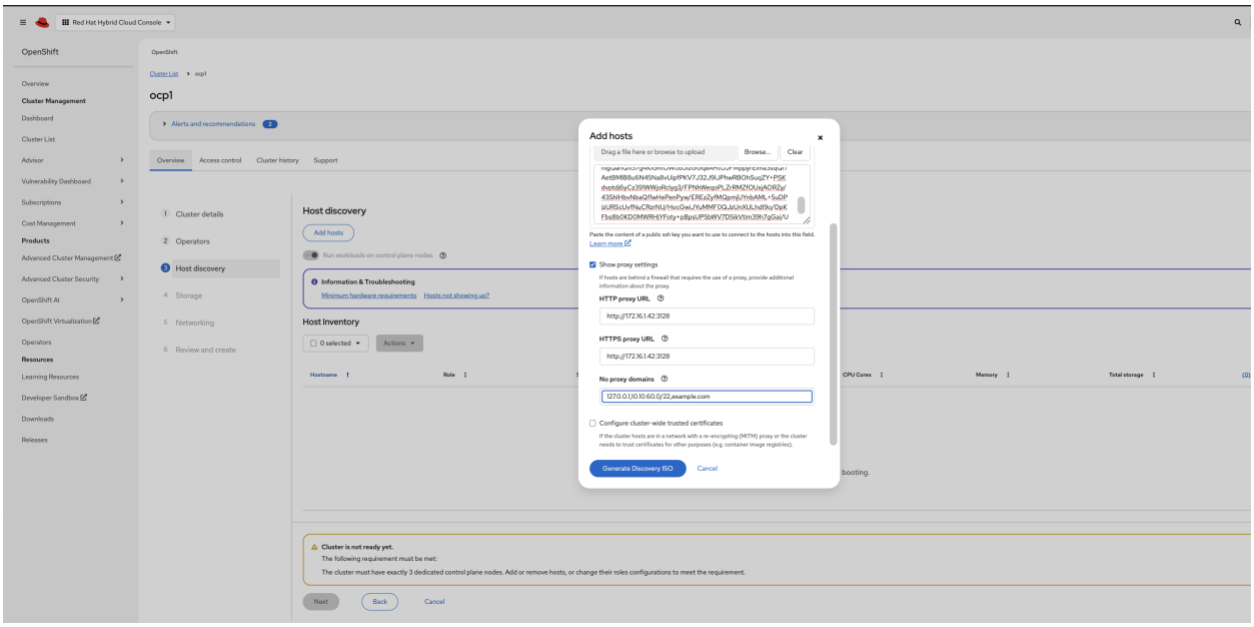


Generate an SSH key and create a backup copy to ensure it is safely preserved.

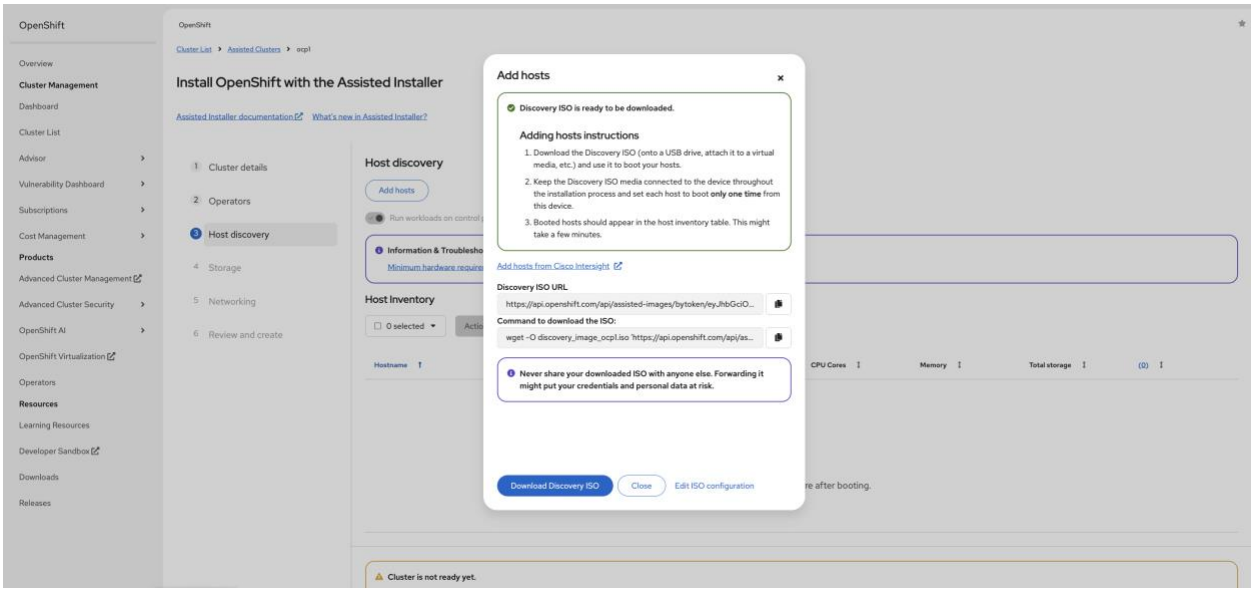
```
# ssh-keygen -b 2048 -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
.our identification has been saved in
.pub.public key has been saved in
The key fingerprint is:
SHA256:SUzkQcyB+gG3qeSYT+LRL+ZbrOe8Xe4pqI+OhYrad5s root@oda.example.com

The key's randomart image is:
+---[RSA 2048]----+
|      **.        |
|    . o+o.       |
|     + o+        |
|    o +. .       |
|   * o .S        |
|  =.=..          |
| ..=..o.  .      |
|.o.o===o.o .     |
|= o=BOEo.o+      |

+----[SHA256]-----+
```


Upload SSH key.





Download the Discovery ISO.




```
[root@oda script]# wget -O discovery_image_ocp1.iso 'https://api.openshift.com/api/assisted-images/bytoken/eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE3NjUzNTQ3MjYsInN1YiI6IjFmZDkyN2NmLTNkMTQtNGRiYy04N2QxLWZiZjc2MjYyNzRhMiJ9.jRZ46uFG16Zfy8I-8qr0Yuo6zKgnHK857wzIcEYDGPY/4.19/x86_64/full.iso'
--2025-12-10 00:19:05--  https://api.openshift.com/api/assisted-images/bytoken/eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE3NjUzNTQ3MjYsInN1YiI6IjFmZDkyN2NmLTNkMTQtNGRiYy04N2QxLWZiZjc2MjYyNzRhMiJ9.jRZ46uFG16Zfy8I-8qr0Yuo6zKgnHK857wzIcEYDGPY/4.19/x86_64/full.iso
Resolving api.openshift.com (api.openshift.com)... 54.243.47.190, 50.19.111.26, 54.209.234.152
Connecting to api.openshift.com (api.openshift.com)|54.243.47.190|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1347420160 (1.3G) [application/octet-stream]
Saving to: 'discovery_image_ocp1.iso'

discovery_image_ocp1.iso          100%[=====================================================>]   1.25G  2.78MB/s    in 6m 17s

2025-12-10 00:25:23 (3.41 MB/s) - 'discovery_image_ocp1.iso' saved [1347420160/1347420160]
```

3.4. Script to create control nodes

Execute the script below to create the Control Plane nodes
=== control.sh ===

```
#!/bin/bash

# Function to monitor VM creation
monitor_vm_creation() {
    while true; do
        # Get the last line containing both "VM" and "creation"
        line=$(odacli list-jobs | grep 'VM' | grep 'creation' | tail -n 1)

        # Extract the status (assumes status is the last word in the line)
        status=$(echo "$line" | awk '{print $NF}')

        # Print current status (optional)
```



```
        echo "Current status: $status"

        # Check status
        if [[ "$status" == "Success" ]]; then
            echo "VM creation completed successfully."
            break
        elif [[ "$status" == "Running" ]]; then
            # Continue waiting
            sleep 30
        else
            echo "VM creation failed or has an unexpected status: $status"
            echo " You can use odacli describe-job -i <jobId> to help identify the reason for the failure. "
            return 1  # Abort the script with a failure code
        fi
    done
}

# Function to create a VM and assign CPU pool
create_vm() {
    local vm_name=$1
    local cpupool=$2
    echo "Creating VM: $vm_name in CPU pool: $cpupool"
    # Create the VM
    odacli create-vm --name "$vm_name" --memory 24G --vcpus 20 -vms kvmvmstore -s 200G |
        --os-variant rhel9.0 -vn pubnet --source /var/lib/libvirt/images/discovery_image_ocp1.iso |
        --graphics vnc,listen=0.0.0.0 -cp $cpupool
}
# Create OpenShift VMs and assign to different CPU pools
create_vm "oda-os03" "controlplane1"
monitor_vm_creation
create_vm "oda-os04" "controlplane2"
monitor_vm_creation
create_vm "oda-os05" "controlplane3"
monitor_vm_creation

# List all created VMs
echo "Listing all created VMs..."
odacli list-vms
=== EOF ===
```


3.5. Script to create worker nodes

Execute the script below to create the Worker nodes
=== worker.sh ===

```
#!/bin/bash

# Function to monitor VM creation
monitor_vm_creation() {
   while true; do
      # Get the last line containing both "VM" and "creation"
      line=$(odacli list-jobs | grep 'VM' | grep 'creation' | tail -n 1)
      # Extract the status (assumes status is the last word in the line)
      status=$(echo "$line" | awk '{print $NF}')
      # Print current status (optional)
      echo "Current status: $status"
      # Check status
      if [[ "$status" == "Success" ]]; then
         echo "VM creation completed successfully."
         break
      elif [[ "$status" == "Running" ]]; then
         # Continue waiting
         sleep 30
      else
         echo "VM creation failed or has an unexpected status: $status"
         echo " You can use odacli describe-job -i <jobId> to help identify the reason for the failure. "
         return 1  # Abort the script with a failure code
      fi
   done
}

# Function to create a VM and assign CPU pool
create_vm() {
   local vm_name=$1
   local cpupool=$2
   echo "Creating VM: $vm_name in CPU pool: $cpupool"
   # Create the VM
   odacli create-vm --name "$vm_name" --memory 24G --vcpus 16 -vms kvmvmstore -s 200G |
      --os-variant rhel9.0 -vn pubnet --source /var/lib/libvirt/images/discovery_image_ocp1.iso |
      --graphics vnc,listen=0.0.0.0 -cp $cpupool
}
```



```
# Create OpenShift VMs and assign to different CPU pools
create_vm "oda-os06" "worker1"
monitor_vm_creation
create_vm "oda-os07" "worker2"
monitor_vm_creation
create_vm "oda-os08" "worker3"
monitor_vm_creation

# List all created VMs
echo "Listing all created VMs..."
odacli list-vms
```

=== EOF ===

3.6. Update the MAC address for each node in dhcpd.conf

=== get.sh ===

```
#!/bin/bash

# Define the list of VM names
VM_NAMES=("oda-os03" "oda-os04" "oda-os05" "oda-os06" "oda-os07" "oda-os08")

# Loop through each VM and fetch its MAC address
for VM in "${VM_NAMES[@]}"; do
   MAC=$(odacli describe-vm -n "$VM" | tail -n 2 | grep -oP '(?<=pubnet:)[0-9a-fA-F:]{17}' | head -n 1)

   if [[ -z "$MAC" ]]; then
      echo "$VM: MAC address not found"
   else
      echo "$VM: $MAC"
   fi
done
```

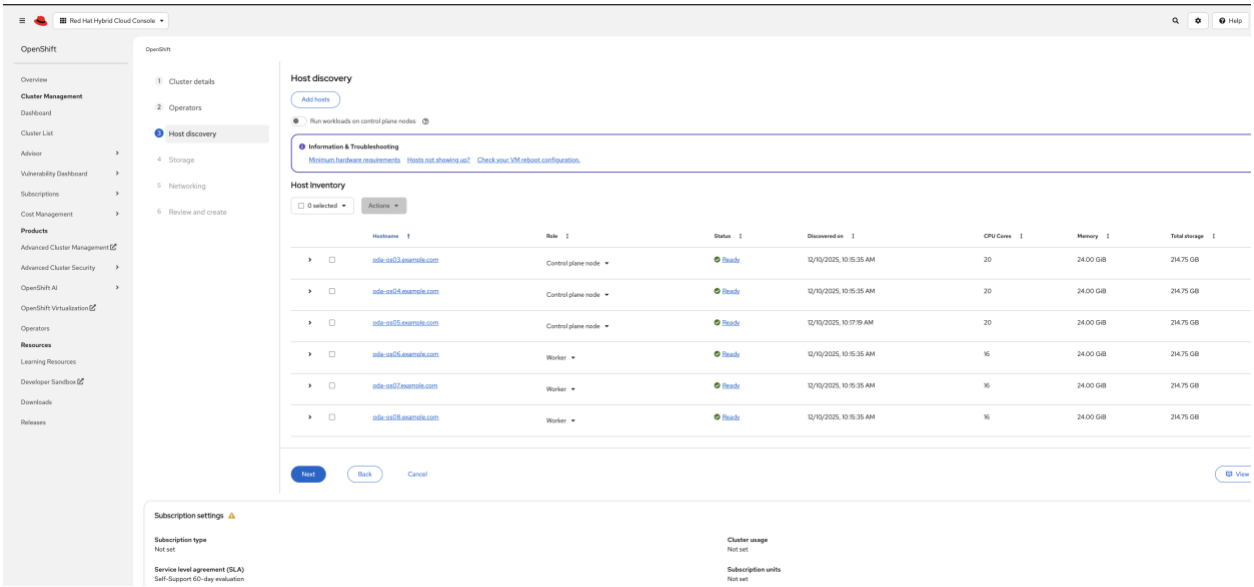
=== EOF ===

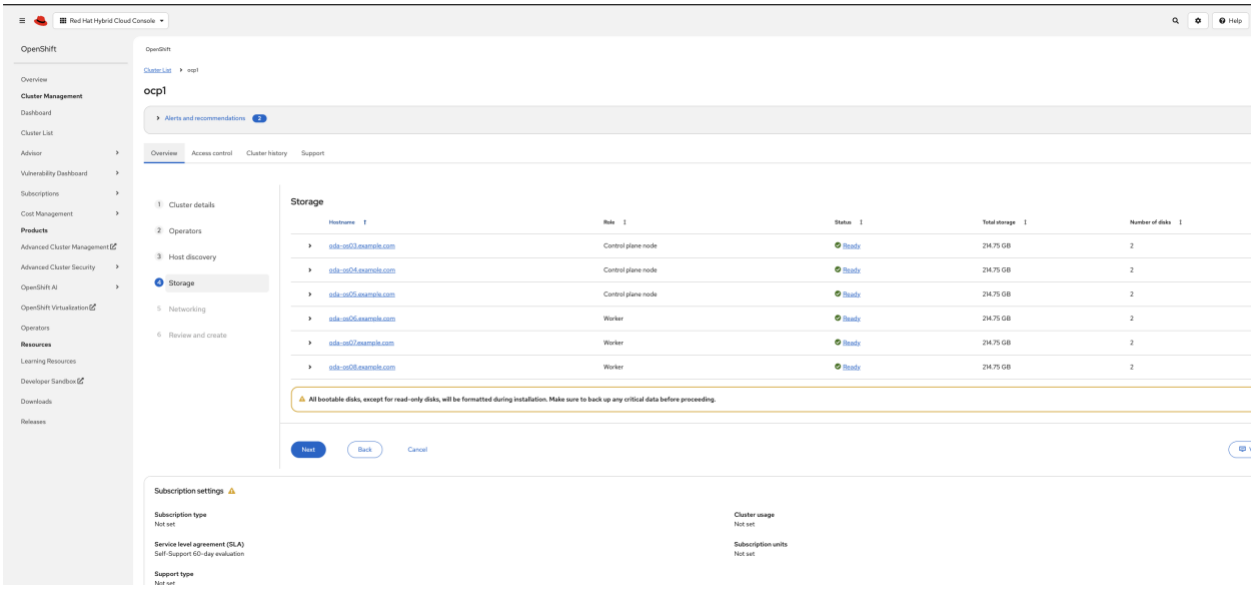
Execute get.sh to determine which MAC address corresponds to each host.

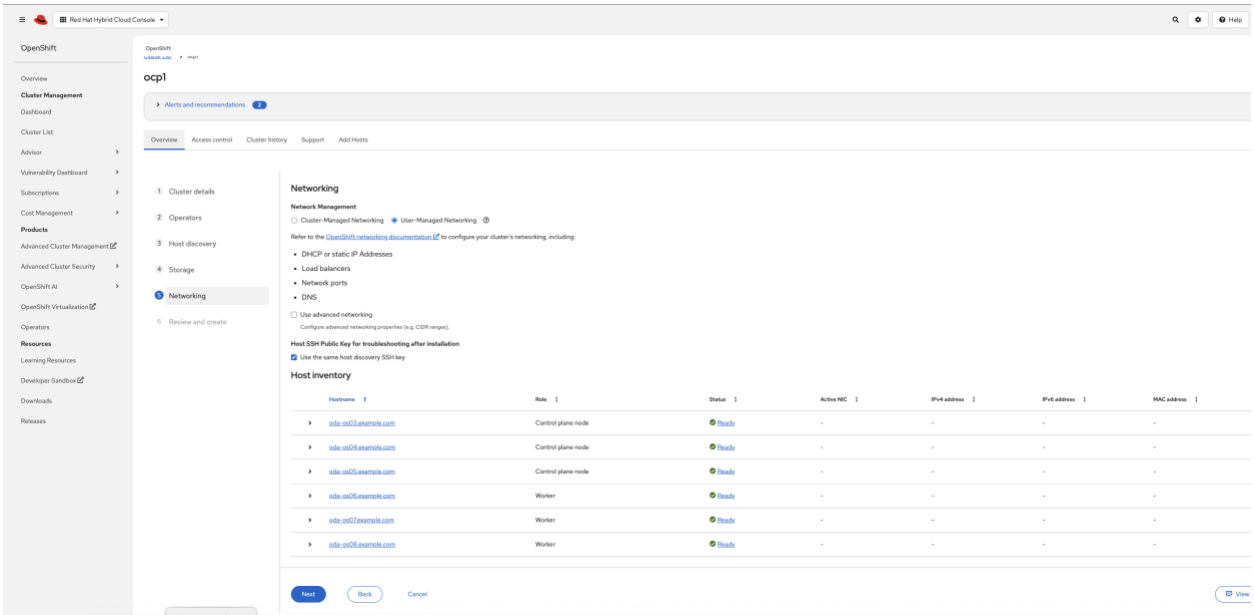
```
# ./get.sh
oda-os03: 52:54:00:94:96:f1
oda-os04: 52:54:00:53:d0:59
oda-os05: 52:54:00:69:a2:93
oda-os06: 52:54:00:04:e2:91
oda-os07: 52:54:00:b3:e4:c6
oda-os08: 52:54:00:17:98:83
```


Update the MAC addresses in /etc/dhcp/dhcpd.conf, then start the DHCPD service.
#systemctl start dhcpd

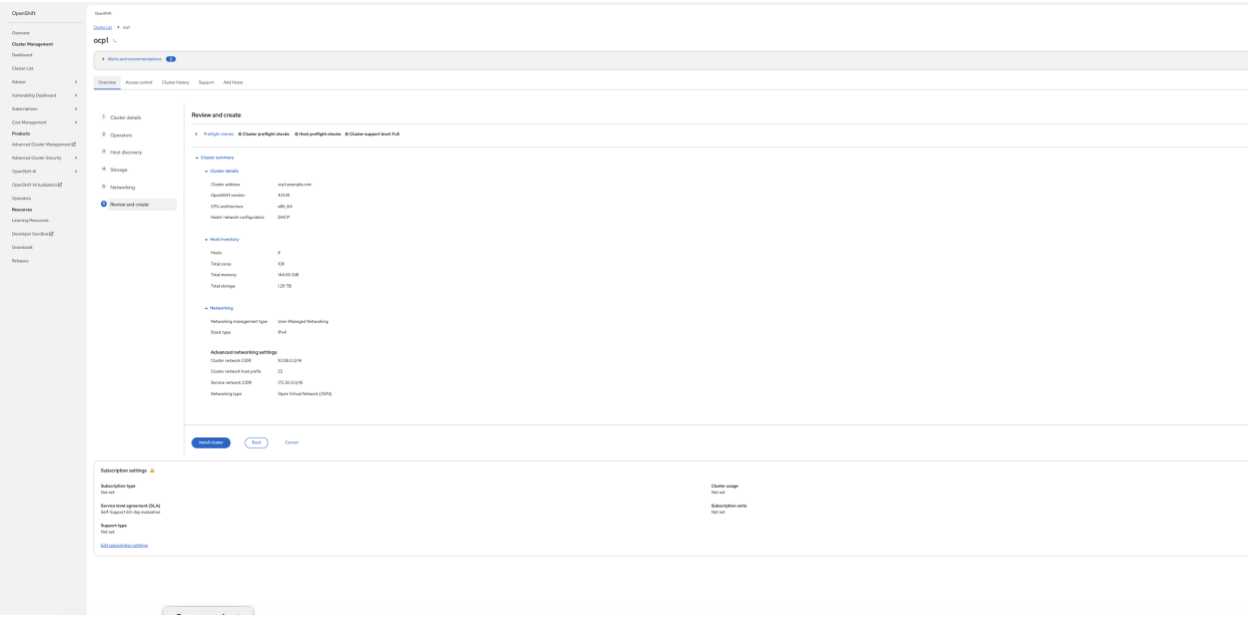
The hosts will become visible in the OpenShift Console after a few minutes.
Assign roles (Control Plane / Worker).







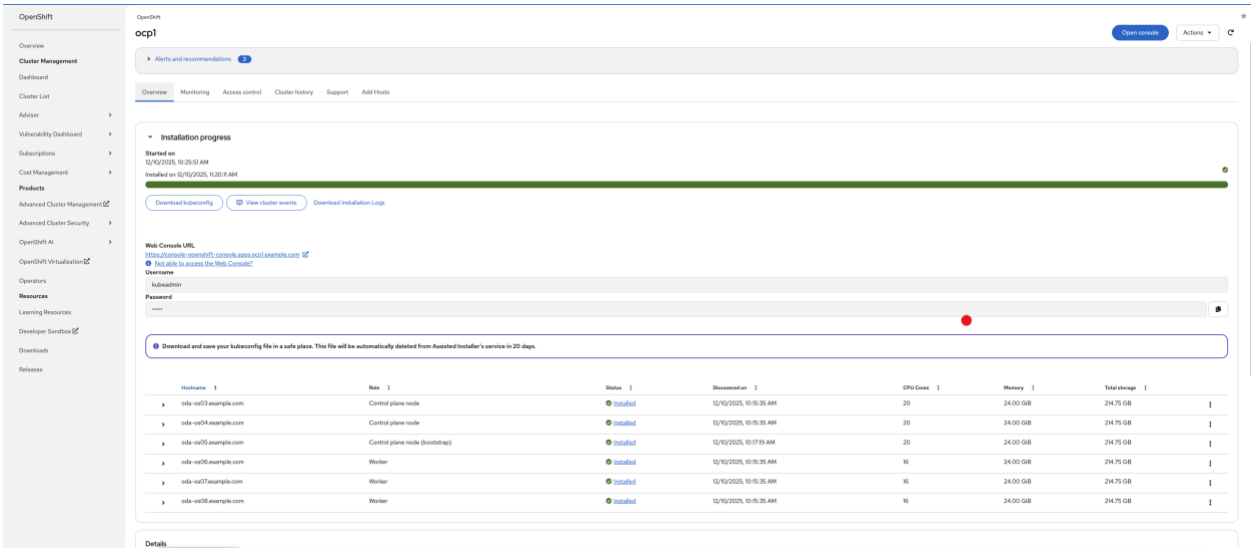
Validate networking and disk checks and start the installation.



3.7. Monitor Deployment

Using the Assisted Installer dashboard:

- Wait for all nodes to reach Ready state.
- Installation completes automatically.



4. Conclusion

Deploying Red Hat OpenShift on Oracle Database Appliance provides a unified platform capable of running mission-critical Oracle Database workloads alongside enterprise containerized applications. By leveraging ODA's engineered performance, integrated management tools, and guaranteed resource isolation via CPU pools, organizations can modernize applications without additional hardware.

