
# Deploying Installer Provisioned Infrastructure (IPI) of OpenShift on Bare Metal - 4.6

Deployment Integration Team

DRAFT

DRAFT

*Draft documentation*

This document is considered a DRAFT:

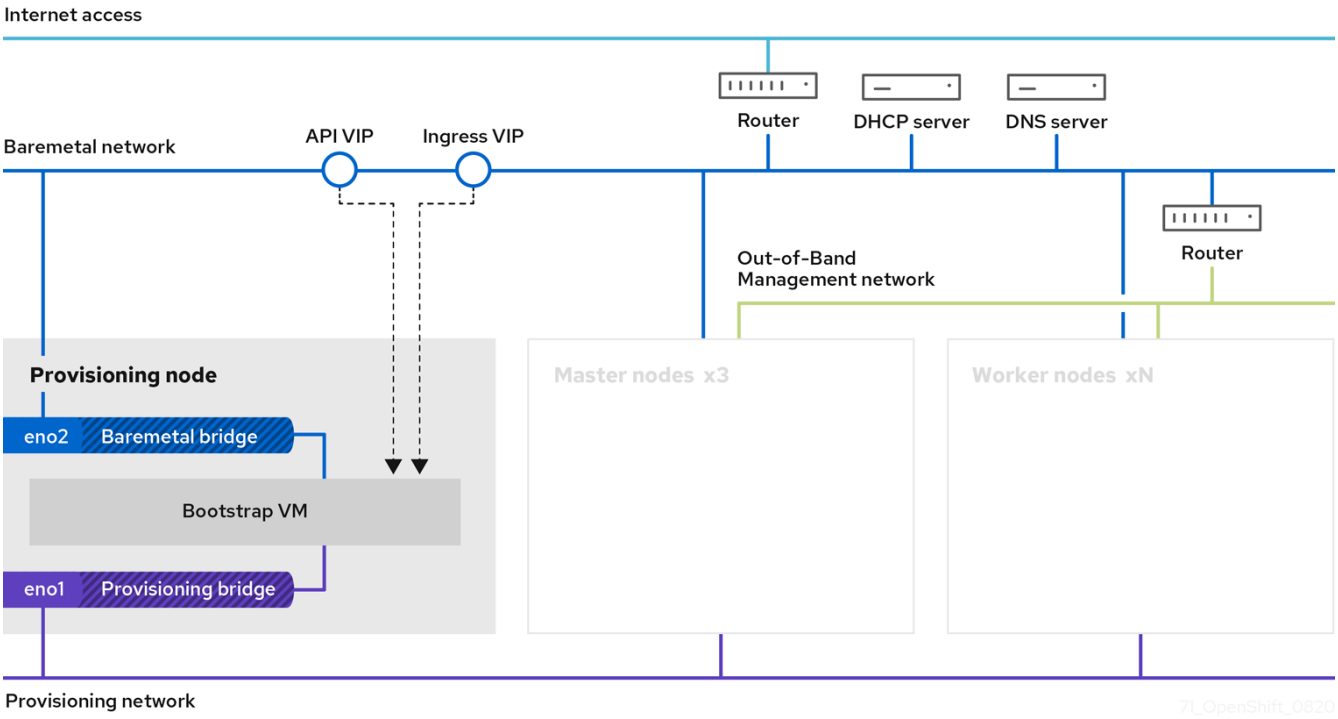
1. It might not be complete
2. It might be not accurate
3. It might break your environment

Download the PDF version of this document or visit https://openshift-kni.github.io/baremetal-deploy/

# Chapter 1. Deploying IPI Bare Metal

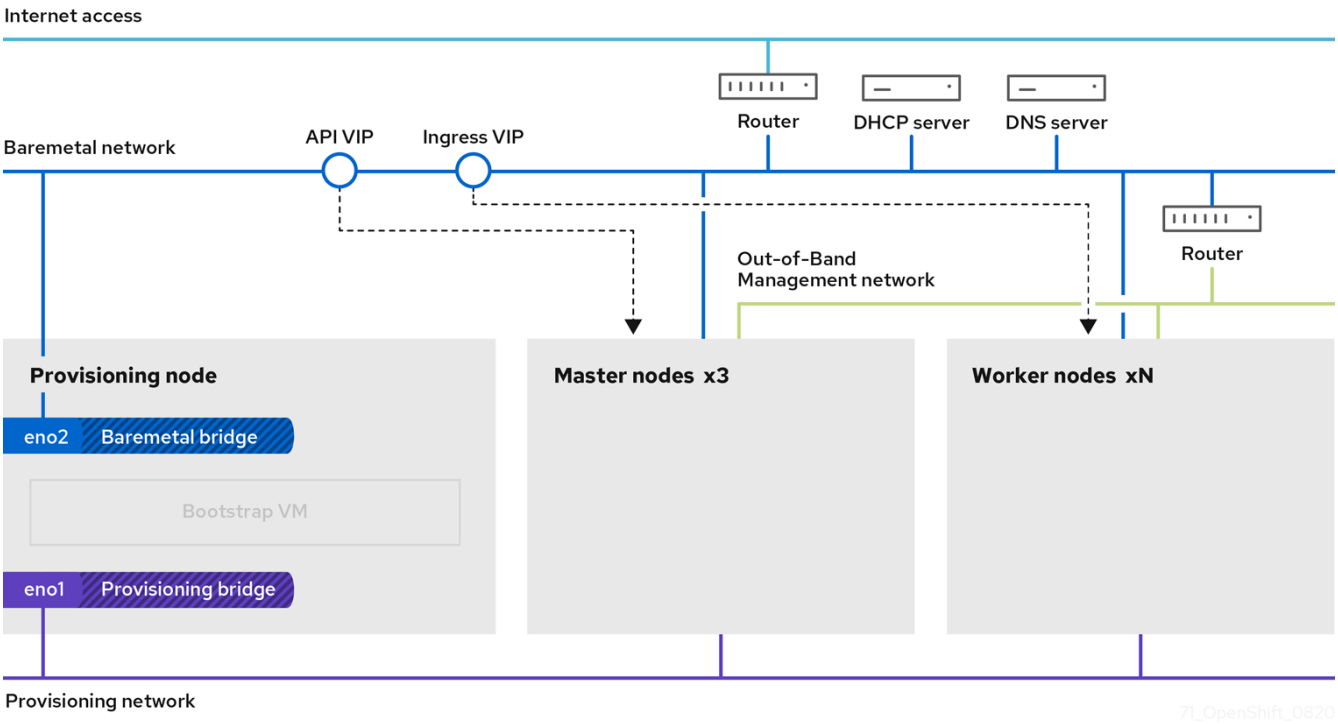
Installer Provisioned Infrastructure (IPI) installation provides support for installing OpenShift Container Platform on bare metal nodes. This guide provides a methodology to achieving a successful installation.

The bare metal node labeled as `provisioner` contains two network bridges: provisioning and baremetal, each one connected to a different network. During installation of IPI on baremetal, a bootstrap VM is created and connected to both the provisioning and baremetal network via those bridges. The role of the VM is to assist in the process of deploying an OpenShift Container Platform cluster.




When the installation of OpenShift control plane nodes, or master nodes, is complete and fully operational, the bootstrap VM is destroyed automatically and the appropriate VIPs are moved accordingly.

The API VIPs move into the control plane nodes and the Ingress VIP services applications that reside within the worker nodes.


Internet access
Router
DHCP server
DNS server
Baremetal network
API VIP
Ingress VIP
Router
Out-of-Band
Management network
Provisioning node
Master nodes x3
Worker nodes xN
eno2
Baremetal bridge
Bootstrap VM
eno1
Provisioning bridge
Provisioning network

DRAFT

# Chapter 2. Prerequisites

Before installing OpenShift Container Platform, ensure the hardware environment meets the following requirements.

## 2.1. Node requirements

IPI installation involves a number of hardware node requirements:

- **CPU architecture:** All nodes **must** use `x86_64` CPU architecture.
- **Similar nodes:** Nodes **should** have an identical configuration per role. That is, control plane nodes **should** be the same brand and model with the same CPU, RAM and storage configuration. Worker nodes should be **identical**.
- **Baseboard Management Controller:** The provisioner node must be able to access the baseboard management controller (BMC) of each OpenShift Container Platform cluster node. You may use IPMI, RedFish, or a proprietary protocol.
- **Latest generation:** Nodes should be of the most recent generation. IPI installation relies on IPMI, which should be compatible across nodes. Additionally, RHEL 8 ships with the most recent drivers for RAID controllers. Ensure that the nodes are recent enough to support RHEL 8 for the provisioning node and all the OpenShift Container Platform nodes.
- **Network interfaces:** Each node **must** have at least two network interfaces (NICs)- one for the `provisioning` network and one for the public `baremetal` network. Network interface names **must** follow the same naming convention across all nodes. For example, the first NIC name on a node, such as `eth0` or `eno1`, should be the same name on all of the other nodes. The same principle applies to the remaining NICs on each node. It is recommended these network interfaces be 10 GB NICs.
- **Provisioning node:** IPI installation requires one provisioning node.
- **Control plane:** IPI installation requires three control plane (master) nodes for high availability.
- **Worker nodes (Optional):** A typical production cluster may include worker nodes. IPI installation in a high availability environment provides the flexibility of deploying an initial cluster with or without worker nodes. If worker nodes are deployed, at **least** two worker nodes are required for the initial cluster deployment.
- **Registry node:** (Optional) If setting up a disconnected mirrored registry, it is recommended this reside in its own node.

DRAFT

*Additional Node Considerations*

- **Unified Extensible Firmware Interface (UEFI):** UEFI boot is required on all OpenShift Container Platform nodes when using IPv6 addressing on the `provisioning` network. In addition, UEFI Device PXE Settings must be set to use the IPv6 protocol on the `provisioning` network NIC.

## 2.2. Network requirements

IPI installation involves several network requirements. First, IPI installation involves a non-routable `provisioning` network for provisioning the OS on each bare metal node and a routable

`baremetal` network for access to the public network. Since IPI installation deploys `ironic-dnsmasq`, the networks should have no other DHCP servers running on the same broadcast domain. Network administrators **must** reserve IP addresses for each node in the OpenShift Container Platform cluster.

*Network Time Protocol (NTP)*

Each OpenShift Container Platform node in the cluster must have access to an NTP server.

*Configuring NICs*

OpenShift Container Platform deploys with two networks:

- `provisioning`: The `provisioning` network is a non-routable network used for provisioning the underlying operating system on each node that is a part of the OpenShift Container Platform cluster. The first NIC on each node, such as `eth0` or `eno1`, **must** interface with the `provisioning` network.
- `baremetal`: The `baremetal` network is a routable network used for external network access to the outside world. The second NIC on each node, such as `eth1` or `eno2`, **must** interface with the `baremetal` network.

Each NIC should be on a separate VLAN corresponding to the appropriate network.

*Configuring the DNS server*

Clients access the OpenShift Container Platform cluster nodes over the `baremetal` network. A network administrator **must** configure a subdomain or subzone where the canonical name extension is the cluster name.

```
<cluster-name>.<domain-name>
```

For example:

```
test-cluster.example.com
```

For assistance in configuring the DNS server, check Appendix section for:

- Creating DNS Records with Bind (Option 1)
- Creating DNS Records with dnsmasq (Option 2)

*Reserving IP Addresses for Nodes with the DHCP Server*

For the `baremetal` network, a network administrator must reserve a number of IP addresses, including:

1. Three virtual IP addresses.
    - 1 IP address for the API endpoint
    - 1 IP address for the wildcard ingress endpoint
    - 1 IP address for the name server

2. One IP Address for the Provisioning node.
3. One IP address for each Control Plane (Master) node.
4. One IP address for each worker node.

The following table provides an exemplary embodiment of hostnames for each node in the OpenShift Container Platform cluster.

| Usage | Hostname | IP |
|---|---|---|
| API | *api.<cluster-name>.<domain>* | *<ip>* |
| Ingress LB (apps) | **.apps.<cluster-name>.<domain>* | *<ip>* |
| Nameserver | *ns1.<cluster-name>.<domain>* | *<ip>* |
| Provisioning node | *provisioner.<cluster-name>.<domain>* | *<ip>* |
| Master-0 | *openshift-master-0.<cluster-name>.<domain>* | *<ip>* |
| Master-1 | *openshift-master-1.<cluster-name>-.<domain>* | *<ip>* |
| Master-2 | *openshift-master-2.<cluster-name>.<domain>* | *<ip>* |
| Worker-0 | *openshift-worker-0.<cluster-name>.<domain>* | *<ip>* |
| Worker-1 | *openshift-worker-1.<cluster-name>.<domain>* | *<ip>* |
| Worker-n | *openshift-worker-n.<cluster-name>.<domain>* | *<ip>* |

For assistance in configuring the DHCP server, check Appendix section for:

- Creating DHCP reservations with dhcpd (Option 1)
- Creating DHCP reservations with dnsmasq (Option 2)

## IPv6 considerations

*SLAAC Addressing*

If you don't plan to use SLAAC [1] addresses on your OpenShift node, then it should be disabled for `baremetal` networks, that means that if your network equipment is configured to send SLAAC addresses when replying to Route Advertisements that behavior should be changed, so it only sends the route and not the SLAAC address.

You can install `ndptool` on your system in order to check what your RAs look like:

DRAFT

```
# Turn down/up baremetal iface on a master Node
$ sudo nmcli con down "Wired connection 5" && sudo nmcli con up "Wired connection 5"
Connection 'Wired connection 5' successfully deactivated (D-Bus active path:
/org/freedesktop/NetworkManager/ActiveConnection/1983)
Connection successfully activated (D-Bus active path:
/org/freedesktop/NetworkManager/ActiveConnection/2044)

# ndptool monitor on Helper node
$ sudo ndptool monitor -t ra
NDP payload len 80, from addr: fe80::c0a4:6464:bcb3:d657, iface: baremetal.153
  Type: RA
  Hop limit: 64
  Managed address configuration: yes
  Other configuration: no
  Default router preference: medium
  Router lifetime: 0s
  Reachable time: unspecified
  Retransmit time: unspecified
  Source linkaddr: 1c:40:24:1b:0c:34
  Prefix: 2620:52:0:1303::/64, valid_time: 86400s, preferred_time: 14400s, on_link:
yes, autonomous_addr_conf: no, router_addr: no
  Route: ::/0, lifetime: 0s, preference: low
```

The `ndptool monitor` should report `Managed address configuration: yes`.

*Network Ranges and Configurations*

Different `baremetal` and `provisioning` networks are required for each environment, each environment will have a different IPv6 range for each one of those networks.

In our configuration we used subinterfaces attached to two different physical interfaces, VLAN tagging was done at O.S. level (this required switch ports configured with `trunk` mode).

Our different IPv6 networks were all routable but usually, the only routable networks are the `baremetal` ones.

Keep in mind that `provisioning` networks cannot be in the same broadcast domain, since there will be services such as DHCP running.

*Route Advertisemnt*

Route Advertisement must be enabled for both networks `baremetal` and `provisioning`.

*Route Advertisements*

As mentioned previously, both the `baremetal` and the `provisioning` networks need to have Route Advertisement enabled. For the `baremetal` network, `radvd` daemon was used, while the `provisioning` network has RA enabled in the Metal³ dnsmasq, so no configuration is needed.

DRAFT

## 2.3. Configuring nodes

Each node in the cluster requires the following configuration for proper installation.

A mismatch between nodes will cause an installation failure.

While the cluster nodes can contain more than two NICs, the installation process only focuses on the first two NICs:

| NIC | Network | VLAN |
|---|---|---|
| NIC1 | `provisioning` | <provisioning-vlan> |
| NIC2 | `baremetal` | <baremetal-vlan> |

The RHEL 8.1 installation process on the Provisioning node may vary. For this procedure, NIC2 is PXE-enabled to ensure easy installation using a local Satellite server.

NIC1 is a non-routable network (`provisioning`) that is only used for the installation of the OpenShift Container Platform cluster.

| PXE | Boot order |
|---|---|
| NIC1 PXE-enabled (provisioning network) | 1 |
| NIC2 PXE-enabled (baremetal network) | 2 |

Ensure PXE is disabled on all other NICs.

Configure the control plane (master) and worker nodes as follows:

| PXE | Boot order |
|---|---|
| NIC1 PXE-enabled (provisioning network) | 1 |

## 2.4. Out-of-band management

Nodes will typically have an additional NIC used by the Baseboard Management Controllers (BMCs). These BMCs must be accessible from the provisioning node.

Each node must be accessible via out-of-band management. The provisioning node requires access to the out-of-band management network for a successful OpenShift Container Platform 4 installation.

The out-of-band management setup is out of scope for this document. We recommend setting up a separate management network for out-of-band management. However, using the `provisioning` network or the `baremetal` network are valid options.

## 2.5. Required data for installation

Prior to the installation of the OpenShift Container Platform cluster, gather the following information from all cluster nodes:

- Out-of-band management IP and credentials
  - Examples
    - Dell (iDRAC) IP
    - HP (iLO) IP
- NIC1 (`provisioning`) MAC address

## 2.6. Validation checklist for nodes

- ☐ NIC1 VLAN is configured on the provisioning network.
- ☐ NIC2 VLAN is configured on the baremetal network.
- ☐ NIC1 is PXE-enabled on the provisioning network for the control plane (master) and worker nodes.
- ☐ (Optional) NIC2 is PXE-enabled on **only** the provisioning node for easy deployment of RHEL operating system via a local Satellite server or PXE server
- ☐ Aside from provisioning network NIC1, PXE has been disabled on all other NICs for all OpenShift Container Platform nodes
- ☐ DNS records configured for the OpenShift Container Platform nodes using a subdomain or subzone and cluster name i.e. `<cluster-name>.<domain-name>`
- ☐ DHCP reservations for the OpenShift Container Platform nodes, API, ingress endpoint, and nameserver.
- ☐ A separate out-of-band management network has been created.
- ☐ All nodes accessible via out-of-band management network.
- ☐ Required data for installation (out-of-band management IP and credentials, provisioning NIC1 MAC address for all OpenShift Container Platform nodes.

After an environment has been prepared according to the documented prerequisites, the installation process is the same as other IPI-based platforms.

[1] Stateless Address AutoConfiguration

DRAFT

# Chapter 3. Installing RHEL on the provision node

With the networking portions complete, the next step in installing the OpenShift Container Platform cluster is to install RHEL 8 on the provision node. This node will be used as the orchestrator in installing the OCP cluster on the three master and two worker nodes. For the purposes of this document, installing RHEL on the provision node is out of scope. However, options include, but are not limited to, using a RHEL Satellite server, PXE, or installation media.

# Chapter 4. Preparing the provisioner node for OpenShift Container Platform installation

Perform the following steps to prepare the environment.

*Procedure*

1. Log in to the provisioner node via `ssh`.
2. Create a user (for example, `kni`) to deploy as non-root and provide that user `sudo` privileges.

```
[root@provisioner ~]# useradd kni
[root@provisioner ~]# passwd kni
[root@provisioner ~]# echo "kni ALL=(root) NOPASSWD:ALL" | tee -a
/etc/sudoers.d/kni
[root@provisioner ~]# chmod 0440 /etc/sudoers.d/kni
```

3. Create an `ssh` key for the new user.

```
[root@provisioner ~]# su - kni -c "ssh-keygen -t rsa -f /home/kni/.ssh/id_rsa -N
''"
```

4. Log in in as the new user on the provision node.

```
[root@provisioner ~]# su - kni
[kni@provisioner ~]$
```

5. Use Red Hat Subscription Manager to register your environment.

```
[kni@provisioner ~]$ sudo subscription-manager register --username=<user>
--password=<pass> --auto-attach
[kni@provisioner ~]$ sudo subscription-manager repos --enable=rhel-8-for-x86_64-
appstream-rpms --enable=rhel-8-for-x86_64-baseos-rpms
```

For more information about Red Hat Subscription Manager, see Using and Configuring Red Hat Subscription Manager.

6. Install the following packages.

```
[kni@provisioner ~]$ sudo dnf install -y libvirt qemu-kvm mkisofs python3-devel jq
ipmitool tar
```

7. Modify the user to add the `libvirt` group to the newly created user.

```
[kni@provisioner ~]$ sudo usermod --append --groups libvirt <user>
```

8. Restart `firewalld` and enable the `http` service.

```
[kni@provisioner ~]$ sudo systemctl start firewalld
[kni@provisioner ~]$ sudo firewall-cmd --zone=public --add-service=http --permanent
[kni@provisioner ~]$ sudo firewall-cmd --reload
```

9. Start and enable the `libvirtd` service.

```
[kni@provisioner ~]$ sudo systemctl start libvirtd
[kni@provisioner ~]$ sudo systemctl enable libvirtd --now
```

10. Create the `default` storage pool and start it.

```
[kni@provisioner ~]$ sudo virsh pool-define-as --name default --type dir --target
/var/lib/libvirt/images
[kni@provisioner ~]$ sudo virsh pool-start default
[kni@provisioner ~]$ sudo virsh pool-autostart default
```

11. Configure network bridges for both `provisioning` and `baremetal` networks.

The `provisioning` bridge can be setup using an IPv4 address or an IPv6 address. Either option is supported and the selection should be based upon your environmental requirements.

```
[kni@provisioner ~]$ export PUB_CONN=<baremetal_nic_name>
[kni@provisioner ~]$ export PROV_CONN=<prov_nic_name>
[kni@provisioner ~]$ sudo nohup bash -c '
    nmcli con down "$PUB_CONN"
    nmcli con delete "$PUB_CONN"
    # RHEL 8.1 appends the word "System" in front of the connection, delete in case
it exists
    nmcli con down "System $PUB_CONN"
    nmcli con delete "System $PUB_CONN"
    nmcli connection add ifname baremetal type bridge con-name baremetal
    nmcli con add type bridge-slave ifname "$PUB_CONN" master baremetal
    nmcli con down baremetal
    nmcli con up baremetal
'
```

The `ssh` connection may disconnect after executing this step. You will want to have some sort of out-of-band connection to your host (eg., a serial console, local keyboard, or dedicated management interface) in the event that something goes wrong while executing these commands.

*Provisioning Network (IPv4 address)*

```
[kni@provisioner ~]$ sudo nohup bash -c '
    nmcli con down "$PROV_CONN"
    nmcli con delete "$PROV_CONN"
    # RHEL 8.1 appends the word "System" in front of the connection, delete in case
it exists
    nmcli con down "System $PROV_CONN"
    nmcli con delete "System $PROV_CONN"
    nmcli connection add ifname provisioning type bridge con-name provisioning
    nmcli con add type bridge-slave ifname "$PROV_CONN" master provisioning
    nmcli connection modify provisioning ipv4.addresses 172.22.0.1/24 ipv4.method
manual
    nmcli con down provisioning
    nmcli con up provisioning
'
```

The IPv4 address may be any address as long as it is not routable via the `baremetal` network.

*Provisioning Network (IPv6 address)*

```
[kni@provisioner ~]$ sudo nohup bash -c '
    nmcli con down "$PROV_CONN"
    nmcli con delete "$PROV_CONN"
    # RHEL 8.1 appends the word "System" in front of the connection, delete in case
it exists
    nmcli con down "System $PROV_CONN"
    nmcli con delete "System $PROV_CONN"
    nmcli connection add ifname provisioning type bridge con-name provisioning
    nmcli con add type bridge-slave ifname "$PROV_CONN" master provisioning
    nmcli connection modify provisioning ipv6.addresses fd00:1101::1/64 ipv6.method
manual
    nmcli con down provisioning
    nmcli con up provisioning
'
```

The IPv6 address may be any address as long as it is not routable via the `baremetal` network.

Ensure that UEFI is enabled and UEFI PXE settings are set to the IPv6 protocol when using IPv6 addressing.

12. `ssh` back into the `provisioner` node (if required).

```
# ssh provisioner.<cluster-name>.<domain>
```

13. Verify the connection bridges have been properly created.

```
[kni@provisioner ~]$ sudo nmcli con show
```

```
NAME               UUID                                  TYPE      DEVICE
baremetal          4d5133a5-8351-4bb9-bfd4-3af264801530  bridge    baremetal
provisioning       43942805-017f-4d7d-a2c2-7cb3324482ed  bridge    provisioning
virbr0             d9bca40f-eee1-410b-8879-a2d4bb0465e7  bridge    virbr0
bridge-slave-eno1  76a8ed50-c7e5-4999-b4f6-6d9014dd0812  ethernet  eno1
bridge-slave-eno2  f31c3353-54b7-48de-893a-02d2b34c4736  ethernet  eno2
```

14. Create a `pull-secret.txt` file.

```
[kni@provisioner ~]$ vim pull-secret.txt
```

In a web browser, navigate to Install on Bare Metal with user-provisioned infrastructure, and scroll down to the **Downloads** section. Click **Copy pull secret**. Paste the contents into the `pull-secret.txt` file and save the contents in the `kni` user's home directory.

DRAFT

# Chapter 5. Retrieving OpenShift Installer

The following sections describe how to properly retrieve and extract the OpenShift Container Platform for either upstream or downstream. Choose the appropriate Installer for your use case.

- Development version
- GA version

## 5.1. Select Development version of installer

You can choose from the following approaches:

- Choose a successfully deployed release that passed CI
- Deploy the latest development version

### 5.1.1. Select an OpenShift installer release from CI (Development)

*Procedure*

1. Go to Release Status and choose a release that has passed the tests for metal.
2. Verify that the release is available in the OpenShift mirror Index of /pub/openshift-v4/clients/ocp-dev-preview.
3. Save the release name. For example, `4.6.0-0.nightly-2019-12-09-035405`.
4. Configure VARS.

```
export VERSION="4.6.0-0.nightly-2019-12-09-035405"
export RELEASE_IMAGE=$(curl -s https://mirror.openshift.com/pub/openshift-
v4/clients/ocp-dev-preview/$VERSION/release.txt
| grep 'Pull From: quay.io' | awk -F ' ' '{print $3}' )
```

### 5.1.2. Retrieving the latest OpenShift installer (Development)

*Procedure*

Export the following variables `VERSION` and `RELEASE_IMAGE`

```
export VERSION=$(curl -s https://mirror.openshift.com/pub/openshift-v4/clients/ocp-
dev-preview/latest/release.txt
| grep 'Name:' | awk -F: '{print $2}')
export RELEASE_IMAGE=$(curl -s https://mirror.openshift.com/pub/openshift-
v4/clients/ocp-dev-preview/latest/release.txt
| grep 'Pull From: quay.io' | awk -F ' ' '{print $3}')
```

### 5.1.3. Extracting the OpenShift Container Platform installer (Development)

*Procedure*

After choosing the installer, the next step is to extract it.

```
export cmd=openshift-baremetal-install
export pullsecret_file=~/pull-secret.txt
export extract_dir=$(pwd)
# Get the oc binary
curl -s https://mirror.openshift.com/pub/openshift-v4/clients/ocp-dev-preview/
$VERSION/openshift-client-linux-$VERSION.tar.gz | tar zxvf - oc
sudo cp oc /usr/local/bin
# Extract the baremetal installer
oc adm release extract --registry-config "${pullsecret_file}" --command=$cmd --to "
${extract_dir}" ${RELEASE_IMAGE}
sudo cp ./openshift-baremetal-install /usr/local/bin/
```

## 5.2. Retrieving OpenShift Installer GA

### 5.2.1. Retrieving the OpenShift Container Platform installer (GA Release)

The latest-4.x can be used to deploy the latest Generally Available version of OpenShift Container Platform:

```
[kni@provisioner ~]$ export VERSION=latest-4.6
export RELEASE_IMAGE=$(curl -s https://mirror.openshift.com/pub/openshift-
v4/clients/ocp/$VERSION/release.txt | grep 'Pull From: quay.io' | awk -F ' ' '{print
$3}')
```

### 5.2.2. Extracting the OpenShift Container Platform installer (GA Release)

After retrieving the installer, the next step is to extract it.

*Procedure*

1. Set the environment variables:

```
[kni@provisioner ~]$ export cmd=openshift-baremetal-install
[kni@provisioner ~]$ export pullsecret_file=~/pull-secret.txt
[kni@provisioner ~]$ export extract_dir=$(pwd)
```

2. Get the `oc` binary:

```
[kni@provisioner ~]$ curl -s https://mirror.openshift.com/pub/openshift-
v4/clients/ocp/$VERSION/openshift-client-linux.tar.gz | tar zxvf - oc
```

3. Extract the installer:

```
[kni@provisioner ~]$ sudo cp oc /usr/local/bin
[kni@provisioner ~]$ oc adm release extract --registry-config "${pullsecret_file}"
--command=$cmd --to "${extract_dir}" ${RELEASE_IMAGE}
[kni@provisioner ~]$ sudo cp openshift-baremetal-install /usr/local/bin
```

# Chapter 6. Creating an RHCOS images cache (Optional)

To employ image caching, you must download two images: the RHCOS image used by the bootstrap VM and the RHCOS image used by the installer to provision the different nodes. Image caching is optional, but especially useful when running the installer on a network with limited bandwidth.

If you are running the installer on a network with limited bandwidth and the RHCOS images download takes more than 15 to 20 minutes, the installer will timeout. Caching images on a web server will help in such scenarios.

Use the following steps to install a container that contains the images.

1. Install `podman` and `policycoreutils-python-utils`.

```
[kni@provisioner ~]$ sudo dnf install -y podman policycoreutils-python-utils
```

2. Open firewall port `8080` to be used for RHCOS Image caching.

```
[kni@provisioner ~]$ sudo firewall-cmd --add-port=8080/tcp --zone=public
--permanent
```

3. Create a directory to store the `bootstraposimage` and `clusterosimage`.

```
[kni@provisioner ~]$ mkdir /home/kni/rhcos_image_cache
```

4. Set the appropriate SELinux context for the newly created directory.

```
[kni@provisioner ~]$ sudo semanage fcontext -a -t httpd_sys_content_t
"/home/kni/rhcos_image_cache(/.*)?"
[kni@provisioner ~]$ sudo restorecon -Rv rhcos_image_cache/
```

5. Get the commit ID from the installer. The ID determines which images the installer needs to download.

```
[kni@provisioner ~]$ export COMMIT_ID=$(/usr/local/bin/openshift-baremetal-install
version | grep '^built from commit' | awk '{print $4}')
```

6. Get the URI for the RHCOS image that the installer will deploy on the nodes.

```
[kni@provisioner ~]$ export RHCOS_OPENSTACK_URI=$(curl -s -S
https://raw.githubusercontent.com/openshift/installer/$COMMIT_ID/data/data/rhcos.js
on  | jq .images.openstack.path | sed 's/"//g')
```

7. Get the URI for the RHCOS image that the installer will deploy on the bootstrap VM.

```
[kni@provisioner ~]$ export RHCOS_QEMU_URI=$(curl -s -S
https://raw.githubusercontent.com/openshift/installer/$COMMIT_ID/data/data/rhcos.js
on  | jq .images.qemu.path | sed 's/"//g')
```

8. Get the path where the images are published.

```
[kni@provisioner ~]$ export RHCOS_PATH=$(curl -s -S
https://raw.githubusercontent.com/openshift/installer/$COMMIT_ID/data/data/rhcos.js
on | jq .baseURI | sed 's/"//g')
```

9. Get the SHA hash for the RHCOS image that will be deployed on the bootstrap VM.

```
[kni@provisioner ~]$ export RHCOS_QEMU_SHA_UNCOMPRESSED=$(curl -s -S
https://raw.githubusercontent.com/openshift/installer/$COMMIT_ID/data/data/rhcos.js
on  | jq -r '.images.qemu["uncompressed-sha256"]')
```

10. Get the SHA hash for the RHCOS image that will be deployed on the nodes.

```
[kni@provisioner ~]$ export RHCOS_OPENSTACK_SHA_COMPRESSED=$(curl -s -S
https://raw.githubusercontent.com/openshift/installer/$COMMIT_ID/data/data/rhcos.js
on  | jq -r '.images.openstack.sha256')
```

11. Download the images and place them in the `/home/kni/rhcos_image_cache` directory.

```
[kni@provisioner ~]$ curl -L ${RHCOS_PATH}${RHCOS_QEMU_URI} -o
/home/kni/rhcos_image_cache
[kni@provisioner ~]$ curl -L ${RHCOS_PATH}${RHCOS_OPENSTACK_URI} -o
/home/kni/rhcos_image_cache
```

12. Confirm SELinux type is of `httpd_sys_content_t` for the newly created files.

```
[kni@provisioner ~]$ ls -Z /home/kni/rhcos_image_cache
```

13. Create the pod.

```
[kni@provisioner ~]$ podman run -d --name rhcos_image_cache \
-v /home/kni/rhcos_image_cache:/var/www/html \
-p 8080:8080/tcp \
registry.centos.org/centos/httpd-24-centos7:latest
```

14. The above command creates a caching webserver with the name `rhcos_image_cache` which will be serving the images for deployment. The first image `${RHCOS_PATH}${RHCOS_QEMU_URI}?sha256=${RHCOS_QEMU_SHA_UNCOMPRESSED}` will be used as `bootstrapOSImage` and the second image `${RHCOS_PATH}${RHCOS_OPENSTACK_URI}?sha256=${RHCOS_OPENSTACK_SHA_COMPRESSED}` will be used as `clusterOSImage` in `install-config.yaml` file as shown in Additional install-config parameters section.

# Chapter 7. Configuration Files

In this section of the document, we'll be covering the set-up of the different configuration files

## 7.1. Configuring the `install-config.yaml` file

The `install-config.yaml` file requires some additional details. Most of the information is teaching the installer and the resulting cluster enough about the available hardware so that it is able to fully manage it.

1. Configure `install-config.yaml`. Change the appropriate variables to match your environment, including `pullSecret` and `sshKey`.

```
apiVersion: v1
<a anchor="basedomain">basedomain</a>: &lt;domain&gt;
metadata:
  <a anchor="metadataname">name</a>: &lt;cluster-name&gt;
networking:
  <a anchor="machinecidr">machineCIDR</a>: &lt;public-cidr&gt;
  networkType: OVNKubernetes
compute:
- <a anchor="workername">name</a>: worker
  <a anchor="computereplicas">replicas</a>: 2
controlPlane:
  <a anchor="controlplanename">name</a>: master
  <a anchor="controlplanereplicas">replicas</a>: 3
  platform:
    baremetal: {}
platform:
  baremetal:
    <a anchor="apivip">apiVIP</a>: &lt;api-ip&gt;
    <a anchor="ingressvip">ingressVIP</a>: &lt;wildcard-ip&gt;
    <a anchor="provisioningNetworkInterface">provisioningNetworkInterface</a>:
&lt;NIC1&gt;
    <a anchor="provisioningNetworkCIDR">provisioningNetworkCIDR</a>: &lt;CIDR&gt;
    <a anchor="hoststable">hosts</a>:
      - <a anchor="name">name</a>: openshift-master-0
        <a anchor="role">role</a>: master
        <a anchor="bmcadressing">bmc</a>:
          address: ipmi://&lt;out-of-band-ip&gt; ①
          username: &lt;user&gt;
          password: &lt;password&gt;
        <a anchor="bootMACAddress">bootMACAddress</a>: &lt;NIC1-mac-address&gt;
        <a anchor="hardwareProfile">hardwareProfile</a>: default
      - <a anchor="name">name</a>: openshift-master-1
        <a anchor="role">role</a>: master
        <a anchor="bmcadressing">bmc</a>:
          address: ipmi://&lt;out-of-band-ip&gt;
          username: &lt;user&gt;
```

```
          password: &lt;password&gt;
        <a anchor="bootMACAddress">bootMACAddress</a>: &lt;NIC1-mac-address&gt;
        <a anchor="hardwareProfile">hardwareProfile</a>: default
      - <a anchor="name">name</a>: openshift-master-2
        <a anchor="role">role</a>: master
        <a anchor="bmcadressing">bmc</a>:
          address: ipmi://<out-of-band-ip>
          username: &lt;user&gt;
          password: &lt;password&gt;
        <a anchor="bootMACAddress">bootMACAddress</a>: &lt;NIC1-mac-address&gt;
        <a anchor="hardwareProfile">hardwareProfile</a>: default
      - <a anchor="name">name</a>: openshift-worker-0
        <a anchor="role">role</a>: worker
        <a anchor="bmcadressing">bmc</a>:
          address: ipmi://&lt;out-of-band-ip&gt;
          username: &lt;user&gt;
          password: &lt;password&gt;
        <a anchor="bootMACAddress">bootMACAddress</a>: &lt;NIC1-mac-address&gt;
        <a anchor="hardwareProfile">hardwareProfile</a>: unknown
      - <a anchor="name">name</a>: openshift-worker-1
        <a anchor="role">role</a>: worker
        <a anchor="bmcadressing">bmc</a>:
          address: ipmi://&lt;out-of-band-ip&gt;
          username: &lt;user&gt;
          password: &lt;password&gt;
        <a anchor="bootMACAddress">bootMACAddress</a>: &lt;NIC1-mac-address&gt;
        <a anchor="hardwareProfile">hardwareProfile</a>: unknown
pullSecret: '&lt;pull_secret&gt;'
sshKey: '&lt;ssh_pub_key&gt;'
```

① Refer to the BMC adressing for more options

2. Create a directory to store cluster configs.

```
[kni@provisioner ~]$ mkdir ~/clusterconfigs
[kni@provisioner ~]$ cp install-config.yaml ~/clusterconfigs
```

## 7.2. Additional `install-config` parameters

This topic describes the required parameters, the `hosts` parameter, and the `bmc address` parameter for the `install-config.yaml` file.

*Table 1. Required parameters*

| Parameters | Default | Description |
|---|---|---|
| `baseDomain` | | The domain name for your cluster, e.g. `example.com` |

| Parameters | Default | Description |
| --- | --- | --- |
| `metadata:`<br>`  name:` | | The name to be given to your OpenShift Container Platform cluster. e.g. `openshift` |
| `networking:`<br>`  machineCIDR:` | | The public CIDR (Classless Inter-Domain Routing) of your external network, e.g. `10.0.0.0/24` or `2620:52:0:1302::/64`. |
| `compute:`<br>`  - name: worker` | | The OpenShift Container Platform cluster requires a name be provided for compute nodes even if no compute nodes are to be used. |
| `compute:`<br>`  replicas: 2` | | Replicas sets the number of compute nodes that are to be included as part of your OpenShift Container Platform cluster. |
| `controlPlane:`<br>`  name: master` | | The OpenShift Container Platform cluster requires a name be provided for control plane (master) nodes. |
| `controlPlane:`<br>`  replicas: 3` | | Replicas sets the number of control plane (master) nodes that are to be included as part of your OpenShift Container Platform cluster. |
| `provisioningNetworkInterface` | | The name of the network interface on control plane nodes connected to the provisioning network. (OpenShift Container Platform 4.4 only) |
| `hosts` | | Details about bare metal hosts to use to build the cluster. |
| `defaultMachinePlatform` | | The default configuration used for machine pools without a platform configuration. |

| Parameters | Default | Description |
|---|---|---|
| apiVIP | api.<clustername.clusterdomain> | The VIP to use for internal API communication.<br><br>This setting must either be provided or pre-configured in DNS so that the default name resolve correctly. |
| disableCertificateVerification | False | This parameter is needed when we use redfish or redfish-virtualmedia to manage BMC addresses. The value should be True if you are using self-signed certificate for BMC addresses. |
| ingressVIP | test.apps.<clustername.clusterdomain> | The VIP to use for ingress traffic. |

*Table 2. Optional Parameters*

| Parameters | Default | Description |
|---|---|---|
| provisioningDHCPExternal | false | Defines if external DHCP will be used or the one configured by installer |
| provisioningDHCPRange | 172.22.0.10,172.22.0.100 | Defines the IP range to use for hosts on the provisioning network. |
| provisioningNetworkCIDR | 172.22.0.0/24 | The CIDR for the network to use for provisioning. This option is required when using IPv6 addressing on the provision network. |
| clusterProvisioningIP | 3rd IP of provisioningNetworkCIDR | The IP within the cluster where the provisioning services run. Defaults to the 3rd IP of the provision subnet, e.g. 172.22.0.3 |
| bootstrapProvisioningIP | 2nd IP of provisioningNetworkCIDR | The IP on the bootstrap VM where the provisioning services run while the control plane is being deployed. Defaults to the 2nd IP of the provision subnet, e.g. 172.22.0.2 or 2620:52:0:1307::2. |
| externalBridge | baremetal | The name of the baremetal bridge of the hypervisor attached to the baremetal network. |
| provisioningBridge | provisioning | The name of the provisioning bridge on the provision host attached to the provisioning network. |
| defaultMachinePlatform | | The default configuration used for machine pools without a platform configuration. |

| Parameters | Default | Description |
| --- | --- | --- |
| `bootstrapOSImage` | | A URL to override the default operating system image for the bootstrap node. The URL must contain a sha256 hash of the image. Example <a href="https://mirror.openshift.com/rhcos-<version>-qemu.qcow2.gz?sha256=<uncompressed_sha256&gt" class="bare">https://mirror.openshift.com/rhcos-<version>-qemu.qcow2.gz?sha256=<uncompressed_sha256&gt</a>; or <code>http://[2620:52:0:1307::1]/rhcos-<version>-qemu.x86_64.qcow2.gz?sha256=<uncompressed_sha256></code>. |
| `clusterOSImage` | | A URL to override the default operating system for cluster nodes. The URL must include a sha256 hash of the image. Example <a href="https://mirror.openshift.com/images/rhcos-<version>-openstack.qcow2.gz?sha256=<compressed_sha256&gt" class="bare">https://mirror.openshift.com/images/rhcos-<version>-openstack.qcow2.gz?sha256=<compressed_sha256&gt</a>; |

*Hosts*

The `hosts` parameter is a list of separate bare metal assets that should be used to build the cluster.

| Name | Default | Description |
| --- | --- | --- |
| `name` | | The name of the BareMetalHost resource to associate with the details. e.g. `openshift-master-0` |
| `role` | | Either `master` or `worker`. |
| `bmc` | | Connection details for the baseboard management controller. See below for details. |
| `bootMACAddress` | | The MAC address of the NIC the host will use to boot on the provisioning network. |

| | | |
|---|---|---|
| `hardwareProfile` | `default` | This parameter exposes the device name that the installer attempts to deploy the OpenShift Container Platform for the master and worker nodes. The value defaults to `default` for masters and `unknown` for workers. The list of profiles includes: `default`, `libvirt`, `dell`, `dell-raid`, `openstack`. The `default` parameter attempts to install on `/dev/sda` of your cluster node. |

The `bmc` parameter for each host is a set of values for accessing the baseboard management controller in the host.

| Name | Default | Description |
|---|---|---|
| `username` | | The username for authenticating to the BMC. |
| `password` | | The password associated with `username`. |
| `address` | | The URL for communicating with the BMC controller, based on the provider being used. See below for details. See BMC Addressing for details. |

### 7.2.1. BMC addressing

The `address` field for each `bmc` entry is a URL with details how to connect to the OpenShift Container Platform cluster nodes, including the type of controller in the URL scheme and its location on the network.

*IPMI*

IPMI hosts use `ipmi://<out-of-band-ip>:<port>` and defaults to port `623` if not specified. Example output of using IPMI within your `install-config.yaml` file.

DRAFT

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: ipmi://<out-of-band-ip>
          username: <user>
          password: <password>
```

*RedFish*

For RedFish, use `redfish://` (or `redfish+http://` to disable TLS). The hostname (or IP address) and the path to the system ID are both required. Example output of using RedFish within your `install-config.yaml` file.

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: redfish://<out-of-band-ip>/redfish/v1/Systems/1
          username: <user>
          password: <password>
```

While it is recommended to have a certificate of authority for your out of band management addresses, if using self-signed certificates ensure to include an additional parameter of `disableCertificateVerification: True`. Example output of using RedFish with `disableCertificateVerification: True` within your `install-config.yaml` file.

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: redfish://<out-of-band-ip>/redfish/v1/Systems/1
          username: <user>
          password: <password>
          disableCertificateVerification: True
```

Currently RedFish is only supported on HPE hardware and Dell with iDRAC firmware version `4.20.20.20` or higher for IPI on Bare metal deployments. We are working with other vendors to enable RedFish capabilities across the board.

*RedFish Virtual Media*

For RedFish virtual media, use `redfish-virtualmedia://`

Example output of using RedFish Virtual Media within your `install-config.yaml` file.

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: redfish-virtualmedia://<out-of-band-ip>/redfish/v1/Systems/1
          username: <user>
          password: <password>
```

Currently RedFish is only supported on HPE hardware for IPI on Bare metal deployments. We are working with other vendors to enable RedFish capabilities across the board.

## 7.3. Creating the OpenShift Container Platform manifests

1. Create the OpenShift Container Platform manifests.

```
[kni@provisioner ~]$ ./openshift-baremetal-install --dir ~/clusterconfigs create
manifests
```

```
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true for
Scheduler cluster settings
WARNING Discarding the Openshift Manifest that was provided in the target directory
because its dependencies are dirty and it needs to be regenerated
```

# Chapter 8. Creating a disconnected registry (optional)

In some cases, you may want to install an Openshift KNI cluster using a local copy of the installation registry. This could be for enhancing network efficiency because the cluster nodes are on a network that does not have access to the internet.

A local, or mirrored, copy of the registry requires the following:

- A certificate for the registry node. This can be a self-signed certificate.
- A webserver - this will be served by a container on a system.
- An updated pull secret that contains the certificate and local repository information.

Creating a disconnected registry on a registry node is optional. The subsequent sections indicate that they are optional since they are steps you need to execute only when creating a disconnected registry on a registry node. You should execute all of the subsequent sub-sections labeled "(optional)" when creating a disconnected registry on a registry node.

## 8.1. Preparing the registry node to host the mirrored registry (optional)

Make the following changes to the registry node.

*Procedure*

1. Open the firewall port on the registry node.

```
[user@registry ~]$ sudo firewall-cmd --add-port=5000/tcp --zone=libvirt
--permanent
[user@registry ~]$ sudo firewall-cmd --add-port=5000/tcp --zone=public
--permanent
[user@registry ~]$ sudo firewall-cmd --reload
```

2. Install the required packages for the registry node.

```
[user@registry ~]$ sudo yum -y install python3 podman httpd httpd-tools jq
```

3. Create the directory structure where the repository information will be held.

```
[user@registry ~]$ sudo mkdir -p /opt/registry/{auth,certs,data}
```

## 8.2. Generating the self-signed certificate (optional)

Generate a self-signed certificate for the registry node and put it in the `/opt/registry/certs` directory.

*Procedure*

1. Adjust the certificate information as appropriate.

```
[user@registry ~]$ host_fqdn=$( hostname --long )
[user@registry ~]$ cert_c="<Country Name>"   # Certificate Country Name (C)
[user@registry ~]$ cert_s="<State>"          # Certificate State (S)
[user@registry ~]$ cert_l="<Locality>"       # Certificate Locality (L)
[user@registry ~]$ cert_o="<Organization>"   # Certificate Organization (O)
[user@registry ~]$ cert_ou="<Org Unit>"      # Certificate Organizational Unit (OU)
[user@registry ~]$ cert_cn="${host_fqdn}"    # Certificate Common Name (CN)

[user@registry ~]$ openssl req \
    -newkey rsa:4096 \
    -nodes \
    -sha256 \
    -keyout /opt/registry/certs/domain.key \
    -x509 \
    -days 365 \
    -out /opt/registry/certs/domain.crt \
    -subj "/C=${cert_c}/ST=${cert_s}/L=${cert_l}/O=${cert_o}/OU=${cert_ou}/CN=
${cert_cn}"
```

When replacing `<Country Name>`, ensure it only contains two letters. For example, `US`.

2. Update the registry node's `ca-trust` with the new certificate.

```
[user@registry ~]$ sudo cp /opt/registry/certs/domain.crt /etc/pki/ca-
trust/source/anchors/
[user@registry ~]$ sudo update-ca-trust extract
```

## 8.3. Creating the registry podman container (optional)

The registry container uses the `/opt/registry` directory for certificates, authentication files, and to store its data files.

The registry container uses `httpd` and needs an `htpasswd` file for authentication.

*Procedure*

1. Create an `htpasswd` file in `/opt/registry/auth` for the container to use.

```
[user@registry ~]$ htpasswd -bBc /opt/registry/auth/htpasswd <user> <passwd>
```

Replace `<user>` with the user name and `<passwd>` with the password.

2. Create and start the registry container.

```
[user@registry ~]$ podman create \
  --name ocpdiscon-registry \
  -p 5000:5000 \
  -e "REGISTRY_AUTH=htpasswd" \
  -e "REGISTRY_AUTH_HTPASSWD_REALM=Registry" \
  -e "REGISTRY_HTTP_SECRET=ALongRandomSecretForRegistry" \
  -e "REGISTRY_AUTH_HTPASSWD_PATH=/auth/htpasswd" \
  -e "REGISTRY_HTTP_TLS_CERTIFICATE=/certs/domain.crt" \
  -e "REGISTRY_HTTP_TLS_KEY=/certs/domain.key" \
  -v /opt/registry/data:/var/lib/registry:z \
  -v /opt/registry/auth:/auth:z \
  -v /opt/registry/certs:/certs:z \
  docker.io/library/registry:2
```

```
[user@registry ~]$ podman start ocpdiscon-registry
```

## 8.4. Copy and update the pull-secret (optional)

Copy the pull secret file from the provisioner node to the registry node and modify it to include the authentication information for the new registry node.

*Procedure*

1. Copy the `pull-secret.txt` file.

```
[user@registry ~]$ scp kni@provisioner:/home/kni/pull-secret.txt pull-secret.txt
```

2. Update the `host_fqdn` environment variable with the fully qualified domain name of the registry node.

```
[user@registry ~]$ host_fqdn=$( hostname --long )
```

3. Update the `b64auth` environment variable with the base64 encoding of the `http` credentials used to create the `htpasswd` file.

```
[user@registry ~]$ b64auth=$( echo -n '<username>:<passwd>' | openssl base64 )
```

Replace <username> with the user name and <passwd> with the password.

4. Set the AUTHSTRING environment variable to use the base64 authorization string. The $USER variable is an environment variable containing the name of the current user.

```
[user@registry ~]$ AUTHSTRING="{\"$host_fqdn:5000\": {\"auth\": \"$b64auth\",
\"email\": \"$USER@redhat.com\"}}"
```

5. Update the pull-secret file.

```
[user@registry ~]$ jq ".auths += $AUTHSTRING" < pull-secret.json > pull-secret-
update.json
```

## 8.5. Mirroring the repository (optional)

*Procedure*

1. Copy the oc binary from the provisioner node to the registry node.

```
[user@registry ~]$ sudo scp kni@provisioner:/usr/local/bin/oc /usr/local/bin
```

2. Mirror the remote install images to the local repository.

```
[user@registry ~]$ /usr/local/bin/oc adm release mirror \
  -a pull-secret-update.json
  --from=$UPSTREAM_REPO \
  --to-release-image=$LOCAL_REG/$LOCAL_REPO:${VERSION} \
  --to=$LOCAL_REG/$LOCAL_REPO
```

Example output of the variables used to mirror the install images

```
UPSTREAM_REPO=${RELEASE_IMAGE}
LOCAL_REG=<registry_FQDN>:<registry_port>
LOCAL_REPO='ocp4/openshift4'
```

The values of RELEASE_IMAGE and VERSION were set during Retrieving OpenShift Installer

## 8.6. Modify the install-config.yaml file to use the disconnected registry (optional)

On the provisioner node, the install-config.yaml file should use the newly created pull-secret from

the `pull-secret-update.json` file. The `install-config.yaml` file must also contain the disconnected registry node's certificate and registry information.

*Procedure*

1. Add the disconnected registry node's certificate to the `install-config.yaml` file. The certificate should follow the `"additionalTrustBundle: |"` line and be properly indented, usually by two spaces.

```
[kni@provisioner ~]$ scp user@registry.example.com:/opt/registry/certs/domain.crt
/home/kni
[kni@provisioner ~]$ echo "additionalTrustBundle: |" >> install-config.yaml
[kni@provisioner ~]$ sed -e 's/^/  /' /home/kni/domain.crt >> install-config.yaml
```

2. Add the mirror information for the registry to the `install-config.yaml` file.

```
[kni@provisioner ~]$ echo "imageContentSources:" >> install-config.yaml
[kni@provisioner ~]$ echo "- mirrors:" >> install-config.yaml
[kni@provisioner ~]$ echo "  - registry.example.com:5000/ocp4/openshift4" >>
install-config.yaml
[kni@provisioner ~]$ echo "  source: quay.io/openshift-release-dev/ocp-v4.0-art-
dev" >> install-config.yaml
[kni@provisioner ~]$ echo "- mirrors:" >> install-config.yaml
[kni@provisioner ~]$ echo "  - registry.example.com:5000/ocp4/openshift4" >>
install-config.yaml
[kni@provisioner ~]$ echo "  source: registry.svc.ci.openshift.org/ocp/release" >>
install-config.yaml
[kni@provisioner ~]$ echo "- mirrors:" >> install-config.yaml
[kni@provisioner ~]$ echo "  - registry.example.com:5000/ocp4/openshift4" >>
install-config.yaml
[kni@provisioner ~]$ echo "  source: quay.io/openshift-release-dev/ocp-release" >>
install-config.yaml
```

Replace `registry.example.com` with your registry's fully qualified domain name.

# Chapter 9. Deploying routers on worker nodes

During the installation of an OpenShift cluster, router pods are deployed on worker nodes (the default is two router pods). In the event that an installation only has one worker node or additional routers are required in order to handle external traffic destined for services within your OpenShift cluster, a `yaml` file can be created to set the appropriate amount of router replicas.

By default two routers are deployed. If you already have two worker nodes you can skip this section. For more information on the Ingress Operator see: Ingress Operator in OpenShift Container Platform.

If you have an environment where no workers are deployed and only has master nodes, by default two routers are deployed on the master nodes. If this is the case, you can skip this section.

*Procedure*

1. Create the `router-replicas.yaml` file.

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  name: default
  namespace: openshift-ingress-operator
spec:
  replicas: <num-of-router-pods>
  endpointPublishingStrategy:
    type: HostNetwork
  nodePlacement:
    nodeSelector:
      matchLabels:
        node-role.kubernetes.io/worker: "<value>" ①
```

① Setting for workers:

- When working with just one worker node, set this value to 1.
- When working with more than 3+ workers, additional router pods (default 2) may be recommended.

1. Save and copy the `router-replicas.yaml` file to the `clusterconfigs/openshift` directory.

```
cp ~/router-replicas.yaml clusterconfigs/openshift/99_router-replicas.yaml
```

# Chapter 10. Validation checklist for installation

- ▢ OpenShift Container Platform installer has been retrieved.
- ▢ OpenShift Container Platform installer has been extracted.
- ▢ Required parameters for the `install-config.yaml` have been configured.
- ▢ The `hosts` parameter for the `install-config.yaml` has been configured.
- ▢ The `bmc` parameter for the `install-config.yaml` has been configured.
- ▢ Conventions for the values configured in the `bmc address` field have been applied.
- ▢ (optional) Validate disconnected registry settings if in use
- ▢ (optional) Deployed routers on worker nodes

# Chapter 11. Deploying the cluster via the OpenShift Container Platform installer

1. Ensure all bare metal nodes are powered off prior to installing the OpenShift Container Platform cluster.

```
[kni@provisioner ~]$ ipmitool -I lanplus -U <user> -P <password> -H <management-
server-ip> power off
```

2. Run the OpenShift Container Platform installer:

```
[kni@provisioner ~]$ ./openshift-baremetal-install --dir ~/clusterconfigs --log
-level debug create cluster
```

# Chapter 12. Following the Installation

During the deployment process, we can check its overall status by issuing the `tail` command to the log file `.openshift_install.log` which will be under the install directory folder.

```
[kni@provisioner ~]$ tail -f ~/clusterconfigs/.openshift_install.log
```

# Chapter 13. Day 2 operations

The following sections are optional, but may be of interest after the initial deployment has been completed.

## 13.1. Backing up the cluster configuration

At this point you have a working OpenShift 4 cluster on baremetal. In order to take advantage of the baremetal hardware that was the provision node, you can repurpose the provisioning node as a worker. Prior to reprovisioning the node, it is recommended to backup some existing files.

*Procedure*

1. Tar the `clusterconfig` folder and download it to your local machine.

```
tar cvfz clusterconfig.tar.gz ~/clusterconfig
```

2. Copy the Private part for the SSH Key configured on the `install-config.yaml` file to your local machine.

```
tar cvfz clusterconfigsh.tar.gz ~/.ssh/id_rsa*
```

3. Copy the `install-config.yaml` and `metal3-config.yaml` files.

```
tar cvfz yamlconfigs.tar.gz install-config.yaml metal3-config.yaml
```

## 13.2. Preparing the provisioner node to be deployed as a worker node

*Procedure*

Perform the following steps prior to converting the provisioner node to a worker node.

1. `ssh` to a system (for example, a laptop) that can access the out of band management network of the current provisioner node.
2. Copy the backups `clusterconfig.tar.gz`, `clusterconfigsh.tar.gz`, and `amlconfigs.tar.gz` to the new system.
3. Copy the `oc` binary from the existing provisioning node to the new system.
4. Make a note of the mac addresses, the baremetal network IP used for the provisioner node, and the IP address of the Out of band Management Network.
5. Reboot the system and ensure that PXE is enabled on the provisioning network and PXE is disabled for all other NICs.
6. If installation was performed using a Satellite server, remove the Host entry for the existing

provisioning node.

7. Install the `ipmitool` on the new system in order to power off the provisioner node.

### 13.2.1. Appending DNS records

#### Configuring Bind (Option 1)

*Procedure*

1. Login to the DNS server using `ssh`.
2. Suspend updates to all dynamic zones: `rndc freeze`.
3. Edit `/var/named/dynamic/example.com`.

```
$ORIGIN openshift.example.com.
<OUTPUT_OMITTED>
openshift-worker-1      A       <ip-of-worker-1>
openshift-worker-2      A       <ip-of-worker-2>
```

Remove the provisioner as it is replaced by openshift-worker-2.

4. Increase the SERIAL value by `1`.
5. Edit `/var/named/dynamic/1.0.10.in-addr.arpa`.

The filename `1.0.10.in-addr.arpa` is the reverse of the public CIDR example `10.0.1.0/24`.

6. Increase the SERIAL value by `1`.
7. Enable updates to all dynamic zones and reload them: `rndc thaw`.

#### Configuring dnsmasq (Option 2)

*Procedure*

Append the following DNS record to the `/etc/hosts` file on the server hosting the `dnsmasq` service.

```
<OUTPUT_OMITTED>
<NIC2-IP> openshift-worker-1.openshift.example.com openshift-worker-1
<NIC2-IP> openshift-worker-2.openshift.example.com openshift-worker-2
```

Remove the `provisioner.openshift.example.com` entry as it is replaced by worker-2

### 13.2.2. Appending DHCP reservations

#### Configuring dhcpd (Option 1)

*Procedure*

1. Login to the DHCP server using `ssh`.
2. Edit `/etc/dhcp/dhcpd.hosts`.

```
host openshift-worker-2 {
     option host-name "worker-2";
     hardware ethernet <NIC2-mac-address>;
     option domain-search "openshift.example.com";
     fixed-address <ip-address-of-NIC2>;
  }
```

Remove the provisioner as it is replaced by openshift-worker-2.

3. Restart the `dhcpd` service.

```
systemctl restart dhcpd
```

**Configuring dnsmasq (Option 2)**

*Procedure*

1. Append the following DHCP reservation to the `/etc/dnsmasq.d/example.dns` file on the server hosting the `dnsmasq` service.

```
<OUTPUT_OMITTED>
dhcp-host=<NIC2-mac-address>,openshift-worker-1.openshift.example.com,<ip-of-
worker-1>
dhcp-host=<NIC2-mac-address>,openshift-worker-2.openshift.example.com,<ip-of-
worker-2>
```

Remove the `provisioner.openshift.example.com` entry as it is replaced by worker-2

2. Restart the `dnsmasq` service.

```
systemctl restart dnsmasq
```

/ Module included in the following assemblies:

## 13.2.3. Deploying the provisioner node as a worker node using Metal3

After you have completed the prerequisites, perform the deployment process.

*Procedure*

1. Power off the node using `ipmitool` and confirm the provisioning node is powered off.

```
ssh <server-with-access-to-management-net>
# Use the user, password and Management net IP adddress to shutdown the system
ipmitool -I lanplus -U <user> -P <password> -H <management-server-ip> power off
# Confirm the server is powered down
ipmitool -I lanplus -U <user> -P <password> -H <management-server-ip> power status
Chassis Power is off
```

2. Get `base64` strings for the Out of band Management credentials. In this example, the user is `root` and the password is `calvin`.

```
# Use echo -ne, otherwise you will get your secrets with \n which will cause issues
# Get root username in base64
echo -ne "root" | base64
# Get root password in base64
echo -ne "calvin" | base64
```

3. Configure the BaremetalHost `bmh.yaml` file.

```
---
apiVersion: v1
kind: Secret
metadata:
  name: openshift-worker-2-bmc-secret
type: Opaque
data:
  username: ca2vdAo=
  password: MWAwTWdtdC0K
---
apiVersion: metal3.io/v1alpha1
kind: BareMetalHost
metadata:
  name: openshift-worker-2
spec:
  online: true
  bootMACAddress: <NIC1-mac-address>
  bmc:
    address: ipmi://<out-of-band-ip>
    credentialsName: openshift-worker-2-bmc-secret
```

4. Create the BaremetalHost.

```
./oc -n openshift-machine-api create -f bmh.yaml
secret/openshift-worker-2-bmc-secret created
baremetalhost.metal3.io/openshift-worker-2 created
```

5. Power up and inspect the node.

```
./oc -n openshift-machine-api get bmh openshift-worker-2

NAME                 STATUS   PROVISIONING STATUS   CONSUMER   BMC
HARDWARE PROFILE   ONLINE   ERROR
openshift-worker-2   OK       inspecting                       ipmi://<out-of-band-
ip>                      true
```

6. After finishing the inspection, the node is ready to be provisioned.

```
./oc -n openshift-machine-api get bmh openshift-worker-2

NAME                 STATUS   PROVISIONING STATUS   CONSUMER   BMC
HARDWARE PROFILE   ONLINE   ERROR
openshift-worker-2   OK       ready                            ipmi://<out-of-band-
ip>   unknown            true
```

7. Scale the workers machineset. Previously, there were two replicas during original installation.

```
./oc get machineset -n openshift-machine-api
NAME            DESIRED   CURRENT   READY   AVAILABLE   AGE
openshift-worker-2   0         0                             21h

./oc -n openshift-machine-api scale machineset openshift-worker-2 --replicas=3
```

8. The baremetal host moves to provisioning status. This can take as long as 30 minutes. You can follow the status from the node console.

```
oc -n openshift-machine-api get bmh openshift-worker-2

NAME                 STATUS   PROVISIONING STATUS   CONSUMER                  BMC
HARDWARE PROFILE   ONLINE   ERROR
openshift-worker-2   OK       provisioning          openshift-worker-0-65tjz
ipmi://<out-of-band-ip>   unknown            true
```

9. When the node is provisioned it moves to provisioned status.

```
oc -n openshift-machine-api get bmh openshift-worker-2

NAME                 STATUS   PROVISIONING STATUS   CONSUMER                  BMC
HARDWARE PROFILE   ONLINE   ERROR
openshift-worker-2   OK       provisioned           openshift-worker-2-65tjz
ipmi://<out-of-band-ip>   unknown            true
```

10. When the `kubelet` finishes initialization the node is ready for use. You can connect to the node and run `journalctl -fu kubelet` to check the process.

```
oc get node
NAME                                          STATUS   ROLES          AGE
VERSION
openshift-master-0.openshift.example.com      Ready    master         30h
v1.16.2
openshift-master-1.openshift.example.com      Ready    master         30h
v1.16.2
openshift-master-2.openshift.example.com      Ready    master         30h
v1.16.2
openshift-worker-0.openshift.example.com      Ready    worker         3m27s
v1.16.2
openshift-worker-1.openshift.example.com      Ready    worker         3m27s
v1.16.2
openshift-worker-2.openshift.example.com      Ready    worker         3m27s
v1.16.2
```

# Chapter 14. Appendix

In this section of the document, extra information is provided that is outside of the regular workflow.

## 14.1. Troubleshooting

Troubleshooting the installation is out of scope of the Deployment Guide. For more details on troubleshooting deployment, refer to our Troubleshooting guide.

## 14.2. Creating DNS Records

Two options are documented for configuring DNS records:

- On a DNS Server (Bind)
- Using dnsmasq

### 14.2.1. Configuring Bind (Option 1)

Use Option 1 if access to the appropriate DNS server for the baremetal network is accessible or a request to your network admin to create the DNS records is an option. If this is not an option, skip this section and go to section Create DNS records using dnsmasq (Option 2).

Create a subzone with the name of the cluster that is going to be used on your domain. In our example, the domain used is `example.com` and the cluster name used is `openshift`. Make sure to change these according to your environment specifics.

*Procedure*

1. Login to the DNS server using `ssh`.
2. Suspend updates to all dynamic zones: `rndc freeze`.
3. Edit `/var/named/dynamic/example.com`.

DRAFT

```
$ORIGIN openshift.example.com.
$TTL 300        ; 5 minutes
@  IN  SOA  dns1.example.com.  hostmaster.example.com. (
       2001062501  ; serial
       21600       ; refresh after 6 hours
       3600        ; retry after 1 hour
       604800      ; expire after 1 week
       86400 )     ; minimum TTL of 1 day
;
api                     A       <api-ip>
ns1                     A       <dns-vip-ip>
$ORIGIN apps.openshift.example.com.
*                       A       <wildcard-ingress-lb-ip>
$ORIGIN openshift.example.com.
provisioner             A       <NIC2-ip-of-provision>
openshift-master-0      A       <NIC2-ip-of-openshift-master-0>
openshift-master-1      A       <NIC2-ip-of-openshift-master-1>
openshift-master-2      A       <NIC2-ip-of-openshift-master-2>
openshift-worker-0      A       <NIC2-ip-of-openshift-worker-0>
openshift-worker-1      A       <NIC2-ip-of-openshift-worker-1>
```

4. Increase the `serial` value by 1.
5. Edit `/var/named/dynamic/1.0.10.in-addr.arpa`.

```
$ORIGIN 1.0.10.in-addr.arpa.
$TTL 300
@  IN  SOA  dns1.example.com.  hostmaster.example.com. (
       2001062501  ; serial
       21600       ; refresh after 6 hours
       3600        ; retry after 1 hour
       604800      ; expire after 1 week
       86400 )     ; minimum TTL of 1 day
;
126 IN      PTR      provisioner.openshift.example.com.
127 IN          PTR     openshift-master-0.openshift.example.com.
128 IN          PTR     openshift-master-1.openshift.example.com.
129 IN          PTR     openshift-master-2.openshift.example.com.
130 IN          PTR     openshift-worker-0.openshift.example.com.
131 IN          PTR     openshift-worker-1.openshift.example.com.
132 IN      PTR     api.openshift.example.com.
133 IN      PTR     ns1.openshift.example.com.
```

In this example, the IP addresses 10.0.1.126-133 are pointed to the corresponding fully qualified domain name.

The filename `1.0.10.in-addr.arpa` is the reverse of the public CIDR example `10.0.1.0/24`.

6. Increase the `serial` value by 1.
7. Enable updates to all dynamic zones and reload them: `rndc thaw`.

### 14.2.2. Configuring dnsmasq (Option 2)

To create DNS records, open the `/etc/hosts` file and add the NIC2 (baremetal net) IP followed by the hostname. In our example, the domain used is `example.com` and the cluster name used is `openshift`. Make sure to change these according to your environment specifics.

*Procedure*

1. Edit `/etc/hosts` and add the NIC2 (baremetal net) IP followed by the hostname.

```
cat /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
<NIC2-IP> provisioner.openshift.example.com provisioner
<NIC2-IP> openshift-master-0.openshift.example.com openshift-master-0
<NIC2-IP> openshift-master-1.openshift.example.com openshift-master-1
<NIC2-IP> openshift-master-2.openshift.example.com openshift-master-2
<NIC2-IP> openshift-worker-0.openshift.example.com openshift-worker-0
<NIC2-IP> openshift-worker-1.openshift.example.com openshift-worker-1
<API-IP>  api.openshift.example.com api
<DNS-VIP-IP> ns1.openshift.example.com ns1
```

2. Open the appropriate `firewalld` DNS service and reload the rules.

```
systemctl restart firewalld
firewall-cmd --add-service=dns --permanent
firewall-cmd --reload
```

## 14.3. Creating DHCP reservations

Two options are documented for configuring DHCP:

- On dhcpd (Option 1)
- Using dnsmasq (Option 2)

### 14.3.1. Configuring dhcpd (Option 1)

Use Option 1 if access to the appropriate DHCP server for the baremetal network is accessible or a request to your network admin to create the DHCP reservations is an option. If this is not an option, skip this section and go to section Create DHCP records using dnsmasq (Option 2).

1. Login to the DHCP server using `ssh`.
2. Edit `/etc/dhcp/dhcpd.hosts`.

```
host provisioner {
     option host-name "provisioner";
     hardware ethernet <mac-address-of-NIC2>;
     option domain-search "openshift.example.com";
     fixed-address <ip-address-of-NIC2>;
  }
host openshift-master-0 {
     option host-name "openshift-master-0";
     hardware ethernet <mac-address-of-NIC2>;
     option domain-search "openshift.example.com";
     fixed-address <ip-address-of-NIC2>;
  }

host openshift-master-1 {
     option host-name "openshift-master-1";
     hardware ethernet <mac-address-of-NIC2>;
     option domain-search "openshift.example.com";
     fixed-address <ip-address-of-NIC2>;
  }

host openshift-master-2 {
     option host-name "openshift-master-2";
     hardware ethernet <mac-address-of-NIC2>;
     option domain-search "openshift.example.com";
     fixed-address <ip-address-of-NIC2>;
  }
host openshift-worker-0 {
     option host-name "openshift-worker-0";
     hardware ethernet <mac-address-of-NIC2>;
     option domain-search "openshift.example.com";
     fixed-address <ip-address-of-NIC2>;
  }
host openshift-worker-1 {
     option host-name "openshift-worker-1";
     hardware ethernet <mac-address-of-NIC2>;
     option domain-search "openshift.example.com";
     fixed-address <ip-address-of-NIC2>;
  }
```

3. Restart the `dhcpd` service.

```
systemctl restart dhcpd
```

### 14.3.2. Configuring dnsmasq (Option 2)

Set up `dnsmasq` on a server that can access the baremetal network.

*Procedure*

1. Install `dnsmasq`.

```
dnf install -y dnsmasq
```

2. Change to the `/etc/dnsmasq.d` directory.

```
cd /etc/dnsmasq.d
```

3. Create a file that reflects your OpenShift cluster appended by `.dns`.

```
touch <filename>.dns
```

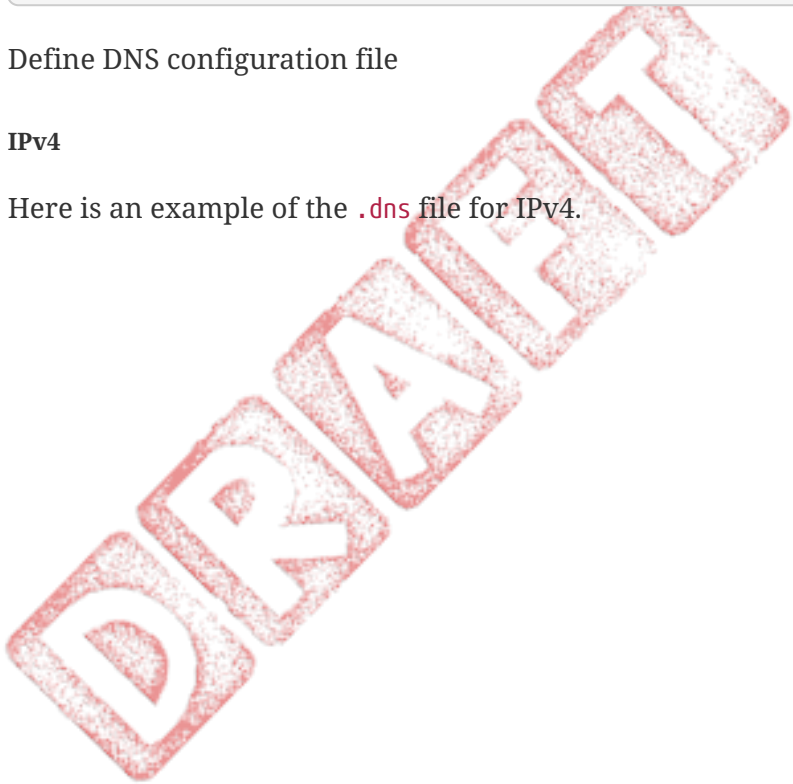
4. Open the appropriate `firewalld` DHCP service.

```
systemctl restart firewalld
firewall-cmd --add-service=dhcp --permanent
firewall-cmd --reload
```

5. Define DNS configuration file

#### IPv4

Here is an example of the `.dns` file for IPv4.



```
domain-needed
bind-dynamic
bogus-priv
domain=openshift.example.com
dhcp-range=<baremetal-net-starting-ip,baremetal-net-ending-ip>
#dhcp-range=10.0.1.4,10.0.14
dhcp-option=3,<baremetal-net-gateway-ip>
#dhcp-option=3,10.0.1.254
resolv-file=/etc/resolv.conf.upstream
interface=<nic-with-access-to-baremetal-net>
#interface=em2
server=<ip-of-existing-server-on-baremetal-net>


#Wildcard for apps -- make changes to cluster-name (openshift) and domain
(example.com)
address=/.apps.openshift.example.com/<wildcard-ingress-lb-ip>

#Static IPs for Masters
dhcp-host=<NIC2-mac-address>,provisioner.openshift.example.com,<ip-of-provisioner>
dhcp-host=<NIC2-mac-address>,openshift-master-0.openshift.example.com,<ip-of-
openshift-master-0>
dhcp-host=<NIC2-mac-address>,openshift-master-1.openshift.example.com,<ip-of-
openshift-master-1>
dhcp-host=<NIC2-mac-address>,openshift-master-2.openshift.example.com,<ip-of-
openshift-master-2>
dhcp-host=<NIC2-mac-address>,openshift-worker-0.openshift.example.com,<ip-of-
openshift-worker-0>
dhcp-host=<NIC2-mac-address>,openshift-worker-1.openshift.example.com,<ip-of-
openshift-worker-1>
```

#### IPv6

Here is an example of the `.dns` file for IPv6.

```
strict-order
bind-dynamic
bogus-priv
dhcp-authoritative
dhcp-range=baremetal,<baremetal-IPv6-dhcp-range-start>,<baremetal-IPv6-dhcp-range-
end>,<range-prefix>
dhcp-option=baremetal,option6:dns-server,[<IPv6-DNS-Server>]

resolv-file=/etc/resolv.conf.upstream
except-interface=lo
dhcp-lease-max=81
log-dhcp

domain=openshift.example.com,<baremetal-IPv6-cidr>,local

# static host-records
address=/apps.openshift.example.com/<wildcard-ingress-lb-ip>
host-record=api.openshift.example.com,<api-ip>
host-record=ns1.openshift.example.com,<dns-ip>
host-record=openshift-master-0.openshift.example.com,<ip-of-openshift-master-0>
host-record=openshift-master-1.openshift.example.com,<ip-of-openshift-master-1>
host-record=openshift-master-2.openshift.example.com,<ip-of-openshift-master-1>
# Registry
host-record=registry.openshift.example.com,<ip-of-registry-server>

#Static IPs for Masters
dhcp-host=<baremetal-nic-duid>,openshift-master-0.openshift.example.com,<ip-of-
openshift-master-0>
dhcp-host=<baremetal-nic-duid>,openshift-master-1.openshift.example.com,<ip-of-
openshift-master-1>
dhcp-host=<baremetal-nic-duid>,openshift-master-2.openshift.example.com,<ip-of-
openshift-master-2>
```

6. Create the `resolv.conf.upstream` file to provide DNS fowarding to an existing DNS server for resolution to the outside world.

```
search <domain.com>
nameserver <ip-of-my-existing-dns-nameserver>
```

7. Restart the `dnsmasq` service.

```
systemctl restart dnsmasq
```

8. Verify the `dnsmasq` service is running.

```
systemctl status dnsmasq
```