# Fully Disconnected Deployment of IPI on BM using the Ansible Playbook

Deployment Integration Team

| 1. Introduction 2  |
|--|
| 2. Prerequisites   |
| 3. Using an Existing Registry  |
| 4. Contents of the Webserver   |
| 5. Fully Disconnected Prerequiste Checklist                                |
| 5.1. Validation checklist for nodes  |
| 5.2. Validation checklist for Ansible playbook installation                |
| 6. Running the playbook.yml. 12  |
| 6.1. git clone the Ansible playbook  |
| 6.2. The ansible.cfg file  |
| 6.3. Ansible version   |
| 6.4. Copy local SSH key to provision node                                  |
| 6.5. Modifying the inventory/hosts file for Fully Disconnected Deployments |
| 6.6. The Ansible playbook.yml  |
| Appendix A: Setup a local RHEL8 repository using an ISO                    |
| Appendix B: Environment Variable Script                                    |
| Appendix C: Helper Script  |



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

# **Chapter 1. Introduction**

This write-up will guide you through the process of deploying a fully-disconnected<sup>[1]</sup> Baremetal IPI installation of OpenShift Container Platform 4 via the Ansible playbook.

[1] Fully disconnected infers that no system in the OpenShift Container Platform has deployment access to the internet.

# **Chapter 2. Prerequisites**

- Best Practice Minimum Setup: 6 Physical servers (1 provision node, 3 master and 2 worker nodes)
- Best Practice Minimum Setup for disconnected environments: 7 Physical servers (1 provision node, 1 registry node<sup>[1]</sup>, 3 master and 2 worker nodes)
- Minimum Setup: 4 Physical servers (1 provision node, 3 master nodes)
- Minimum Setup for disconnected environments: 5 Physical servers (1 provision node, 1 registry node<sup>[1]</sup>, 3 master nodes)
- Each server needs 2 NICs pre-configured. NIC1 for the private network and NIC2 for the baremetal network. NIC interface names must be identical across all nodes<sup>[2]</sup>
- It is recommended each server have a RAID-1 configured and initialized (though not enforced)
- Each server must have IPMI configured
- Each server must have DHCP setup for the baremetal NICs
- Each server must have DNS setup for the API, wildcard applications
- A DNS VIP is IP on the baremetal network is required for reservation. Reservation is done via our DHCP server (though not required).
- Optional Include DNS entries for the hostnames for each of the servers
- Download a copy of your Pull Secret

Due to the complexities of properly configuring an environment, it is recommended to review the following steps prior to running the Ansible playbook as without proper setup, the Ansible playbook won't work.

The section to review and ensure proper configuration are as follows:

- Validation checklist for nodes
- One of the Create DNS records sections
  - Create DNS records on a DNS server (Option 1)
  - Create DNS records using dnsmasq (Option 2)
- One of the Create DHCP reservation sections
  - Create DHCP reservations (Option 1)
  - Create DHCP reservations using dnsmasq (Option 2)
- An existing Registry node (details on creating a registry if required below)
  - Create a disconnected registry
- An existing webserver to cache required files and the RHCOS images (details on creating a webserver if required below)
  - Webserver

Once the above is complete, install Red Hat Enterprise Linux (RHEL) 8.x on your provision node

and create a user (i.e. kni) to deploy as non-root and provide that user sudo privileges.

For simplicity, the steps to create the user named kni is as follows:

- 1. Login into the provision node via ssh
- 2. Create a user (i.e kni) to deploy as non-root and provide that user sudo privileges

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

3. Enable a dnf local repository on the provision host

[1] If creating the mirrored registry, this system will require online access. The registry node may be a virtual machine in order to reduce physical server footprint.

[2] https://github.com/openshift/installer/issues/2762

# **Chapter 3. Using an Existing Registry**



If no existing registry is already existing for your fully disconnected environment, visit Creating a New Disconnected Registry section.

When using an existing registry, two variables labeled disconnected\_registry\_auths\_file and the disconnected\_registry\_mirrors\_file must be set. These variables are located within your inventory/hosts file and the inventory/hosts.sample file can be used as reference.

The disconnected\_registry\_auths\_file variable should point to a file containing json data regarding your registry information. This will be appended to the auths section of the pull secret by the Ansible playbook itself.

An example of the contents of the disconnected\_registry\_auths\_file is shown below.

```
cat /path/to/registry-auths.json
{"registry.example.com:5000": {"auth": "ZHVtbXk6ZHsFVtbXk=", "email":
"user@example.com" } }
```

The auth password given base64 encoding of the http credentials used to create the htpasswd file.



### Example:

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

The disconnected\_registry\_mirrors\_file variable should point to a file containing the additionalTrustBundle and imageContentSources (OpenShift 4.13 and below) or imageDigestSources (OpenShift 4.14 and above) for the disconnected registry. The certificate that goes within the additional trust bundle is the disconnected registry node's certificate. The imageContentSources adds the mirrored information of the registry. The below content from the install-config-appends.yml file gets automatically appended by the Ansible playbook.

```
cat /path/to/install-config-appends.yml
additionalTrustBundle: |
    -----BEGIN CERTIFICATE-----
MIIGPDCCBCSgAwIBAgIUWr1DxDq53hrsk6XVLRXUjfF9m+swDQYJKoZIhvcNAQEL
BQAwgZAxCzAJBgNVBAYTA1VTMRAwDgYDVQQIDAdNeVN0YXR1MQ8wDQYDVQQHDAZN
eUNpdHkxEjAQBgNVBAoMCU15Q29tcGFueTEVMBMGA1UECwwMTX1EZXBhcnRtZW50
.
.
[ABBREVIATED CERTIFICATE FOR BREVITY]
.
MTMwMQYDVQQDDCpyZWdpc3RyeS5rbmk3LmNsb3VkLmxhYi51bmcuYm9zLnJ1ZGhh
dC5jb20wHhcNMjAwNDA3MjM1MzI2WhcNMzAwNDA1MjM1MzI2WjCBkDELMAkGA1UE
-----END CERTIFICATE-----
```

```
<image-config>: ①
- mirrors:
    registry.example.com:5000/ocp4/openshift4
    source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
- mirrors:
    registry.example.com:5000/ocp4/openshift4
    source: registry.svc.ci.openshift.org/ocp/release
- mirrors:
    registry.example.com:5000/ocp4/openshift4
    source: quay.io/openshift-release-dev/ocp-release
```

### Where:

+ <1> <image-config> is either imageContentSources for OpenShift 4.13 and below, or imageDigestSources for Openshift 4.14 and above.



Indentation is important in the yml file. Ensure your copy of the install-configappends.yml is properly indented as in the example above.

# **Chapter 4. Contents of the Webserver**

When following the details on how to create a webserver, if one not already in place, there is still additional content required for a fully disconnected environment to be successfully deployed with the Ansible playbook.

The Ansible playbook requires the end user to additionally include the following to there already existing webserver.

The example provided below showcases how a user adds the required prerequisites to the webserver in order install the latest OpenShift Container Platform version 4.9.

Automatic Procedure

1. Change to the webserver directory that is to store your OpenShift related binaries

[user@webserver ~]\$ cd /path/to/webserver/dir

2. Create a local copy of environment variables script and make the script executable.

[user@webserver ~]\$ chmod +x /path/to/webserver/dir/env\_vars.sh

3. Create a local copy of helper script that downloads all the prerequisites to the webserver

[user@webserver ~]\$ chmod +x /path/to/webserver/dir/helper\_script.sh

- 4. Open the the env\_vars.sh script and fill out the appopriate environment variable values
- 5. Run the helper\_script.sh script

[user@webserver ~]\$ /path/to/webserver/dir/helper\_script.sh



Using the helper\_script.sh has some caveats. Extracting the openshiftbaremetal-install binary does not pull from a local registry when given a local registry, BZ#1823143 Due to this, in order to properly extract the installer, the OpenShift disconnected mirrored registry that is to be used must be available and have access to quay.io temporary to properly extract the binary.



The following manual procedure can be skipped if used the helper script.

#### Manual Procedure

1. Download the OpenShift Container Platform version 4.9 latest release.txt file

[user@webserver ~]\$ cd /path/to/webserver/dir [user@webserver ~]\$ wget https://mirror.openshift.com/pub/openshift-



When working with a development version of OpenShift Container Platform, use the following link for the development version of the release.txt

2. Create a directory with the explict release version of the captured release.txt file

```
export OCP_RELEASE=`cat release.txt | grep Name | awk {'print $2'}`
[user@webserver ~]$ mkdir $OCP_RELEASE
```

3. Move the release.txt file to the newly created release version directory

```
[user@webserver ~]$ mv release.txt $OCP_RELEASE/
```

4. Download the oc client and untar its contents

```
[user@webserver ~]$ wget https://mirror.openshift.com/pub/openshift-
v4/clients/ocp/$0CP_RELEASE/openshift-client-linux-$0CP_RELEASE.tar.gz | tar zxvf -
oc
```

5. Extract the Installer



Extracting the installer currently has some caveats. Extracting the openshiftbaremetal-install binary does not pull from a local registry when given a local registry, BZ#1823143 Due to this, in order to properly extract the installer, the OpenShift disconnected mirrored registry that is to be used must be available and have access to quay.io temporary to properly extract the binary. The following step assumes this.

```
[user@webserver ~]$ export LOCAL_REPOSITORY='ocp4'
[user@webserver ~]$ export LOCAL_REGISTRY='registry.example.com:5000'
[user@webserver ~]$ export cmd=openshift-baremetal-install
[user@webserver ~]$ export pullsecret_file=~/pull-secret.txt
[user@webserver ~]$ export extract_dir=$(pwd)
[user@webserver ~]$ oc adm release extract --registry-config "${pullsecret_file}"
--command="${cmd}" --to `pwd` ${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}
```

6. Ensure the openshift-baremetal-install binary points to the appopriate release image (i.e. registry.example.com)

```
[user@webserver ~]$ ./openshift-baremetal-install version
openshift-baremetal-install 4.4.3
built from commit 78b817ceb7657f81176bbe182cc6efc73004c841
release image
```

registry.example.com:5000/ocp4/openshift4@sha256:e805d6a36762e22ecf66fd3f3642e609a0
0ed25ab44f89f064b5138cf3f0f554

7. The rhcos.json file is required for the disconnected installs as it contains the appropriate image name and SHA hash



This assumes the openshift-baremetal-install has been extracted

```
[user@webserver ~]$ export COMMIT_ID=$(./openshift-baremetal-install version | grep
'^built from commit' | awk '{print $4}')
[user@webserver ~]$ curl -s -S
https://raw.githubusercontent.com/openshift/installer/$COMMIT_ID/data/data/rhcos.js
on > rhcos.json
```

8. Clean up the oc and kubelet binary extraction as no longer required

```
[user@webserver ~]$ rm -f /path/to/$OCP_RELEASE/oc /path/to/$OCP_RELEASE/kubelet
```

9. Confirm all four files have been captured within your \$OCP\_RELEASE directory

```
[user@webserver ~]$ ls -latr /path/to/$OCP_RELEASE
openshift-baremetal-install openshift-client-linux-$OCP_RELEASE.tar.gz rhcos.json
release.txt
```

# Chapter 5. Fully Disconnected Prerequiste Checklist

## 5.1. Validation checklist for nodes

When using the provisioning network

- □ DHCP reservations use infinite leases to deploy the cluster with static IP addresses. (optional)
- □ NIC1 VLAN is configured for the provisioning network.
- □ NIC2 VLAN is configured for the baremetal network.
- □ NIC1 is PXE-enabled on the provisioner, Control Plane (master), and worker nodes.
- $\hfill\square$  PXE has been disabled on all other NICs.
- □ Control plane and worker nodes are configured.
- □ All nodes accessible via out-of-band management.
- □ A separate management network has been created. (optional)
- □ Required data for installation.

### When omitting the provisioning network

- □ DHCP reservations use infinite leases to deploy the cluster with static IP addresses. (optional)
- □ NICx VLAN is configured for the baremetal network.
- □ Control plane and worker nodes are configured.
- □ All nodes accessible via out-of-band management.
- □ A separate management network has been created. (optional)
- □ Required data for installation.

### Summary

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

# 5.2. Validation checklist for Ansible playbook installation

- □ Create a local repository using a RHEL 8 Installation DVD to install packages
- Suppress Unable to read consumer identity messages when using subscription-manager via /etc/yum.conf
- □ Ensure release.txt file exists within the webserver path/to/webserver/<ocp\_release\_version>
- □ Ensure rhcos.json file exists within the webserver path/to/webserver/<ocp\_release\_version>
- Ensure openshift-baremetal-install binary exists within the webserver path/to/webserver/<ocp\_release\_version>

- Ensure the openshift-baremetal-install binary points to the appopriate release image registry (i.e. registry.example.com)
- □ Ensure release.txt file exists within the webserver path/to/webserver/<ocp\_release\_version>
- □ Ensure openshift-client-linux-<ocp\_release\_version>.tar.gz tar.gz exists within the webserver path/to/webserver/<ocp\_release\_version>
- □ Create registry-auths.json
- □ Create install-config-appends.json

# Chapter 6. Running the playbook.yml

The following are the steps to successfully run the Ansible playbook.

## 6.1. git clone the Ansible playbook

The first step to using the Ansible playbook is to clone the baremetal-deploy repository.



This should be done on a system that can access the provision host

1. Clone the git repository

[user@laptop ~]\$ git clone https://github.com/openshift-kni/baremetal-deploy.git

2. Change to the ansible-ipi-install directory

[user@laptop ~]\$ cd /path/to/git/repo/baremetal-deploy/ansible-ipi-install

## 6.2. The ansible.cfg file

While the ansible.cfg may vary upon your environment a sample is provided in the repository.

```
[defaults]
inventory=./inventory
remote_user=kni
callback_whitelist = profile_tasks
```

[privilege\_escalation]
become\_method=sudo



Ensure to change the remote\_user as deemed appropriate for your environment. The remote\_user is the user previously created on the provision node.

### 6.3. Ansible version

Ensure that your environment is using Ansible 2.9 or greater. The following command can be used to verify.

```
ansible --version
ansible 2.9.1
config file = /path/to/baremetal-deploy/ansible-ipi-install/ansible.cfg
configured module search path = ['/path/to/.ansible/plugins/modules',
'/usr/share/ansible/plugins/modules']
```

```
ansible python module location = /usr/lib/python3.7/site-packages/ansible
executable location = /usr/bin/ansible
python version = 3.7.2 (default, Jan 16 2019, 19:49:22) [GCC 8.2.1 20181215 (Red Hat
8.2.1-6)]
```



The config file section should point to the path of your ansible.cfg

## 6.4. Copy local SSH key to provision node

With the ansible.cfg file in place, the next step is to ensure to copy your public ssh key to your provision node using ssh-copy-id.

From the system that is to run the playbook,

\$ ssh-copy-id <user>@provisioner.example.com



<user> should be the user previously created on the provision node (i.e. kni)

# 6.5. Modifying the inventory/hosts file for Fully Disconnected Deployments

While there are many options that may be set when deploying IPI on baremetal using the Ansible playbook. This portion will strictly focus on what are the requirements for including your existing webserver and registry node for a successful deployment.

A sample of the required variables with regards to the existing webserver and registry node are shown below

```
# Provide the webserver URL as shown below if using fully disconnected
webserver_url=http://example.com:8080'
[registry_host]
registry.example.com
[registry_host:vars]
disconnected_registry_auths_file=/path/to/registry-auths.json
disconnected_registry_mirrors_file=/path/to/install-config-appends.json
```

### 6.6. The Ansible playbook.yml

The Ansible playbook connects to your provision host and runs through the redhatci.ocp.node\_prep role and the redhatci.ocp.installer role. No modification is necessary. All modifications of variables may be done within the inventory/hosts file. A sample file is located in this repository under inventory/hosts.sample. From the system that is to run the playbook, Sample playbook.yml

With the playbook.yml set and in-place, run the playbook.yml

\$ ansible-playbook -i inventory/hosts playbook.yml

## Appendix A: Setup a local RHEL8 repository using an ISO

1. On the provision host, mount your RHEL8 ISO

[user@provisioner ~]\$ sudo mount -o loop rhel-8.0-x86\_64-dvd.iso /mnt/

2. Copy media.repo file from mounted directory to /etc/yum.repos.d/

[user@provisioner ~]\$ sudo cp /mnt/media.repo /etc/yum.repos.d/rhel8.repo

3. Set permissions of the newly created rhel8.repo file

```
[user@provisioner ~]$ sudo chmod 644 /etc/yum.repos.d/rhel8.repo
```

4. Edit the rhel8.repo file to match the following

```
[InstallMedia-BaseOS]
name=Red Hat Enterprise Linux 8 - BaseOS
metadata_expire=-1
gpgcheck=1
enabled=1
baseurl=file:///mnt/BaseOS/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
[InstallMedia-AppStream]
name=Red Hat Enterprise Linux 8 - AppStream
metadata_expire=-1
gpgcheck=1
enabled=1
baseurl=file:///mnt/AppStream/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
```

5. Clear the subscription-manager cache

[user@provisioner ~]\$ sudo dnf clean all

6. Modify the /etc/yum.conf file and set plugins to zero

[user@provisioner ~]\$ sudo echo "plugins=0" >> /etc/yum.conf



This is required as certain plugins won't properly load when not directly

subscripted with subscription-manager and may give the error of Unable to read consumer identity

7. Verify the BaseOS and AppStream repos are available

```
[user@provisioner ~]$ sudo dnf repolist
$ sudo dnf repolist
Last metadata expiration check: 0:29:59 ago on Tue 12 May 2020 08:15:46 PM UTC.
repo id repo name
status
InstallMedia-AppStream Red Hat Enterprise Linux 8 - AppStream
4,820
InstallMedia-BaseOS Red Hat Enterprise Linux 8 - BaseOS
1,661
```

# **Appendix B: Environment Variable Script**

```
#!/bin/bash
#Enter 'dev' for development or 'ga' for Generally Available version of OCP
export release=''
#Provide build version, i.e. 4.3.18, 4.4.4, nightly build: 4.3.0-0.nightly-2019-10-29-
073252
export build_version='<desired-build-version>'
export LOCAL_REPOSITORY='ocp4'
export LOCAL_REGISTRY='registry.example.com'
export REGISTRY_PORT='5000'
export OCP_RELEASE='4.4.3'
export COCAL_PULL_SECRET='<Path-to-your-pull-secret.txt'
export cmd=openshift-baremetal-install</pre>
```

# **Appendix C: Helper Script**

```
#!/bin/bash
echo "***This script downloads the files needed for Ansible Automation****"
echo "***Downloads
      1. Release.txt
      'openshift-client-linux-$build_version.tar.gz'
      3. openshift-baremetal-install binary
      4. rhcos.json****"
. ./source_env_vars.sh
code=$(curl -sL -w "%{http_code}\\n" "https://mirror.openshift.com/pub/" -o /dev/null)
if [[ $code != 200 ]]; then
    echo "Did not receive a successful 200 code, exiting..."
    exit
fi
if [ $release == 'dev' ]
then
  export release_version='ocp-dev-preview'
elif [ $release == 'qa' ]
then
   export release_version='ocp'
else
   echo Provide either dev or ga as a value for release.
fi
rm -f release.txt rhcos.json oc kubelet openshift-client-linux-$build_version.tar.gz
echo "****Below are the values that has been set****"
echo Local Repo = $LOCAL_REPOSITORY
echo Local Registry = $LOCAL REGISTRY
echo Registry Port = $REGISTRY PORT
echo Release = $OCP_RELEASE
echo Pull-Secret File = $LOCAL PULL SECRET
echo Build Version = $build version
GREEN='\033[0;32m'
NC='\033[0m'
echo -e "**** Download the release.txt for ${GREEN}$build_version${NC}******"
wget https://mirror.openshift.com/pub/openshift-v4/clients/$release_version
/$build_version/release.txt
echo "****Download the openshift-client-linux-$build_version.tar.gz for the
$build version*******"
wget https://mirror.openshift.com/pub/openshift-v4/clients/$release_version
/$build version/openshift-client-linux-$build version.tar.gz
```

```
tar -xvzf openshift-client-linux-$build_version.tar.gz
echo "*****Download the 'openshift-baremetal-install' binary for the $build version
and extract it******"
web_url=$(curl -sL -w "%{http_code}\\n" "http://${LOCAL_REGISTRY}/${RHCOS_QEMU_URI}"
-o /dev/null)
if [[ $web_url != 200 ]]; then
    echo "Did not receive a successful 200 code, exiting..."
    echo "****Extracting the installer currently has some caveats. Extracting the
openshift-baremetal-install binary does not pull from a local registry when given a
local registry, BZ#1823143 Due to this, in order to properly extract the installer,
the OpenShift disconnected mirrored registry that is to be used must be available and
have access to quay.io temporary to properly extract the binary. The following step
assumes this.****"
    exit # other actions
fi
oc adm release extract --registry-config "${LOCAL_PULL_SECRET}" --command="${cmd}"
--to `pwd` ${LOCAL_REGISTRY}:${REGISTRY_PORT}/${LOCAL_REPOSITORY}:${OCP_RELEASE}
echo "*****Download the rhcos.json file for the $build_version******"
export COMMIT_ID=$(./openshift-baremetal-install version | grep '^built from commit' |
awk '{print $4}')
curl -s -S https://raw.githubusercontent.com/openshift/installer/
$COMMIT_ID/data/data/rhcos.json > rhcos.json
ls -ltr release.txt rhcos.json openshift-baremetal-install openshift-client-linux-
$build_version.tar.gz
echo "****Confirm the version*****"
```

```
./openshift-baremetal-install version
```