OpenShift 4.2 Disconnected Install

SEPTEMBER 13, 2019
BY CHRISTIAN HERNANDEZ
(HTTPS://BLOG.OPENSHIFT.COM/AUTHOR/CHRISTIANHERNANDEZ/)

In a previous blog (https://blog.openshift.com/introducing-red-hat-openshift-4-2-in-developer-preview-releasing-nightly-builds/), it was announced that Red Hat is making the OpenShift nightly builds available to everyone. This gives users a chance to test upcoming features before their general availability. One of the features planned for OpenShift 4.2 is the ability to perform an install in a restricted environment with limited access to the Internet or outside world.

NOTE: OpenShift 4.2 has been released! Find supported instructions on the official documentation site. (https://docs.openshift.com/container-platform/4.2/installing/installing_restricted_networks/installing-restricted-networks-preparations.html)

In this blog I will be going over how to perform an install in a lab environment with restricted access to the Internet. I will also give an overview of my environment, how to mirror the needed images, and any other tips and tricks I’ve learned along the way.

Environment Overview

In my environment, I have two networks. One network is limited between OpenShift infrastructure and the Bastion host. The other connects Bastion host to the general internet with full access. This bastion host will perform the following functions.

- Registry server (where I will mirror the content)
- Apache web server (where I will store installation artifacts)
- Installation host (where I will be performing the installation from)

In my environment, I have already set up DNS, DHCP, and other ancillary services for my network. Also, it’s important to get familiar with the OpenShift 4 prerequisites (https://docs.openshift.com/container-platform/4.1/installing/installing_bare_metal/installing-bare-metal.html#installation-infrastructure-user-infra_installing-bare-metal) before attempting an install.

Doing a this type of install can be challenging, so I recommend trying a fully connected OpenShift 4 install (https://blog.openshift.com/openshift-4-bare-metal-install-quickstart/) first to familiarize yourself with the install process (as they are quite similar).

Registry Set Up

You can use your own registry or build one from scratch. I used the following steps to build one from scratch. Since I'll be using a container for my registry, and Apache for my webserver, I will need `podman` and `httpd` on my host.

```
yum -y install podman httpd httpd-tools
```

Create the directories you'll need to run the registry. These directories will be mounted in the container running the registry.

```
mkdir -p /opt/registry/{auth,certs,data}
```

Next, generate an SSL certificate for the registry. This can, optionally, be self-signed if you don't have an existing, trusted, certificate authority. I'll be using `registry.ocp4.example.com` as the hostname of my registry. Make sure your hostname is in DNS and resolves to the correct IP.

```
cd /opt/registry/certs
openssl req -newkey rsa:4096 -nodes -sha256 -keyout domain.key -x509
-days 365 -out domain.crt
```

Generate a username and password (must use `bcrypt` formatted passwords), for access to your registry.

```
htpasswd -bBc /opt/registry/auth/htpasswd dummy dummy
```

Make sure to open port `5000` on your host, as this is the default port for the registry. Since I am using Apache to stage the files I need for installation, I will open port `80` as well.

```
firewall-cmd --add-port=5000/tcp --zone=internal --permanent
firewall-cmd --add-port=5000/tcp --zone=public   --permanent
firewall-cmd --add-service=http  --permanent
firewall-cmd --reload
```

Now you're ready to run the container. Here I specify the directories I want to mount inside the container. I also specify I want to run on port `5000`. I recommend you put this in a shell script for ease of starting.

```
podman run --name poc-registry -p 5000:5000 \
-v /opt/registry/data:/var/lib/registry:z \
-v /opt/registry/auth:/auth:z \
-e "REGISTRY_AUTH=htpasswd" \
-e "REGISTRY_AUTH_HTPASSWD_REALM=Registry" \
-e "REGISTRY_HTTP_SECRET=ALongRandomSecretForRegistry" \
-e REGISTRY_AUTH_HTPASSWD_PATH=/auth/htpasswd \
-v /opt/registry/certs:/certs:z \
-e REGISTRY_HTTP_TLS_CERTIFICATE=/certs/domain.crt \
-e REGISTRY_HTTP_TLS_KEY=/certs/domain.key \
docker.io/library/registry:2
```

Verify connectivity to your registry with `curl`. Provide it the username and password you created.

```
curl -u dummy:dummy -k https://registry.ocp4.example.com:5000/v2/_ca
talog
```

Note, this should return an “empty” repo

If you have issues connecting try to stop the container.

```
podman stop poc-registry
```

Once it’s down, you can start it back up using the following `podman` command:

```
podman start poc-registry
```

Obtaining Artifacts

You will need the preview builds for 4.2 in order to do this type of install. Specifically, you will need the client binaries along with the install artifacts. This can be found in the dev preview links provided below.

- Client Binaries (https://mirror.openshift.com/pub/openshift-v4/clients/ocp-dev-preview/latest/)
- Install Artifacts (https://mirror.openshift.com/pub/openshift-v4/dependencies/rhcos/pre-release/latest/)

Download the binaries and any installation artifacts you may need for the installation. The file names will differ depending on when you choose to download the preview builds (they get updated often).

You can inspect the nightly release notes and extract the build number from there. I did this with the `curl` command.

```
export BUILDNUMBER=$(curl -s https://mirror.openshift.com/pub/opensh
ift-v4/clients/ocp-dev-preview/latest/release.txt | grep 'Name:' | a
wk '{print $NF}')
echo ${BUILDNUMBER}
```

To download the client binaries to your staging server/area (in my case, it's the registry server itself) use `wget`:

```
wget https://mirror.openshift.com/pub/openshift-v4/clients/ocp-dev-p
review/latest/openshift-client-linux-${BUILDNUMBER}.tar.gz -P /var/w
ww/html/
wget https://mirror.openshift.com/pub/openshift-v4/clients/ocp-dev-p
review/latest/openshift-install-linux-${BUILDNUMBER}.tar.gz -P /var/
www/html/
```

You'll also need these clients on your registry host, so feel free to un-tar them now.

```
tar -xzf /var/www/html/openshift-client-linux-${BUILDNUMBER}.tar.gz
-C /usr/local/bin/
tar -xzf /var/www/html/openshift-install-linux-${BUILDNUMBER}.tar.gz
-C /usr/local/bin/
```

Depending on what kind of install you will do, you would need to do one of the following.

PXE Install

If you’re doing a PXE install, you’ll need the *BIOS*, *initramfs*, and the *kernel* files. For example:

```
wget https://mirror.openshift.com/pub/openshift-v4/dependencies/rhco
s/pre-release/latest/rhcos-${BUILDNUMBER}-metal-bios.raw.gz -P /var/
www/html/
wget https://mirror.openshift.com/pub/openshift-v4/dependencies/rhco
s/pre-release/latest/rhcos-${BUILDNUMBER}-installer-initramfs.img -P
/var/www/html/
wget https://mirror.openshift.com/pub/openshift-v4/dependencies/rhco
s/pre-release/latest/rhcos-${BUILDNUMBER}-installer-kernel -P /var/w
ww/html/
```

Once you have staged these, copy them over into your environment. Once they are in your PXE install server and your configuration updated, you can proceed to mirror your images.

ISO Install

If you’re doing an ISO install, you’ll still need the `BIOS` file but only the `ISO` for the install.

```
wget https://mirror.openshift.com/pub/openshift-v4/dependencies/rhco
s/pre-release/latest/rhcos-${BUILDNUMBER}-metal-bios.raw.gz -P /var/
www/html/
wget https://mirror.openshift.com/pub/openshift-v4/dependencies/rhco
s/pre-release/latest/rhcos-${BUILDNUMBER}-installer.iso -P /var/www/
html/
```

Once these are staged, copy them over to where you’ll need them for the installation. The BIOS file will need to be on a web server accessible to the OpenShift nodes. The ISO can be burned onto a disk/usb drive or mounted via your virtualization platform.

Once that’s done, you can proceed to mirror the container images.

Mirroring Images

The installation images will need to be mirrored in order to complete the installation. Before you begin you need to make sure you have the following in place.

- An internal registry to mirror the images to (like the one I just built)
    - You'll also need the certificate of this registry
    - The username/password for access
- A pullsecret obtained at https://cloud.redhat.com/openshift/install/pre-release (https://cloud.redhat.com/openshift/install/pre-release)
    - I downloaded mine and saved it as **~/pull-secret.json**
- The **oc** and **openshift-install** CLI tools installed
- The **jq** command is also helpful

First, you will need to get the information to mirror. This information can be obtained via the dev-preview release notes (https://mirror.openshift.com/pub/openshift-v4/clients/ocp-dev-preview/latest/release.txt). With this information, I constructed the following environment variables.

```
export OCP_RELEASE="4.2.0-0.nightly-2019-08-29-062233"
export LOCAL_REG='registry.ocp4.example.com:5000'
export LOCAL_REPO='ocp4/openshift4'
export UPSTREAM_REPO='openshift-release-dev'
export LOCAL_SECRET_JSON="${HOME}/pull-secret-2.json"
export OPENSHIFT_INSTALL_RELEASE_IMAGE_OVERRIDE=${LOCAL_REG}/${LOCAL
_REPO}:${OCP_RELEASE}
export RELEASE_NAME="ocp-release-nightly"
```

I will now go over how to construct these environment variables from the release notes (https://mirror.openshift.com/pub/openshift-v4/clients/ocp-dev-preview/latest/release.txt)

- **OCP_RELEASE** – Can be obtained by the Release Metadata.Version section of the release page.
- **LOCAL_REG** – This is your registry's hostname with port
- **LOCAL_REPO** – This is the name of the repo in your registry (you don't have to create it beforehand)
- **UPSTREAM_REPO** – Can be obtained from the Pull From section of the release page.
- **LOCAL_SECRET_JSON** – This is the path to your pull secret with your registry's information (which we will create later)
- **OPENSHIFT_INSTALL_RELEASE_IMAGE_OVERRIDE** – This environment variable is set so the installer knows to use your registry.

- **RELEASE_NAME** – This can be obtained in the Pull From section of the release page.

Before you can mirror the images, you'll need to add the authentication for your registry to your pull secret file (the one you got from try.openshift.com (https://try.openshift.com)). This will look something like this:

```
{
    "registry.ocp4.example.com:5000":
        {
            "auth": "ZHVtbXk6ZHVtbXk=",
            "email": "noemail@localhost"
        }
}
```

The *base64* is a construction of the registry's auth in the *username:password* format. For example, with the `username` of `dummy` and password of `dummy`; I created the base64 by running:

```
echo -n 'dummy:dummy' | base64 -w0
```

You can add your registry's information to your pull secret by using `jq` and the pull secret you downloaded (thus creating a new pull secret file with your registry's information).

```
jq '.auths += {"registry.ocp4.example.com:5000": {"auth": "ZHVtbXk6Z
HVtbXk=","email": "noemail@localhost"}}' &lt; ~/pull-secret.json &g
t; ~/pull-secret-2.json
```

Also, if needed and you haven't done so already, make sure you trust the self-signed certificate. This is needed in order for `oc` to be able to login to your registry during the mirror process.

```
cp /opt/registry/certs/domain.crt /etc/pki/ca-trust/source/anchors/
update-ca-trust extract
```

With this in place, you can mirror the images with the following command.

```
oc adm release mirror -a ${LOCAL_SECRET_JSON} \
--from=quay.io/${UPSTREAM_REPO}/${RELEASE_NAME}:${OCP_RELEASE} \
--to-release-image=${LOCAL_REG}/${LOCAL_REPO}:${OCP_RELEASE} \
--to=${LOCAL_REG}/${LOCAL_REPO}
```

Part of the output will have an example `imageContentSources` to put in your `install-config.yaml` file. It'll look something like this.

```
imageContentSources:
- mirrors:
  - registry.ocp4.example.com:5000/ocp4/openshift4
  source: quay.io/openshift-release-dev/ocp-release-nightly
- mirrors:
  - registry.ocp4.example.com:5000/ocp4/openshift4
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

Save this output, as you'll need it later

Installation

At this point you can proceed with the normal installation procedure, with the main difference being what you specify in the `install-config.yaml` file when you create the ignition configs.

Please refer to the official documentation (https://docs.openshift.com/container-platform/4.1/welcome/index.html) for specific installation information. You’re most likely doing a Bare Metal install, so my previous blog (https://blog.openshift.com/openshift-4-bare-metal-install-quickstart/) would be helpful to look over as well.

When creating an `install-config.yaml` file, you need to specify additional parameters like the example below.

```yaml
apiVersion: v1
baseDomain: example.com
compute:
- hyperthreading: Enabled
  name: worker
  replicas: 0
controlPlane:
  hyperthreading: Enabled
  name: master
  replicas: 3
metadata:
  name: ocp4
networking:
  clusterNetworks:
  - cidr: 10.254.0.0/16
    hostPrefix: 24
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  none: {}
```

```
pullSecret: '{"auths":{"registry.ocp4.example.com:5000": {"auth": "Z
HVtbXk6ZHVtbXk=","email": "noemail@localhost"}}}'
sshKey: 'ssh-rsa .... root@helper'
additionalTrustBundle: |
  -----BEGIN CERTIFICATE-----
  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
imageContentSources:
- mirrors:
  - registry.ocp4.example.com:5000/ocp4/openshift4
  source: quay.io/openshift-release-dev/ocp-release-nightly
- mirrors:
  - registry.ocp4.example.com:5000/ocp4/openshift4
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

Some things to note here:

- **pullSecret** – only the information about your registry is needed.
- **sshKey** – the contents of your **id_rsa.pub** file (or another ssh public key that you want to use)

- **additionalTrustBundle** – this is your crt file for your registry. (i.e. the output of **cat domain.crt**)
- **imageContentSources** – What is the local registry is and the expected original source that should be in the metadata (otherwise they should be considered as tampered)

You will also need to export the `OPENSHIFT_INSTALL_RELEASE_IMAGE_OVERRIDE` environment variable. This tells OpenShift which image to use for bootstrapping. This is in the form of `${LOCAL_REG}/${LOCAL_REPO}:${OCP_RELEASE}`. It looked like this in my environment:

```
export OPENSHIFT_INSTALL_RELEASE_IMAGE_OVERRIDE=registry.ocp4.exampl
e.com:5000/ocp4/openshift4:4.2.0-0.nightly-2019-08-29-062233
```

I created my `install-config.yaml` under my `~/ocp4` install directory. At this point you can create your Ignition configs as you would normally.

```
# openshift-install create ignition-configs --dir=/root/ocp4
INFO Consuming "Install Config" from target directory
WARNING Making control-plane schedulable by setting MastersSchedulab
le to true for Scheduler cluster settings
WARNING Found override for ReleaseImage. Please be warned, this is n
ot advised
```

> Please note that it warns you about overriding the image and that, for the 4.2 dev preview, the masters are schedulable.

At this point, you can proceed with the installation as you would normally (https://docs.openshift.com/container-platform/4.1/installing/installing_bare_metal/installing-bare-metal.html#installation-generate-ignition-configs_installing-bare-metal).

Troubleshooting

A good thing to do during the bootstrapping process is to login to the bootstrap server and tail the journal logs as the bootstrapping process progresses. Many errors or misconfigurations can be seen immediately when tailing this log.

```
[core@bootstrap ~]$ journalctl -b -f -u bootkube.service
```

There are times where you might have to approve the worker/master node’s CSR. You can check pending CSRs with the `oc get csr` command. This is important to check since the cluster operators won’t finish without any worker nodes added. You can approve all the pending CSRs in one shot with the following command.

```
[user@bastion ~]$ oc get csr --no-headers | awk '{print $1}' | xargs
oc adm certificate approve
```

After the bootstrap process is done, it’s helpful to see your cluster operators running. You can do this with the `oc get co` command. It’s helpful to have this in a watch in a separate window.

```
[user@bastion ~]$ watch oc get co
```

The two most common issues are that the `openshift-install` command is waiting for the `image-registry` and `ingress` to come up before it considers the install a success. Make sure you've approved the CSRs for your machines (https://docs.openshift.com/container-platform/4.1/installing/installing_bare_metal/installing-bare-metal.html#installation-approve-csrs_installing-bare-metal) and you've configured storage for your `image-registry` (https://docs.openshift.com/container-platform/4.1/installing/installing_bare_metal/installing-bare-metal.html#installation-registry-storage-config_installing-bare-metal). The commands I've provided should help you navigate any issues you may have.

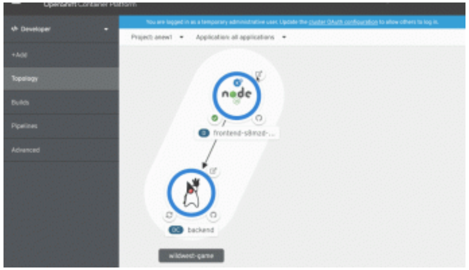
Conclusion

In this blog, I went over how you can prepare for an install in a restricted environment and how to perform an install in this environment using the nightly developer preview of OpenShift 4. These types of installs were a highly popular request for OpenShift 4, and we are excited to bring you a preview build.

Nightly builds are a great way to preview what’s up and coming with OpenShift, so you can test things before the GA release. We are excited to bring you this capability and hope that you find it useful. If you have any questions or comments, feel free to use the comment section below!

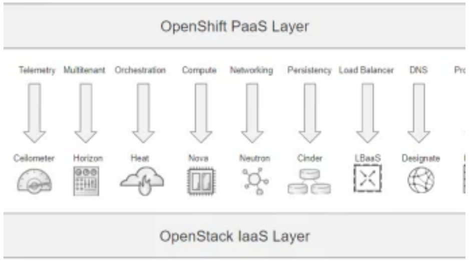
(https://blog.openshift.com/openshift-4-2-disconnected-install/?share=twitter&nb=1)

(https://blog.openshift.com/openshift-4-2-disconnected-install/?share=facebook&nb=1)



(https://blog.openshift.com/introducing-red-hat-openshift-4-2-developers-get-an-expanded-and-improved-toolbox/)

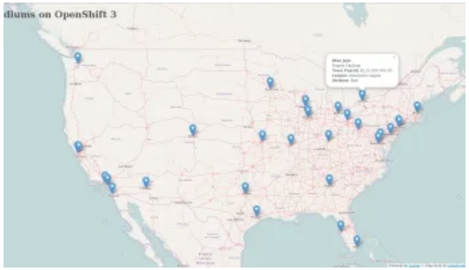
Introducing Red Hat OpenShift 4.2: Developers get an expanded and improved toolbox



(https://blog.openshift.com/openshift-on-openstack-1-2-3-bringing-iaas-paas-together/)

OpenShift on OpenStack 1-2-3: Bringing IaaS and PaaS Together

(https://blog.openshift.com/openshift-



(https://blog.openshift.com/openshift-ecosystem-unleashing-mongodb-with-your-openshift-applications/)

OpenShift Ecosystem: Unleashing MongoDB with your OpenShift

(https://blog.openshift.com/introducing-red-hat-openshift-4-2-developers-get-an-expanded-and-improved-toolbox/)

openstack-1-2-3-bringing-iaas-paas-together/)

Applications (https://blog.openshift.com/op
ecosystem-unleashing-mongodb-with-your-openshift-applications/)

Categories:
Kubernetes (https://blog.openshift.com/category/kubernetes/), News (https://blog.openshift.com/category/news/), OpenShift Container Platform (https://blog.openshift.com/category/products/openshift-container-platform/), Products (https://blog.openshift.com/category/products/)
Tags:

disconnected installs (https://blog.openshift.com/tag/disconnected-installs/), install (https://blog.openshift.com/tag/install/), installation (https://blog.openshift.com/tag/installation/), offline (https://blog.openshift.com/tag/offline/)

Paul Kauffmann
• 14 days ago

Looks like the dependencies file nightly builds are not being published, only the client builds?

^ | ˅ • Reply • Share ›

Keaton Willis
• 25 days ago

If there's ZERO connection between the bastion host and your openshift infrastructure target(s), how can the repository be replicated

Search the blog

CATEGORIES

Containers (https://blog.openshift.com/category/containers/) (120)

CoreOS (https://blog.openshift.com/category/coreos/) (11)

Educators (https://blog.openshift.com/category/educators/) (14)

Events (https://blog.openshift.com/category/events/) (331)

Istio (https://blog.openshift.com/category/istio/) (12)

Kubernetes (https://blog.openshift.com/category/kubernetes/) (152)

News (https://blog.openshift.com/category/news/) (483)

OpenShift Commons (https://blog.openshift.com/category/openshift-commons/) (181)

OpenShift Container Engine (https://blog.openshift.com/category/openshift-container-engine/) (7)

OpenShift Ecosystem (https://blog.openshift.com/category/openshift-ecosystem/) (329)

Operator Framework (https://blog.openshift.com/category/operator-framework/) (16)

Products (https://blog.openshift.com/category/products/) (778)

OpenShift Container Platform (https://blog.openshift.com/category/products/openshift-container-platform/)

(484)

OpenShift Dedicated (https://blog.openshift.com/category/products/openshift-dedicated/)

(162)

OpenShift Online (https://blog.openshift.com/category/products/openshift-online/)

(292)

OpenShift Origin (https://blog.openshift.com/category/products/openshift-origin/)

(358)

OpenStack Platform (https://blog.openshift.com/category/products/openstack-platform/)
(3)
Red Hat CoreOS (https://blog.openshift.com/category/products/red-hat-coreos/)
(5)
Red Hat Quay Registry (https://blog.openshift.com/category/products/red-hat-quay-registry/)
(5)

Security (https://blog.openshift.com/category/security/) (23)

Serverless (https://blog.openshift.com/category/serverless/) (5)

Technologies (https://blog.openshift.com/category/technologies/)
(275)

.NET (https://blog.openshift.com/category/technologies/dotnet/)
(18)

Java (https://blog.openshift.com/category/technologies/java/)
(68)

JBoss (https://blog.openshift.com/category/technologies/jboss/)
(40)

Jenkins
(https://blog.openshift.com/category/technologies/jenkins/)
(23)

MongoDB
(https://blog.openshift.com/category/technologies/mongodb/)
(28)

MySQL
(https://blog.openshift.com/category/technologies/mysql/)
(21)

Node.js (https://blog.openshift.com/category/technologies/node-js/)
(35)

Operators (https://blog.openshift.com/category/technologies/operators/)
(22)

Perl (https://blog.openshift.com/category/technologies/perl/) (4)

PHP (https://blog.openshift.com/category/technologies/php/)
(32)

PostgreSQL (https://blog.openshift.com/category/technologies/postgresql/)
(18)

preview (https://blog.openshift.com/category/technologies/preview/)
(1)

Python (https://blog.openshift.com/category/technologies/python/)

(36)

Ruby (https://blog.openshift.com/category/technologies/ruby/)

(18)

Service Catalog (https://blog.openshift.com/category/technologies/service-catalog/)

(16)

Tomcat (https://blog.openshift.com/category/technologies/tomcat/)

(9)

Zend Server (https://blog.openshift.com/category/technologies/zend-server/)

(2)

Thought Leadership (https://blog.openshift.com/category/thought-leadership/) (197)

Videos (https://blog.openshift.com/category/videos/) (212)

USEFUL LINKS

App Gallery (https://www.openshift.com/application-gallery)

Developer Portal (https://developers.openshift.com/)

Help Center (https://help.openshift.com/)

OpenShift Hub (https://hub.openshift.com/)

Plans & Pricing (https://hub.openshift.com/)

Resource Grants (https://www.openshift.com/startup)

Startup Program (https://www.openshift.com/startup)

What is OpenShift? (https://www.openshift.com/products)

ARCHIVES

Select Month ▼

OLD POSTS

← Older posts (https://blog.openshift.com/page/2/)