How To Open a Shell Prompt on an OpenShift Node

By **Josphat Mutai** - March 28, 2020 941 0

You can support us by downloading this article as PDF from the Link below.

Download the guide as PDF

OpenShift Worker and Master Nodes as of OpenShift Container Platform 4.0 users Red Hat CoreOS(RHCOS) as the base operating system. For OKD, this will be Fedora CoreOS(FCOS). It is not recommended to ssh directly to the OpenShift nodes, not forgetting only ssh keys can be used for access by default. This SSH key is the one provided when deploying an OpenShift / OKD Container platform. So how then can you access a Shell Prompt of an OpenShift Node.

Open a Shell Prompt on an OpenShift Node

OpenShift provides an **oc** client which can be used to access the Shell prompt of an OpenShift Node.

Once you have an OpenShift Cluster up and running, you can install oc client with below commands:

```
 --- Linux ---
wget https://mirror.openshift.com/pub/openshift-v4/clients/oc/latest
tar xvf oc.tar.gz
```

```
chmod +x oc
sudo mv oc /usr/local/bin

--- macOS ---
wget https://mirror.openshift.com/pub/openshift-v4/clients/oc/latest
tar xvf oc.tar.gz
chmod +x oc
sudo mv oc /usr/local/bin
```

Confirm installation by checking the version:

```
$ oc version
Client Version: 4.5.0-202003270516-ad76834
Kubernetes Version: v1.17.4
```

Access OpenShift Node Shell

You'll use the `oc debug node` command to open a shell prompt in any node of your cluster. The syntax is:

```
 $ oc debug node/<node-name>
```

But first list your cluster nodes:

```
 $ oc get nodes
NAME                                STATUS   ROLES          AGE   VE
mas01.ocp.computingforgeeks.com     Ready    master         10d   v1
mas02.ocp.computingforgeeks.com     Ready    master         10d   v1
mas03.ocp.computingforgeeks.com     Ready    master         10d   v1
infra03.ocp.computingforgeeks.com   Ready    infra,worker   10d   v1
infra03.ocp.computingforgeeks.com   Ready    infra,worker   10d   v1
node01.ocp.computingforgeeks.com    Ready    worker         10d   v1
node02.ocp.computingforgeeks.com    Ready    worker         10d   v1
node03.ocp.computingforgeeks.com    Ready    worker         10d   v1
ocs01.ocp.computingforgeeks.com     Ready    worker         10d   v1
```

```
ocs02.ocp.computingforgeeks.com         Ready    worker         10d   v1
ocs03.ocp.computingforgeeks.com         Ready    worker         10d   v1
```

Let’s say you want to start a shell session to **ocs01.ocp.computingforgeeks.com** node, you’ll run:

```
 $ oc debug node/node01.ocp.computingforgeeks.com
Starting pod/node01ocpcomputingforgeekscom-debug ...
To use host binaries, run `chroot /host`
```

The prompt comes from a special-purpose tools container that mounts the node root file system at the **/host** folder, and allows you to inspect any files from the node.

You need to start a *chroot* shell in the */host* folder as shown in the command output. This will enable you to use host binaries in the shell.

```
 chroot /host
```

You’ll see output like below:

```
 chroot /host
Pod IP: 10.184.48.235
If you don't see a command prompt, try pressing enter.
sh-4.2# chroot /host
sh-4.4#
```

Try running command e.g. checking OS version:

```
sh-4.4# cat /etc/redhat-release
Red Hat Enterprise Linux CoreOS release 4.3

sh-4.4# nmcli con show
NAME                UUID                                  TYPE
Wired connection 1  dcd75b54-c1d9-39b7-b7e9-6996a182b53a  ethernet
```

The oc debug shell session uses the same tunneling technology that allows opening a shell prompt inside a running pod – **oc rsh**.