Supermicro X11 BigTwin® OpenShift® 4.8 Deployment Guide

Revision 1.0

The information in this USER'S GUIDE has been carefully reviewed and is believed to be accurate. The vendor assumes no responsibility for any inaccuracies that may be contained in this document, makes no commitment to update or to keep current the information in this manual, or to notify any person organization of the updates. Please Note: For the most up-to-date version of this manual, please see our web site at www.supermicro.com.

Super Micro Computer, Inc. ("Supermicro") reserves the right to make changes to the product described in this manual at any time and without notice. This product, including software, if any, and documentation may not, in whole or in part, be copied, photocopied, reproduced, translated or reduced to any medium or machine without prior written consent.

DISCLAIMER OF WARRANTY ON SOFTWARE AND MATERIALS. You expressly acknowledge and agree that use of the Software and Materials is at your sole risk. FURTHERMORE, SUPER MICRO COMPUTER INC. DOES NOT WARRANT OR MAKE ANY REPRESENTATIONS REGARDING THE USE OR THE RESULTS OF THE USE OF THE SOFTWARE OR MATERIALS IN TERMS OF THEIR CORRECTNESS, ACCURACY, RELIABILITY, OR OTHERWISE. NO ORAL OR WRITTEN INFORMATION OR ADVICE GIVEN BY SUPER MICRO COMPUTER INC. OR SUPER MICRO COMPUTER INC. AUTHORIZED REPRESENTATIVE SHALL CREATE A WARRANTY OR IN ANY WAY INCREASE THE SCOPE OF THIS WARRANTY. SHOULD THE SOFTWARE AND/OR MATERIALS PROVE DEFECTIVE, YOU (AND NOT SUPER MICRO COMPUTER INC. OR A SUPER MICRO COMPUTER INC. AUTHORIZED REPRESENTATIVE) ASSUME THE ENTIRE COST OF ALL NECESSARY SERVICE, REPAIR, OR CORRECTION.

LIMITATION OF LIABILITY. UNDER NO CIRCUMSTANCES INCLUDING NEGLIGENCE, SHALL SUPER MICRO COMPUTER INC. BE LIABLE FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES THAT RESULT FROM THE USE OR INABILITY TO USE THE SOFTWARE OR MATERIALS, EVEN IF SUPER MICRO COMPUTER INC. OR A SUPER MICRO COMPUTER INC. AUTHORIZED REPRESENTATIVE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Any disputes arising between manufacturer and customer shall be governed by the laws of Santa Clara County in the State of California, USA. The State of California, County of Santa Clara shall be the exclusive venue for the resolution of any such disputes. Super Micro's total liability for all claims will not exceed the price paid for the hardware product.

Manual Revision 1.0
Release Date: 11/29/2021

Unless you request and receive written permission from Super Micro Computer, Inc., you may not copy any part of this document.

Information in this document is subject to change without notice. Other products and companies referred to herein are trademarks or registered trademarks of their respective companies or mark holders.

Copyright © 2021 by Super Micro Computer, Inc.
All rights reserved.
Printed in the United States of America

Document Revision History

Date	Revision	Description
11/29/2021	1.0	Initial document.

Contents

Overview 5
Step 1. Design Network Topology 8
Step 2. Preparing Bastion Node with RHEL 8 10
Step 3. Preparing Target Hardware 13
Step 4. Preparing the Provision Environment 14
Step 5. Post-Installation Configuration 22
Appendix 34
List of Hardware Used in Lab 34
List of Supported Network Switches 34
References 34
Contacting Supermicro 35

Overview

There are two basic types of Red Hat OpenShift® clusters – IPI (Installer-Provisioned Infrastructure) and UPI (User-Provisioned Infrastructure) clusters.

This guide is intended to deploy an IPI OpenShift cluster on bare metal for compact clusters on X11 BigTwin® servers and provide instructions for setting up and configuring a bastion node to deploy an OpenShift cluster in a data center or demo lab.

This and other documents may be found on Supermicro's OpenShift landing page:
https://www.supermicro.com/en/solutions/red-hat-openshift

BigTwin

The 2U BigTwin is a breakthrough multi-node server system with a multitude of innovations and industry firsts, designed to support the highest performance CPUs, maximum memory capacity, flexible networking, and hot-swap storage (SATA/SAS/NVMe).

More information and details about BigTwin may be found here:
https://www.supermicro.com/en/products/bigtwin/

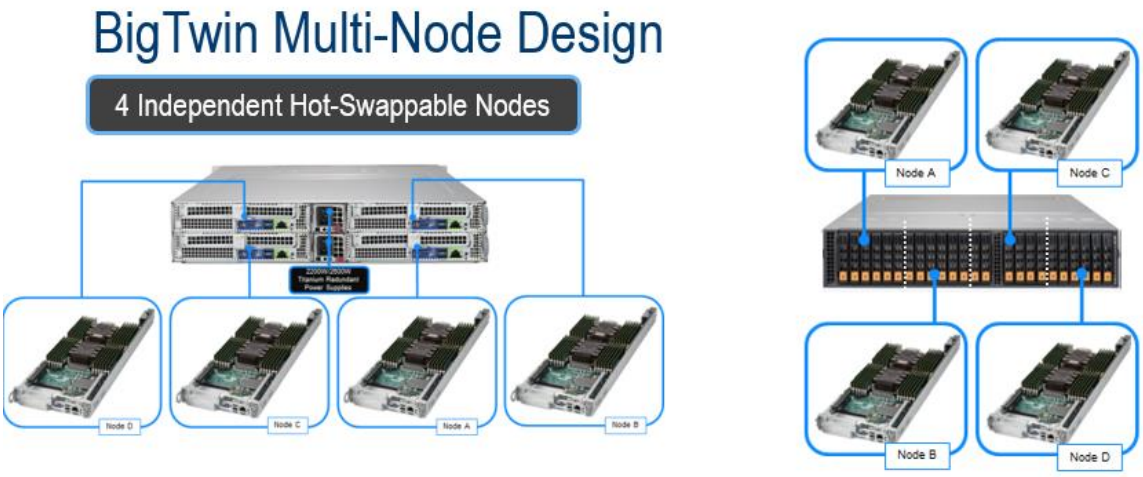

Figure 0-1. BigTwin System Architecture

Figure 0-2. BigTwin Node with Dedicated I/O

Super I/O Module (SIOM) Networking

Supermicro® Super I/O Modules provide flexible I/O networking options. The AOC-MGP-i2 is the most flexible and scalable GbE SIOM controller available. Based on the Intel® i350 GbE network controller, it is designed with performance-enhancing features and power-management technologies.

Figure 0-3. AOC-MGP-i2 Super I/O Module

Figure 0-4. AOC-S100G-b2C Add-on Card

Add-on Card (AOC) Networking

The Supermicro AOC-S100G-b2C is a high-performance, dual-port 100Gb/s Ethernet adapter utilizing the Broadcom® NetXtreme E-Series BCM57508 controller with PCIe Gen 4 support and backward compatibility to PCIe Gen 3. This is a highly scalable NIC that is suitable for cloud-scale networking, storage, machine learning, data analytics and HPC. It is also a feature-rich adapter that provides support to DCDB, VXLAN, NVGRE, Geneve and RoCE. Supermicro Asset Management and thermal detection provide an extra layer of controller health management and peace of mind. The Supermicro AOC-S100G-b2C 100GbE adapter is an excellent choice for your enterprise computing infrastructure.

Server (BMC) Management Switch

The SSE-G3648B Layer 2/3 Ethernet Switch complements the growing family of switch offerings by Supermicro in the Open Networking arena. Offering 48 1Gbps RJ45 Ethernet ports and four 10Gbps SFP+ uplinks, the SSE-G3648B switch allows connectivity at 1Gbps to servers while aggregating traffic for connection to higher speed servers, routers, and other backbone network switches. The compact 1U form factor gives users the ability to optimize deployment in stand-alone or top-of-rack environments. This switch is ideal for managing and maintaining infrastructure via the dedicated BMC port on each OpenShift node.

Figure 0-5. SSE-G3648B/SSE-G3648BR Layer 2/3 Ethernet Switch

Storage Fabric

The SSE-X3348T and its companion the SSE-X3348TR Layer 3 Ethernet Switch both offer forty-eight 10-Gigabit Ethernet ports using the popular new 10GBase-T connection option to allow even more flexibility in providing data-center-friendly connectivity to 10GE routers, servers, and backbones. They also offer four ports running at 40Gbps for access to high-speed backbone networks or storage servers. The 1U form factor gives users the ability to optimize deployment in standalone or top-of-rack environments. An optional rail kit facilitates rack-mounting installations. The SSE-X3348TR model provides a data-center-friendly reverse air-flow design for improved cooling when installed in the rear of a rack.

Figure 0-6. *SSE-X3348T 10Gbps Ethernet Switch*

Step 1. Designing Network Topology

The following resources were used in the demo lab:

- 1G Switch: SSE-G3648B
- 10G Switch: SSE-X3348T
- 100G Switch: SSE-C3632S
- 2U 4-Node BigTwin: SYS-2029BT-HNC0R

Note: Nodes A, B, and C were used for a 3-node compact cluster.

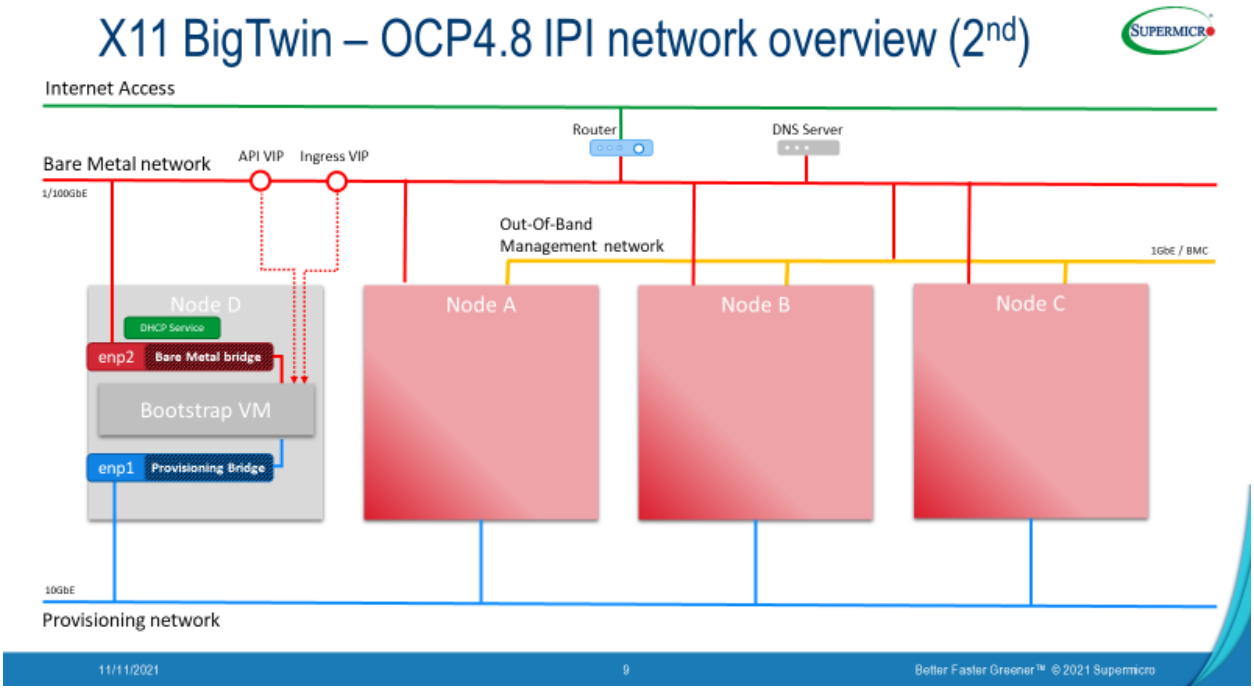

Figure 1-1. OCP 4.8 IPI Network Overview

1. Network switches assigned to each network:
   - BMC network (1G switch, SSE-G3648B, 172.27.34.0/24, Internet access)
   - 10G network (10G switch, SSE-X3348T, no Internet access)
   - 100G network (100G switch, SSE-C3632S, 172.27.34.0/24, Internet access)

2. IP addresses assigned to the BMC LAN ports on an X11 BigTwin®:
   - Node A 172.27.34.111
   - Node B 172.27.34.112
   - Node C 172.27.34.113
   - Node D 172.27.34.114

3. DNS records setup for OpenShift API VIP, Ingress VIP, and the nodes.

Domain Name	IP Address
nodea.compact2.openshift.smc	172.27.34.11
nodeb.compact2.openshift.smc	172.27.34.12
nodec.compact2.openshift.smc	172.27.34.13
noded.compact2.openshift.smc	172.27.34.14
*.apps.compact2.openshift.smc	172.27.34.7
api.compact2.openshift.smc	172.27.34.8

Table 2-1. DNS Records

4. Reserved the DHCP IP addresses for the node BMC and OS LAN ports.

Node	OS IP Address	BMC IP Address	Note
Node A	172.27.34.11	172.27.34.111	
Node B	172.27.34.12	172.27.34.112	
Node C	172.27.34.13	172.27.34.113	
Node D	172.27.34.14	172.27.34.114	Bastion node

Table 3-2. DCHP IP Addresses

Reference documents for provisioning Red Hat OpenShift 3-node compact cluster:

- https://docs.openshift.com/container-platform/4.8/installing/installing_bare_metal/installing-bare-metal.html#installation-three-node-cluster_installing-bare-metal
- https://docs.openshift.com/container-platform/4.9/installing/installing_bare_metal/installing-bare-metal.html#installation-three-node-cluster_installing-bare-metal

Step 2. Preparing Bastion Node with RHEL 8

1. Install RHEL 8.3 to M.2 SATA RAID VD on node D by BMC Virtual Media with ISO image via iKVM.

2. Set up the network 172.27.34.14/255.255.255.0, Gateway 172.27.34.1, DNS 172.27.88.10, 8.8.8.8, 1.1.1.1

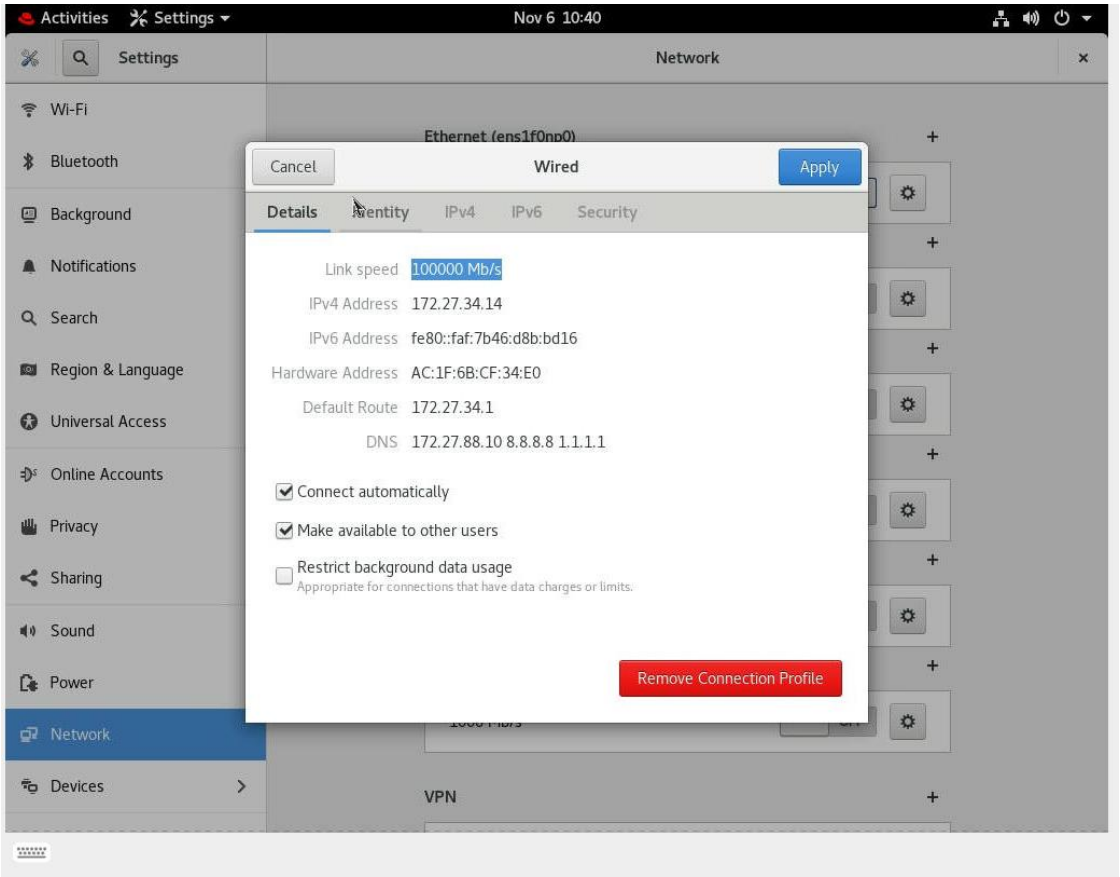

Figure 2-1. Bastion Network Configuration

3. Use Red Hat Subscription Manager to register the provisioner node:
   $ sudo subscription-manager register --username=<username> --password=<pass> --auto-attach

4. Add OCP/ODF 4.8 & ansible-2.9 into repolist, then check
   $ sudo subscription-manager repos \
   --disable='*' \
   --enable=rhel-8-for-x86_64-appstream-rpms \
   --enable=rhel-8-for-x86_64-baseos-rpms \
   --enable=rhel-8-for-x86_64-highavailability-rpms \
   --enable=ansible-2.9-for-rhel-8-x86_64-rpms \

--enable=rhocp-4.8-for-rhel-8-x86_64-rpms

$ sudo yum repolist

```
Updating Subscription Management repositories.
repo id                                  repo name
ansible-2.9-for-rhel-8-x86_64-rpms       Red Hat Ansible Engine 2.9 for RHEL 8 x86_64 (RPMs)
rhel-8-for-x86_64-appstream-rpms         Red Hat Enterprise Linux 8 for x86_64 - AppStream (RPMs)
rhel-8-for-x86_64-baseos-rpms            Red Hat Enterprise Linux 8 for x86_64 - BaseOS (RPMs)
rhel-8-for-x86_64-highavailability-rpms Red Hat Enterprise Linux 8 for x86_64 - High Availability (RPMs)
rhocp-4.8-for-rhel-8-x86_64-rpms         Red Hat OpenShift Container Platform 4.8 for RHEL 8 x86_64 (RPMs)
```

5. Log in to the Bastion node (node D) via ssh

6. Run yum update -y for latest patches and update repos
   $ sudo yum update -y

7. Enable Cockpit on RHEL8
   $ sudo systemctl enable cockpit.socket
   $ sudo systemctl start cockpit.socket

8. Add Virtual Machine, Container applications on RHEL Cockpit web UI.

9. Upload the CentOS 8.3 LiveCD ISO image for the next step.

10. Generate an SSH key pair and share the local SSH pub key to node D.

11. Create a non-root user (kni) and provide that user with sudo privileges:
    $ sudo su -
    # useradd kni
    # passwd kni
    # echo "kni ALL=(root) NOPASSWD:ALL" | tee -a /etc/sudoers.d/kni
    # chmod 0440 /etc/sudoers.d/kni

12. Create an SSH key for the new user and login as kni
    # su - kni -c "ssh-keygen -t ed25519 -f /home/kni/.ssh/id_rsa -N ''"
    # su - kni

13. Install the following packages:
    $ sudo dnf install -y libvirt qemu-kvm mkisofs python3-devel jq ipmitool

14. Modify the user to add the libvirt group to the newly created user:
    $ sudo usermod --append --groups libvirt kni
    $ sudo usermod --append --groups libvirt twinuser

15. Restart firewalld and enable the http service:
    $ sudo systemctl start firewalld
    $ sudo firewall-cmd --zone=public --add-service=http --permanent
    $ sudo firewall-cmd --reload

16. Start and enable the libvirtd service:
    $ sudo systemctl enable libvirtd --now
    $ sudo systemctl status libvirtd

```
● libvirtd.service - Virtualization daemon
   Loaded: loaded (/usr/lib/systemd/system/libvirtd.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2021-11-08 13:56:42 PST; 15s ago
     Docs: man:libvirtd(8)
           https://libvirt.org
 Main PID: 18184 (libvirtd)
    Tasks: 19 (limit: 32768)
   Memory: 61.1M
   CGroup: /system.slice/libvirtd.service
           ├─ 4169 /usr/sbin/dnsmasq --conf-file=/var/lib/libvirt/dnsmasq/default.conf --
leasefile-ro --dhcp-script=/usr/libexec/libvirt_leaseshelper
           ├─ 4170 /usr/sbin/dnsmasq --conf-file=/var/lib/libvirt/dnsmasq/default.conf --
leasefile-ro --dhcp-script=/usr/libexec/libvirt_leaseshelper
           └─18184 /usr/sbin/libvirtd --timeout 120

Nov 08 13:56:42 bastion2.openshift.smc systemd[1]: Starting Virtualization daemon...
Nov 08 13:56:42 bastion2.openshift.smc systemd[1]: Started Virtualization daemon.
Nov 08 13:56:42 bastion2.openshift.smc dnsmasq[4169]: read /etc/hosts - 2 addresses
Nov 08 13:56:42 bastion2.openshift.smc dnsmasq[4169]: read
/var/lib/libvirt/dnsmasq/default.addnhosts - 0 addresses
Nov 08 13:56:42 bastion2.openshift.smc dnsmasq-dhcp[4169]: read
/var/lib/libvirt/dnsmasq/default.hostsfile
```

17. Create the default storage pool and start it:
    $ sudo virsh pool-define-as --name default --type dir --target /var/lib/libvirt/images
    $ sudo virsh pool-start default
    $ sudo virsh pool-autostart default

18. Install a DHCP server
    $ sudo yum install dhcp-server
    $ sudo vi /etc/dhcp/dhcpd.conf #### Copy etc-dhcp-dhcpd.conf to /etc/dhcp/dhcpd.conf
    $sudo systemctl enable --now dhcpd

19. If you have the firewalld running, allow the service port to be accessible from your network.
    $ sudo firewall-cmd --add-service=dhcp --permanent
    $ sudo firewall-cmd --reload

Step 3. Preparing Target Hardware

1. Update SYS-2029BT-HNC0R BMC and BIOS firmware
   - BMC version 1.73.14 (built on 11/8/2021)
   - BIOS version 3.5 (built on 5/15/2021)
   - AOC-SMG2-2TM2 Firmware 2.3.212.1006 with UEFI 1.1.21.1002 (Package 2.3.21.2007, release date: 8/27/2019)

2. Wipe the OS drive on the target node with CentOS 8.3 LiveCD by BMC VirtualMedia
   $ sudo wipefs -a /dev/sda
   ```
   /dev/sda: 2 bytes were erased at offset 0x000001fe (dos): 55 aa
   /dev/sda: calling ioctl to re-read partition table: Success
   ```
   $ sudo wipefs -a /dev/nvme0n1
   ```
   /dev/nvme0n1: 8 bytes were erased at offset 0x00000200 (gpt): 45 46 49 20 50 41 52 54
   /dev/nvme0n1: 8 bytes were erased at offset 0x6fc7d255e00 (gpt): 45 46 49 20 50 41 52 54
   /dev/nvme0n1: 2 bytes were erased at offset 0x000001fe (PMBR): 55 aa
   /dev/nvme0n1: calling ioctl to re-read partition table: Success
   ```
   $ sudo wipefs -a /dev/nvme1n1
   ```
   /dev/nvme1n1: 8 bytes were erased at offset 0x00000200 (gpt): 45 46 49 20 50 41 52 54
   /dev/nvme1n1: 8 bytes were erased at offset 0x6fc7d255e00 (gpt): 45 46 49 20 50 41 52 54
   /dev/nvme1n1: 2 bytes were erased at offset 0x000001fe (PMBR): 55 aa
   /dev/nvme1n1: calling ioctl to re-read partition table: Success
   ```

3. Collect MAC addresses for NIC1 (connect to 100G switch) and NIC2 (connect to 10G switch)
   $ ip a

4. Reboot the system and enter BIOS.

5. Switch Virtual Media -- "UEFI USB CD/DVD:UEFI: ATEN Virtual CDROM YS0J" as 2nd boot order.

6. Set Boot mode as Dual (prefer UEFI).

7. Clean up PXE list.

Step 4. Preparing the Provision Environment

1. Configure network bridges for OpenShift Bootstrap VM

2. Export the bare metal network NIC name:
   $ export PUB_CONN=baremetal

3. Configure the bare metal network:
   $ 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
   pkill dhclient;dhclient baremetal"

4. Export the provisioning network:
   $ export PROV_CONN=provisioning

5. Configure the provisioning network:
   sudo nohup bash -c "
   nmcli con down \"$PROV_CONN\"
   nmcli con delete \"$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"

6. Verify the connection bridges have been properly created.
   $ sudo nmcli con show

```
NAME          UUID                                  TYPE      DEVICE
baremetal     0d71b126-b708-4017-92f0-31b5f3216fa8  bridge    baremetal
virbr0        b6e0fce6-eb34-4884-bd89-624dc2a6f749  bridge    virbr0
ens1f0np0     ef5cecad-1342-419c-969a-89a3cc13375d  ethernet  ens1f0np0
ens1f1np1     ab147953-b5b5-4802-804d-833d9e339246  ethernet  --
ens2f0np0     60b157f5-64ae-4737-baf9-fbe6996287c1  ethernet  --
ens2f1np1     a29ea056-930d-41c8-bd46-fe1eb0e1a203  ethernet  --
ens6f0        f767f61d-8757-48d7-80f4-51067c671fab  ethernet  --
ens6f1        88021319-e1d1-4650-a2b3-faf9bcf43d0f  ethernet  --
provisioning  28e02d04-fb3a-4d7d-949c-5469232b6400  bridge    --
```

7. Get pull-secret.txt from Red Hat Cloud console.

8. Retrieve the OpenShift Container Platform installer using the following steps:
   $ export VERSION=4.8.14
   $ 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}')

9. Extract the OpenShift Container Platform installer.

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

11. Set the environment variables:

    $ export cmd=openshift-baremetal-install

    $ export pullsecret_file=~/pull-secret.txt

    $ export extract_dir=$(pwd)

12. Get oc binary:
    $ curl -s https://mirror.openshift.com/pub/openshift-v4/clients/ocp/$VERSION/openshift-client-linux.tar.gz | tar zxvf - oc
    $ sudo cp oc /usr/local/bin

13. Get the installer:
    $ oc adm release extract --registry-config "${pullsecret_file}" --command=$cmd --to "${extract_dir}" ${RELEASE_IMAGE}
    $ sudo cp openshift-baremetal-install /usr/local/bin

14. Create an RHCOS images cache:
    a. Install Podman
       $ sudo dnf install -y podman

    b. Open firewall port 8080 to be used for RHCOS image caching:
       $ sudo firewall-cmd --add-port=8080/tcp --zone=public –permanent
       $ sudo firewall-cmd –reload

    c. Create a directory to store the bootstraposimage and clusterosimage:
       $ mkdir /home/kni/rhcos_image_cache

d. Check the storage devices' capacity:
   $ df -h

```
Filesystem              Size  Used Avail Use% Mounted on
devtmpfs                 87G     0   87G   0% /dev
tmpfs                    87G     0   87G   0% /dev/shm
tmpfs                    87G   19M   87G   1% /run
tmpfs                    87G     0   87G   0% /sys/fs/cgroup
/dev/mapper/rhel-root    70G  7.3G   63G  11% /
/dev/nvme1n1p1          3.1T   22G  3.1T   1% /var/lib/libvirt/images
/dev/sda1              1014M  326M  689M  33% /boot
/dev/mapper/rhel-home   149G  1.9G  147G   2% /home
tmpfs                    18G   20K   18G   1% /run/user/42
tmpfs                    18G  4.0K   18G   1% /run/user/1000
```

e. Set the appropriate SELinux context for the newly created directory:
   $ sudo semanage fcontext -a -t httpd_sys_content_t "/home/kni/rhcos_image_cache(/.*)?"
   $ sudo restorecon -Rv rhcos_image_cache/

f. Relabel /home/kni/rhcos_image_cache from unconfined_u:object_r:user_home_t:s0 to unconfined_u:object_r:httpd_sys_content_t:s0

g. Get the commit ID from the installer:
   $ export COMMIT_ID=$(/usr/local/bin/openshift-baremetal-install version | grep '^built from commit' | awk '{print $4}')

   The ID determines which images the installer needs to download.

h. Get the URI for the RHCOS image that the installer will deploy on the nodes:
   $ export RHCOS_OPENSTACK_URI=$(curl -s -S https://raw.githubusercontent.com/openshift/installer/$COMMIT_ID/data/data/rhcos.json | jq .images.openstack.path | sed 's/"//g')

i. Get the URI for the RHCOS image that the installer will deploy on the bootstrap VM:
   $ export RHCOS_QEMU_URI=$(curl -s -S https://raw.githubusercontent.com/openshift/installer/$COMMIT_ID/data/data/rhcos.json | jq .images.qemu.path | sed 's/"//g')

j. Get the path where the images are published:
   $ export RHCOS_PATH=$(curl -s -S https://raw.githubusercontent.com/openshift/installer/$COMMIT_ID/data/data/rhcos.json | jq .baseURI | sed 's/"//g')

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

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

l. Get the SHA hash for the RHCOS image that will be deployed on the nodes
$ export RHCOS_OPENSTACK_SHA_COMPRESSED=$(curl -s -S https://raw.githubusercontent.com/openshift/installer/$COMMIT_ID/data/data/rhcos.json | jq -r '.images.openstack.sha256')

m. Download the images and place them in the /home/kni/rhcos_image_cache directory:
$ curl -L ${RHCOS_PATH}${RHCOS_QEMU_URI} -o /home/kni/rhcos_image_cache/${RHCOS_QEMU_URI}

$ curl -L ${RHCOS_PATH}${RHCOS_OPENSTACK_URI} -o /home/kni/rhcos_image_cache/${RHCOS_OPENSTACK_URI}

n. Confirm SELinux type is of httpd_sys_content_t for the newly created files:
$ ls -Z /home/kni/rhcos_image_cache

o. Create the pod
$ 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

#####
The above command creates a caching webserver with the name rhcos_image_cache, which serves the images for deployment. The first image ${RHCOS_PATH}${RHCOS_QEMU_URI}?sha256=${RHCOS_QEMU_SHA_UNCOMPRESSED} is the bootstrapOSImage and the second image ${RHCOS_PATH}${RHCOS_OPENSTACK_URI}?sha256=${RHCOS_OPENSTACK_SHA_COMPRESSED} is the clusterOSImage in the install-config.yaml file.
#####

p. Generate the bootstrapOSImage and clusterOSImage configuration:
$ export BAREMETAL_IP=$(ip addr show dev baremetal | awk '/inet /{print $2}' | cut -d"/" -f1)

$ export RHCOS_OPENSTACK_SHA256=$(zcat /home/kni/rhcos_image_cache/${RHCOS_OPENSTACK_URI} | sha256sum | awk '{print $1}')

$ export RHCOS_QEMU_SHA256=$(zcat /home/kni/rhcos_image_cache/${RHCOS_QEMU_URI} | sha256sum | awk '{print $1}')
$ export CLUSTER_OS_IMAGE="http://${BAREMETAL_IP}:8080/${RHCOS_OPENSTACK_URI}?sha256=${RHCOS_OPENSTACK_SHA256}"

$ export BOOTSTRAP_OS_IMAGE="http://${BAREMETAL_IP}:8080/${RHCOS_QEMU_URI}?sha256=${RHCOS_QEMU_SHA256}"

$ echo "${RHCOS_OPENSTACK_SHA256} ${RHCOS_OPENSTACK_URI}" > /home/kni/rhcos_image_cache/rhcos-ootpa-latest.qcow2.md5sum

$ echo "   bootstrapOSImage: ${BOOTSTRAP_OS_IMAGE}"

```
  bootstrapOSImage: http://172.27.34.14:8080/rhcos-48.84.202106301921-0-qemu.x86_64.qcow2.gz?sha256=fe7f5cb0bf0a46d8a7a1b58f658cb759732ada5ed875d9ef0f51574bf426b061
```

$ echo "   clusterOSImage: ${CLUSTER_OS_IMAGE}"

```
  clusterOSImage: http://172.27.34.14:8080/rhcos-48.84.202106301921-0-openstack.x86_64.qcow2.gz?sha256=5a75df7b4d4dc1861093e520187a133eda3439019f280dc6e2f57edf70eb089d
```

q. Add the required configuration to the install-config.yaml file under platform.baremetal:

```
platform:
  baremetal:
    bootstrapOSImage:
     http://<BAREMETAL_IP>:8080/<RHCOS_QEMU_URI>?sha256=<RHCOS_QEMU_SHA256>
    clusterOSImage:
     http://<BAREMETAL_IP>:8080/<RHCOS_OPENSTACK_URI>?sha256=<RHCOS_OPENSTACK_SHA256>
```

For example:

```
platform:
  baremetal:
    bootstrapOSImage: http://172.27.34.14:8080/rhcos-48.84.202106301921-0-
      qemu.x86_64.qcow2.gz?sha256=fe7f5cb0bf0a46d8a7a1b58f658cb759732ada5ed875d9ef0f5
      1574bf426b061
    clusterOSImage: http://172.27.34.14:8080/rhcos-48.84.202106301921-0-
      openstack.x86_64.qcow2.gz?sha256=5a75df7b4d4dc1861093e520187a133eda3439019f280dc
      6e2f57edf70eb089d
```

15. Prepare the configuration files
    a. Prepare the config yaml file as install-config.yaml. In this deployment, Redfish Virtual Media was used on a bare metal network.
       Hence, the “ipmi” will be replaced by “redfish-virtualmedia”
    b. Add the provisioningNetwork: "Disabled" below the ingressVIP attribute.
    c. Add “disableCertificateVerification: True” below the password attribute since our BMC uses self-signed SSL certificates.
    d. Put the NIC1 (connected to bare metal network) MAC address on the bootMACAddress attribute.
    e. Put the worker’s replicas attribute to be 0 for the compact cluster as **Figure 3-1**

```
compute:
- name: worker
```

```
replicas: 0
```

```yaml
apiVersion: v1
baseDomain: openshift.smc
metadata:
  name: compact2
networking:
  machineCIDR: 172.27.34.0/24
  networkType: OpenShiftSDN
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  serviceNetwork:
  - 172.30.0.0/16
compute:
- name: worker
  replicas: 0
controlPlane:
  hyperthreading: Enabled
  name: master
  replicas: 3
  platform:
    baremetal: {}
platform:
  baremetal:
```

Figure 4-1. Worker Replica Values = 0

```yaml
platform:
  baremetal:
    apiVIP: 172.27.34.8
    ingressVIP: 172.27.34.7
    provisioningNetwork: "Disabled"  #use redfish-virtualmedia
    bootstrapOSImage: http://172.27.34.14:8080/rhcos-48.84.202109241901-0-qemu.x86_64.qcow2.gz?sha256=50377ba9c5cb92c649c7d9e31b5
    clusterOSImage: http://172.27.34.14:8080/rhcos-48.84.202109241901-0-openstack.x86_64.qcow2.gz?sha256=e0a1d8a99c5869150a56b8de
    hosts:
      - name: nodea
        role: master
        bmc:
          address: redfish-virtualmedia://172.27.34.111/redfish/v1/Systems/1
          username: ADMIN
          password: Supermicro123!
          disableCertificateVerification: True
        bootMACAddress: AC:1F:6B:CF:34:D0  # NIC2: 0C:C4:7A:F9:96:E5
        rootDeviceHints:
         deviceName: "/dev/sda"
        #bootMode: UEFISecureBoot
      - name: nodeb
        role: master
        bmc:
          address: redfish-virtualmedia://172.27.34.112/redfish/v1/Systems/1
          username: ADMIN
          password: Supermicro123!
          disableCertificateVerification: True
        bootMACAddress: AC:1F:6B:CF:34:50 # NIC2: 0C:C4:7A:FB:5D:2D
        rootDeviceHints:
         deviceName: "/dev/sda"
        #bootMode: UEFISecureBoot
      - name: nodec
        role: master
        bmc:
          address: redfish-virtualmedia://172.27.34.113/redfish/v1/Systems/1
          username: ADMIN
          password: Supermicro123!
          disableCertificateVerification: True
        bootMACAddress: AC:1F:6B:CF:38:8A # NIC2: 0C:C4:7A:FB:5C:AD
        rootDeviceHints:
         deviceName: "/dev/sda"
        #bootMode: UEFISecureBoot
pullSecret: '{"auths":{"cloud.openshift.com":{"auth":"b3BlbnNoaWZ0LXJlbGVhc2UtZGV2K2Rvam9qbzIwMjAxeG1kNmVqY2FibTZuZGFzMXRjbnR1Ymh
```

Figure 4-2. 3-Node Configuration in YAML File

f. Create a directory to store the cluster configurations.
   $ mkdir ~/clusterconfigs
   $ cp install-config.yaml ~/clusterconfigs

16. Ensure all bare metal nodes are powered off prior to installing the OpenShift Container Platform cluster.
    $ ipmitool -I lanplus -U <user> -P <password> -H <management-server-ip> power off

17. Remove old bootstrap resources if any are left over from a previous deployment attempt.

```
$ for i in $(sudo virsh list | tail -n +3 | grep bootstrap | awk {'print $2'});
do
  sudo virsh destroy $i;
  sudo virsh undefine $i;
  sudo virsh vol-delete $i --pool $i;
  sudo virsh vol-delete $i.ign --pool $i;
  sudo virsh pool-destroy $i;
  sudo virsh pool-undefine $i;
done
```

18. Clean up and prepare env.:
    $ openshift-baremetal-install --dir ~/clusterconfigs --log-level debug destroy cluster
    $ cp install-config.yaml ./clusterconfigs/install-config.yaml

19. Create a cluster with openshift-baremetal-install:
    $ openshift-baremetal-install --dir ~/clusterconfigs --log-level debug create cluster

20. If the installer times out, use the following command to wait for bootstrap VM to complete jobs:
    Suggest using iKVM to run the command to avoid the SSH session timeout.
    $ openshift-baremetal-install --dir ~/clusterconfigs wait-for bootstrap-complete

21. Once the installer completes, it will show a message like the following;

```
DEBUG Cluster is initialized
INFO Waiting up to 10m0s for the openshift-console route to be created...
DEBUG Route found in openshift-console namespace: console
DEBUG OpenShift console route is admitted
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/kni/clusterconfigs/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.compact2.openshift.smc
INFO Login to the console with user: "kubeadmin", and password: "XXXXX-yyyyy-ZZZZZ-aaaaa"
DEBUG Time elapsed per stage:
DEBUG Cluster Operators: 18m0s
INFO Time elapsed: 18m0s
```

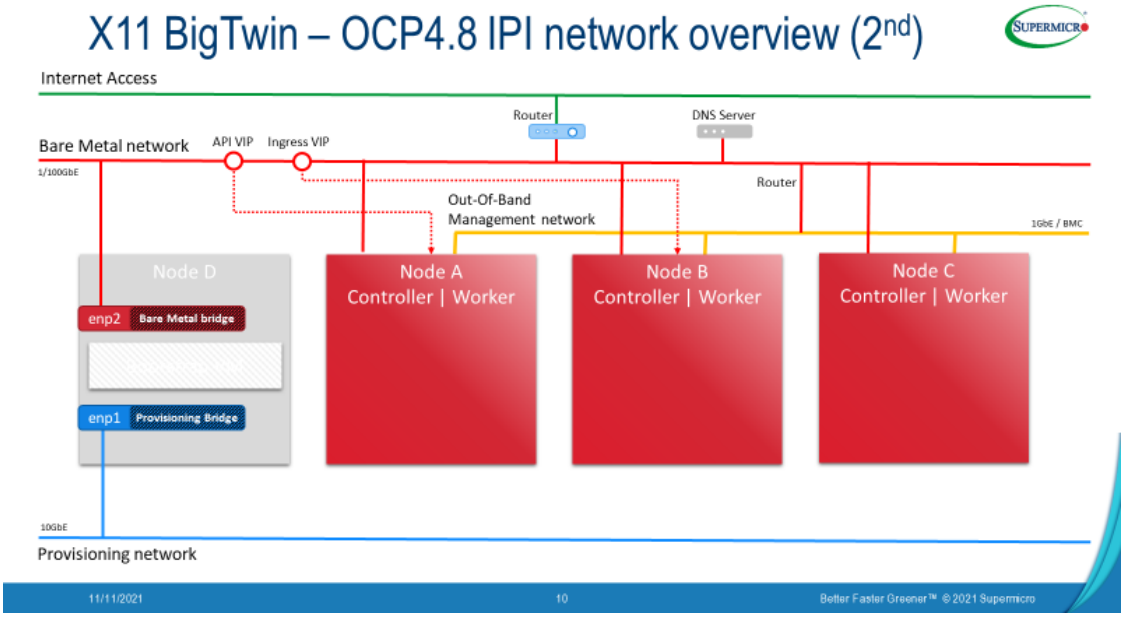

Figure 4-3. Network Overview of Completed Installation

Step 5. Post-Installation Configuration

1. Login to the OpenShift web console with the kubeadmin credential

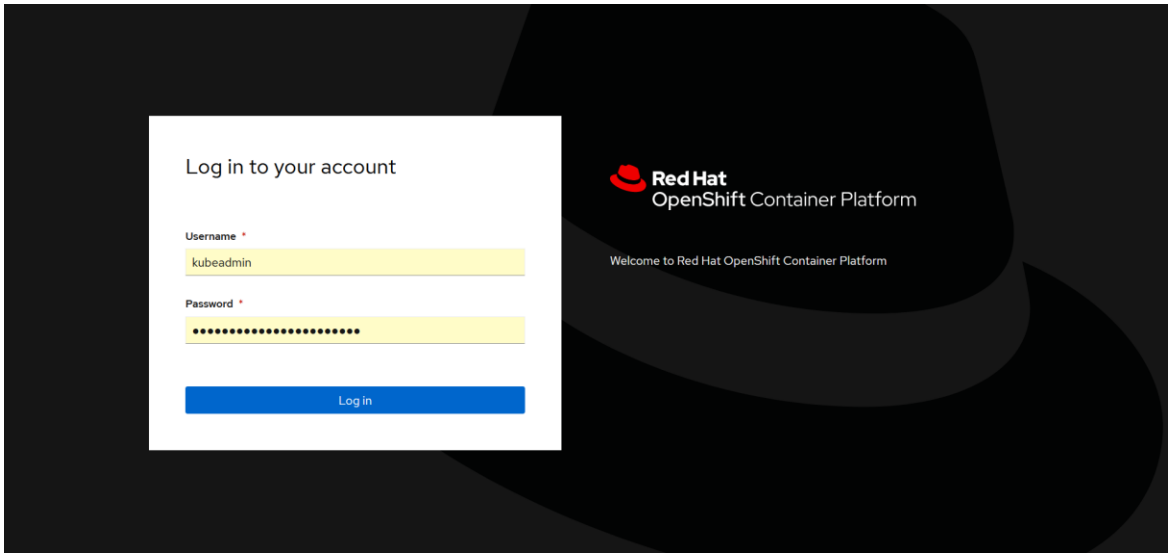

Figure 5-1. OpenShift Web Console View

```
$ oc get nodes
NAME    STATUS   ROLES           AGE   VERSION
nodea   Ready    master,worker   20d   v1.21.1+a620f50
nodeb   Ready    master,worker   20d   v1.21.1+a620f50
nodec   Ready    master,worker   20d   v1.21.1+a620f50
```

Figure 5-2. Node Roles of OpenShift Cluster

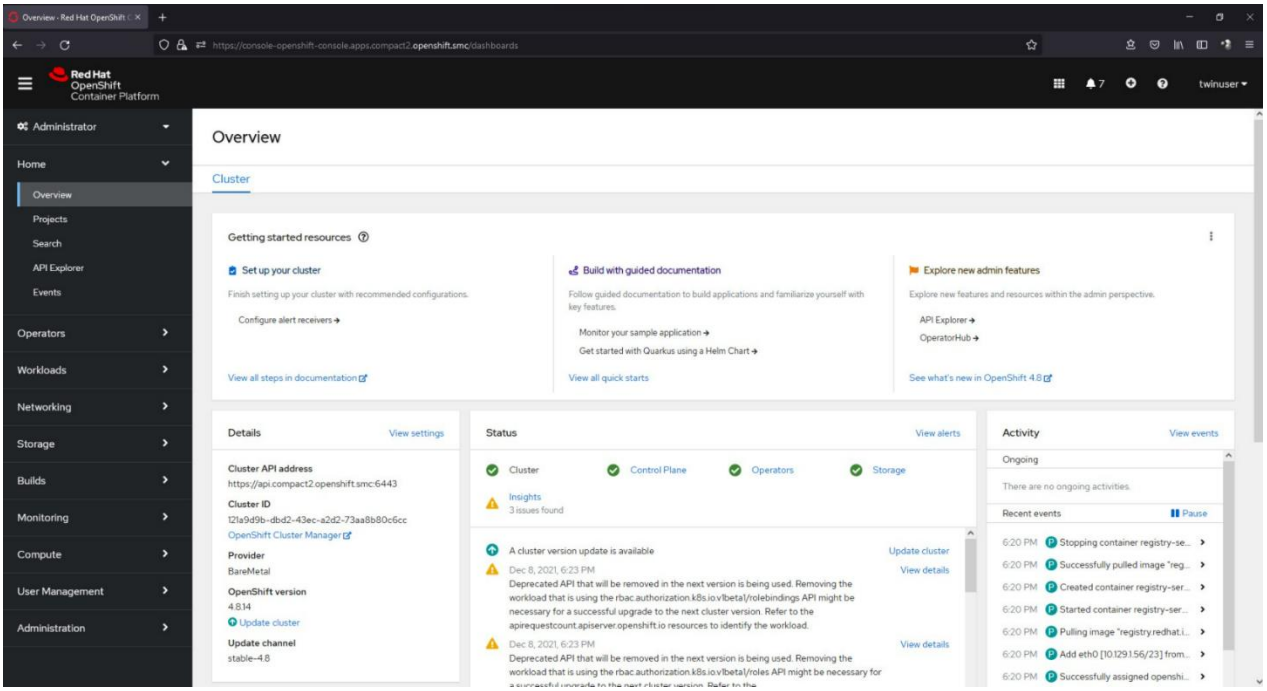

Figure 5-3. OpenShift Administrator Overview

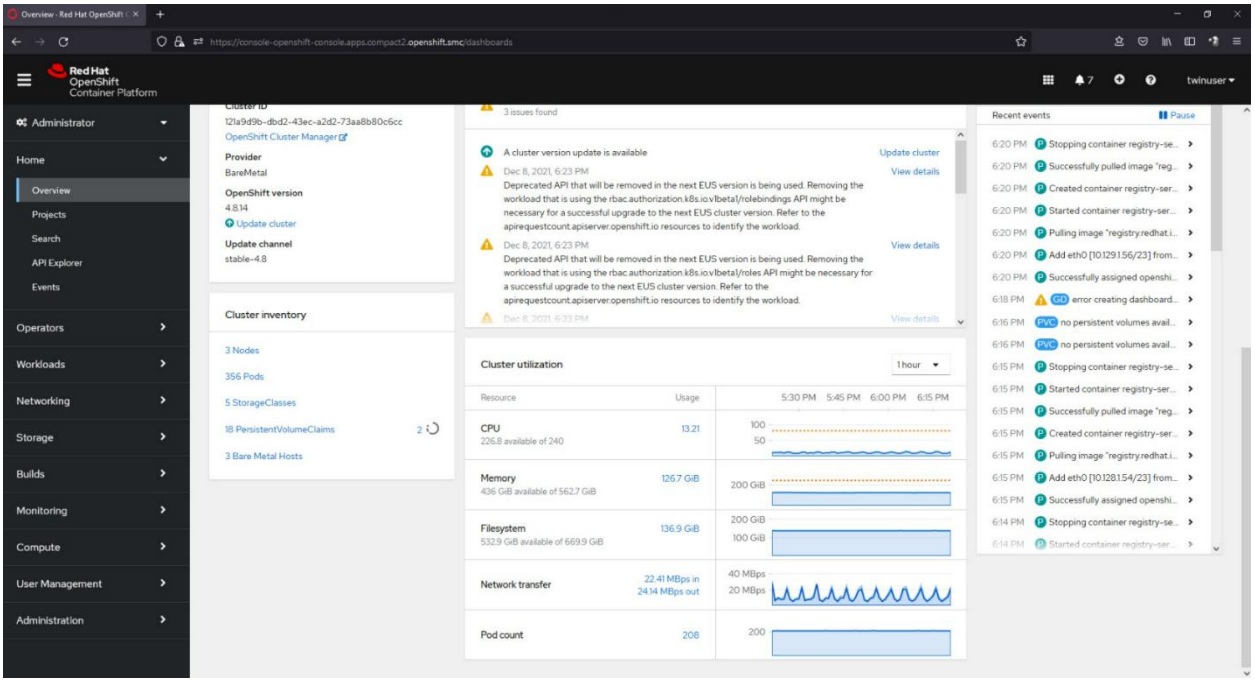

Figure 5-4. OpenShift Cluster Inventory

2. Install httpd-tools for creating an htpasswd-ocp file for a cluster OAuth configuration as the identity provider

   $ sudo dnf -y install httpd-tools

   $ htpasswd -c ~/installer/htpasswd-ocp twinuser

   $ htpasswd -c ~/installer/htpasswd-ocp twinuser

3. Add Identity Provider – HTPasswd

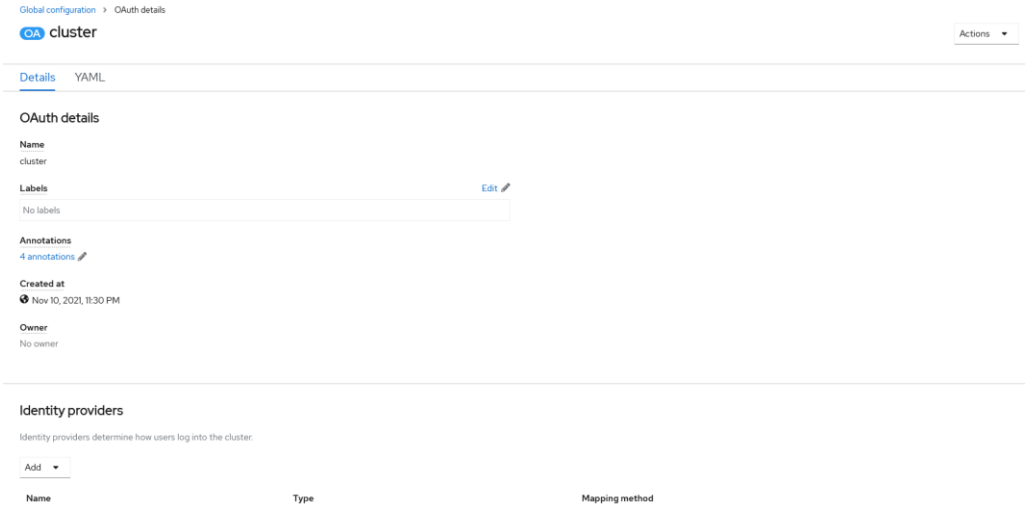

Figure 5-5. Add Identity Provider

4. Upload htpasswd-ocp

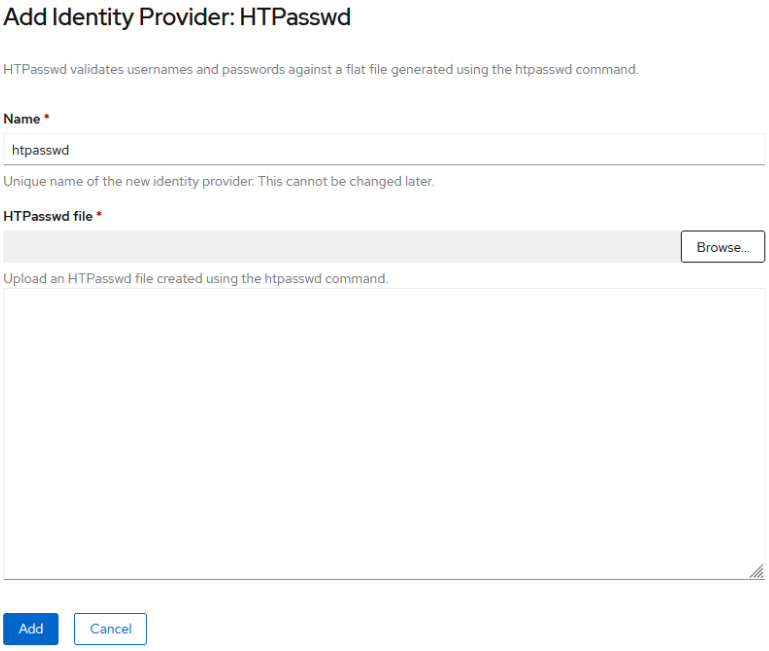

Figure 5-6. Upload Password File

5. Add role-binding to the users
   $ oc adm policy add-cluster-role-to-user cluster-admin twinuser
   $ oc adm policy add-cluster-role-to-user cluster-admin redisuser

6. Install OpenShift Container Storage Operator.

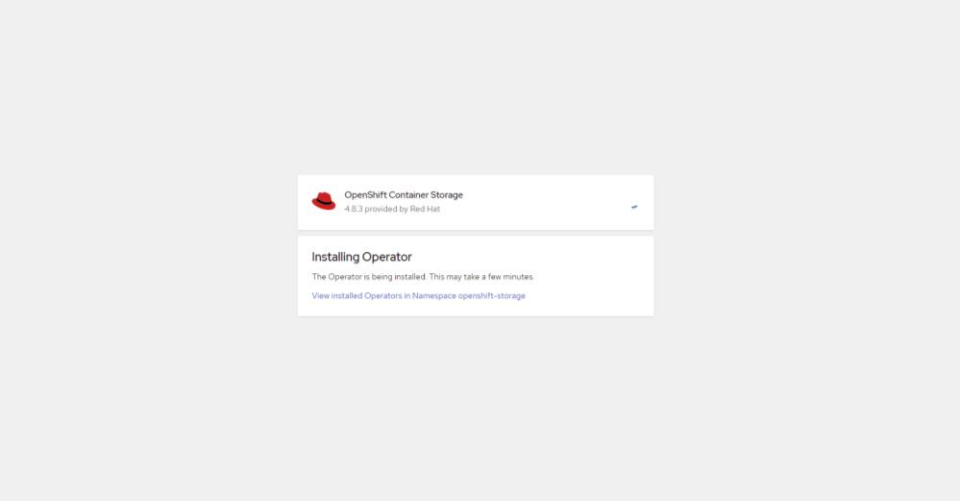

Figure 5-7. Installing Container Storage Operator

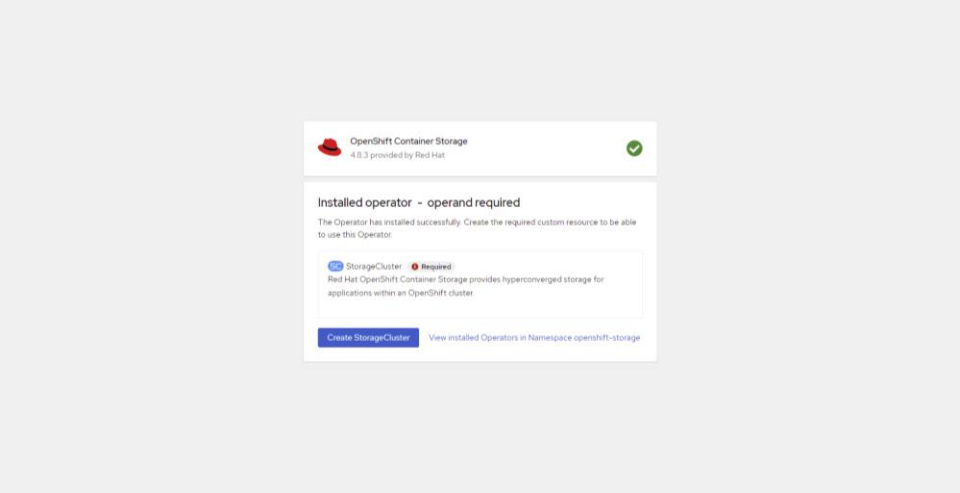

Figure 5-8. Installing Hyperconverged Storage

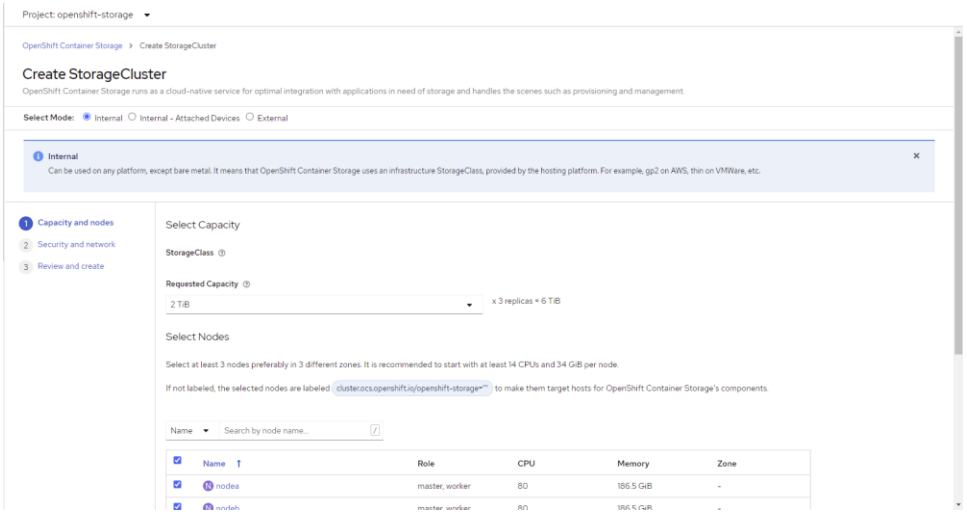

Figure 5-9. Create Storage Cluster

7. Install LocalStorage Operator for Local Storage.

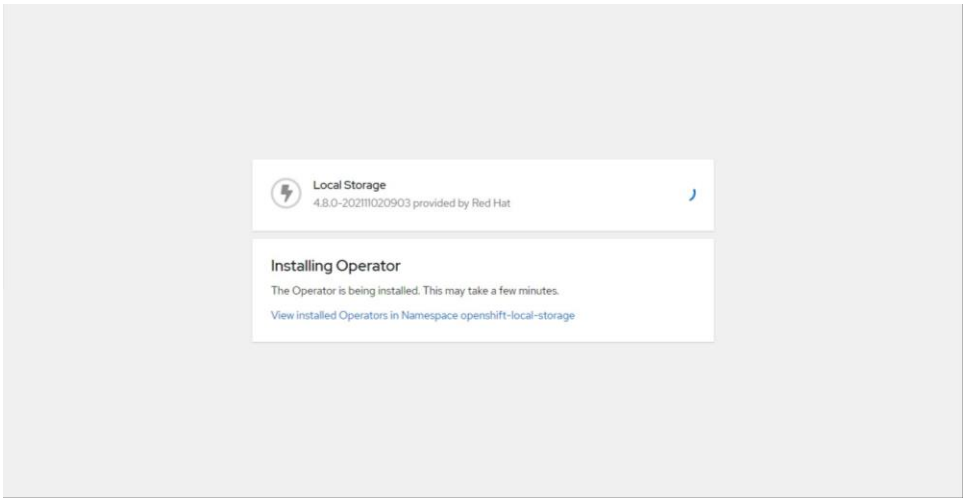

Figure 5-10. Create Storage Cluster

8. Create StorageCluster with the ”Internal – Attached Devices”.

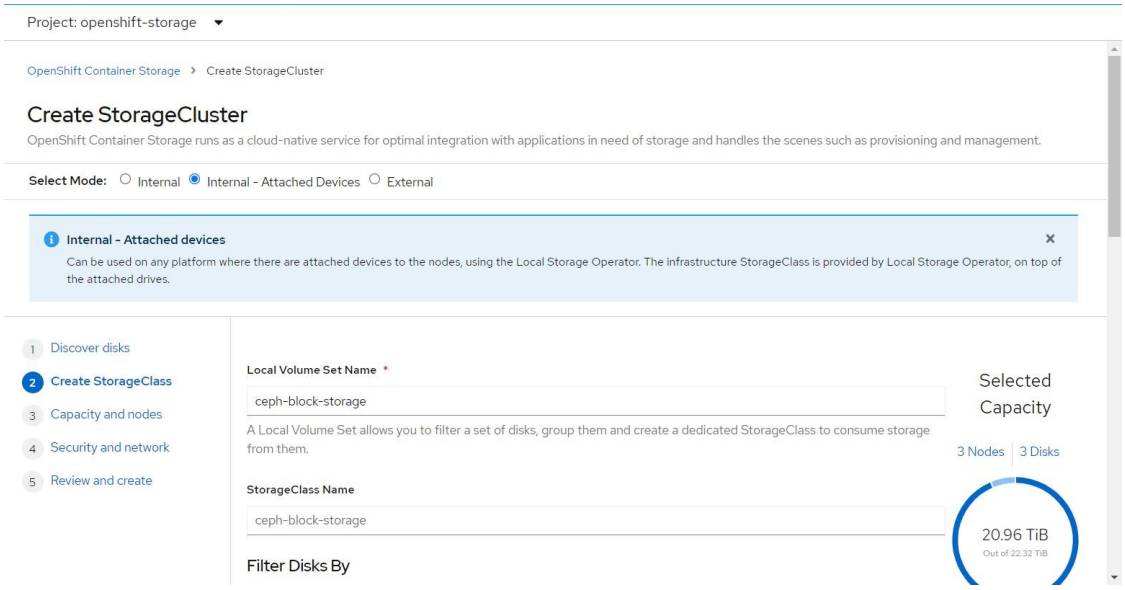

Figure 5-11. Choosing Internal – Attached Devices

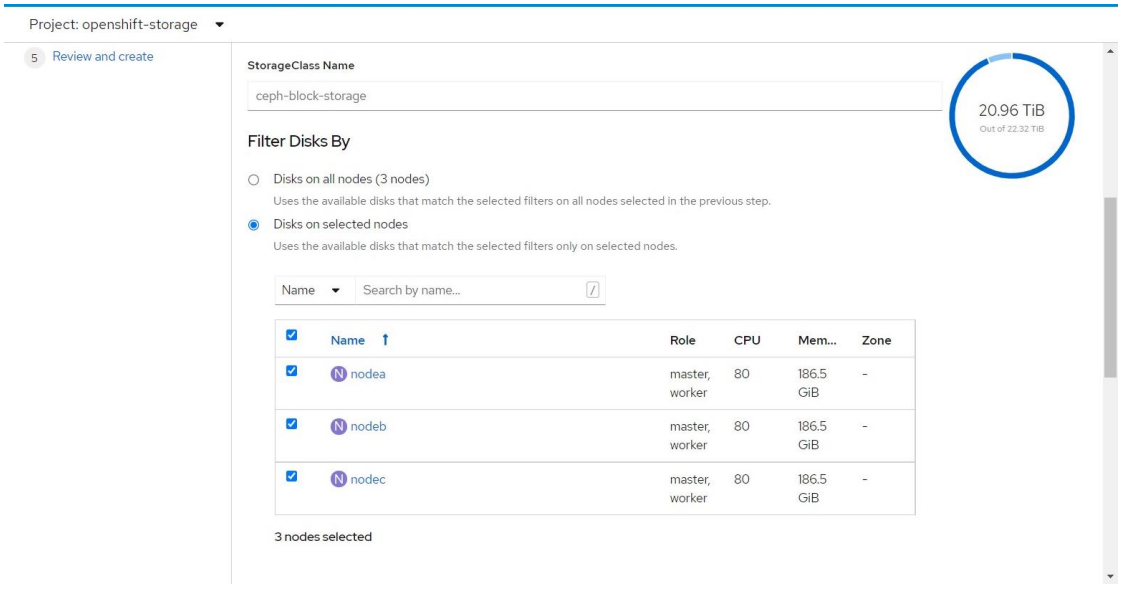

Figure 5-12. Choosing Disks on Selected Nodes

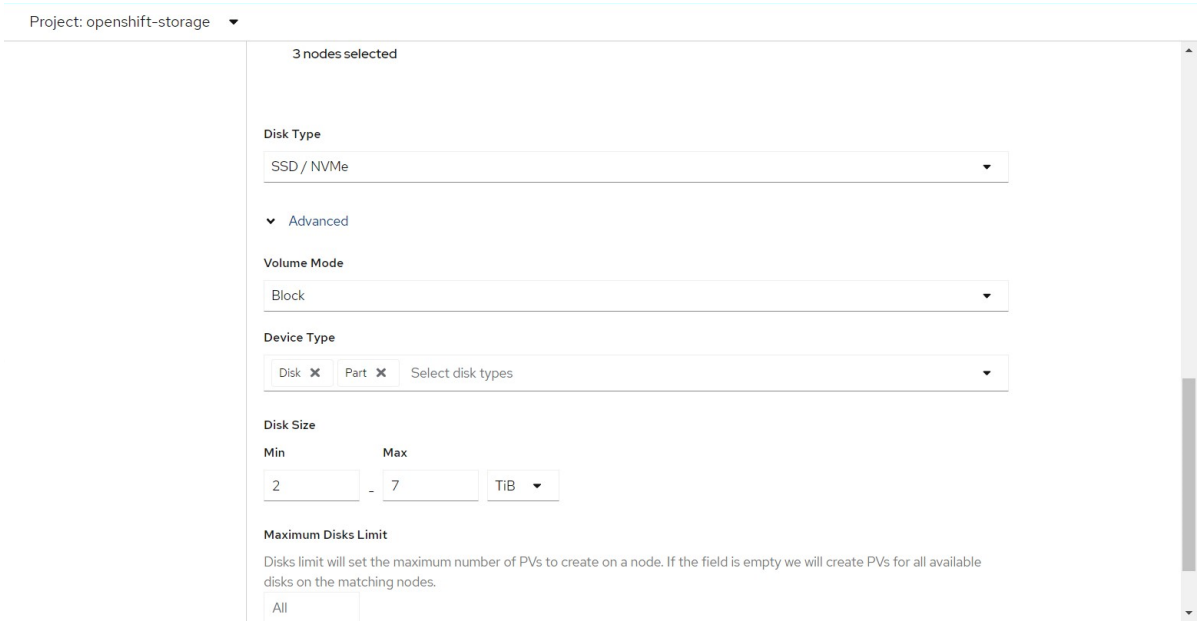

Figure 5-13. Choosing Disks on Selected Nodes (continued)

9. OCS 4.8.3 operator creates Block Pools.

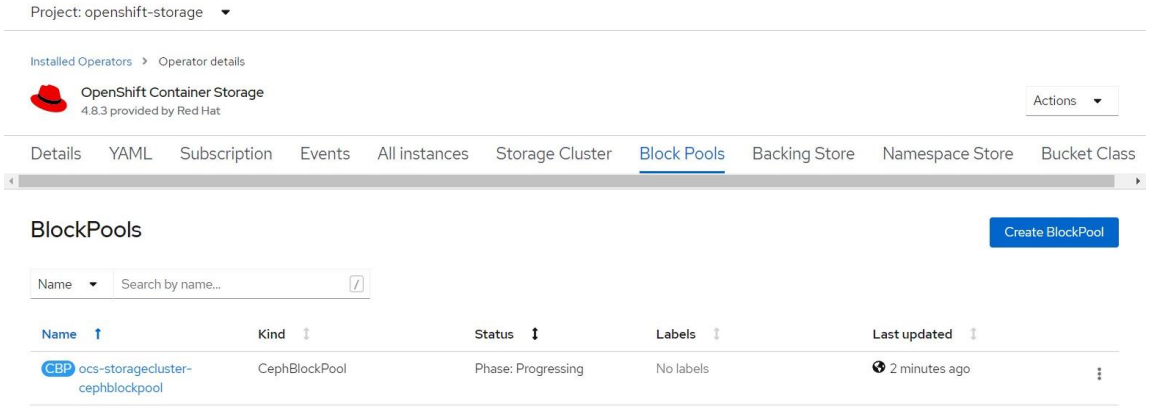

Figure 5-14. Block Pools Creation

10. OCS 4.8.3 operator creates Backing Store for Object Storage service.

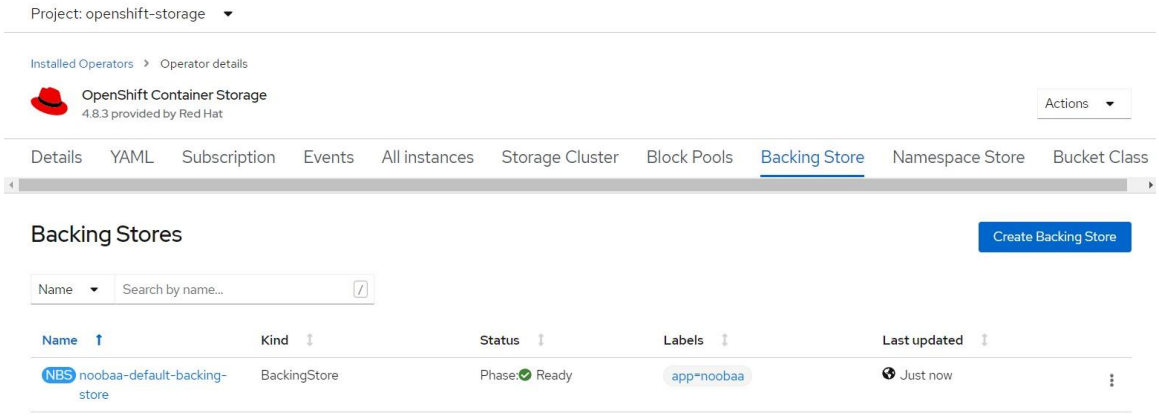

Figure 5-15. Backing Store Creation

11. Check the OpenShift Container Storage Overview.

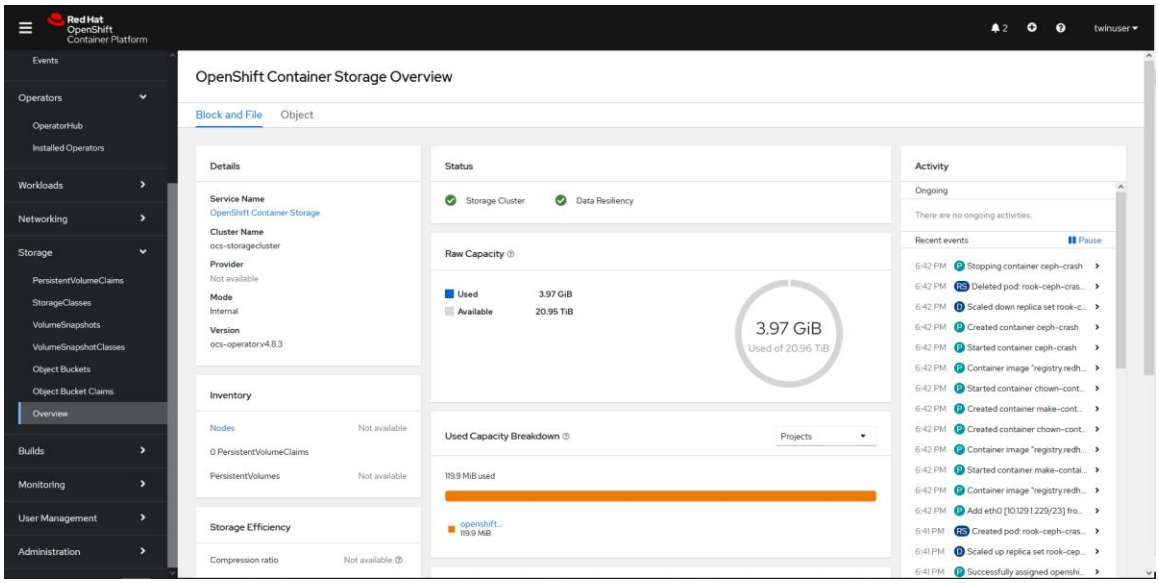

Figure 5-16. OpenShift Container Storage: Block and File View

12. Check Object Service.

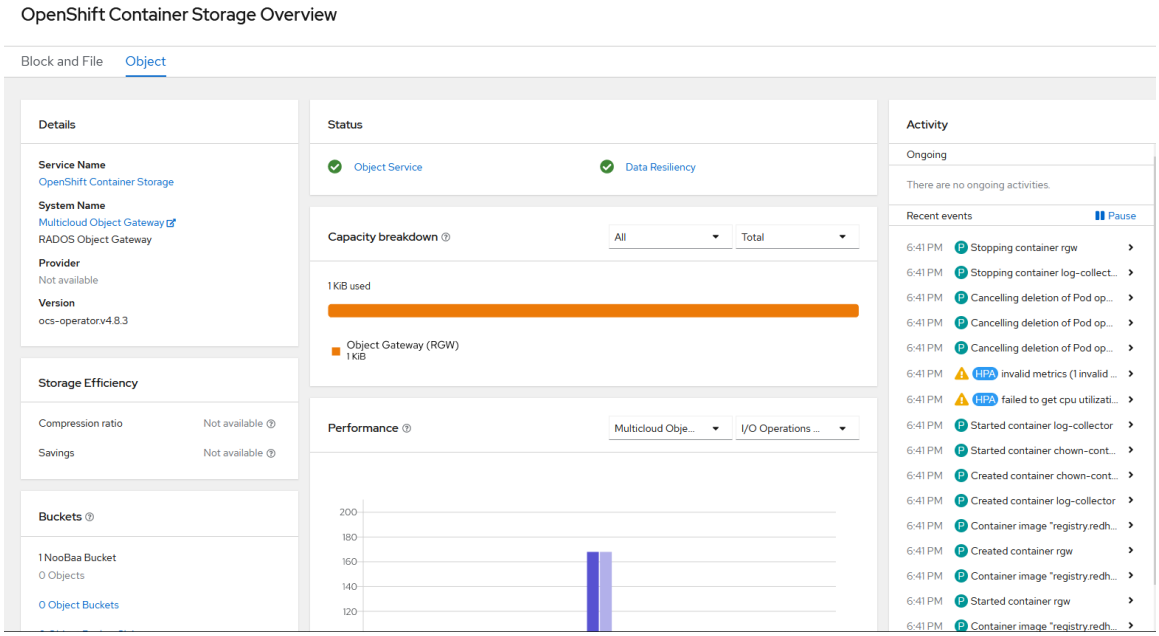

Figure 5-17. OpenShift Container Storage: Object View

13. Configure a persistent storage for the monitoring stack – Prometheus

Running cluster monitoring with persistent storage means that the metrics are stored to a persistent volume (PV) and can survive a pod being restarted or recreated. The is recommended if you require your metrics or altering data to be guarded from data loss. For production environments, it is highly recommended to configure persistent storage. Because of the high I/O demands, it's advantageous to use local storage.

The steps:

a. Create the following YAML manifest for ConfigMap, cluster-monitoring-config, under the namespace of openshift-monitoring. In this case, the file is named `cluster-monitoring-config.yaml`:

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    prometheusK8s:
      volumeClaimTemplate:
       spec:
         storageClassName: ocs-storagecluster-cephfs
         volumeMode: Filesystem
         resources:
           requests:
             storage: 100Gi
```

b. Apply the configuration to create the ConfigMap object:

```
$ oc apply -f cluster-monitoring-config.yaml
```

c. After a few seconds, there will be two persistent storages (PVCs) created for Prometheus under the openshift-monitoring project.

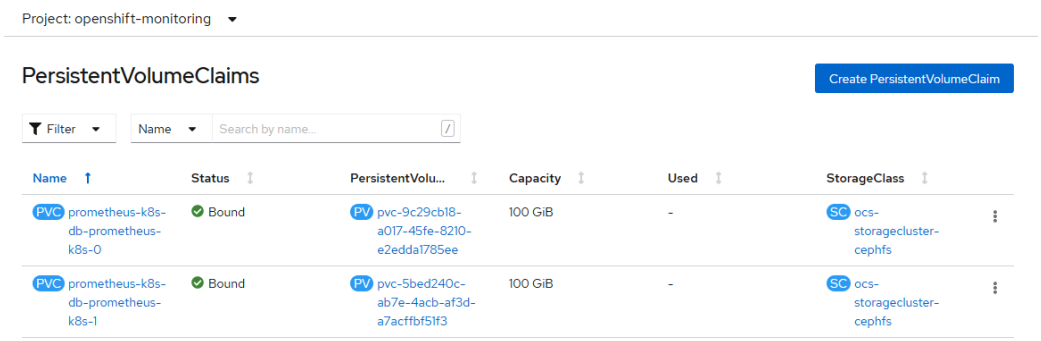

Figure 5-18. 2 PVCs for Prometheus Persistent Storages

Afterwards, the OpenShift monitoring stack is going to collect metrics for each project (namespaces) and the OpenShift cluster.

14. Monitoring – Dashboards

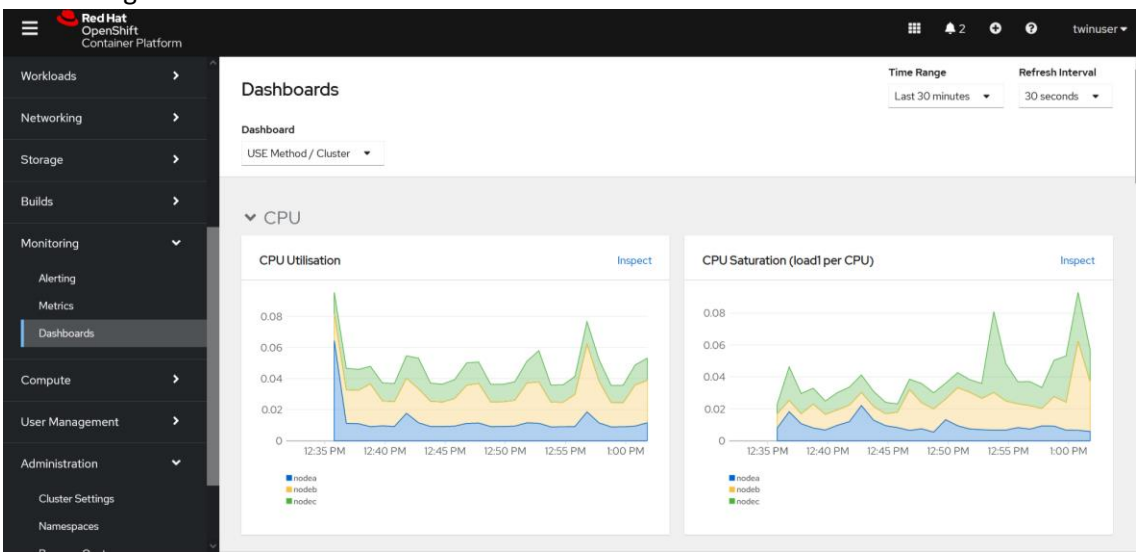

Figure 5-19. CPU Metrics under USE Method / Cluster

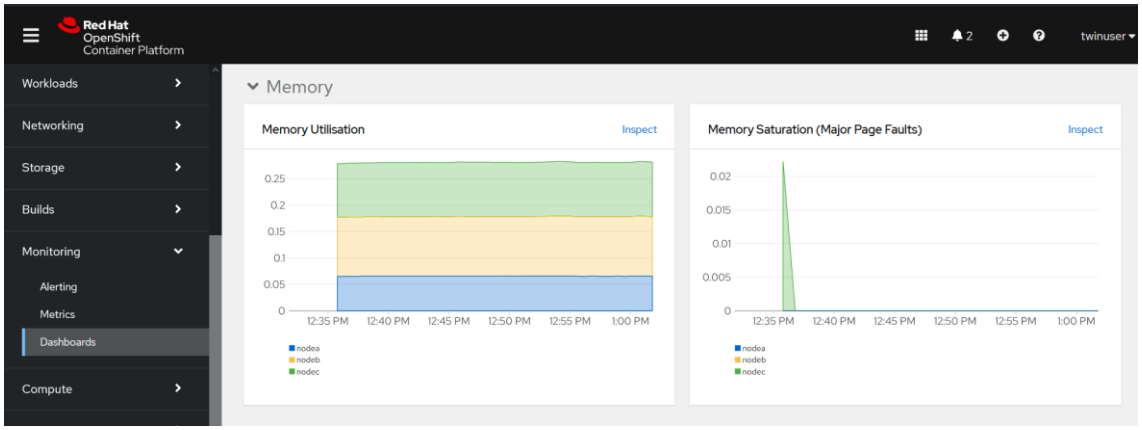

Figure 5-20. Memory Metrics under USE Method / Cluster

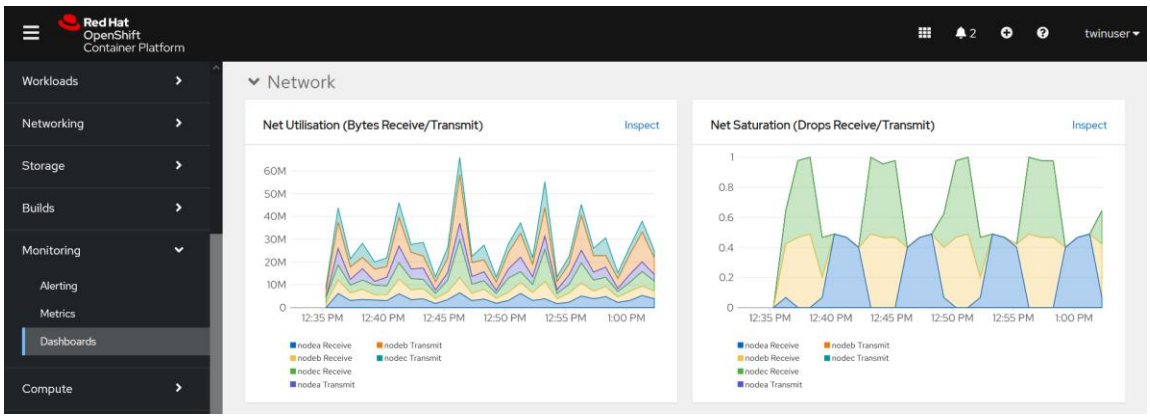

Figure 5-21. Network Metrics under USE Method / Cluster

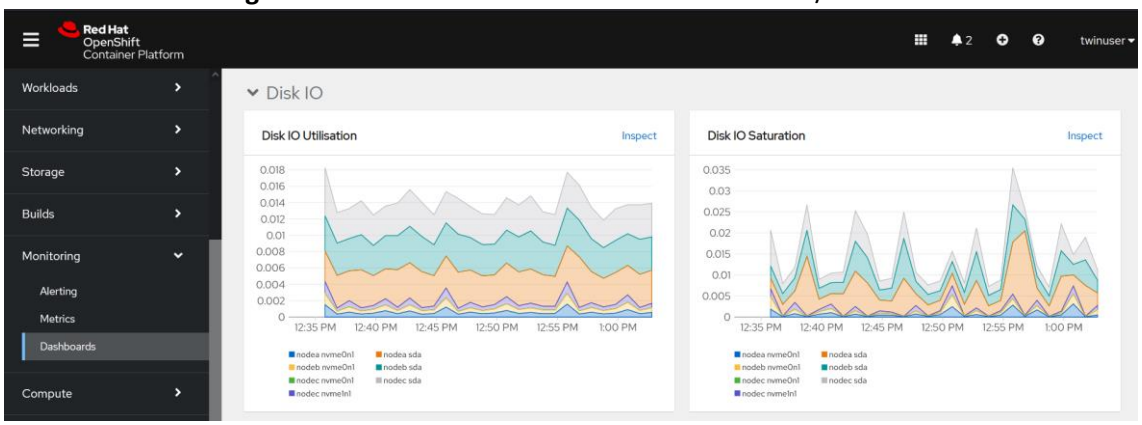

Figure 5-22. Disk IO Metrics under USE Method / Cluster

Appendix

List of Hardware Used in Lab

SYS-2029BT-HNC0R: High-performance building block for OpenShift Container Platform

AOC-M25G-i2SM: Primary NIC: SIOM - Flexible on-motherboard I/O controllers

AOC-S25G-i2S: Secondary NIC: AOC – Industry-standard PCIe I/O card

SSE-C3632S(R): 100GbE QSFP28 Cumulus-interfaced switch (Data/OS Management Switch)

SSE-G3648B(R): 1 GbE 1GBaseT Cumulus-interfaced switch (BMC Management Switch)

List of Supported Network Switches

SSE-G3648B(R): 1 Gigabit 1GBaseT Cumulus-interfaced switches

SSE-X3648S(R): 10 Gigabit Cumulus-interfaces switches

SSE-C3632S(R): 100 Gigabit, 25G, 50G, 10G, QSFP28 Cumulus-interfaced switches

SSE-X3548S(R): 10 Gigabit SFP+ Layer3-interfaced switches

SSE-F3548S(R): 25 Gigabit SFP+ Layer3-interfaced switches

SSE-X3348S(R): 10 Gigabit SFP+ Layer3-interfaced switches

SSE-X3348T(R): 10 Gigabit 10GBaseT RJ45 Layer3-interfaced switches

References

Rapidly Deploy Hybrid Cloud Infrastructure: https://www.supermicro.com/solutions/Solution-Brief_RedHat-OpenShift_202102.pdf

Supermicro BigTwin Multi-Node Servers: https://www.supermicro.com/en/products/bigtwin/

Contacting Supermicro

Headquarters

Address:	Super Micro Computer, Inc. 980 Rock Ave. San Jose, CA 95131 U.S.A.
Tel:	+1 (408) 503-8000
Fax:	+1 (408) 503-8008
Email:	marketing@supermicro.com (General Information) support@supermicro.com (Technical Support)
Web Site:	www.supermicro.com

Europe

Address:	Super Micro Computer B.V. Het Sterrenbeeld 28, 5215 ML 's-Hertogenbosch, The Netherlands
Tel:	+31 (0) 73-6400390
Fax:	+31 (0) 73-6416525
Email:	sales@supermicro.nl (General Information) support@supermicro.nl (Technical Support) rma@supermicro.nl (Customer Support)
Web Site:	www.supermicro.com.nl

Asia-Pacific

Address:	Super Micro Computer, Inc. 3F, No. 150, Jian 1st Rd. Zhonghe Dist., New Taipei City 235 Taiwan (R.O.C)
Tel:	+886-(2) 8226-3990
Fax:	+886-(2) 8226-3992
Email:	support@supermicro.com.tw
Web Site:	www.supermicro.com.tw