Redbooks
ibm.com/redbooks

Red Hat OpenShift on IBM Z Installation Guide

Subhajit Maitra

Eric Marins

Cloud

IBM Z

IBM

Redpaper

IBM Redbooks

Red Hat OpenShift on IBM Z: Installation Guide

September 2020

REDP-5605-00

Note: Before using this information and the product it supports, read the information in “Notices” on page v.

First Edition (September 2020)

This edition applies to Version 4, Release 4, Modification 0 of Red Hat OpenShift Container Platform.

This document was created or updated on September 29, 2020.

Contents

Notices

Trademarks

Preface

This IBM® Redpaper publication provides all the necessary steps to successfully install Red Hat OpenShift 4.4 on IBM Z® or LinuxONE servers.

It also provides an introduction to OpenShift nodes, Red Hat Enterprise Linux CoreOS, and Ansible.

The steps that are described in this paper are taken from the official pages of the Red Hat website.

This IBM Redpaper publication was written for IT architects, IT specialists, and others who are interested in installing Red Hat OpenShift on IBM Z.

Authors

This paper was produced by a team of specialists from around the world working at IBM Redbooks, Poughkeepsie Center.

Subhajit Maitra is a Consulting IT Specialist and member of the IBM America's System Z Hybrid Cloud Technical Sales team based in Hartford, CT. In addition to Red Hat OpenShift on System Z, his expertise includes zCX, IBM App Connect Enterprise, IBM Operational Decision Manager, IBM Integration Bus, IBM MQ, IBM Event Streams (Kafka), IBM API Connect®, IBM DB2®, and CICS® on Systems Z. Subhajit, who has worked in IT for over 27 years, has been an IBM Global Technical Ambassador for Central and Eastern Europe, where he helped IBM clients implement business-critical solutions on IBM Z servers. He has written several IBM Redbooks® publications about z/OS® Container Extensions, IBM Middleware, CICS, and DevOps. He holds a master’s degree in computer science from Jadavpur University, in Kolkata, India, and is an IBM zChampion.

Eric Marins is a Senior IT Architect in Brazil, focused on hybrid cloud solutions, Infrastructure and Platform solutions and competencies, including High Availability, Disaster Recovery, Networking, Linux, and Cloud. Eric works with CIO Private Cloud designing and implementing complex hybrid cloud solutions involving Infrastructure-as-a-Service (IaaS) and Platform-as-a-Service (PaaS) capabilities. Eric has co-authored several IBM Redbooks publications, including *Advanced Networking Concepts Applied Using Linux on IBM System z*, SG24-7995; *Security for Linux on System z*, SG24-7728; *Scale up for Linux on IBM Z*, REDP-5461; *Getting Started with Docker Enterprise Edition on IBM Z*, SG24-8429, and *Getting started with z/OS Container Extensions and Docker*, SG24-8457.

Thanks to the following people for their contributions to this project:

Lydia Parziale
IBM Redbooks, Poughkeepsie Center

Robert Haimowitz
IBM WW Client Experience Center, Poughkeepsie

Sandor Imes, Yuri Gelfand, Fernando Aragao Costa. Livio Sousa, Anderson Augusto, Wilhelm Mild, Stev Glodowski and Silke Niemann
IBM

Murthy Garimella
Red Hat

Now you can become a published author, too!

Comments welcome

Stay connected to IBM Redbooks

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  `https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm`

- Stay current on recent Redbooks publications with RSS Feeds:

  `http://www.redbooks.ibm.com/rss.html`

1

Introduction

Before you install an OpenShift cluster, you must understand some architectural components and how they interact with each other. In this chapter, we describe the nodes that are installed and configured during the OpenShift cluster installation process.

This chapter includes the following topics:

1.1 Overview

Application modernization is a term that describes the process of transitioning existing applications to use new approaches on the cloud. Customers today are seeking innovative, efficient approaches that help them make this transition that is based on business and application complexity.

Business pressures demand faster time to market. Your existing estate includes not only applications, but data, processes, business logic, and user interfaces, all of which must adapt to keep up with new business demands.

Application modernization has the following benefits:

- Improved developer productivity
- Increased operational efficiency
- Reduced cost to build new capabilities
- Expanded capacity that is delivered in a short time

The Red Hat OpenShift Container Platform is a platform that is designed to orchestrate containerized workloads across a cluster of nodes. The platform uses Kubernetes as the core container orchestration engine, which manages the container images and their lifecycle.

Red Hat OpenShift Container Platform 4.2 was released for IBM Z and IBM LinuxONE servers in February of 2020. Since then, organizations can use enterprise server capabilities, such as security features, scalability, and reliability to run cloud-native applications and accelerate their digital transformation.

Red Hat OpenShift is a container-based application platform that is built on open source technologies and principles, such as Kubernetes. You can now deploy and manage applications by way of a centralized web interface (portal) that is managed by OpenShift clusters. Also, your application operations can be automated and have 24 x 7 global support.

Running Red Hat OpenShift on IBM Z brings many advantages to your application, such as:

- Vertical and horizontal scalability
- 99.999% availability
- Great performance
- Reduced latency

Application developers and operations teams can use OpenShift as a powerful tool to easily manage their container environment and see how they are organized. In fact, Red Hat OpenShift makes their lives easier by providing an intuitive web interface.

Use the built-in Red Hat OpenShift, Kubernetes, and IBM Z features to make your OpenShift cluster more highly available and protect your application from downtime.

Some restricted networks might not have access to the internet, even by way of proxy servers. However, you can still install OpenShift Container Platform in these environments. You must download that required software and images and make them available to the disconnected environment. For more information about air-gap installation (disconnected), see this web page.

If your OpenShift nodes have direct internet access, they can:

- Access the OpenShift Infrastructure Providers page to download the installation program.
- Access `quay.io` to obtain the packages that are required to install the cluster.
- Obtain the packages that are required to perform cluster updates.
- Access Red Hat’s software as a service page to perform subscription management.

1.2 Red Hat Enterprise Linux CoreOS

The OpenShift Container Platform uses Red Hat Enterprise Linux CoreOS (RHCOS), a container-oriented operating system. RHCOS is specifically designed for running containerized applications from OpenShift Container Platform and works with new tools to provide fast installation, operator-based management, and simplified upgrades.

This RHCOS technology combines the advantages of Red Hat Enterprise Linux 8 (RHEL) and core libraries with the automated, remote upgrade features from containers to build a robust cloud infrastructure for your applications. The use of RHCOS is advantageous, as shown in the following examples:

- Red Hat OpenShift on Z supports RHCOS for all nodes.
- Red Hat OpenShift is never managed directly; the OpenShift client or web console is used to manage it.
- Red Hat OpenShift is provisioned from a virtual machine image template, or installed by way of ISO that is provided by Red Hat.
- Red Hat OpenShift is managed entirely by the OpenShift cluster. You do not create more accounts on it.
- RHCOS updates are performed during OpenShift upgrades as part of an amazingly seamless and hands-off upgrade process.
- Red Hat OpenShift is more secure. You do not have direct access by using a root ID, even on the virtual machine (VM) console.

1.3 Red Hat OpenShift architecture

The Red Hat OpenShift architecture that is shown in Figure 1-1 on page 4 provides an application platform for developing and managing containerized applications. Control, worker, and bootstrap nodes are deployed as Linux guests on the IBM Z platform.

The minimum OpenShift architecture consists of five Linux guests (bootstrap, three control nodes, and one worker node) that are deployed on top of IBM z/VM® 7.1. In our scenario, we installed a bastion node to provide DNS, FTP, and load balancer functions because these services were not available in our network.

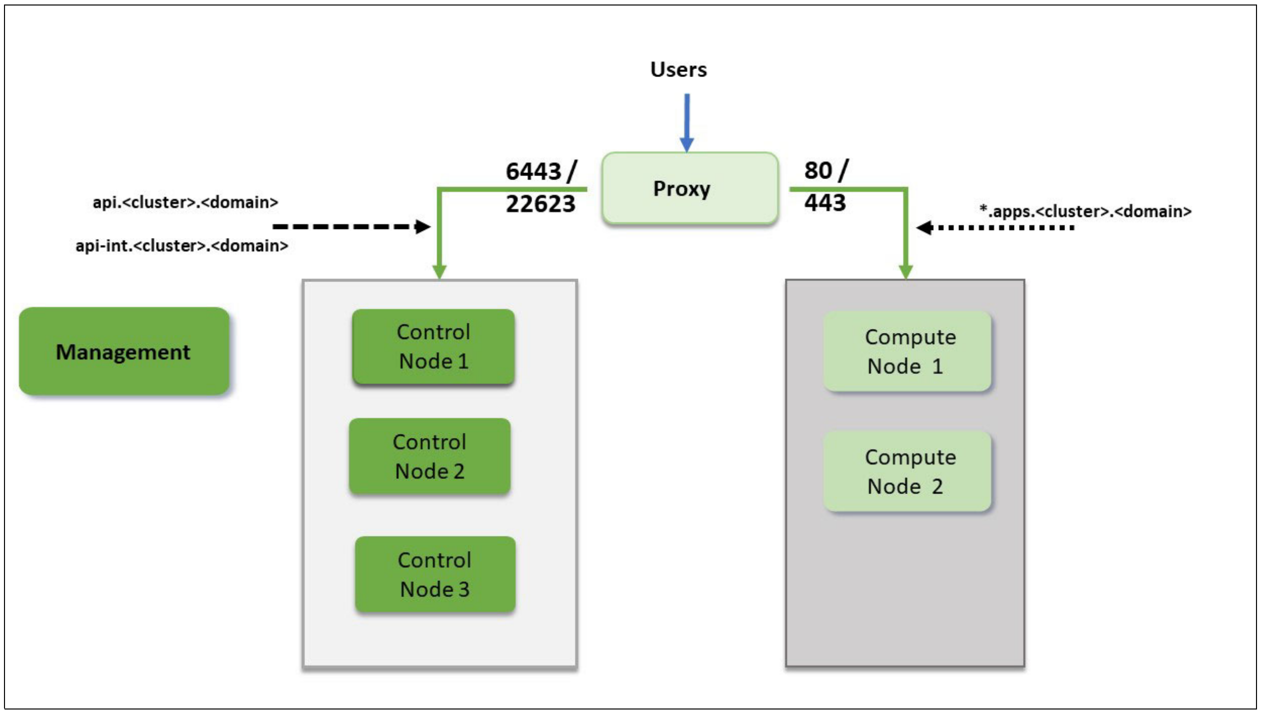


Figure 1-1 OpenShift architecture

For more information about Red Hat OpenShift software and hardware requirements, see Chapter 2, "Planning the environment" on page 7.

Starting with Red Hat OpenShift releases 4.2, IBM Z supports RHCOS only as an operating system for the entire cluster. Therefore, the bootstrap, control plane, and compute nodes are be installed by using RHCOS, *not* Red Hat Enterprise Linux.

1.4 Bootstrap host

The bootstrap node host uses RHCOS. This host is responsible for deploying the OpenShift Container Platform cluster on the control-plane machines. The node is a temporary node that is used to create the controller nodes that make up the control-plane. The control-plane nodes then create the worker nodes. After the cluster initializes, the bootstrap node can be released if you need to use the resources to perform another installation or increase cluster capacity.

1.5 Control plane hosts

The control-plane hosts (also known as controller hosts) are using Red Hat Enterprise Linux CoreOS (RHCOS), a container-oriented operating system designed for running containers.

These nodes use Kubernetes services in the background and they are responsible for managing and controlling the entire OpenShift cluster while taking advantage of this new OS technology to deploy container workloads on IBM Z.

A list of the essential functionality provided to the cluster includes:

- API server
- Authentication
- Replication
- Schedulers
- etcd
- Monitoring

Although you can run containers on these machines, you should not run pods on the controller nodes. A *pod* is a group of containers that operate together and are deployed on the same host. The terms *controller* and *control-plane* are used interchangeably.

For the controller API endpoints, a load balancer is needed to expose the api and api-int functions on ports 6443 and 22623. We chose HAProxy to act as a load balancer for our cluster. Appliance-based load balancers can be used, such as F5 or Datapower.

HAProxy is an open source Transmission Control Protocol (TCP) or HTTP load balancer. You can use HAProxy to add load balancing capabilities to your OpenShift architecture if you do not have an appliance acting as a load balancer available.

HAProxy is an operating system package that you must install by using a tool such, as YUM. For more information about installing HAProxy, see Chapter 3, “Installation” on page 15.

For more information about HAProxy, see this web page. this web page

1.6 Compute nodes

The compute node hosts (also known as *worker nodes*) also use Red Hat Enterprise Linux CoreOS (RHCOS).

The compute nodes are known as the worker nodes, where the Kubernetes workloads are deployed. They are responsible for running workloads for the cluster users.

A load balancer is configured on the bastion node to spread load across these VMs for ports 80 and 443, which exposes the wildcard DNS and *.apps.

The compute nodes make their capacity known to the control-plane nodes.

1.7 Bastion node

The bastion node can be any Linux distribution (SLES, RHEL, or Ubuntu) and it can be on any platform (x86, Power or Z). Also, it is not mandatory if the required services are from the enterprise FTP, load balancer, or DNS.

In our architecture, we chose to use the bastion node to enable the build process. It is accessed by way of SSH and It also hosts some network services to help build the OpenShift cluster process.

In our lab environment, the bastion node runs Red Hat Enterprise Linux 8 (RHEL8) on an IBM Z server, and it is used to host the scripts, files, and tools to provision the bootstrap, control-plane, and compute nodes.

After the deployment, we recommend keeping the bastion node for the following reasons:

- Administrative node for the cluster
- Load balancer capability (HAProxy)
- Local DNS (named)
- Share Files (FTP)

Within the OpenShift environment, a load balancer is required for accessing the controller nodes and the worker nodes. The HAProxy is enabled on a bastion node to use load balancing and is configured with load-balancing to point to the OpenShift control-plane and OpenShift workers.

A load balancer is configured to spread load across these VMs for ports 6443 and 22623, which exposes the api and api-int functions, while ports 80 and 443 are configured to point to the compute nodes (see Figure 1-1 on page 4).

2

Planning the environment

This chapter provides general guidance for planning the installation of Red Hat OpenShift on an IBM Z or LinuxONE platform. To effectively plan an installation, you must select the hardware, number of nodes and network, and prepare your environment for the OpenShift installation.

This chapter includes the following topics:

- 2.1, “Required virtual machine resources” on page 8
- 2.2, “Users and disks” on page 9
- 2.3, “Required DNS IP addresses and names” on page 13

2.1 Required virtual machine resources

The physical infrastructure that is required to deploy a Red Hat OpenShift production instance onto an IBM Z requires the following minimum specifications:

- For the installation of OpenShift Container Platform version 4.4, ensure use of one of the following IBM servers:
  - IBM Z: z13®, z13s®, all z14 models, all z15™ models.
  - LinuxONE: all models.

  Other hardware requirements include:
  - One LPAR with 3 IFLs that supports SMT2.
  - One OSA or RoCE network adapter.
- Operating system: One instance of IBM z/VM 7.1.
- Network connectivity: A z/VM VSwitch is set up.
- Minimum OpenShift Container Platform deployment on IBM Z requires several z/VM virtual machines (VMs), between them:
  - One bootstrap machine
  - Three control plane machines (master nodes)
  - Two compute machine (worker nodes)
  - One bastion server (DNS, FTP, and HAProxy).

Our environment contains seven Linux guests, as shown in Figure 2-1.

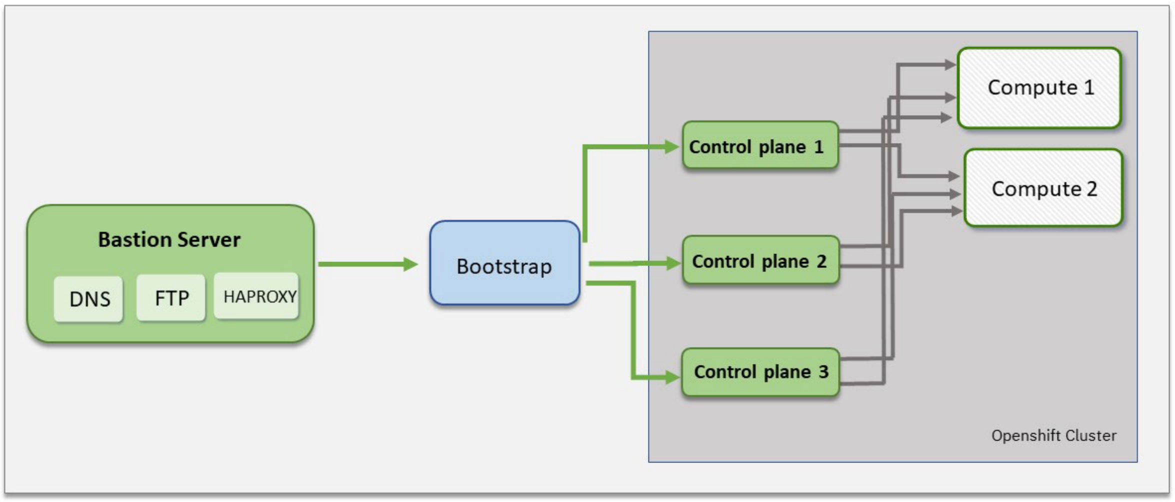


Figure 2-1 Linux guests

The required hardware resources are listed in Table 2-1.

Table 2-1 Hardware resources

Role	Node number	Operating system	vCPU number	Memory	Storage
Bootstrap	1	RHCOS	4	16 GB	120 GB
Control-plane (master node)	3	RHCOS	4	16 GB	120 GB
Compute (worker nodes)	2	RHCOS	2	8 GB	120 GB
Bastion	1	RHEL 8	2	8 GB	120 GB

In addition to the Red Hat OpenShift hardware requirements, you must create persistent volumes in the Red Hat OpenShift environment to store images for the container registry or customer workloads. We added 120 GB for the bastion node to store these files.

Also, disk storage for the z/VM guest VMs includes IBM FICON® attached disk storage (DASD), full pack minidisks, or dedicated DASD. To reach the minimum size for the RHCOS installations, you need extended address volumes (EAV). If available, use HyperPAV to ensure optimal performance.

2.2 Users and disks

On your z/VM instance, you must define the following z/VM user IDs and default Linux profile:

- `RDBKO3M1`
- `RDBKO3M2`
- `RDBKO3M3`
- `RDBKO3W1`
- `RDBKO3W2`
- `DFLTLNX` (default profile)

z/VM user ID management is not described in this IBM Redpaper publication. Complete the following steps to define the user ID and default Linux profile:

Note: Use your preferred x3270 terminal to access the z/VM console.

1. Connect to your z/VM LPAR by using your x3270 terminal and create the following files by using XEDIT:
   - RDBKO3M1.DIRECT as shown in Example 2-1 on page 10.
   - RDBKO3M2.DIRECT as shown in Example 2-2 on page 10.
   - RDBKO3M3.DIRECT as shown in Example 2-3 on page 10.
   - RDBKO3W1.DIRECT as shown in Example 2-4 on page 10.
   - RDBKO3W2.DIRECT as shown in Example 2-5 on page 10.
   - DFLTLNX.DIRECT as shown in Example 2-6 on page 11.

Note: Remember to update the NICDEF tags to fit your network. In our environment, we defined VSWITCH1 and VLAN number 0001.

Example 2-1 RDBKO3M1.DIRECT file

```
USER RDBKO3M1 REDHAT 16G 16G G
   INCLUDE DFLTLNX
   CPU 00 BASE
   CPU 01
   CPU 02
   CPU 03
   IPL 100
   POSIXINFO   UID 100754
   NICDEF 1000 TYPE QDIO LAN SYSTEM VSWITCH1
   NICDEF 1000 VLAN 0001
```

Example 2-2 RDBKO3M2.DIRECT file

```
USER RDBKO3M2 REDHAT 16G 16G G
   INCLUDE DFLTLNX
   CPU 00 BASE
   CPU 01
   CPU 02
   CPU 03
   IPL 100
   POSIXINFO   UID 100754
   NICDEF 1000 TYPE QDIO LAN SYSTEM VSWITCH1
   NICDEF 1000 VLAN 0001
```

Example 2-3 RDBKO3M3.DIRECT file

```
USER RDBKO3M3 REDHAT 16G 16G G
   INCLUDE DFLTLNX
   CPU 00 BASE
   CPU 01
   CPU 02
   CPU 03
   IPL 100
   POSIXINFO   UID 100754
   NICDEF 1000 TYPE QDIO LAN SYSTEM VSWITCH1
   NICDEF 1000 VLAN 0001
```

Example 2-4 RDBKO3W1.DIRECT file

```
USER RDBKO3W1 REDHAT 8G 8G G
   INCLUDE DFLTLNX
   CPU 00 BASE
   CPU 01
   IPL 100
   POSIXINFO   UID 100754
   NICDEF 1000 TYPE QDIO LAN SYSTEM VSWITCH1
   NICDEF 1000 VLAN 0001
```

Example 2-5 RDBKO3W2.DIRECT file

```
USER RDBKO3W2 REDHAT 8G 8G G
   INCLUDE DFLTLNX
   CPU 00 BASE
   CPU 01
   IPL 100
   POSIXINFO   UID 100754
   NICDEF 1000 TYPE QDIO LAN SYSTEM VSWITCH1
   NICDEF 1000 VLAN 0001
```

Example 2-6 DFLTLNX.DIRECT file

```
PROFILE DFLTLNX
CLASS G
STORAGE 256M
MAXSTORAGE 2G
CPU 00 BASE
CRYPTO    APVIRTUAL
IPL CMS
MACHINE ESA 120
OPTION  APPLMON
CONSOLE 0009 3215 T LNXCONS
SPOOL 000C 2540 READER *
SPOOL 000D 2540 PUNCH A
SPOOL 000E 1403 A
LINK MAINT 0190 0190 RR
LINK MAINT 019D 019D RR
LINK MAINT 019E 019E RR
```

2. Use the following **`DIRM ADD`** commands to add the new Linux user IDs and default Linux profile into the z/VM directory:

   ```
   DIRM ADD DFLTLNX
   DIRM ADD RDBKO3M1
   DIRM ADD RDBKO3M2
   DIRM ADD RDBKO3M3
   DIRM ADD RDBKO3W1
   DIRM ADD RDBKO3W2
   ```

   Important: You must create the DFLTLNX profile first. Otherwise, the ID creation fails.

3. To check whether the user IDs were successfully created, run the command that shown in Example 2-7.

   Example 2-7 Review the new Linux guest IDs

   ```
   dirm for rdbko3m1 review
   DVHXMT1191I Your REVIEW request has been sent for processing to DIRMAINT
   DVHXMT1191I at LNXVM64 via DIRMSAT2.
   Ready; T=0.01/0.01 21:35:27
    DVHREQ2288I Your REVIEW request for RDBKO3M1 at *
    DVHREQ2288I has been accepted.
    DVHREQ2289I Your REVIEW request for RDBKO3M1 at *
    DVHREQ2289I has completed; with RC = 0.
   RDR FILE 0250 SENT FROM DIRMAINT PUN WAS 2626 RECS 0040 CPY  001 A NOHOLD
   NOKEEP
   ```

The command that is shown in Example 2-7 on page 11 takes the user directory for the guest and puts the results into the reader. Run the **peek** command to read this file. In our example, the file that was created in the reader has the number `0250`. Therefore, we ran the **peek 0250** command and received the output that is shown in Example 2-8.

Example 2-8 Output of peek command

```
USER RDBKO3M1 REDHAT 16G 16G G
DVHRXV3355I The following records are included from profile: DFLTLNX
   PROFILE DFLTLNX
   CLASS G
   STORAGE 256M
   MAXSTORAGE 2G
   CPU 00 BASE CPUID A06400
   IPL CMS
   MACHINE ESA 20
   OPTION APPLMON
   CONSOLE 0009 3215 T LNXCONS
   SPOOL 000C 2540 READER *
   SPOOL 000D 2540 PUNCH A
   SPOOL 000E 1403 A
   LINK $MAINT 0190 0190 RR
   LINK $MAINT 019D 019D RR
   LINK $MAINT 019E 019E RR
   LINK LINUXMON 0291 0192 RR
   LINK LINUXMON 0192 0191 RR
*DVHOPT LNK0 LOG1 RCM1 SMS0 NPW1 LNGAMENG PWC20070129 CRCèï
DVHRXV3355I The preceding records are included from profile: DFLTLNX
   CPU 00 BASE
   CPU 01
   CPU 02
   CPU 03
   IPL 100
   NICDEF 1000 TYPE QDIO LAN SYSTEM VSWITCH1
   NICDEF 1000 VLAN 0001
*DVHOPT LNK0 LOG1 RCM1 SMS0 NPW1 LNGAMENG PWC20190604 CRC")
DVHREV3356I The following are your user option settings:
DVHREV3356I Links DISABLED Logging ON RcvMsg ON Smsg OFF NeedPW ON
DVHREV3356I Lang
AMENG
```

Run the commands that are shown in Example 2-7 on page 11 and Example 2-8 against each Linux user ID.

4. Define an IPL disk for each guest ID. We use a DASD model A that supports large disks. In our example, we used the following commands to define the IPL disk for each OpenShift node:

```
DIRM FOR RDBKO3M1 AMDISK 100 3390 AUTOG 271000 MYPOOL
DIRM FOR RDBKO3M2 AMDISK 100 3390 AUTOG 271000 MYPOOL
DIRM FOR RDBKO3M3 AMDISK 100 3390 AUTOG 271000 MYPOOL
DIRM FOR RDBKO3W1 AMDISK 100 3390 AUTOG 271000 MYPOOL
DIRM FOR RDBKO3W2 AMDISK 100 3390 AUTOG 271000 MYPOOL
```

These commands add a virtual disk with 271000 cylinders (approximately 180 GB) to each Linux guest. The disk number that is chosen is `100`.

2.3 Required DNS IP addresses and names

Red Hat OpenShift depends on a domain name server (DNS) to function correctly. You must define a wildcard zone to register all OpenShift records, which are configured under the bastion server. This wildcard zone resolves all names to the IP address of the Red Hat OpenShift cluster application. This way, all applications that run within Red Hat OpenShift are routed through the load balancer, as needed.

A wildcard DNS record (in our case, `*.apps.rdbk1.pok.ibm.local`) points to the load balancer server that targets the machines that run the Ingress router pods, which are the worker nodes by default. This record must be resolvable by clients that are external to the cluster and from all the nodes within the cluster. It allows users to access applications inside the OpenShift cluster.

Choosing a DNS cluster and domain name are important tasks. Therefore, ensure that you use a consistent naming convention when deploying your OpenShift cluster. In this section, you see an example of how to use a good naming convention.

The physical infrastructure of our OpenShift cluster is shown in Figure 2-2.

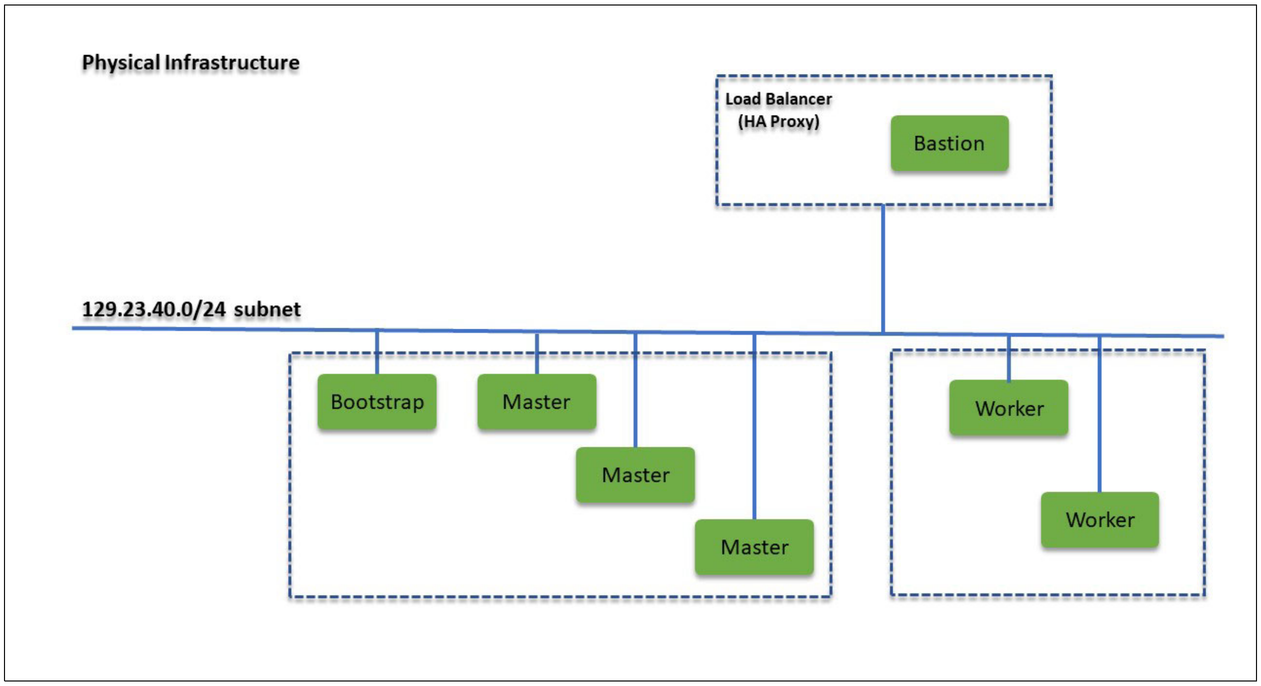


Figure 2-2 Physical Infrastructure

You can use the following format for DNS naming standards:

```
HostName.ClusterName.SubDomain.DomainName
```

Where:

HostName:	Name of the VM or host, for example, `control-plane-1`
ClusterName:	OpenShift cluster name, for example, `rdbk1`
SubDomain:	Subdomain of the OpenShift deployment, for example, `pok`
DomainName:	Domain name of the OpenShift deployment, for example, `ibm.local`

In our example, the fully qualified domain name (FQDN) is:

```
control-plane-1.rdbk1.pok.ibm.local
```

Table 2-2 lists our example deployment. Use your own values in your deployment.

Table 2-2 DNS Records for OpenShift implementation

DNS Description	DNS Example Name	DNS Example IP address / CNAME
DNS Reverse Lookup for OpenShift network	N/A	129.40.23.0/24
API record	api.rdbk1.pok.ibm.local	129.40.23.109 (IP address of Bastion)
API int record	api-int.rdbk1.pok.ibm.local	129.40.23.109 (IP address of Bastion)
Bastion server (HAProxy)	rdbkbas3.rdbk1.pok.ibm.local	129.40.23.109
Bootstrap Server	rdbko3b1.rdbk1.pok.ibm.local	129.40.23.110
Control plane 1 Server	rdbko3m1.rdbk1.pok.ibm.local	129.40.23.111
Control plane 2 Server	rdbko3m2.rdbk1.pok.ibm.local	129.40.23.112
Control plane 3 Server	rdbko3m3.rdbk1.pok.ibm.local	129.40.23.113
Compute 1 Server	rdbko3w1.rdbk1.pok.ibm.local	129.40.23.114
Compute 2 Server	rdbko3w2.rdbk1.pok.ibm.local	129.40.23.115
Application Wildcard DNS (Load Balancer)	*.apps.rdbk1.pok.ibm.local	129.40.23.109 (IP address of Bastion)
etcd Node 0	etcd-0.rdbk1.pok.ibm.local	129.40.23.111 (IP address of Control Plane 1)
etcd Node1	etcd-1.rdbk1.pok.ibm.local	129.40.23.112 (IP address of Control Plane 2)
etcd Node 2	etcd-2.rdbk1.pok.ibm.local	129.40.23.113 (IP address of Control Plane 3)
etcd Service Record Node 0	etcd-server-ssl._tcp.rdbk1.pok.ibm.local	etcd-0.rdbk1.pok.ibm.local
etcd Service Record Node 1	etcd-server-ssl._tcp.rdbk1.pok.ibm.local	etcd-1.rdbk1.pok.ibm.local
etcd Service Record Node 2	etcd-server-ssl._tcp.rdbk1.pok.ibm.local	etcd-2.rdbk1.pok.ibm.local

As described in “Bastion server setup” on page 16, we use the information that is listed in Table 2-2 to set up our DNS service.

3

Installation

This chapter describes how to install Red Hat OpenShift on IBM Z and Includes the following topics:

Important: Red Hat OpenShift uses kubelet client log-ins that must be rotated periodically for security purposes. The initial log-ins that are created during the installation process expire 24 hours after they are created. Red Hat OpenShift mainly automates the rotation process and starts the initial log-in rotation. It is important not to restart any of the Red Hat OpenShift cluster virtual machines (VMs) or the bastion VM until the first rotation is complete.

3.1 Bastion server setup

The bastion server is an instance that is built with RHEL 8 and can be accessed by using SSH for administrative tasks. After it is set up, the bastion server acts as load balancer front end to the OpenShift APIs (internal and external) and the OpenShift router. Red Hat does not make an official recommendation as to which load balancer to use. Therefore, we selected the IBM HAProxy package that supports Server Name Indication (SNI).

The first thing that you must do is provision an RHEL 8 virtual machine (VM). This IBM Redpaper publication does *not* describe this process. For more information about installing RHEL 8 for IBM Z, see this web page.

After you provision an RHEL 8 VM, we outline bastion server setup, which consists of the following steps:

- Install a load balancer (HAProxy)
- Install bind (DNS Server)
- Install vsftpd (FTP Server)

3.1.1 Installing a load balancer (HAProxy)

The load balancer is one of the most important components in the OpenShift Architecture. The HAProxy service that is available with the RHEL 8 distribution is used as a front end to the Red Hat OpenShift APIs (internal and external) and the OpenShift router. The load balancer is configured so that port 6443 and 22623 point to the bootstrap and control-plane nodes, while ports 80 and 443 are configured to point to the worker nodes.

This service is used to load balance the traffic to the cluster.

Although you can disable SELinux, we discourage this for security reasons. The steps that are described in this publication provide information about how to enable HAProxy when SELinux service is enabled on RHEL 8 server.

To install the HAProxy service, complete the following steps after you are connected to the bastion node and have root privileges:

1. Run the **`yum -y install haproxy`** command, as shown in Example 3-1 with its expected output.

Example 3-1 Install the HAProxy

```
[root@rdbkbas3 ~]# yum -y install haproxy
Updating Subscription Management repositories.
Unable to read consumer identity
This system is not registered to Red Hat Subscription Management. You can use
subscription-manager to register.
Last metadata expiration check: 0:00:57 ago on Tue 30 Jun 2020 09:57:48 AM EDT.
Dependencies resolved.
===============================================================================
 Package                                Architecture
Version                             Repository
Size
===============================================================================
Installing:
 haproxy                              s390x
1.8.15-6.el8_1.1                 local-rhn-server-appstream
1.3 M

Transaction Summary
===============================================================================
Install  1 Package

Total size: 1.3 M
Installed size: 4.5 M
Downloading Packages:
[SKIPPED] haproxy-1.8.15-6.el8_1.1.s390x.rpm: Already downloaded
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
 Preparing   :
1/1
 Runningscriptlet:haproxy-1.8.15-6.el8_1.1.s390x
1/1
 Installing  :haproxy-1.8.15-6.el8_1.1.s390x
1/1
 Runningscriptlet:haproxy-1.8.15-6.el8_1.1.s390x
1/1
 Verifying  :haproxy-1.8.15-6.el8_1.1.s390x
1/1
Installed products updated.

Installed:
  haproxy-1.8.15-6.el8_1.1.s390x

Complete!
```

2. Verify whether SELinux is active by running the **`sestatus`** command.

   You should receive the output that is shown in Example 3-2.

Example 3-2 Output of sestatus command

```
SELinux status:                 enabled
SELinuxfs mount:                /sys/fs/selinux
SELinux root directory:         /etc/selinux
Loaded policy name:             targeted
Current mode:                   permissive
Mode from config file:          enforcing
Policy MLS status:              enabled
Policy deny_unknown status:     allowed
Memory protection checking:     actual (secure)
Max kernel policy version:      31
```

3. Because SELinux is enabled for only the current boolean value, the boot-time default settings must be changed. Use the following SELinux boolean option and make it persistent during reboots by using the `-P` option as shown in the following example:

```
setsebool -P haproxy_connect_any on
```

4. To confirm the **haproxy_connect_any** parameter was changed, run the following command:

   `semanage boolean -lC`

   You should receive the output that is shown in Example 3-3.

   Example 3-3 semanage output

   ```
   SELinux boolean                State  Default Description
   haproxy_connect_any            (on   ,   on)  Allow haproxy to connect any
   ```

5. Rename the `/etc/haproxy/haproxy.cfg` file by running the following command:

   ```
   mv /etc/haproxy/haproxy.cfg /etc/haproxy/haproxy.cfg.original
   ```

6. Create the `/etc/haproxy/haproxy.cfg` file by using the `vim` editor (or any other Linux text editor). Ensure that you have the content that is shown in Example 3-4.

   Highlighted in red in Example 3-4 are the values that must be updated to fit your OpenShift IP addresses and node names. Use Table 2-2 on page 14 that was created during the planning phase for your values.

   Example 3-4 /etc/haproxy/haproxy.cfg file

   ```
   #---------------------------------------------------------------------
   # common defaults that all the 'listen' and 'backend' sections will
   # use if not designated in their block
   #---------------------------------------------------------------------
   defaults
       mode                    http
       log                     global
       option                  httplog
       option                  dontlognull
       option http-server-close
       option forwardfor       except 127.0.0.0/8
       option                  redispatch
       retries                 3
       timeout http-request    10s
       timeout queue           1m
       timeout connect         10s
       timeout client          1m
       timeout server          1m
       timeout http-keep-alive 10s
       timeout check           10s
       maxconn                 3000

   # Enable stats
   frontend stats
       bind *:8404
       stats enable
       stats uri /stats
       stats refresh 10s
       stats admin if LOCALHOST
       # REMEMBER TO UPDATE USER (admin) AND PASSWORD (redhot)  BELOW TO PROTECT
   YOUR PROXY STATS PAGE
       stats auth admin:redhat

   frontend openshift-api-server-6443
       bind *:6443
       default_backend openshift-api-server-6443
   ```

```
    mode tcp
    option tcplog

backend openshift-api-server-6443
    balance source
    mode tcp
    server rdbko3m1 129.40.23.111:6443 check
    server rdbko3m2 129.40.23.112:6443 check
    server rdbko3m3 129.40.23.113:6443 check
    server rdbko3b1 129.40.23.110:6443 check

frontend machine-config-server-22623
    bind *:22623
    default_backend machine-config-server-22623
    mode tcp
    option tcplog

backend machine-config-server-22623
    balance source
    mode tcp
    server rdbko3m1 129.40.23.111:22623 check
    server rdbko3m2 129.40.23.112:22623 check
    server rdbko3m3 129.40.23.113:22623 check
    server rdbko3b1 129.40.23.110:22623 check

frontend hafrontend-http-80
    bind *:80
    mode tcp
    default_backend habackend-http-80
    option tcplog

frontend hafrontend-https-443
    bind *:443
    default_backend habackend-https-443
    mode tcp
    option tcplog

backend habackend-http-80
    mode tcp
    balance roundrobin
    server rdbko3w1 129.40.23.114:80  check
    server rdbko3w2 129.40.23.115:80  check

backend habackend-https-443
    balance roundrobin
    mode tcp
    server rdbko3w1 129.40.23.114:443  check
    server rdbko3w2 129.40.23.115:443  check
```

HAProxy Stats

HAProxy Stats provides the status of the OpenShift cluster ports, including information about data transfer, total connection, and server state. You can access HAProxy Stats by using the URL that is shown in Example 3-5. Change the `<BASTION IP>` to your bastion node IP address.

Example 3-5 Access HAProxy stats

```
URL: http://<BASTION_IP>:8404/stats
 Login user: admin
 Login password: redhat
```

It is highly recommended that you change the login details of HAProxy Stats. To make this change, edit the configuration file (`/etc/haproxy/haproxy.cfg`) and update the “stats auth” value that is shown in Example 3-6.

We added the lines that are shown in Example 3-6 in the `/etc/haproxy/haproxy.cfg` file to enable the HAProxy Stats page. It was a helpful resource during the installation process and also to monitor the cluster during normal support activities.

Example 3-6 Enable the HAProxy statistics report page

```

# Enable stats
frontend stats
    bind *:8404
    stats enable
    stats uri /stats
    stats refresh 10s
    stats admin if LOCALHOST
    # REMEMBER TO UPDATE USER (admin) AND PASSWORD (redhot)  BELOW TO PROTECT YOUR
PROXY STATS PAGE
    stats auth admin:redhat
```

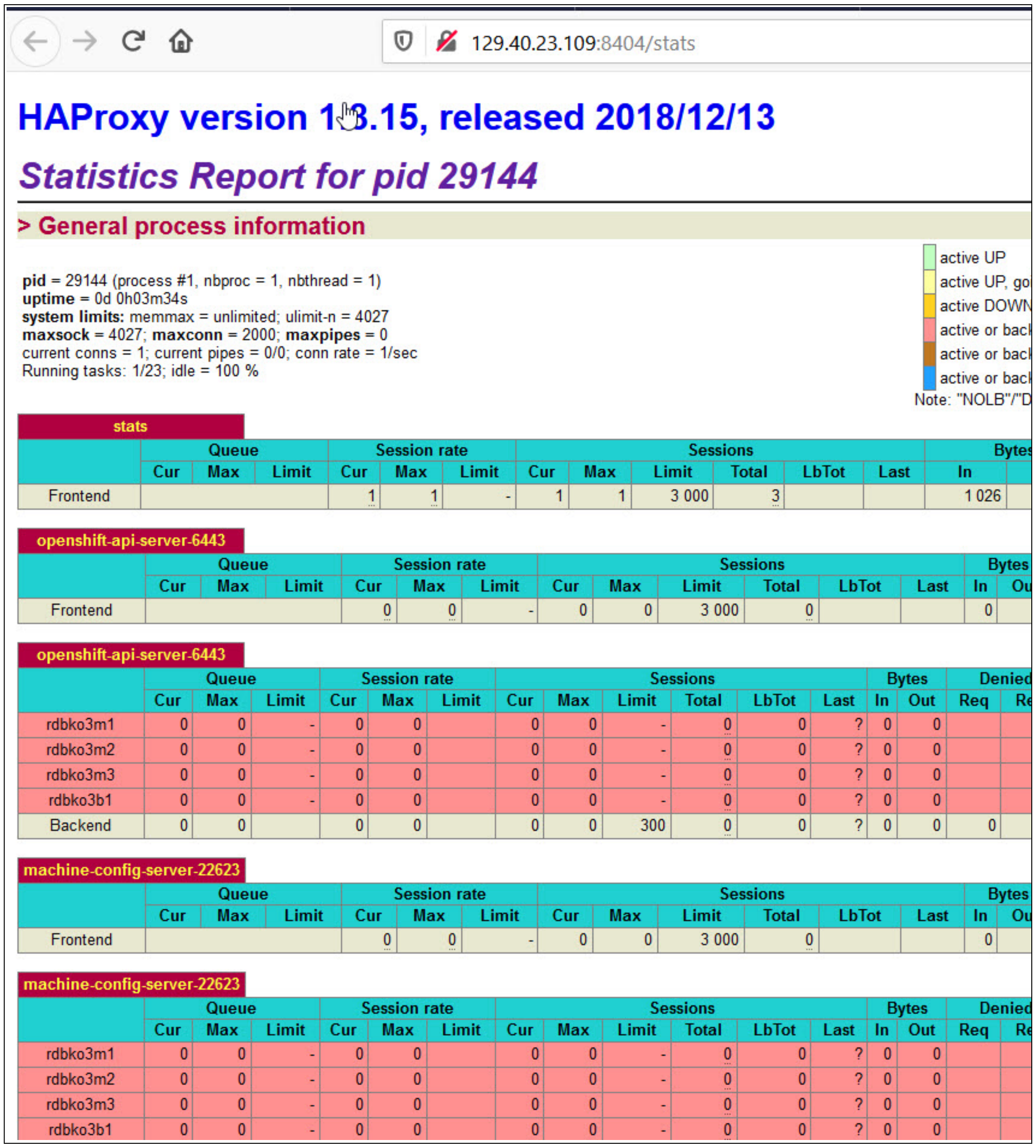
Change the user (admin) and password (redhat) configuration for the status page as a security precaution.

To access the status page, open an internet browser and access the following URL:

`https://BASTION_NODE_IP:8404/stats`

Where `BASTION_NODE_IP` is the IP address of your bastion node.

Save the configuration file and restart the HAProxy to update the service. Figure 3-1 shows an example of the HAProxy statistics report.



129.40.23.109:8404/stats

HAProxy version 1.8.15, released 2018/12/13

Statistics Report for pid 29144

> General process information

pid = 29144 (process #1, nbproc = 1, nbthread = 1)
uptime = 0d 0h03m34s
system limits: memmax = unlimited; ulimit-n = 4027
maxsock = 4027; **maxconn** = 2000; **maxpipes** = 0
current conns = 1; current pipes = 0/0; conn rate = 1/sec
Running tasks: 1/23; idle = 100 %

🟩	active UP
🟨	active UP, go
🟧	active DOWN
🟥	active or back
🟫	active or back
🟦	active or back

Note: "NOLB"/"D

stats

	Queue			Session rate			Sessions						Bytes
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In
Frontend				1	1	-	1	1	3 000	3			1 026

openshift-api-server-6443

	Queue			Session rate			Sessions						Bytes	
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Ou
Frontend				0	0	-	0	0	3 000	0			0	

openshift-api-server-6443

	Queue			Session rate			Sessions						Bytes		Denied	
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Re
rdbko3m1	0	0	-	0	0		0	0	-	0	0	?	0	0		
rdbko3m2	0	0	-	0	0		0	0	-	0	0	?	0	0		
rdbko3m3	0	0	-	0	0		0	0	-	0	0	?	0	0		
rdbko3b1	0	0	-	0	0		0	0	-	0	0	?	0	0		
Backend	0	0		0	0		0	0	300	0	0	?	0	0	0	

machine-config-server-22623

	Queue			Session rate			Sessions						Bytes	
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Ou
Frontend				0	0	-	0	0	3 000	0			0	

machine-config-server-22623

	Queue			Session rate			Sessions						Bytes		Denied	
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Re
rdbko3m1	0	0	-	0	0		0	0	-	0	0	?	0	0		
rdbko3m2	0	0	-	0	0		0	0	-	0	0	?	0	0		
rdbko3m3	0	0	-	0	0		0	0	-	0	0	?	0	0		
rdbko3b1	0	0	-	0	0		0	0	-	0	0	?	0	0		

Figure 3-1 HAProxy Stats page

7. Continuing on with the installation of the load balancer, open any required firewall rules for the local firewall by using the commands that are shown in Example 3-7.

Example 3-7 Open firewall rules

```
firewall-cmd --permanent --add-service=https
firewall-cmd --permanent --add-service=http
firewall-cmd --permanent --add-port=6443/tcp
firewall-cmd --permanent --add-port=22623/tcp
firewall-cmd --permanent --add-port=443/tcp
firewall-cmd --permanent --add-port=8404/tcp
firewall-cmd --reload
```

Note: To list all firewall rules, run the **`firewall-cmd --list-all`** command.

8. Run the following command to enable and start the HAProxy:

   `systemctl enable --now haproxy`

   The output of the command is shown in Example 3-8. It confirms that the HAProxy was enabled.

Example 3-8 Enable HAProxy

```
[root@rdbkbas3 ~]# systemctl enable --now haproxy
Created symlink /etc/systemd/system/multi-user.target.wants/haproxy.service ·
/usr/lib/systemd/system/haproxy.service.
```

9. Run the following command to confirm that the HAProxy is running:

   `systemctl status haproxy`

   You receive the output that is shown in Example 3-9.

Example 3-9 HAProxy status

```
   haproxy.service - HAProxy Load Balancer
      Loaded: loaded (/usr/lib/systemd/system/haproxy.service; enabled; vendor
   preset: disabled)
      Active: active (running) since Mon 2020-06-08 08:31:09 CDT; 2 weeks 6 days
   ago
    Main PID: 19042 (haproxy)
       Tasks: 2 (limit: 24483)
      Memory: 13.1M
      CGroup: /system.slice/haproxy.service
              ··19042 /usr/sbin/haproxy -Ws -f /etc/haproxy/haproxy.cfg -p
   /run/haproxy.pid
              ··19043 /usr/sbin/haproxy -Ws -f /etc/haproxy/haproxy.cfg -p
   /run/haproxy.pid
```

HAProxy is now configured correctly.

3.1.2 Installing bind (DNS service)

The domain name service (DNS) is another service that is required for the OpenShift infrastructure. If you have a DNS infrastructure, it is recommended that you configure the DNS delegation so that queries for hostnames within the OpenShift domain zone are handled by your local DNS server. For redundancy purposes, you can install a second RHEL guest and configure it as secondary DNS server for your OpenShift DNS domain.

The bastion node is configured with a DNS server. The following DNS components are defined.

- DNS Forward Lookup Zone
- DNS Reverse Lookup Zone
- DNS A Records, with PTR
- DNS Service record for etcd
- DNS SRV record for etcd

For more information about DNS records with examples, see the table that is listed under "User-provisioned DNS requirements" at this web page.

To install the DNS server, complete the following steps after you are connected to the bastion node and have root privileges. To run the command with the privileges of the root user, you can run the **sudo** command:

1. Install the DNS packages by using the **yum -y install bind bind-utils** command.

   The output of the command is shown in Example 3-10.

Example 3-10 Install bind-utils

```
[root@rdbkbas3 ~]# yum -y install bind bind-utils
Updating Subscription Management repositories.
Unable to read consumer identity
This system is not registered to Red Hat Subscription Management. You can use
subscription-manager to register.
Last metadata expiration check: 0:54:59 ago on Tue 30 Jun 2020 09:57:48 AM EDT.
Package bind-utils-32:9.11.4-26.P2.el8.s390x is already installed.
Dependencies resolved.
=================================================================================
=================================================================================
=========================
 Package                              Architecture
Version                                Repository
Size
=================================================================================
=================================================================================
=========================
Installing:
 bind                                 s390x
32:9.11.4-26.P2.el8                     InstallMedia-AppStream
2.1 M

Transaction Summary
=================================================================================
=================================================================================
=========================
Install  1 Package

Total size: 2.1 M
Installed size: 4.8 M
Downloading Packages:
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
 Preparing    :
1/1
 Running scriptlet: bind-32:9.11.4-26.P2.el8.s390x
1/1
 Installing   : bind-32:9.11.4-26.P2.el8.s390x
1/1
 Running scriptlet: bind-32:9.11.4-26.P2.el8.s390x
1/1
```

```
 Verifying   :bind-32:9.11.4-26.P2.el8.s390x
1/1
Installed products updated.

Installed:
  bind-32:9.11.4-26.P2.el8.s390x

Complete!
```

2. Update the `/etc/named.conf` file. Our updates are highlighted in red in Example 3-11.

Example 3-11 /etc/named.conf file

```
//
// named.conf
//
// Provided by Red Hat bind package to configure the ISC BIND named(8) DNS
// server as a caching only nameserver (as a localhost DNS resolver only).
//
// See /usr/share/doc/bind*/sample/ for example named configuration files.
//

options {
// Update with the Bastion node IP address
listen-on port 53 { 127.0.0.1; 129.40.23.109; };
listen-on-v6 port 53 { ::1; };
directory "/var/named";
dump-file "/var/named/data/cache_dump.db";
statistics-file "/var/named/data/named_stats.txt";
memstatistics-file "/var/named/data/named_mem_stats.txt";
secroots-file "/var/named/data/named.secroots";
recursing-file "/var/named/data/named.recursing";

// Use Network address
allow-query     { localhost; 129.40.23.0/24; };

// Use Network address
allow-recursion { localhost; 129.40.23.0/24; };

// Use Network address
allow-query-cache { localhost; 129.40.23.0/24; };

// Update the IP addresses below with the official primary and secondary DNS
servers of your Network
forwarders { 129.40.106.1; 129.40.106.2; };
/*
- If you are building an AUTHORITATIVE DNS server, do NOT enable recursion.
- If you are building a RECURSIVE (caching) DNS server, you need to enable
  recursion.
- If your recursive DNS server has a public IP address, you MUST enable access
  control to limit queries to your legitimate users. Failing to do so will
  cause your server to become part of large scale DNS amplification
  attacks. Implementing BCP38 within your network would greatly
  reduce such attack surface
*/
recursion yes;
dnssec-enable no;
dnssec-validation no;
managed-keys-directory "/var/named/dynamic";
pid-file "/run/named/named.pid";
session-keyfile "/run/named/session.key";
/* https://fedoraproject.org/wiki/Changes/CryptoPolicy */
include "/etc/crypto-policies/back-ends/bind.config";
};

logging {
        channel default_debug {
                file "data/named.run";
                severity dynamic;
        };
};

zone "." IN {
type hint;
file "named.ca";
};
include "/etc/named.rfc1912.zones";
include "/etc/named.root.key";
```

3. Update the `/etc/named.rfc1912.zones` file, as shown in Example 3-12.

Example 3-12 /etc/named.rfc1912.zones file

```
// named.rfc1912.zones:
//
// Provided by Red Hat caching-nameserver package
//
// ISC BIND named zone configuration for zones recommended by
// RFC 1912 section 4.1 : localhost TLDs and address zones
// and
http://www.ietf.org/internet-drafts/draft-ietf-dnsop-default-local-zones-02.txt
// (c)2007 R W Franks
//
// See /usr/share/doc/bind*/sample/ for example named configuration files.
//

zone "localhost.localdomain" IN {
type master;
file "named.localhost";
allow-update { none; };
};

zone "localhost" IN {
type master;
file "named.localhost";
allow-update { none; };
};

zone "1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.ip6.arpa" IN
{
type master;
file "named.loopback";
allow-update { none; };
};

zone "1.0.0.127.in-addr.arpa" IN {
type master;
file "named.loopback";
allow-update { none; };
};

zone "0.in-addr.arpa" IN {
type master;
file "named.empty";
allow-update { none; };
};

zone "ibm.local" IN {
               type master;
               file "forward.zone";
               allow-update { none; };
               forwarders {};
};

zone "23.40.129.in-addr.arpa" IN {
               type master;
               file "reverse.zone";
              allow-update { none; };
};
```

4. Create the `/var/named/forward.zone` file with the content that is shown in Example 3-13. Update the content to fit your OpenShift cluster names.

Example 3-13 /var/named/forward.zone

```
$TTL     1D
@        IN  SOA rdbkbas3.rdbk1.pok.ibm.local. root.ibm.local. (
                      2019022400 ; serial
                      3h         ; refresh
                      15         ; retry
                      1w         ; expire
                      3h         ; minimum
                                                                    )
        IN NS rdbkbas3.rdbk1.pok.ibm.local.
rdbkbas1.rdbk1.pok IN A 129.40.23.109


; The api points to the IP of your load balancer
api.rdbk1.pok IN A 129.40.23.109
api-int.rdbk1.pok IN A 129.40.23.109
;
; The wildcard also points to the load balancer
*.apps.rdbk1.pok IN A 129.40.23.109
;
; Create entry for the bootstrap host
rdbko3b1.rdbk1.pok IN A 129.40.23.110
```

```
;
; Create entries for the worker hosts
rdbko3w1.rdbk1.pok IN A 129.40.23.114
rdbko3w2.rdbk1.pok IN A 129.40.23.115

; Create entries for the controller hosts
rdbko3m1.rdbk1.pok IN A 129.40.23.111
rdbko3m2.rdbk1.pok IN A 129.40.23.112
rdbko3m3.rdbk1.pok IN A 129.40.23.113

; etcd records
etcd-0.rdbk1.pok IN A 129.40.23.111
etcd-1.rdbk1.pok IN A 129.40.23.112
etcd-2.rdbk1.pok IN A 129.40.23.113

; OpenShift Container Platform also requires an SRV DNS record for etcd server on
that machine with priority 0, weight 10 and port 2380. A cluster that uses three
control plane machines requires the following records:
; _service._proto.name. TTL class SRV priority weight port target.
_etcd-server-ssl._tcp.rdbk1.pok 86400 IN SRV 0 10 2380 etcd-0.rdbk1.pok.ibm.local.
_etcd-server-ssl._tcp.rdbk1.pok 86400 IN SRV 0 10 2380 etcd-1.rdbk1.pok.ibm.local.
_etcd-server-ssl._tcp.rdbk1.pok 86400 IN SRV 0 10 2380 etcd-2.rdbk1.pok.ibm.local.
```

5. Create the `/var/named/reverse.zone` file, as shown in Example 3-14.

Example 3-14 Reverse zone file

```
$TTL     1D
@        IN  SOA rdbkbas3.rdbk1.pok.ibm.local. root.ibm.local. (
                     2019022400 ; serial
                     3h         ; refresh
                     15         ; retry
                     1w         ; expire
                     3h         ; minimum
                                                                  )

         IN NS rdbkbas3.rdbk1.pok.ibm.local.

; Create entry for the bastion host
109 IN PTR rdbkbas3.rdbk1.pok.ibm.local.

;
; Create entry for the bootstrap host
110 IN PTR rdbko3b1.rdbk1.pok.ibm.local.

; Create entries for the worker hosts
114 IN PTR rdbko3w1.rdbk1.pok.ibm.local.
115 IN PTR rdbko3w2.rdbk1.pok.ibm.local.

; Create entries for the controller hosts
111 IN PTR rdbko3m1.rdbk1.pok.ibm.local.
112 IN PTR rdbko3m2.rdbk1.pok.ibm.local.
113 IN PTR rdbko3m3.rdbk1.pok.ibm.local.
```

6. Update file ownership and file permissions by using the commands that are shown in Example 3-15.

Example 3-15 Update file ownership and permissions

```
chown named:named /var/named/forward.zone /var/named/reverse.zone
chmod 640  /var/named/forward.zone /var/named/reverse.zone
```

7. Enable and start the `named.service` daemon by using the following command:

   ```
   systemctl enable –now named.service
   ```

   The output of the command is shown in Example 3-16.

Example 3-16 Enable named service

```
[root@rdbkbas3 ~]# systemctl enable --now named.service
Created symlink /etc/systemd/system/multi-user.target.wants/named.service ·
/usr/lib/systemd/system/named.service.
```

8. Update the local firewall to accept connections to DNS traffic by using the following commands:

   ```
   firewall-cmd --permanent --add-service=dns
   firewall-cmd –reload
   ```

9. Confirm if DNS forward records are working correctly and responding to OpenShift required DNS entries. Run the commands that are shown in Example 3-17.

Example 3-17 Verify DNS forward zone

```
dig @localhost rdbko3m1.rdbk1.pok.ibm.local +short
dig @localhost rdbko3m2.rdbk1.pok.ibm.local +short
dig @localhost rdbko3m3.rdbk1.pok.ibm.local +short
dig @localhost rdbko3w1.rdbk1.pok.ibm.local +short
dig @localhost rdbko3w2.rdbk1.pok.ibm.local +short
dig @localhost etcd-0.rdbk1.pok.ibm.local +short
dig @localhost etcd-1.rdbk1.pok.ibm.local +short
dig @localhost etcd-2.rdbk1.pok.ibm.local +short
```

   If you receive output with the IP address for each OpenShift resource, the DNS forward zone configuration is working correctly.

10. Confirm the OpenShift DNS reverse zone by running the commands that are shown in Example 3-18

Example 3-18 Verify DNS reverse zone

```
dig @localhost -x 129.40.23.109 +short
dig @localhost -x 129.40.23.110 +short
dig @localhost -x 129.40.23.111 +short
dig @localhost -x 129.40.23.112 +short
dig @localhost -x 129.40.23.113 +short
dig @localhost -x 129.40.23.114 +short
dig @localhost -x 129.40.23.115 +short
```

   If you receive output with the node names for each OpenShift resource, the DNS reverse zone configuration is correctly defined.

11.Confirm the OpenShift DNS SRV resources by running the command that is shown in Example 3-19.

Example 3-19 Verify OpenShift DNS SRV

```
dig @localhost _etcd-server-ssl._tcp.rdbk1.pok.ibm.local SRV +short
```

The output appears similar to the output that is shown in Example 3-20.

Example 3-20 Output of verify DNS SRV

```
0 10 2380 etcd-2.rdbk1.pok.ibm.local.
0 10 2380 etcd-0.rdbk1.pok.ibm.local.
0 10 2380 etcd-1.rdbk1.pok.ibm.local.
```

The DNS service is now configured.

Note: If you update the DNS file after you start it, remember to reload the service by running the **`systemctl restart named`** command.

3.1.3 File transfer protocol server

A file transfer protocol (FTP) server is needed to hold the ignition configuration files and installation images for the installation of RHEL CoreOS. The VSFTPD package that comes with the RHEL distribution is installed on the bastion node to provide this function.

To install an FTP server, complete the following steps after you are connected to the bastion node and have root privileges:

1. Install the FTP operating system package by using the yum:

   ```
   yum -y  install vsftpd
   ```

   The output of the command is shown in Example 3-21.

Example 3-21 Install vsftpd

```
[root@rdbkbas3 ~]# yum -y install vsftpd
Updating Subscription Management repositories.
Unable to read consumer identity
This system is not registered to Red Hat Subscription Management. You can use
subscription-manager to register.
Extra Packages for Enterprise Linux 8 - s390x
95 kB/s |  18 kB     00:00
Jenkins-stable
589  B/s | 2.9 kB     00:05
Poughkeepsie Client Center Local RHN - RHEL 8.1 s390x Server RPMs
601  B/s | 2.9 kB     00:05
Poughkeepsie Client Center Local RHN - RHEL 8.1 s390x Server Supplementary RPMs
157 kB/s | 2.9 kB     00:00
Poughkeepsie Client Center Local RHN - RHEL 8.1 s390x Server Optional RPMs
171 kB/s | 2.9 kB     00:00
Dependencies resolved.
==================================================================================
 Package                                 Architecture
Version                              Repository
Size
==================================================================================
```

```
Installing:
 vsftpd                                  s390x
3.0.3-28.el8                        InstallMedia-AppStream
175 k

Transaction Summary
==================================================================================
Install  1 Package

Total size: 175 k
Installed size: 370 k
Downloading Packages:
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
 Preparing   :
1/1
 Installing  :vsftpd-3.0.3-28.el8.s390x
1/1
 Running scriptlet:vsftpd-3.0.3-28.el8.s390x
1/1
 Verifying   :vsftpd-3.0.3-28.el8.s390x
1/1
Installed products updated.

Installed:
  vsftpd-3.0.3-28.el8.s390x

Complete!
```

2. Update the `/etc/vsftpd/vsftpd.conf` file to allow anonymous connections for FTP server. Update the highlighted section in red that is shown in Example 3-22.

Example 3-22 update vsftpd.conff

```
anonymous_enable=YES
local_enable=YES
write_enable=YES
local_umask=022
dirmessage_enable=YES
xferlog_enable=YES
connect_from_port_20=YES
xferlog_std_format=YES
listen=NO
listen_ipv6=YES
pam_service_name=vsftpd
userlist_enable=YES
pasv_enable=Yes
```

3. Start and enable vsftpd service by using the following command:

```
systemctl enable —now vsftpd.service
```

The output of the command is shown in Example 3-23.

Example 3-23 enable vsftpd service

```
[root@rdbkbas3 ~]# systemctl enable --now vsftpd.service
Created symlink /etc/systemd/system/multi-user.target.wants/vsftpd.service ·
/usr/lib/systemd/system/vsftpd.service.
```

4. Run the commands that are shown in Example 3-24 to update the local firewall to allow FTP connections.

Example 3-24 Update local firewall

```
firewall-cmd --permanent --add-service=ftp
firewall-cmd --permanent --add-port=21/tcp
firewall-cmd –reload
```

5. Run the following command to set SELinux `ftpd_use_passive_mode`. Use the Boolean option to "**on**" to enable passive FTP and make it persistent during reboots by using the **-P** option:

   ```
   setsebool -P ftpd_use_passive_mode on
   ```

6. To confirm that the `ftpd_use_passive_mode` was changed, run the following command:

   ```
   semanage boolean -lC
   ```

   You should receive the output that is shown in Example 3-25.

Example 3-25 Output of semanage

```
SELinux boolean               State  Default Description
ftpd_use_passive_mode         (on   ,   on)  Allow ftpd to use passive mode
haproxy_connect_any           (on   ,   on)  Allow haproxy to connect any
```

7. Create the `/var/ftp/pub` directory by running the following command:

   ```
   mkdir -p /var/ftp/pub
   ```

8. Set permissions on the `/var/ftp/pub` folder by running the following command:

   ```
   chmod 755 /var/ftp/pub
   ```

9. Ensuring that this FTP server can be reached by the bootstrap, control plane, and compute machines during installation is an important task before starting downloads. Therefore, access the FTP server by using your preferred browser or run the FTP following command:

   ```
   ftp <BASTION_IP_ADDRESS>
   ```

   Where `BASTION_IP_ADDRESS` is the IP address or IP name of your bastion server.

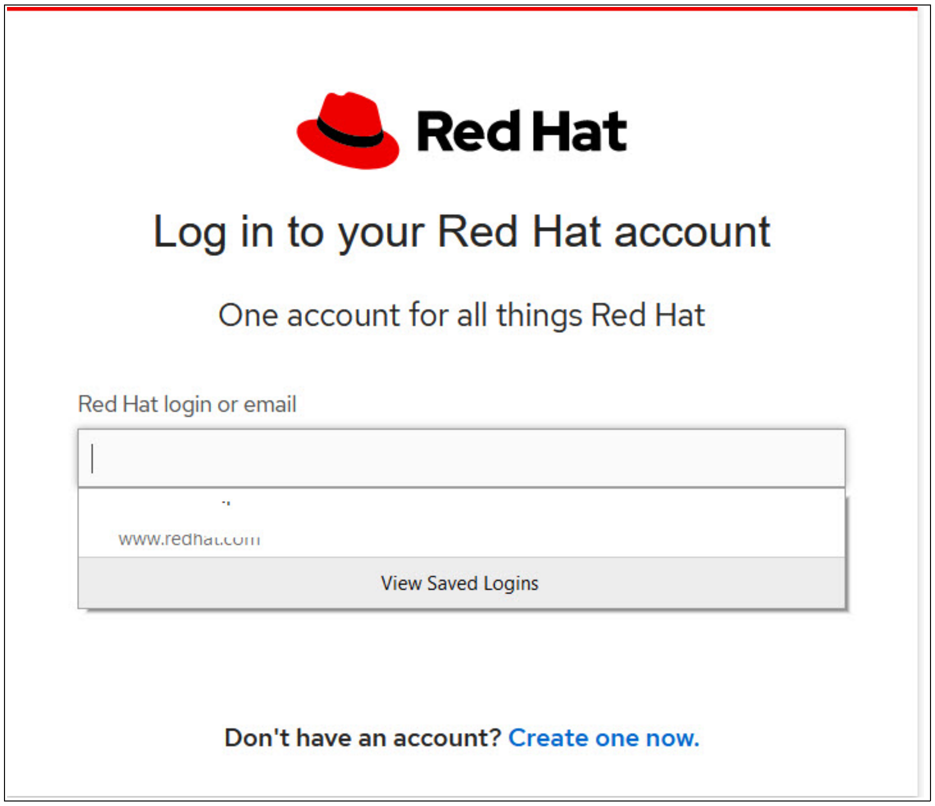
The configuration for VSFTPD is complete if you can access the FTP server.

3.2 Downloading OpenShift files

In this section, we describe the process of downloading the files that are required for a successful installation of Red Hat OpenShift 4.4 and RHCOS.

To download Red Hat OpenShift 4.4 and RHCOS, complete the following steps:

1. Browse to the OpenShift Infrastructure Providers web page (see Figure 3-2).



Log in to your Red Hat account

One account for all things Red Hat

Red Hat login or email

Figure 3-2 Red Hat Log on page

2. Authenticate by using your Red Hat credentials (or create an account if you do not have credentials yet).

Note: You must log in to your Red Hat account to download the files.

3. Multiple platforms are displayed. Click **Run on IBM Z**, as shown in Figure 3-3.

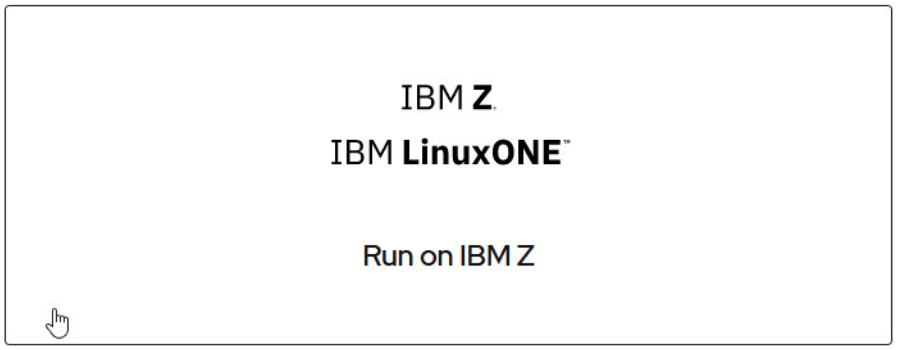
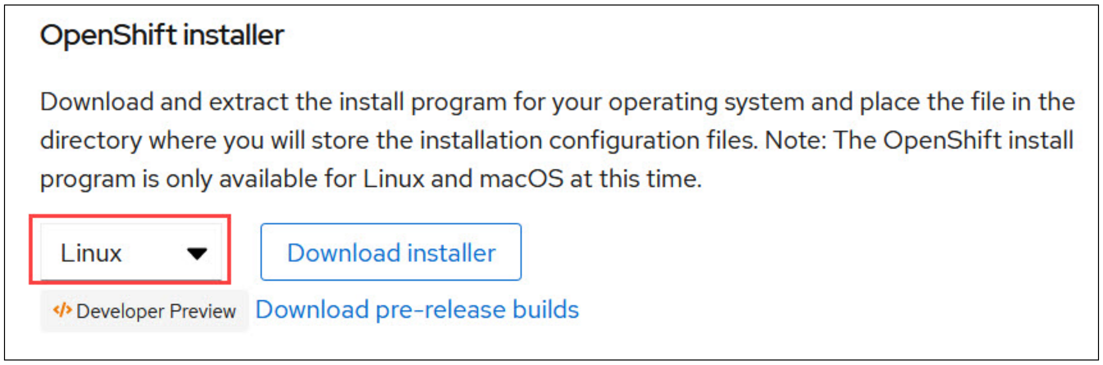


Figure 3-3 Run on IBM Z

4. A window opens. In the Download section:

   a. Download the OpenShift installer (Linux version). Select **Linux** from the drop-down menu, as shown in Figure 3-4. Click **Download Installer** (the file name is `openshift-install-linux.tar.gz`).

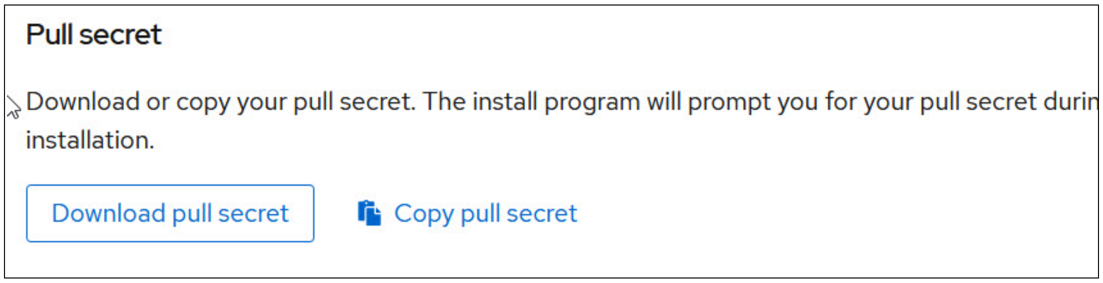


OpenShift installer

Download and extract the install program for your operating system and place the file in the directory where you will store the installation configuration files. Note: The OpenShift install program is only available for Linux and macOS at this time.

Figure 3-4 Download installer

   b. Download the pull secret, as shown in Figure 3-5. Save the content to a text file because the OpenShift installer prompts you for a pull secret during the installation process (use `pull-secret` as the file name).



Pull secret

Download or copy your pull secret. The install program will prompt you for your pull secret durin installation.

Figure 3-5 Download Pull Secret

Note: This pull secret is used to associate the new OpenShift cluster with your Red Hat account.

c. Download the command-line tools if you want to run the commands from a desktop or outside of the bastion server (`openshift-client-linux.tar.gz`).

d. Download the Red Hat Enterprise Linux CoreOS (RHEL CoreOS) image for IBM Z. Go to the bastion server and create a folder called `/opt/downloads` by running the following command:

```
mkdir /opt/downloads
```

e. Browse to `/opt/downloads` and download the files that are required for OpenShift installation. For DASD installations, download the `rhcos-4.4.9-s390x-dasd.s390x.raw.gz` file by running the **`wget`** command, as shown in Example 3-26.

Example 3-26 Download rhcos-dasd

```
[root@rdbkbas3 downloads]# wget
https://mirror.openshift.com/pub/openshift-v4/s390x/dependencies/rhcos/4.4/4.4.9/r
hcos-4.4.9-s390x-dasd.s390x.raw.gz
--2020-06-30 16:32:56--
https://mirror.openshift.com/pub/openshift-v4/s390x/dependencies/rhcos/4.4/4.4.9/r
hcos-4.4.9-s390x-dasd.s390x.raw.gz
Resolving mirror.openshift.com (mirror.openshift.com)... 54.173.18.88,
54.172.163.83, 54.172.173.155
Connecting to mirror.openshift.com (mirror.openshift.com)|54.173.18.88|:443...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 764659687 (729M) [application/x-gzip]
Saving to: ‘rhcos-4.4.9-s390x-dasd.s390x.raw.gz’

rhcos-4.4.9-s390x-dasd.s390x.raw.gz
100%[=====================================================================>]
729.24M  76.5MB/s    in 9.6s

2020-06-30 16:33:06 (75.8 MB/s) - ‘rhcos-4.4.9-s390x-dasd.s390x.raw.gz’ saved
[764659687/764659687]
```

For SAN installations, download the `rhcos-4.4.9-s390x-metal.s390x.raw.gz` file by running the command that is shown in Example 3-27.

Example 3-27 Download rhcos-SAN

```
[root@rdbkbas3 downloads]# wget
https://mirror.openshift.com/pub/openshift-v4/s390x/dependencies/rhcos/4.4/4.4.9/r
hcos-4.4.9-s390x-metal.s390x.raw.gz
```

f. Using the same **`wget`** command, download the following Linux files, as shown in Example 3-28:

`rhcos-4.4.9-s390x-installer-initramfs.s390x.img`

`rhcos-4.4.9-s390x-installer-kernel-s390x`

Example 3-28 Download Linux files

```
wget
https://mirror.openshift.com/pub/openshift-v4/s390x/dependencies/rhcos/4.4/4.4.9/r
hcos-4.4.9-s390x-installer-initramfs.s390x.img
```

```
wget
https://mirror.openshift.com/pub/openshift-v4/s390x/dependencies/rhcos/4.4/4.4.9/r
hcos-4.4.9-s390x-installer-kernel-s390x
```

5. At the desktop, copy the following files to your Bastion node in the `/opt/downloads` directory:

   `openshift-install-linux.tar.gz`

   `pill-secret`

   `openshift-client-linux.tar.gz`

   After completing these steps, the `/opt/downloads` directory on the bastion server should look like that which is shown in Example 3-29.

Example 3-29 contents of /opt/downloads

```
[root@rdbkbas3 downloads]# ls -ltr
total 3972504
-rw-r--r--. 1 root root  56227910 Jun 22 13:11
rhcos-4.4.9-s390x-installer-initramfs.s390x.img
-rw-r--r--. 1 root root 126365696 Jun 22 13:11
rhcos-4.4.9-s390x-installer.s390x.iso
-rw-r--r--. 1 root root   5124581 Jun 22 13:11
rhcos-4.4.9-s390x-installer-kernel-s390x
-rw-r--r--. 1 root root 764676689 Jun 22 13:11
rhcos-4.4.9-s390x-metal.s390x.raw.gz
--rw-r--r--. 1 root root 764659687 Jun 22 13:12
rhcos-4.4.9-s390x-dasd.s390x.raw.gz
-rw-r--r--. 1 root root  81947062 Jun 30 15:43 openshift-install-linux.tar.gz
-rw-r--r--. 1 root root  24695307 Jun 30 15:48 openshift-client-linux.tar.gz
-rw-r--r--. 1 root root      2731 Jun 30 16:45 pull-secret
```

3.3 Passwordless SSH configuration

During the RHCOS installation process, a user named `core` is created with the SSH key that is assigned to that user. Direct root access is not allowed by way of server console or SSH. Therefore, the system administrators can log in to the nodes with that user name and SSH key credentials. The following command is an example to access the bootstrap node:

```
ssh -i $HOME/.ssh/id_rsa core@129.40.23.110
```

To allow internal cluster communications and administrative access to the OpenShift server machines, system administrators can use the ssh-key pair.

On the bastion server, generate SSH keys for the `core` user by running the **`ssh-keygen`** command. Do not protect the private key with a passphrase. Run the following command:

```
ssh-keygen -t rsa -N '' -f $HOME/.ssh/id_rsa
```

The output of the command is shown in Example 3-30.

Example 3-30 Output of ssh-keygen

```
[root@rdbkbas3 /]# ssh-keygen -t rsa -N '' -f $HOME/.ssh/id_rsa
Generating public/private rsa key pair.
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:q58dnneTBMaEQdLuGsd1COgC/helloRedbooks root@rdbkbas3.pbm.ihost.com
The key's randomart image is:
+---[RSA 3072]----+
|     welcome     |
|       to        |
|   IBM Redbooks  |
|                 |
|     . OS= o o   |
|      o *.=   .  |
|       ..=.  . . |
|       ..+ o. +  |
|      ..o +. . . |
+----[SHA256]-----+
```

Note: The contents of the `id_rsa.pub` file (`$HOME/.ssh/id_rsa.pub`) is incorporated in the sshKey section of the `install-config.yaml` file later during the cluster installation step.

3.4 Installing and creating the ignition configuration files on the bastion server

The ignition files contain instructions for the RHCOS installer about how to configure the new Linux server. It facilitates silent or unattended installation. Therefore, you do not need to provide any input during the RHCOS installation.

We are going to use the bastion server to generate the necessary ignition files for the cluster installation. Without the ignition files, you cannot install RHCOS.

The openshift-installer that was obtained from the OpenShift Infrastructure Providers page that is found in section 3.2, “Downloading OpenShift files” on page 32 is run to create the ignition configuration files. The openshift-installer expects a YAML-formatted file that is called `install-config.yaml` to generate the cluster configuration information.

For installations of OpenShift Container Platform that use a user-provisioned infrastructure, you must manually generate your installation configuration file. The following steps help you to define an `install-config.yaml` file and create the ignition files.

Complete the following steps after you are connected to the bastion node and have root privileges:

1. Create an installation directory to store the required installation files by running the following commands:

   ```
   mkdir /stage
   ```

   ```
   cd /stage
   ```

2. Create and customize the `/stage/install-config.yaml` file, as shown in Example 3-31. The settings that you must update to your values are highlighted in red.

   Example 3-31 /stage/install-config.yaml file

   ```
   apiVersion: v1
   ```

```
baseDomain: pok.ibm.local
compute:
- hyperthreading: Enabled
  name: worker
  replicas: 2
controlPlane:
  hyperthreading: Enabled
  name: master
  replicas: 3
metadata:
  name: rdbk1
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  none: {}
pullSecret:
'{"auths":{"cloud.openshift.com":{"auth":"b3BlbnNoaWZ0LXJlbGVhc2UtZGV2K2VtYXJpb
nNicmlibWNvbTF6eGRmam1jdTZnb3FseHo2dGdjbXhOMWJ6cjpNVONSWUFDWlZNOTg2ODIxVVFOWTFT
UDVZSk9DMEVXNO4ONEVITUZXVzBVMkk4RFIyQUZaTlFNSTJUQVdTUUZX","email":"xxxxxx@br.ib
m.com"},"quay.io":{"auth":"b3BlbnNoaWZ0LXJlbGVhc2UtZGV2K2VtYXJpbnNicmlibWNvbTF6
eGRmam1jdTZnb3FseHo2dGdjbXhOMWJ6cjpNVONSWUFDWlZNOTg2ODIxVVFOWTFTUDVZSk9DMEVXNO4
ONEVITUZXVzBVMkk4RFIyQUZaTlFNSTJUQVdTUUZX","email":"welcoome"},"registry.connec
t.redhat.com":{"auth":"xxxxxxlk;ajflsdfoeulnl;u","email":"xxxxxxxx@br.ibm.com"}
,"registry.redhat.io":{"auth":"Nzg2MjI4M3x1aGMtMVpYREZKTUN1NmdvUWx4ejZORONNWHQx
YlpyOmV5SmhiR2NpT2lKU1V6VXhNaUo5LmV5SnpkV0lpT2lKbU9UZzRaRGM1TOdFNE1tUTBObU01TOR
OaE56ZzHHKJHHLKJJd1ltUTVaQOo5LkxLc1BUbTZ6VXZjdktwcUg3aFB3UmpmNTk1b3RvWXhpTWVvU2
pZNVBOREJJdHB2NndUdy1pXzEtNkhpalozOU9MQXI3b1JrdWNVSkZZcmIwdGOtaEwzSlVicGtuSHQtV
HRBTHA2SktOVV9KeGOzcVJRLVUzYzFsTm9PSkpySnRfMUp2U3dqNGdYcm5ubzFMTOYydm14dTJTRF9D
QURleWhUSOlIOThOdk9hZ2O3eE1YOUlSOWpIUO1obWVwZS1VVFNoY29pTGNHUFpvVXczTlRuY3NNTFB
RNHcxcDZrR2FhRGVHZGx4SXU2SWVaV3RKRlBHVOQyNzVDejFFWlhxbkZFWFptaOJFQjJFeXZXWGZ3Yl
pHcWFwZnI5SWppLWpjcOEzR1UyUWF5TTFNakVvWmO3Vjh3ODFUZXlOS3ZiUGUtRVpiMTVQMUVRdEVpS
lVjSmlOMVJRbXFmNmEzanZLR2ZFQ1hYdm9mYOhJVWtBZ1JveWZKbzZnT2ZzUzh5ekFhVEVZRjA3dOk2
SzV5X1UxY2l1UERnZTY5LWV4dWM3MzRVbmpBZXdHWVR4RFVOazN2NTZSX3k4YUlURVltcjlyTHFUWFV
IYWVvMGU1STAzSXVUcGdOWDRjRGZQdk9Ka252bOZMZjFZUVFObFJXenIxbi1NNnUyb2N2SnVpVi1NMk
hybGxheHlGaVVkQjR4TEtPUkNxNGJJa1ROTUkxUnExZTVDZmZwckVPUzBzRUs4RzliNmVrbnY2WTg3c
FEtcXhzdDdtYl8tVGNOVEFXxxxxXZWXVLSDhlMkozZUOtM28OaFoxQ2s5VUoxXzExRDdjT1FwUFZnMW
xleDdKQlFHdOh6MUkOalZOSUYtZlFqaO9CTGtqNjVhN3lWa1AwSF9naEhEWlNSSzJV","email":"xx
xxxx@br.ibm.com"}}}'
sshKey: 'ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQCrkGIqQjlx3ui6IgG4ioG2pRMYr2HhxXGVsvmACcu06b8u26N
1F8Z3IeO/1CACXxLTHHqcswl8l1IVJ1zTl5vSh2SZIAwK5/xKz7bUjeeWUePnyspZltmJv7IzcRn3KX
ziXW8OmRedbooksyq+xMrQs4Jku5IhvAUEEk2HYomDbPcDngnoKWoQOsSFrujxL/l6bJIiEzA+HJn9J
52iqlugw+jj/vp4Gdc6Xhf68NX3WDD/ysWzP1tO8Zz3mbn5hHGb+ZuhXCfOgymqdgM/qcHPQv1mLKuf
hsXplLJ2/1CmMwjzQAStGtmu2SUMCpdLWokO5OSW6g6KOAgDnqqdvxqKOk3GP9rJOcNIq+kaj8JhN2U
uMNz2RfmjTq/vXXHhhptijCT2ijve9+A7qaQnlogkoBBvBc1DbxUc+otsvWov4BS2ac22QTBJrDO5+W
2gc9wSFiZO5ZxE25ot/ji/eeBOu2vEnoxyve9wT+Ldet4zlZYO8SEgBfodL8TT/TrlLrNrs=
redbo@br.ibm.com'
```

The values in red in this example represent:

– Your baseDomain, metadata.name, pullSecret, and sshKey tags.

– The pullSecret that should be updated with your pullSecret.
– The sshKey that should be updated with the content of id_rsa.pub.

3. Save a copy of the `install-config.yaml` file so that you can use it to install multiple clusters. The openshift-installer reads the YAML file and deletes the file after the ignition files are created; therefore, you want to ensure that you have a copy of this file:

   ```
   cp /stage/install-config.yaml  /stage/install-config.yam.bkup
   ```

   Important: The `install-config.yaml` file is deleted during the next step of the installation process. You must back it up now.

4. Go to the folder where you put the downloaded Red Hat files (`/opt/downloads`):

   ```
   cd /opt/downloads
   ```

5. Copy the downloaded OpenShift installer `openshift-install-linux.tar.gz to /stage` and unpack the OpenShift installer by running the commands that are shown in Example 3-32.

   Example 3-32 Unpack OpenShift install file

   ```
   cp /opt/downloads/openshift-install-linux.tar.gz /stage
   cd /stage/
   tar zxf openshift-install-linux.tar.gz com
   ```

6. Copy the OpenShift client (`openshift-client-linux.gz`) to `/stage`:

   ```
   cp /opt/downloads/openshift-client-linux.tar.gz /stage
   ```

7. Unpack OpenShift client file (`openshift-client-linux.tar.gz`) and run the following command:

   ```
   tar zxf openshift-client-linux.tar.gz
   ```

8. Copy the oc and kubectl files to `/usr/local/bin` folder:

   ```
   cp -a oc kubectl /usr/local/bin/
   ```

9. Generate the Kubernetes manifests for the cluster, issue:

   ```
   ./openshift-install create manifests --dir=/stage
   ```

   The output of the command is shown in Example 3-33.

   Example 3-33 Output of OpenShift installation

   ```
   [root@rdbkbas3 stage]# ./openshift-install create manifests --dir=/stage
   INFO Consuming Install Config from target directory
   ```

10. Modify the `/stage/manifests/cluster-scheduler-02-config.yml` Kubernetes manifest file to prevent pods from being scheduled on the control plane machines by completing the following steps:

    a. Open the `/stage/manifests/cluster-scheduler-02-config.yml` file.

    b. Locate the mastersSchedulable parameter and set its value to False, as shown in Example 3-34.

    c. Save and exit the file.

    Example 3-34 Change mastersSchedulable parameter

    ```
    apiVersion: config.openshift.io/v1
    kind: Scheduler
    metadata:
    ```

```
  creationTimestamp: null
  name: cluster
spec:
  mastersSchedulable: false
  policy:
    name: ""
status: {}
```

11. Run the following command to generate the ignition configuration files:

    ```
    ./openshift-install create ignition-configs --dir=/stage
    ```

    The output of the command is shown in Example 3-35.

    Example 3-35 Create ignition files

    ```
    [root@rdbkbas3 stage]# ./openshift-install create ignition-configs --dir=/stage
    INFO Consuming Openshift Manifests from target directory
    INFO Consuming Master Machines from target directory
    INFO Consuming Worker Machines from target directory
    INFO Consuming Common Manifests from target directory
    INFO Consuming OpenShift Install (Manifests) from target directory
    ```

    Upon completion, the directories and files that are shown in Example 3-36 are created in the `/stage` directory.

    Example 3-36 Contents of /stage directory

    ```
    [root@rdbkbas3 stage]# ls -ltr
    total 592144
    -rw-r--r--. 1 root root       954 Jun 21 13:54 README.md
    -rwxr-xr-x. 2 root root  75297728 Jun 21 13:54 oc
    -rwxr-xr-x. 2 root root  75297728 Jun 21 13:54 kubectl
    -rwxr-xr-x. 1 root root 348782592 Jun 21 14:29 openshift-install
    -rw-r--r--. 1 root root      3750 Jun 30 17:13 install-config.yaml.44.bkup
    -rw-r--r--. 1 root root  81947062 Jun 30 17:41 openshift-install-linux.tar.gz
    -rw-r--r--. 1 root root  24695307 Jun 30 17:52 openshift-client-linux.tar.gz
    drwxr-x---. 2 root root        50 Jun 30 17:59 auth
    -rw-r-----. 1 root root      1824 Jun 30 17:59 master.ign
    -rw-r-----. 1 root root      1824 Jun 30 17:59 worker.ign
    -rw-r-----. 1 root root    299596 Jun 30 17:59 bootstrap.ign
    -rw-r-----. 1 root root        98 Jun 30 17:59 metadata.json
    ```

    The kubeconfig file, found in the `/state/auth` directory, is used later to set the cluster context for logging in to the cluster by using the command-line interface. In addition, the `/stage/auth/kubeadmin-password` file includes password information for logging in from a browser. The *.ign files that are obtained are used to start each type of cluster nodes.

12. Copy all *.ign files to the `/var/ftp/pub/` folder of the FTP server (the bastion server:

    ```
    cp /stage/*.ign /var/ftp/pub/
    ```

13. Ensure that the files have 644 permissions set and SELinux context is updated. Run the following commands:

    ```
    chmod 644 /var/ftp/pub/*.ign
    restorecon -R -v /var/ftp/pub/
    ```

14. We strongly recommend that you perform a backup of the `/stage` directory. It contains files about your OpenShift installation. Run the following command to back up this directory:

    ```
    cp -a /stage /stage.bkp
    ```

3.5 Creating Red Hat Enterprise Linux CoreOS guests

Before you configure an OpenShift cluster on the IBM Z platform, you must install RHCOS on z/VM guest VMs. The RHCOS installation uses the ignition file to perform the operating system configuration.

The ignition file is read-only early in the boot process (in the initramfs to be exact) and provides a fully automated install, which reduces the complexity for Linux container administrators. You would perform the install by passing the `coreos.inst.<arg>` arguments on the parameter file. This section describes the required steps to complete a successful installation.

The process to generate the ignition files by using the OpenShift installer is described in Example 3.4 and created in the `/stage` folder.

Complete the following steps after you connect to the bastion node and have root privileges:

1. Copy the downloaded Red Hat Enterprise Linux CoreOS installation files to `/var/ftp/pub` and run:

   ```
   cp /opt/downloads/rhcos-4.4.9-s390x-installer-kernel-s390x /var/ftp/pub/
   cp /opt/downloads/rhcos-4.4.9-s390x-installer-initramfs.s390x.img /var/ftp/pub/
   cp /opt/downloads/rhcos-4.4.9-s390x-dasd.s390x.raw.gz /var/ftp/pub/
   ```

2. Verify that the Red Hat Enterprise Linux CoreOS Ignition files (*.ign) exists by running the following command:

   ```
   ls -l /var/ftp/pub/*.ign
   ```

   You should receive output that is similar to the following example:

   ```
   -rw-r--r--. 1 root root 295184 Jun  8 08:32 /var/ftp/pub/bootstrap.ign
   -rw-r--r--. 1 root root   1826 Jun  8 08:31 /var/ftp/pub/master.ign
   -rw-r--r--. 1 root root   1970 Jun  8 10:30 /var/ftp/pub/worker.ign
   ```

3. Create parameter files (parmfiles) for each OpenShift node (bootstrap, controller, and worker nodes). In our environment, the following parmfiles were created:

   – `/var/ftp/pub/bootstrap-0.parm`
   – `/var/ftp/pub/master-1.parm`
   – `/var/ftp/pub/master-2.parm`
   – `/var/ftp/pub/master-3.parm`
   – `/var/ftp/pub/worker-1.parm`
   – `/var/ftp/pub/worker-2.parm`

   Before you can begin the installation, you must configure some boot parameters. When installing through z/VM, these parameters must be configured before you boot by using the parmfiles. Use a Linux editor, such as vim, to add the parameters that are listed in Table 3-1.

Table 3-1 Parm file parameters

Parameter	Description	Example
coreos.inst=yes	Specify whether a coreos will be installed.	coreos.inst=yes

Parameter	Description	Example
coreos.inst.install_dev=sda	Specify the block device of the system to which to install.	For example, setting: coreos.inst.install_dev=dasda Tells installer to extract RHCOS image into a DASDA device. DASDA is the first DASD. We defined only 1 disk.
coreos.inst.image_url=<image_URL>	Specify the URL of the RAW image that you uploaded to your server.	coreos.inst.image_url=ftp://129.40.23.109/pub/rhcos-4.4.9-s390x-dasd.s390x.raw.gz
coreos.inst.ignition_url=http://example.com/config.ign	Specify the URL of the Ignition config file for this machine type. The location of the Ignition config file that you copied to your FTP server.	For example, setting: coreos.inst.ignition_url=ftp://129.40.23.109/pub/bootstrap.ign Tells installation to load bootstrap ignition file by way of FTP
ip=<dhcp or static IP address>	Set ip=dhcp or Set an individual static IP address (ip=) and DNS server (nameserver=) on each node.	For example, setting: ip=129.40.23.110::129.40.23.1:255.255.255.0:::none nameserver=129.40.23.109 Sets: Node's IP address to `129.40.23.110` Gateway address to `129.40.23.1` Netmask to 255.255.255.0 Hostname will get from DNS reverse zone The DNS server address to `129.40.23.109`
rd.znet	Specify a network protocol type, a comma delimited list of subchannels, and, optionally, comma delimited sysfs parameter and value pairs.	rd.znet=qeth,0.0.1000,0.0.1001,0.0.1002,layer2=1
zfcp.allow_lun_scan	Option to control LUN disk scanning. 0 = disable 1 = enable Using DASD, ensure you set it to 0.	For example, setting: zfcp.allow_lun_scan=0 Disables the automatic LUN scan for FCP devices that run NPIV mode.i
cio_ignore	Remove the device from the list of ignored devices and make it visible to Linux,	cio_ignore=all,!condev
rd.dasd	Boot option to activate DASDs early in the boot process.	For example, setting: rd.dasd=0.0.0100 Tells kernel to activate 100 disk.,

Parameter	Description	Example
rd.neednet	Bring up network.	For example, setting: rd.neednet=1 Tells kernel to bring network up

In the following examples, we highlighted in red the values that change from one file to another to help you to quickly create the parm files:

– `/var/ftp/pub/bootstrap-0.parm file`

```
rd.neednet=1 coreos.inst=yes coreos.inst.install_dev=dasda
coreos.inst.image_url=ftp://129.40.23.109/pub/rhcos-4.4.9-s390x-dasd.s390x.r
aw.gz coreos.inst.ignition_url=ftp://129.40.23.109/pub/bootstrap.ign
ip=129.40.23.110::129.40.23.1:255.255.255.0:::none nameserver=129.40.23.109
rd.znet=qeth,0.0.1000,0.0.1001,0.0.1002,layer2=1,portno=0
zfcp.allow_lun_scan=0 cio_ignore=all,!condev rd.dasd=0.0.0100
```

– `/var/ftp/pub/master-1.parm file`

```
rd.neednet=1 coreos.inst=yes coreos.inst.install_dev=dasda
coreos.inst.image_url=ftp://129.40.23.109/pub/rhcos-4.4.9-s390x-dasd.s390x.r
aw.gz coreos.inst.ignition_url=ftp://129.40.23.109/pub/master.ign
.ign ip=129.40.23.111::129.40.23.1:255.255.255.0:::none
nameserver=129.40.23.109
rd.znet=qeth,0.0.1000,0.0.1001,0.0.1002,layer2=1,portno=0
zfcp.allow_lun_scan=0 cio_ignore=all,!condev rd.dasd=0.0.0100
```

– `/var/ftp/pub/master-2.parm`

```
rd.neednet=1 coreos.inst=yes coreos.inst.install_dev=dasda
coreos.inst.image_url=ftp://129.40.23.109/pub/rhcos-4.4.9-s390x-dasd.s390x.r
aw.gz coreos.inst.ignition_url=ftp://129.40.23.109/pub/master.ign
ip=129.40.23.112::9.16.8.1:255.255.255.0:::none nameserver=129.40.23.109
rd.znet=qeth,0.0.1000,0.0.1001,0.0.1002,layer2=1,portno=0
zfcp.allow_lun_scan=0 cio_ignore=all,!condev rd.dasd=0.0.0100
```

– `/var/ftp/pub/master-3.parm`

```
rd.neednet=1 coreos.inst=yes coreos.inst.install_dev=dasda
coreos.inst.image_url=ftp://129.40.23.109/pub/rhcos-4.4.9-s390x-dasd.s390x.r
aw.gz coreos.inst.ignition_url=ftp://129.40.23.109/pub/master.ign
ip=129.40.23.113::129.40.23.1:255.255.255.0:::none nameserver=129.40.23.109
rd.znet=qeth,0.0.1000,0.0.1001,0.0.1002,layer2=1,portno=0
zfcp.allow_lun_scan=0 cio_ignore=all,!condev rd.dasd=0.0.0100
```

– `/var/ftp/pub/worker-1.parm`

```
rd.neednet=1 coreos.inst=yes coreos.inst.install_dev=dasda
coreos.inst.image_url=ftp://129.40.23.109/pub/rhcos-4.4.9-s390x-dasd.s390x.r
aw.gz coreos.inst.ignition_url=ftp://129.40.23.109/pub/worker.ign
ip=129.40.23.114::129.40.23.1:255.255.255.0:::none nameserver=129.40.23.109
rd.znet=qeth,0.0.1000,0.0.1001,0.0.1002,layer2=1,portno=0
zfcp.allow_lun_scan=0 cio_ignore=all,!condev rd.dasd=0.0.0100
```

– /var/ftp/pub/worker-2.parm

```
rd.neednet=1 coreos.inst=yes coreos.inst.install_dev=dasda
coreos.inst.image_url=ftp://129.40.23.109/pub/rhcos-4.4.9-s390x-dasd.s390x.r
aw.gz coreos.inst.ignition_url=ftp://129.40.23.109/pub/worker.ign
ip=129.40.23.115::9.16.8.1:255.255.255.0:::none nameserver=129.40.23.109
rd.znet=qeth,0.0.1000,0.0.1001,0.0.1002,layer2=1,portno=0
zfcp.allow_lun_scan=0 cio_ignore=all,!condev rd.dasd=0.0.0100
```

Note: Although our installation uses a DASD disk, we included `zfcp.allow_lun_scan=0` in the parmfile; otherwise, the installation does not proceed as it tries to scan for LUN disks. Use this option for SAN disks.

Review all of the files to ensure the IP addresses are correct for each node. We experienced a duplicate IP address problem that we had to address before continuing.

4. Run the following commands to create files that use short names. Remember, z/VM has a limitation on the file name length (8 characters). Use following commands to create files in the `/var/ftp/pub` folder:

```
cp -a /var/ftp/pub/rhcos-4.4.9-s390x-installer-kernel-s390x
/var/ftp/pub/KERNEL.IMG
cp -a /var/ftp/pub/rhcos-4.4.9-s390x-installer-initramfs.s390x.img
/var/ftp/pub/INITRD.IMG
```

5. Transfer the initramfs (INITRD.IMG), kernel (KERNEL.IMG) and the parameter files to z/VM by completing the following steps. We use the **vmur** command to accomplish this task:

Note: Ensure that the **vmur** command is available. If it is not available, run the following command to install the `s390utils-base` package on your bastion node:

```
yum install -y s390utils-base
```

a. Load the module and activate the required devices by running the following commands:

```
/sbin/modprobe vmur
/usr/sbin/cio_ignore -r c-e
/usr/sbin/chccwdev -e c-e
```

b. Change directories to the `/var/ftp/pub/` directory:

```
cd /var/ftp/pub/
```

c. Upload files to the bootstrap server by using the commands that are shown in Example 3-37. We specify the VMID (RDBKO3B1) for the bootstrap node when **vmur** required.

Example 3-37 Upload files to bootstrap server

```
/usr/sbin/vmur pun -r -u RDBKO3B1 -N kernel.img /var/ftp/pub/KERNEL.IMG
/usr/sbin/vmur pun -r -u RDBKO3B1 -N generic.parm
/var/ftp/pub/bootstrap-0.parm
/usr/sbin/vmur pun -r -u RDBKO3B1 -N initrd.img /var/ftp/pub/INITRD.IMG
```

d. Upload the files to controller node 1 by using the commands that are shown in Example 3-38 on page 44. We specify the VMID (RDBKO3M1) for controller node 1 when **vmur** is used.

Example 3-38 Upload files to controller node 1

```
/usr/sbin/vmur pun -r -u RDBKO3M1 -N kernel.img /var/ftp/pub/KERNEL.IMG
/usr/sbin/vmur pun -r -u RDBKO3M1 -N generic.parm /var/ftp/pub/master-1.parm
/usr/sbin/vmur pun -r -u RDBKO3M1 -N initrd.img /var/ftp/pub/INITRD.IMG
```

e. Upload files to controller node 2 by using the commands that are shown in Example 3-39. We specify the VMID (RDBKO3M2) for controller node 2 when **vmur** is used.

Example 3-39 Upload files to controller node 2

```
/usr/sbin/vmur pun -r -u RDBKO3M2 -N kernel.img /var/ftp/pub/KERNEL.IMG
/usr/sbin/vmur pun -r -u RDBKO3M2 -N generic.parm /var/ftp/pub/master-2.parm
/usr/sbin/vmur pun -r -u RDBKO3M2 -N initrd.img /var/ftp/pub/INITRD.IMG
```

f. Upload files to controller node 3 by using the commands that are shown in Example 3-40. We specify the VMID (RDBKO3M3) for controller node 3 when **vmur** is used.

Example 3-40 Upload files to controller node 3

```
/usr/sbin/vmur pun -r -u RDBKO3M3 -N kernel.img /var/ftp/pub/KERNEL.IMG
/usr/sbin/vmur pun -r -u RDBKO3M3 -N generic.parm /var/ftp/pub/master-3.parm
/usr/sbin/vmur pun -r -u RDBKO3M3-N initrd.img /var/ftp/pub/INITRD.IMG
```

g. Upload files to worker node 1 by using the commands that are shown in Example 3-41. We specify the VMID f(RDBKO3W1) for worker node 1 when **vmur** is used.

Example 3-41 Upload files to worker node 1

```
/usr/sbin/vmur pun -r -u RDBKO3W1 -N kernel.img /var/ftp/pub/KERNEL.IMG
/usr/sbin/vmur pun -r -u RDBKO3W1 -N generic.parm /var/ftp/pub/worker-1.parm
/usr/sbin/vmur pun -r -u RDBKO3W1 -N initrd.img /var/ftp/pub/INITRD.IMG
```

h. Upload files to worker node 2 by using the commands that are shown in Example 3-42. We specify the VMID (RDBKO3W2) for worker node 2 when **vmur** is used.

Example 3-42 Upload files to worker node 2

```
/usr/sbin/vmur pun -r -u RDBKO3W2 -N kernel.img /var/ftp/pub/KERNEL.IMG
/usr/sbin/vmur pun -r -u RDBKO3W2 -N generic.parm /var/ftp/pub/worker-2.parm
/usr/sbin/vmur pun -r -u RDBKO3W2 -N initrd.img /var/ftp/pub/INITRD.IMG
```

Example 3-43 shows an example of the output you should receive when the **vmur** commands are run.

Example 3-43 Example of vmur output

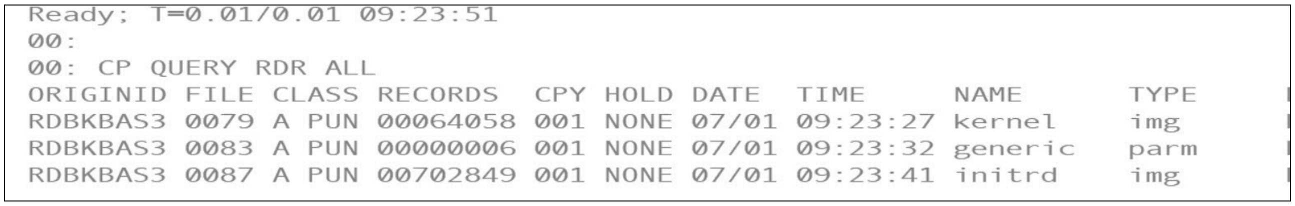
```
/usr/sbin/vmur pun -r -u RHOCPBOO -N kernel.img /var/ftp/pub/KERNEL.IMG
Reader file with spoolid 1057 created and transferred to RHOCPBOO.

/usr/sbin/vmur pun -r -u RHOCPBOO -N generic.parm
/var/ftp/pub/bootstrap-0.parm
Reader file with spoolid 1061 created and transferred to RHOCPBOO.

/usr/sbin/vmur pun -r -u RHOCPBOO -N initrd.img /var/ftp/pub/INITRD.IMG
Reader file with spoolid 1065 created and transferred to RHOCPBOO.
```

6. Verify that files were successfully uploaded on each z/VM guest. Because the bootstrap is the first server to be installed, log on to the z/VM guest VM that you used for the bootstrap installation (RDBKO3B1). You can use an x3270 or c3270 terminal emulator, which is available in your operating system, and perform the following steps:

   a. Query the guest reader by running the following command:

      ```
      CP QUERY RDR ALL
      ```

   The output of the command is shown in Figure 3-6.



```
Ready; T=0.01/0.01 09:23:51
00:
00: CP QUERY RDR ALL
ORIGINID FILE CLASS RECORDS  CPY HOLD DATE  TIME     NAME      TYPE
RDBKBAS3 0079 A PUN 00064058 001 NONE 07/01 09:23:27 kernel    img
RDBKBAS3 0083 A PUN 00000006 001 NONE 07/01 09:23:32 generic   parm
RDBKBAS3 0087 A PUN 00702849 001 NONE 07/01 09:23:41 initrd    img
```

Figure 3-6 cp query reader

   b. If you notice extra files, you must purge them and move the kernel image to the first position and parm file to the second position in the reader queue. To purge the files, run the purge command with the spool ID 0NNN, as shown in Example 3-44. To re-order the files, run the order command as shown in Example 3-44, where 0YYY is the spool ID.

   Example 3-44 Purge and re-order files

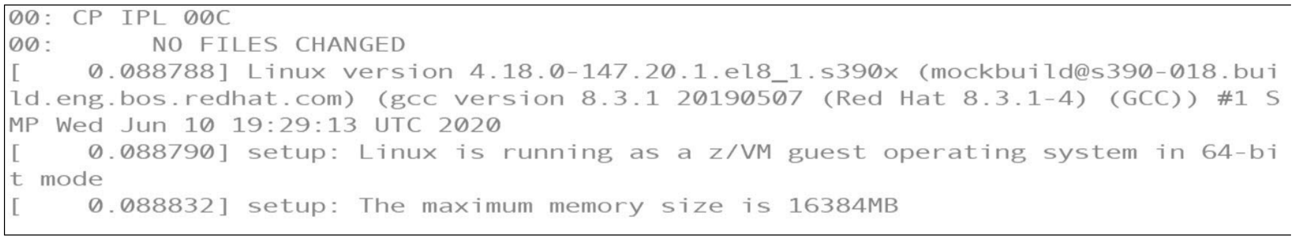
   ```
   purge rdr 0NNN
   order rdr 0YYY
   ```

   If you receive NO RDR FILES, it means that something went wrong and the files were not uploaded to z/VM. You must upload the files to that z/VM guest again by using the punch commands that are shown in Example 3-42.

7. To start the installation, run the following command:

   ```
   CP IPL 00c
   ```

   The installation starts and we see the output that is shown in Figure 3-7



```
00: CP IPL 00C
00:      NO FILES CHANGED
[    0.088788] Linux version 4.18.0-147.20.1.el8_1.s390x (mockbuild@s390-018.bui
ld.eng.bos.redhat.com) (gcc version 8.3.1 20190507 (Red Hat 8.3.1-4) (GCC)) #1 S
MP Wed Jun 10 19:29:13 UTC 2020
[    0.088790] setup: Linux is running as a z/VM guest operating system in 64-bi
t mode
[    0.088832] setup: The maximum memory size is 16384MB
```

Figure 3-7 Start the installation

8. After the installation completes, you see the output in the VM console that indicates that the server started, as shown in Example 3-45.

   Example 3-45 Linux is ready.

   ```
   Red Hat Enterprise Linux CoreOS 44.81.202006171550-0 (Ootpa) 4.4
   SSH host key: SHA256:u5MCV8ejWu+9h4kosogWvDF8EbaMyOyrkEbJBdy86D8 (ED25519)
   SSH host key: SHA256:kScOYNr5IhCjLJP8OridhEUOpyOcgFIXTMDvFUwDDT4 (ECDSA)
   SSH host key: SHA256:GpyRQzFqXaxMlvCSKL//BQU+dTH9JEXddtb6WkBBlzo (RSA)
   enc640: 129.40.23.110 fe80::1:2ff:fe00:1a
   ```

```
rdbko3b1 login: [175316.757707] SELinux: mount invalid.  Same superblock,
different security settings for (dev mqueue, type mqueue)
```

Alternatively, to check whether the server is ready, you can run the **ping** command to check network connectivity.

9. Run the following commands to disconnect from the console:

   ```
   #CP SET RUN ON
   #CP DISC HOLD
   ```

Installation of the bootstrap is completed.

Repeat the steps for the other IDs starting with controller node 1 and ending with worker node 2. You should start with the controller nodes and then go to the worker nodes. In our installation, we used the VMIDs that are listed inTable 3-2 for our installation.

Table 3-2 VMID

Node	**VMID**
Controller Node 1	RDBKO3M1
Controller Node 2	RDBKO3M2
Controller Node 3	RDBKO3M3
Worker Node 1	RDBKO3W1
Worker Node 2	RDBKO3W2

We recommend you monitor the bootstrap installation by following the instructions in 3.6, “Post deployment activities” on page 46.

OpenShift is now deployed.

3.6 Post deployment activities

This section describes the activities that must be performed after OpenShift is deployed. Complete the following steps:

1. After the VMs bootstrapped, run the following command to monitor the installation:

   ```
    cd /stage/
   ./openshift-install --dir=. wait-for bootstrap-complete --log-level debug
   ```

   Wait until you see the following message:

   ```
   INFO It is now safe to remove the bootstrap resources
   ```

2. The cluster image registry does not select a storage backend in User Provisioned Infrastructure (UPI) mode. Therefore, the cluster operator continually waits for an administrator to configure a storage backend. A workaround is to point the image-registry to an empty directory to allow the installation to complete by running the following commands:

   ```
    mkdir /root/.kube/
    cp /stage/auth/kubeconfig ~/.kube/config
    oc patch configs.imageregistry.operator.openshift.io cluster --type merge
   --patch '{"spec":{"storage":{"emptyDir":{}}}}'
   ```

3. After the installation is complete, change the image registry to a suitable location. Follow the steps that are listed in 3.9, “Setting up an image registry” on page 59.
4. Run the following command to monitor the installation:

   ```
   ./openshift-install --dir=. wait-for install-complete
   ```

   Wait until you see the message that is shown in Example 3-46. The system provides the URL and credentials to log in to the OpenShift console after the installation is complete. Your URL and password are different.

Example 3-46 Installation complete messages

```
INFO Install complete!
 INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/stage/auth/kubeconfig'
 INFO Access the OpenShift web-console here:
https://console-openshift-console.rdbk1.pok.ibm.local
 INFO Login to the console with user: kubeadmin, password: my-kube-password
The password for the user that was created during installation can also be found
in the auth subdirectory in the install-dir. It lets you log in through oc login
and also gives you access to the web console. The URL for the console is
https://console-openshift-console.<cluster>.<base_domain>.
```

5. Run the **oc get nodes** command and you see the worker nodes. If the worker nodes are not in the `Ready` state, you must repeatedly check for pending CSRs by using the **oc get csr** command. Then, approve the CSRs by using the **oc adm log in approve <csr_name>** command. Continue to check until the configuration is complete and all worker nodes are in the `Ready` state.
6. Run the following command from the `/stage/` directory:

   ```
   watch -n5 oc get clusteroperators
   ```

   Monitor for cluster completion. Sample output is shown in Example 3-47.

Example 3-47 Sample output from get clusteroperators command

```
NAME                                       VERSION   AVAILABLE   PROGRESSING
DEGRADED   SINCE
authentication                             4.4.9     True        False
False      25d
cloud-credential                           4.4.9     True        False
False      25d
cluster-autoscaler                         4.4.9     True        False
False      25d
console                                    4.4.9     True        False
False      3d14h
csi-snapshot-controller                    4.4.9     True        False
False      3d14h
dns                                        4.4.9     True        False
False      25d
etcd                                       4.4.9     True        False
False      3d15h
image-registry                             4.4.9     True        False
False      3d14h
ingress                                    4.4.9     True        False
False      3d14h
```

```
insights                                    4.4.9     True        False
False      25d
kube-apiserver                              4.4.9     True        False
False      3d15h
kube-controller-manager                     4.4.9     True        False
False      3d15h
kube-scheduler                              4.4.9     True        False
False      3d15h
kube-storage-version-migrator               4.4.9     True        False
False      3d14h
machine-api                                 4.4.9     True        False
False      25d
machine-config                              4.4.9     True        False
False      3d14h
marketplace                                 4.4.9     True        False
False      3d14h
monitoring                                  4.4.9     True        False
False      3d14h
network                                     4.4.9     True        False
False      25d
node-tuning                                 4.4.9     True        False
False      3d15h
openshift-apiserver                         4.4.9     True        False
False      3d14h
openshift-controller-manager                4.4.9     True        False
False      3d15h
openshift-samples                           4.4.9     True        False
False      3d15h
operator-lifecycle-manager                  4.4.9     True        False
False      25d
operator-lifecycle-manager-catalog          4.4.9     True        False
False      25d
operator-lifecycle-manager-packageserver    4.4.9     True        False
False      3d14h
service-ca                                  4.4.9     True        False
False      25d
service-catalog-apiserver                   4.4.9     True        False
False      25d
service-catalog-controller-manager          4.4.9     True        False
False      25d
storage                                     4.4.9     True        False
False      3d15h
```

For more information, see this web page.

3.7 Configuring a persistent volume for use by the OpenShift cluster

Persistent data is data that you want to be available, even if the container is removed. Application developers and container administrators naturally want to know how to preserve data for applications and users. They must also understand the requirements of their applications, the types of data that are involved, and how the data is accessed.

Containers are inherently ephemeral, meaning they last only a short time. They are routinely destroyed and rebuilt from a previously pushed application image. Keep in mind that after a container is removed, all container data is gone. With containers, it is necessary that you take specific actions to deal with this ephemeral behavior. OpenShift offers efficient ways to manage data of an application while respecting the ephemeral nature of containers.

By default, data that is generated inside the container is available only from within the container and only for the lifetime of the container instance. Persistent storage of data, such as application logs and cluster metrics, are not set up by default.

Before you get started with provisioning storage, it is important to understand the Kubernetes concepts of a persistent volume and a persistent volume claim and how they work together in a cluster.

3.7.1 Cluster

By default, every cluster is set up with a plug-in to provision file storage. You can choose to install other add-ons, such as one for block storage. To use storage in a cluster, you must create a persistent volume claim (PVC), a persistent volume (PV), and a physical storage instance. When you delete the cluster, you can delete related storage instances.

3.7.2 Persistent volume claim

A PVC is the request to provision persistent storage with a specific type and configuration. To specify the persistent storage type that you want, you use Kubernetes storage classes. The cluster admin can define storage classes, or you can choose from one of the predefined storage classes in Red Hat OpenShift on IBM Cloud®.

When you create a PVC, the request is sent to the IBM Cloud storage provider. Depending on the configuration that is defined in the storage class, the physical storage device is ordered and provisioned into your IBM Cloud infrastructure account. If the requested configuration does not exist, the storage is not created.

3.7.3 Persistent volume

A PV is a virtual storage instance that is added as a volume to the cluster. The PV points to a physical storage device in your IBM Cloud infrastructure account and abstracts the API that is used to communicate with the storage device. To mount a PV to an application, you must have a matching PVC. Mounted PVs appear as a directory inside the container's file system.

3.7.4 Physical storage

Physical storage is an instance that you can use to persist your data. Examples of physical storage in IBM Cloud include File Storage, Block Storage, Object Storage, and local worker node storage that you can use as software-defined storage (SDS) with Portworx or some other storage platform. IBM Cloud provides high availability for physical storage instances. However, data that is stored on a physical storage instance is not backed up automatically. Depending on the type of storage that you use, different methods exist to set up backup and restore solutions.

Container system administrators can create storage volumes, such as PVCs and PVs, to suit their requirements. The following provisioning options are available:

- Dynamic provisioning

  Container system administrators use dynamic volume provisioning to create storage volumes on-demand. It gives developers the freedom to provision storage when they need it.

- Static provisioning

  In static provisioning, an administrator creates several PVs, which include information about the storage that is available to each user in the cluster. Container system administrators must make the storage instance available to application developers.

In this IBM Redpaper publication, we set up dynamic provisioning by way of a Network File System (NFS). For more information, see this web page.

Note: Only NFS is supported for the OpenShift Container Platform on IBM Z.

NFS is an open-standard persistent file storage solution that is built into Linux (with RFCs for each different version of NFS). NFS allows client computers to access files from a remote server over a network. For more information, see this web page.

NFS uses a server that has the NFS server package that is installed with a volume that client computers can access over the network and uses the NFS client package. This approach allows remote clients to mount remote volumes as local volumes, which enables easy access of files across multiple computers. In a clustered system such as Kubernetes, this allows all nodes to keep up-to-date with the same storage. Thus, a pod can start on any worker node in the cluster and read its data. The NFS server must allow mount access from that node in its `/etc/exports` file.

We describe how to install an NFS server on the bastion node in 3.8, “Installing an NFS server” on page 51.

3.7.5 Application

To read from and write to your storage instance, you must mount the PVC to your application. Different storage types have different read-write rules. For example, you can mount multiple pods to the same PVC for file storage. Block storage comes with a ReadWriteOnce (RWO) access mode so that you can mount the storage to one pod only.

3.8 Installing an NFS server

In this section, we describe how to install an NFS server on the bastion node. To start, ensure that you are connected to the bastion node and ran the **sudo** command to access the system with root privileges.

The following steps are a simplified version of how to setup an NFS server. For more detailed RHEL instructions, see this web page.

Complete the following steps:

1. Install the NFS server package by running the following command:

```
yum install nfs-utils rpcbind
```

2. Create a directory by running the following commands:

```
mkdir -p //exports/nfsshare
chmod 755 /exports/
chmod 755 /exports/nfsshare
```

3. Add the following lines to the `/etc/exports` file:

```
# NFS auto provisioner
/exports/nfsshare *(rw,sync,no_root_squash)
```

Note: You can add the client hostname or IP address for each of the worker nodes. Also, you can use a wildcard with hostnames, such as `rdbko2w*` (if that name was the hostname of each of your workers within your network) or an IP subnet mask (for example, 129.40.23/24). For example:

```
/exports/nfsshare rdbko2w*(rw,sync,no_root_squash)
```

or

```
/exports/nfsshare 129.40.23.0/24(rw,sync,no_root_squash)
```

4. Start the NFS server by running the following commands:

```
systemctl enable --now rpcbind
systemctl enable --now nfs-server
```

5. To confirm that the folder is exported, run the following command:

```
exportfs -v
```

6. Open the local firewall ports by running the following commands:

```
firewall-cmd --permanent --add-service=nfs
firewall-cmd --permanent --add-service=rpc-bind
firewall-cmd --reload
```

Downloading code from the Kubernetes Incubator

Complete the following steps to download code from the Kubernetes Incubator GitHub project that contains the Provisioner interface and ProvisionController, which is a custom Kubernetes controller that watches PersistentVolumes and PersistentVolumeClaims:

1. Create /opt/nfs-client-provisioner folder:

```
mkdir /opt/nfs-client-provisioner
```

2. Go to nfs-client-provisioner:

   ```
   cd /opt/nfs-client-provisioner
   ```

3. Download the file that is named `master.zip` to a file called `kubernetes-incubator.zip` by running the following commands:

   ```
   curl -L -o kubernetes-incubator.zip
   https://github.com/kubernetes-incubator/external-storage/archive/master.zip
   ```

4. Uncompress the downloaded package by running the following command:

   ```
   unzip kubernetes-incubator.zip
   ```

5. Change directories to `external-storage-master/nfs-client/`:

   ```
   cd external-storage-master/nfs-client/
   ```

Configuring a service account and role bindings

A service account provides an identity for processes that run in a pod. A role binding grants the permissions that are defined in a role to a user or set of users. It holds a list of subjects (users, groups, or service accounts), and a reference to the role being granted.

Complete the following steps to configure a service account and the necessary role bindings:

1. Access the OpenShift cluster by using the `kudeadmin` user ID.

   You have two options to log in with the `kubeadmin` account:

   – Use the KUBECONFIG variable with the following commands:

     ```
     export KUBECONFIG=/stage/auth/kubeconfig
     oc get nodes
     ```

   – Use the `kubeadmin` password. Complete the following steps:

     i. Get kubeadmin password by running the following command:

        ```
        cat /stage/auth/kubeadmin-password ; echo
        ```

     ii. Authenticate to OpenShift by running the following command:

        ```
        oc login -u kubeadmin -p KUBEADMIN_PW https://BASTION_IP:6443
        ```

        where `KUBEADMIN_PW` is the password from step i and `BASTION_IP` is the IP address of your bastion node

     iii. Test by running the `oc get nodes` command.

2. Create a namespace called nfs-client-provisioner:

   ```
   oc create namespace nfs-client-provisioner
   ```

   You should receive the following output:

   ```
   namespace/nfs-client-provisioner created
   ```

3. Switch to the nfs-client-provisioner namespace project by running the following command:

   ```
   oc project nfs-client-provisioner
   ```

   You should receive the following output:

   ```
   Now using project "nfs-client-provisioner" on server
   "https://api.z-east1-int.ciocloud.nonprod.intranet.ibm.com:6443".
   ```

4. Set the variable to be used during installation by running the following command:

   ```
   export NAMESPACE=`oc project -q`
   ```

5. Check the variable by running the following command:

   ```
   echo $NAMESPACE
   ```

   You should receive the following output:

   ```
   nfs-client-provisioner
   ```

6. Run the following command to update the `deploy/rbac.yaml` file:

   ```
   sed -i'' "s/namespace:.*/namespace: $NAMESPACE/g" ./deploy/rbac.yaml
   ```

7. Create the required cluster roles by running the following command:

   ```
   oc create -f deploy/rbac.yaml
   ```

   You should receive the following output:

   ```
   serviceaccount/nfs-client-provisioner created
   clusterrole.rbac.authorization.k8s.io/nfs-client-provisioner-runner created
   clusterrolebinding.rbac.authorization.k8s.io/run-nfs-client-provisioner created
   role.rbac.authorization.k8s.io/leader-locking-nfs-client-provisioner created
   rolebinding.rbac.authorization.k8s.io/leader-locking-nfs-client-provisioner
   created
   ```

8. Allow the nfs-client-provisioned to run as root by running the following command:

   ```
   oc adm policy add-scc-to-user hostmount-anyuid
   system:serviceaccount:$NAMESPACE:nfs-client-provisioner
   ```

   You should receive the following output:

   ```
   securitycontextconstraints.security.openshift.io/hostmount-anyuid added to:
   ["system:serviceaccount:nfs-client-provisioner:nfs-client-provisioner"]
   ```

Adding the NFS dynamic provisioner

Complete the following steps:

1. Update the `deploy/deployment.yaml` file with the NFS share information, as shown in Example 3-48.

Note: Update values in red to fit your NFS configuration.

Example 3-48 The deploy/deployment.yaml file

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nfs-client-provisioner
  labels:
    app: nfs-client-provisioner
  # replace with namespace where provisioner is deployed
  namespace: nfs-client-provisioner
spec:
  replicas: 1
  strategy:
    type: Recreate
  selector:
    matchLabels:
      app: nfs-client-provisioner
  template:
    metadata:
      labels:
```

```
        app: nfs-client-provisioner
    spec:
      serviceAccountName: nfs-client-provisioner
      containers:
        - name: nfs-client-provisioner
          image: docker.io/gmoney23/nfs-client-provisioner-s390x:latest
          #image: quay.io/external_storage/nfs-client-provisioner:latest
          volumeMounts:
            - name: nfs-client-root
              mountPath: /persistentvolumes
          env:
            - name: PROVISIONER_NAME
              value: nfs-client-provisioner/nfs
            - name: NFS_SERVER
              value: 129.40.23.109
            - name: NFS_PATH
              value: /exports/nfsshare
      volumes:
        - name: nfs-client-root
          nfs:
            server: 129.40.23.109
            path: /exports/nfsshare
```

2. Update `deploy/class.yaml` file with the correct provisioner name (see Example 3-49).

Note: Update values in red to fit your NFS infrastructure

Example 3-49 The deploy/class.yaml file

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: managed-nfs-storage
provisioner: nfs-client-provisioner/nfs # or choose another name, must match
deployment's env PROVISIONER_NAME'
parameters:
  archiveOnDelete: "false"
```

3. Run the command that is shown in Example 3-50 to deploy the new storage class.

Example 3-50 Creating required resources

```
oc create -f deploy/class.yaml
storageclass.storage.k8s.io/managed-nfs-storage created

oc create -f deploy/deployment.yaml
deployment.apps/nfs-client-provisioner created
```

4. To monitor the nfs-client-provisioner pod creation, run the following command:

```
watch oc get pods
```

Wait until the pod status shows `Running`. You should receive the following results:

```
NAME                                      READY   STATUS    RESTARTS   AGE
nfs-client-provisioner-74d774fbdb-8kcr5   1/1     Running   0          10s
```

Press **CRTL+C** to quit.

Optionally, the pod logs can also assist you in verifying the pod status by completing the following steps:

a. Get name of the nfs-client-provisioner container by running the following command:

```
oc get pods
```

You should receive the following output. Highlighted in bold is the name of the container.

```
NAME                                      READY   STATUS    RESTARTS   AGE
nfs-client-provisioner-74d774fbdb-8kcr5   1/1     Running   0          15s
```

b. Check the pod logs for the nfs-client-provisioner container by running the following command:

```
oc logs  nfs-client-provisioner-74d774fbdb-8kcr5
```

Example 3-51 shows a sample of the output.

Example 3-51 Sample output

```
I0603 17:58:08.910496       1 leaderelection.go:187] attempting to acquire
leader lease  nfs-client-provisioner/nfs-client-provisioner-nfs...
I0603 17:58:08.917086       1 leaderelection.go:196] successfully acquired
lease nfs-client-provisioner/nfs-client-provisioner-nfs
I0603 17:58:08.917405       1 controller.go:571] Starting provisioner
controller
nfs-client-provisioner/nfs_nfs-client-provisioner-6595b9c777-kk76g_c87d9f22-
a5c3-11ea-9a73-0a580a820216!
I0603 17:58:08.917427       1 event.go:221]
Event(v1.ObjectReference{Kind:"Endpoints",
Namespace:"nfs-client-provisioner", Name:"nfs-client-provisioner-nfs",
UID:"3848eeee-fe42-405e-b851-71a9aeb3c95f", APIVersion:"v1",
ResourceVersion:"12157768", FieldPath:""}): type: 'Normal' reason:
'LeaderElection'
nfs-client-provisioner-6595b9c777-kk76g_c87d9f22-a5c3-11ea-9a73-0a580a820216
became leader
I0603 17:58:09.017563       1 controller.go:620] Started provisioner
controller
nfs-client-provisioner/nfs_nfs-client-provisioner-6595b9c777-kk76g_c87d9f22-
a5c3-11ea-9a73-0a580a820216!
```

Testing with a Persistent Volume Claim

Complete the following steps to test the new dynamic NFS storage class:

1. Create a claim resource and test pod to confirm the new storage class:

a. Go to the deploy directory:

```
cd deploy/
```

b. Check the `test-claim.yaml` file to confirm that the annotations option points to the new storage class called `managed-nfs-storage` (see Example 3-52).

Example 3-52 The test-claim.yaml file

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: test-claim
  annotations:
    volume.beta.kubernetes.io/storage-class: "managed-nfs-storage"
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Mi
```

c. Create a PVC resource by running the following command:

```
oc create -f test-claim.yaml
```

d. Check the status of your claim resource by running the following command:

```
oc get pvc
```

You should receive the following output:

```
NAME         STATUS    VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS          AGE
test-claim   Bound     pvc-18bba8d9-9a6d-4f35-bb18-7ac3c811baac   1Mi
RWX            managed-nfs-storage   37s
```

Note: If you see `Pending` for a test-claim resource, it means something is wrong with NFS sharing or nfs-client-provisioner deployment.

2. Complete the following steps to deploy a pod to test:

a. Edit the `test-pod.yaml` file with a valid IBM Z container image for the test (see Example 3-53). We updated our file to use the `docker.io/busybox:1.31` image.

Example 3-53 test-pod.yaml file

```
kind: Pod
apiVersion: v1
metadata:
  name: test-pod
spec:
  containers:
  - name: test-pod
    #image: gcr.io/google_containers/busybox:1.24
    image: docker.io/busybox:1.31
    command:
      - "/bin/sh"
    args:
      - "-c"
      - "touch /mnt/SUCCESS && exit 0 || exit 1"
    volumeMounts:
      - name: nfs-pvc
        mountPath: "/mnt"
  restartPolicy: "Never"
  volumes:
    - name: nfs-pvc
      persistentVolumeClaim:
        claimName: test-claim
```

b. Create the pod by running the following command:

```
oc create -f test-pod.yaml
```

c. Wait for 5 seconds and check the status of your new pod by running the following command:

```
oc get pods
```

You should receive the following output:

```
NAME       READY   STATUS      RESTARTS   AGE
test-pod   0/1     Completed   0          8s
```

Note: If you see any other status, it means something is wrong with NFS sharing or the nfs-client-provisioner deployment.

d. Check the NFS server sharing for the SUCCESS message that is created by the test pod. Run the commands that are shown in Example 3-54.

Example 3-54 Checking NFS server sharing

```
# ls -la /exports/nfsshare/
total 12
drwxr-xr-x. 3 root root 4096 Jun 29 10:55 .
drwxr-xr-x. 9 root root 4096 Jun  3 11:17 ..
drwxrwxrwx. 2 root root 4096 Jun 29 10:52
default-test-claim-pvc-18bba8d9-9a6d-4f35-bb18-7ac3c811baac

# ls -la
/exports/nfsshare/default-test-claim-pvc-18bba8d9-9a6d-4f35-bb18-7ac3c811baa
c/
total 8
drwxrwxrwx. 2 root root 4096 Jun 29 10:52 .
drwxr-xr-x. 3 root root 4096 Jun 29 10:55 ..
-rw-r--r--. 1 root root    0 Jun 29 10:52 SUCCESS
```

Setting default storage class to nfs-client-provisioner

Storage classes can map to a provisioner to dynamically provision PVs based on the volume claim requests that come in as users deploy workloads and services.

Some storage providers support the dynamic provisioning and abstract details so that developers do not need to take many steps to acquire, bind, and claim storage for their services.

A default storage class can dynamically provision storage when a storage class is not specified.

Complete the following steps to set up a default storage class on an OpenShift cluster:

1. Determine which storage class is default by running the command that is shown in Example 3-55.

Example 3-55 Checking Available Storage Classes

```
#oc get storageclass
NAME                            PROVISIONER                    AGE
managed-nfs-storage  nfs-client-provisioner/nfs   60m
```

2. Mark the managed-nfs-storage storageclass as default by completing the following steps:

   a. Run the **`oc edit storageclass/managed-nfs-storage`** command to add the two lines in red in Example 3-56.

Example 3-56 Running the oc edit storageclass/managed-nfs-storage command

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
  creationTimestamp: "2020-06-03T17:59:07Z"
  name: managed-nfs-storage
  resourceVersion: "12167441"
  selfLink: /apis/storage.k8s.io/v1/storageclasses/managed-nfs-storage
  uid: 5a4d3b19-b80e-42a0-b29e-be1cde4b91cd
parameters:
  archiveOnDelete: "false"
provisioner: nfs-client-provisioner/nfs
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

   b. To save this file, run the **`:wq`** command (this command is the same command that is used with the **`vim`** editor).

   c. Run the **`oc get storageclass -o yaml`** command to confirm if it was successfully updated. You should receive the output that is shown in Example 3-57 which shows the updates that you made to this file.

Example 3-57 Updated yaml file

```
apiVersion: v1
items:
- apiVersion: storage.k8s.io/v1
  kind: StorageClass
  metadata:
    annotations:
      storageclass.kubernetes.io/is-default-class: "true"
    creationTimestamp: "2020-06-03T17:06:05Z"
    name: managed-nfs-storage
    resourceVersion: "444482"
    selfLink: /apis/storage.k8s.io/v1/storageclasses/managed-nfs-storage
    uid: d0667b93-5086-4211-a950-c0323da4d8fb
  parameters:
    archiveOnDelete: "false"
  provisioner: default/nfs
  reclaimPolicy: Delete
  volumeBindingMode: Immediate
kind: List
metadata:
  resourceVersion: ""
  selfLink: ""
```

d. Optionally, you can also confirm by running the **oc** command:

```
oc get storageclass
```

You receive the following output:

```
NAME                          PROVISIONER                 AGE
managed-nfs-storage (default)   nfs-client-provisioner/nfs   69m
```

3.9 Setting up an image registry

OpenShift clusters include an internal registry to build, deploy, and manage container images locally. By default, your OpenShift cluster's internal registry is not configured and requires you to manually set it up to store the registry images.

Complete the following steps to set up an internal registry after you are connected to the bastion node with root privileges:

Note: This procedure requires that "Installing an NFS server" on page 51 is completed.

1. Go to the /opt/downloads directory:

   ```
   cd /opt/downloads
   ```

2. Use the `vim` editor to create a PVC file for use with the registry. We named our file `registry_nfs_pvc.yml`. An example of the contents of our file is shown in Example 3-58.

   Note: Ensure your bastion host has at least 120 GB available for the NFS directory (/exports).

   Example 3-58 PVC file for use with the registry.

   ```
   apiVersion: v1
   kind: PersistentVolumeClaim
   metadata:
    name: image-registry-storage
   spec:
     accessModes:
       - ReadWriteMany
     resources:
       requests:
         storage: 100Gi
   ```

3. Run the command that is shown in Example 3-59 to create the PV and PVC resources.

   Example 3-59 Create the resources

   ```
   # oc apply -f registry_nfs_pvc.yml -n openshift-image-registry
   persistentvolumeclaim/image-registry-storage created
   ```

4. Complete the following steps to configure the image registry to use the PVC `openshift-image-registry`:

   a. Edit the OpenShift image registry object by running the following command:

   ```
   oc edit configs.imageregistry.operator.openshift.io
   ```

   b. Update the storage section to use the `openshift-image-registry` PVC and ManagementState from Removed to Managed:

   > **Note:** You must also edit the Image Registry Operator configuration to switch the ManagementState from Removed to Managed.

   ```
   storage:
     pvc:
       claim: image-registry-storage
   ```

   c. Save and Exit by running the **`:wq`** command.

   d. Run the following command to verify private registry pods are running:

   ```
   watch -n5 oc get pods -n openshift-image-registry
   ```

   Monitor for registry completion. A sample output is shown in Example 3-60.

   Example 3-60 Verify private registry pods are running

   ```
   Every 5.0s: oc get clusteroperators
   NAME                                             READY   STATUS    RESTARTS   AGE
   cluster-image-registry-operator-674b45c8b9-lcs2c  2/2       Running       0   2d6h
   image-registry-7b76b55bbd-b9drd                    1/1       Running       0   2d6h
   node-ca-7228s                                     1/1       Running       1   2d7h
   node-ca-78hch                                     1/1       Running       1   2d7h
   node-ca-d6tjx                                     1/1       Running       1   2d7h
   node-ca-w9qgv                                     1/1       Running       1   2d7h
   node-ca-xvhz8                                     1/1       Running       1   2d7h
   ```

   The private registry is verified to be running.

To view volume details, including the storage class and size, run the following command:

```
oc describe pvc -n openshift-image-registry image-registry-storage
```

3.10 Accessing the console

Your system must correctly resolve DNS names before you can connect to the Red Hat OpenShift web console. This resolution must occur because the Red Hat OpenShift web console runs as an application within Red Hat OpenShift. Complete the following steps before you open the Red Hat OpenShift console:

1. Ensure that the system that you are using to connect to the Red Hat OpenShift web console can resolve hostnames in the DNS zone for your OpenShift cluster.

   As an alternative, you can configure your localhosts file with the entries that are shown in Example 3-61 on page 61, replacing our details with your own. This ensures that you can access the OpenShift web console.

Example 3-61 /etc/hosts file

```
# Redpaper OCP Cluster
129.40.23.109 console-openshift-console.apps.rdbk1.pok.ibm.local
129.40.23.109 oauth-openshift.apps.rdbk1.pok.ibm.local
129.40.23.109 api.apps.rdbk1.pok.ibm.local
```

2. Replace the APPLICATION_IP with the bastion node IP address.
3. Replace ClusterName, SubDomain, and DomainName with the information you recorded in 2.3, "Required DNS IP addresses and names" on page 13.

   Note: Our cluster uses the following information:

   - ClusterName: `rdbk1`
   - SubDomain: `pok`
   - DomainName: `ibm.local`

   Your localhosts file should contain the following elements:

   – APPLICATION_IP **`console-openshift-console.apps`**.ClusterName.SubDomain.DomainName
   – APPLICATION_IP **oauth-openshift.apps.**ClusterName.SubDomain.DomainName
   – APPLICATION_IP **api.apps.**ClusterName.SubDomain.DomainName

4. Open the following URL in your internet browser:

   `https://console-openshift-console.apps.ClusterName.SubDomain.DomainName`

5. For our environment, we used the following URL to connect to the Red Hat OpenShift web console:

   `https://console-openshift-console.apps.rdbk1.pok.ibm.local/`

4

Hints and tips

This chapter outlines some hints and tips to diagnose and correct some issues that you might encounter when installing Red Hat OpenShift on IBM Z.

This chapter includes the following topics:

4.1 Verifying RHEL subscription

Verify your RHEL subscriptions by running the following command:

```
subscription-manager list --available --all
```

4.2 Load balancer

To check that load balancing is working, run the following command from the bastion node. This command should return a full set of headers. Replace ClusterName, SubDomain, and DomainName with the information you recorded in section 2.3, "Required DNS IP addresses and names" on page 13:

```
wget https://api.apps.ClusterName.SubDomain.DomainName:6443
```

In our environment, we ran the following command:

```
wget --no-check-log in https://api.rdbk1.pok.ibm.local:6443
or using IP address of Bastion node
wget --no-check-log in https://129.40.23.109:6443
```

Alternatively, you can access the HAProxy page as described in "HAProxy Stats" on page 20

4.3 Red Hat Enterprise Linux CoreOS

There should be no need to connect by using SSH to node. However, if you must connect, you can do so by using the bastion node. As root, connect to the bootstrap server by running the following command:

```
ssh core@129.40.23.110
```

If you receive an error, run the following command that disables checking the fingerprint:

```
ssh -o StrictHostKeyChecking=no core@129.40.23.110
```

The following example shows how to connect to the control-plane node from the bastion node to view the etcd logs (your directory name might be different):

```
ssh -i /root/.ssh/id_rsa core@129.40.23.110
sudo su -
cd /var/log/pods/
ls -ld *etcd*
```

4.4 OpenShift commands

Table 4-1 lists some useful OpenShift commands.

Table 4-1 OpenShift commands

Command	Description
`oc get nodes`	Get a list of nodes and their status
`oc login`	Log in to the cluster

Command	Description
`oc get nodes`	List all nodes and show their roles and status
`oc get all`	Get all resources
`oc get pods --all-namespaces`	Get all PODs in all namespaces
`oc get clusteroperators`	Get cluster operators
`oc logs <POD_NAME> -n <NAMESPACE>`	Show all logs for a specific pod
`oc describe pod <POD NAME> -n <NAMESPACE>`	Show detailed information about a pod
`oc adm node-logs --role=master -u kubelet`	Get adm logs for control-plane nodes

The Kubernetes command-line tool, **`kubectl`**, allows you to run commands against Kubernetes clusters. A kubeconfig file exists behind every working **`kubectl`** command. When you use **`kubectl`**, a preference takes effect when determining which kubeconfig file is used.

Note: A file that is used to configure access to clusters is called a *kubeconfig file*. This term is a generic way of referring or pointing to specific configuration files.

The following commands are available:

- Run the **`--kubeconfig`** command to specify a file directly.

  For example: **`kubectl get pods --kubeconfig=file1`**.

- Use the **`KUBECONFIG`** environment variable, which is an environment variable that is a list of paths to the configuration file.

  For example: **`KUBECONFIG=file1 kubectl get pods`**.

- Use the `$HOME/.kube/config` file. If you have a `$HOME/.kube/config` file and it is not listed in your **`KUBECONFIG`** environment variable, append it to your **`KUBECONFIG`** environment variable.

  For example: **`export KUBECONFIG=$KUBECONFIG:$HOME/.kube/config`**.

  Run the following command to export the kubeconfig that is created by the OpenShift Installer to an environment variable:

  ```
  export KUBECONFIG=/stage/auth/kubeconfig
  ```

4.5 Certificate signing requests

Sometimes you might have to approve the worker and control-plane node's login signing requests (CSRs). If the bootstrap installation process is complete on all nodes and you find that one or more worker nodes are not listed in the command output of **`oc get nodes`**, use the following commands to approve the pending CSRs.

- To view pending CSRs:

  ```
  oc get csr
  ```

- To approve all pending CSRs at once:

  ```
  oc get csr -o go-template='{{range .items}}{{if not
  .status}}{{.metadata.name}}{{"\n"}}{{end}}{{end}}' | xargs oc adm log in
  approve
  ```

- Verify the API log ins. Run the following commands to confirm API log ins are valid:

  Replace ClusterName, SubDomain, and DomainName with the information you recorded in section 2.3, “Required DNS IP addresses and names” on page 13

  ```
  echo | openssl s_client -connect api.apps.ClusterName.SubDomain.DomainName:6443
  | openssl x509 -noout -text
  ```

Red Hat OpenShift uses kubelet client log ins that must be rotated periodically for security purposes. The initial log ins that are created during installation expire 24 hours after they are created. Red Hat OpenShift mainly automates the rotation process and starts the initial log in rotation.

Important: Do not restart any of the Red Hat OpenShift cluster virtual machines (VMs) or the bastion VM until the first log in rotation is done.

If you face an issue with CSRs, you can recover from expired control plane log ins and get the OpenShift cluster operational again. For more information, see this web page.

4.6 Adding a worker to an OpenShift Cluster

This section describes how to add a worker node to an OpenShift Cluster. The following process is used to add a worker node:

1. Create a virtual machine.
2. Create DNS records for the new worker.
3. Update the HAProxy.
4. Create an ignition file.
5. Deploy the new RHCOS.
6. Approve any log in signing requests.

4.6.1 Creating a virtual machine

Complete the following steps to add a VM on z/VM for the RHCOS installation:

1. Connect to the z/VM LPAR by using your x3270 terminal and create the following file: `RDBKO3W3.DIRECT`.

   Use XEDIT to create this file.

   Note: Remember to update the NICDEF tags to fit your network. In our environment, we defined VSWITCH1 and VLAN number 0001, as shown in red in Example 4-1.

   Example 4-1 Example of creating a worker file-RDBKO3W3.DIRECT file

   ```
   USER RDBKO3W3 REDHAT 8G 8G G
      INCLUDE DFLTLNX
      CPU 00 BASE
      CPU 01
      IPL 100
      POSIXINFO   UID 100754
      NICDEF 1000 TYPE QDIO LAN SYSTEM VSWITCH1
      NICDEF 1000 VLAN 0001
   ```

2. Define an IPL disk for this new guest ID. We use a DASD model A that supports large disks. In our example, we used the following commands to define the IPL disk:

   ```
   DIRM FOR RDBKO3W3 AMDISK 100 3390 AUTOG 271000 MYPOOL
   ```

   In this example, one disk with 271000 cylinders (approximately. 180 GB) is added. The disk number is 100.

   For more information about how to create a VM, see 2.1, “Required virtual machine resources” on page 8.

4.6.2 Creating DNS records for the new worker

Complete the following steps to create the A record and associated PTR record:

1. Add the following line into the file `/var/named/forward.zone`:

   ```
   rdbko3w3.rdbk1.pok IN A 129.40.23.116
   ```

Note: In our example, the new worker node uses the following host name and IP address:

- Host name: `rdbko3w3.rdbk1.pok.ibm.local`
- IP address: `129.40.23.116`

2. Add the following line to the `/var/named/reverse.zone` file:

   ```
   116 IN PTR rdbko3w3.rdbk1.pok.ibm.local.
   ```

3. Recycle the DNS by running the following command:

   ```
   systemctl restart named
   ```

4.6.3 Updating the HAProxy

Complete the following steps to add the worker node into the load balancer:

1. Update `/etc/haproxy/haproxy.cfg` file with the information that is shown in red in Example 4-2 to add information for the new worker node.

   Example 4-2 Update the haproxy.cfg file to include the new worker

   ```
   backend habackend-http-80
       mode tcp
       balance roundrobin
       server rdbko3w1 129.40.23.114:80  check
       server rdbko3w2 129.40.23.115:80  check
       server rdbko3w3 129.40.23.116:80  check

   backend habackend-https-443
       balance roundrobin
       mode tcp
       server rdbko3w1 129.40.23.114:443  check
       server rdbko3w2 129.40.23.115:443  check
       server rdbko3w3 129.40.23.116:443  check
   ```

2. After making the update, run the following command to recycle the HAProxy service:

   ```
   systemctl restart haproxy
   ```

3. Verify that the new worker was added by using the HAProxy stats page. The new worker node appears under backend sections.

4.6.4 Creating an ignition file

Complete the following steps to create an ignition file:

1. Go to `/var/ftp/pub` by running the following command:

   ```
   cd /var/ftp/pub
   ```

2. Create a copy of the `worker.ign` file by running the following command:

   ```
   cp -a worker.ign worker.ign.backup
   ```

3. Define a variable that is named MCS and assign the API server name and port to it (in our example, `api-int.rdbk1.pok.ibm.local`):

   ```
   export MCS=api-int.rdbk1.pok.ibm.local:22623
   ```

4. Run the following command to update the ignition file with OpenShift API log in:

   ```
   echo "q" | openssl s_client -connect $MCS -showcerts | awk '/-----BEGIN log
   in-----/,/-----END log in-----/' | base64 --wrap=0 | tee ./api-int.base64 &&
   sed --regexp-extended --in-place=.backup "s%base64,[^,]+%base64,$(cat
   ./api-int.base64)\"%" ./worker.ign
   ```

5. Check whether the file changed. If the file changed, proceed with the steps to deploy the new worker. If the file did not change, verify that the API server is up and repeat the steps.

4.6.5 Deploy the new RHCOS

Complete the following steps to deploy the new RHCOS:

1. Go to `/var/ftp/pub` folder by running the following command:

   ```
   cd /var/ftp/pub
   ```

2. Create the `/var/ftp/pub/worker-3.parm` file and populate it with the content that is shown in Example 4-3.

   Example 4-3 Creating the worker parameter file

   ```
   rd.neednet=1 coreos.inst=yes coreos.inst.install_dev=dasda
   coreos.inst.image_url=ftp://129.40.23.109/pub/rhcos-4.4.9-s390x-dasd.s390x.raw.
   gz coreos.inst.ignition_url=ftp://129.40.23.109/pub/worker.ign
   ip=129.40.23.116::9.16.8.1:255.255.255.0:::none nameserver=129.40.23.109
   rd.znet=qeth,0.0.1000,0.0.1001,0.0.1002,layer2=1,portno=0 zfcp.allow_lun_scan=0
   cio_ignore=all,!condev rd.dasd=0.0.0100
   ```

   Review the file to ensure that the IP address and URLs are correct. Eliminate any duplicate IP addresses.

3. Transfer the initramfs (INITRD.IMG), kernel (KERNEL.IMG), and parameter files to z/VM. We run the **`vmur`** command to make this transfer.

   Note: Ensure that the **`vmur`** command is available. If it is not, run the following command to install the `s390utils-base` package on your bastion node:

   ```
   yum install -y s390utils-base
   ```

a. Load the module and activate required devices by running the following commands:

```
/sbin/modprobe vmur
/usr/sbin/cio_ignore -r c-e
/usr/sbin/chccwdev -e c-e
```

b. Upload files to worker node 3 by running the commands that are shown in Example 4-4. We specify the VMID for worker node 3 when running **vmur**.

Example 4-4 Upload files to worker node 3

```
/usr/sbin/vmur pun -r -u RDBKO3W3 -N kernel.img /var/ftp/pub/KERNEL.IMG
/usr/sbin/vmur pun -r -u RDBKO3W3 -N generic.parm /var/ftp/pub/worker-3.parm
/usr/sbin/vmur pun -r -u RDBKO3W3 -N initrd.img /var/ftp/pub/INITRD.IMG
```

4. Ensure that you verify that the files were successfully uploaded on the target z/VM guest. Log on to the z/VM guest VM that was chosen for the new worker node (RDBKO3W3). You can use a x3270 or c3270 terminal emulator to complete the following steps:

   a. Query the guest reader by running the following command:

   ```
   CP QUERY RDR ALL
   ```

   b. If you notice more files, you must purge them and move the kernel image to the first position and parm file to the second position in the reader queue. To purge the files, run the **purge** command with the spool ID 0NNN, as shown in Example 4-5. To reorder the files, run the **order** command, as shown in Example 4-5, where `0YYY` is the spool ID

   Example 4-5 Purge and reorder files

   ```
   purge rdr 0NNN
   order rdr 0YYY
   ```

   If you receive the `NO RDR FILES` message, it means that something went wrong and the files were not uploaded to z/VM. You must upload the files to that z/VM guest again by running the **punch** commands that are listed in Example 4-4.

5. To start installation, run the following command:

   ```
   CP IPL 00c
   ```

   The installation starts and we see several Linux kernel messages.

6. After you see the Linux log, run the following commands to disconnect from the console:

   ```
   #CP SET RUN ON
   #CP DISC HOLD
   ```

The worker node is now installed.

4.6.6 Approving any log in signing requests

To provision the new worker, you might have to approve CSRs from the bastion server. Complete the following steps:

1. Log in to the bastion as the root user and change to the bastion installation directory.

   Before you can run any commands, you must authenticate to OpenShift.

   If authentication is not configured and you use the default kubeadmin account and password, run the **`export KUBECONFIG=/stage/auth/kubeconfig`** command and verify that you are authenticated by running the **`oc whoami`** command.

If other backends or users are authenticated, log in by running the following command:

```
oc login -u <user_name> -p <user_password) https://api.rdbk1.ibm.local:6443
```

2. Run the **`oc get nodes`** command and you should see the new worker. If it is not yet in the Ready state, you must repeatedly check for pending CSRs by running the **`oc get csr`** command and then approve the CSRs by running the **`oc adm log in approve <csr_name>`** command. Continue to check until the configuration is complete and the new worked is in the Ready state.

   After the new worker is in Ready state, it can be used by OpenShift.

4.7 OpenShift private registry

An OpenShift registry is a storage and distribution system for container images. You can store public images for free and share them with other users.

Red Hat OpenShift includes an internal registry to build, deploy, and manage container images locally. Because we are using an IBM Z, our required set up was to use NFS to store container images.

Complete the following steps to work with an OpenShift private registry to pull, push, and view images:

1. Log in to one of the controller nodes by way of SSH by running the following command:

   ```
   ssh -i $HOME/.ssh/id_rsa core@129.40.23.110
   ```

   Where `129.40.23.110` is the IP address of one of your controller nodes.

2. Authenticate with your cluster. Use your own user ID, password, and `api.clusterName` in the following command:

   ```
   oc login -u kubeadmin -p <password_from_install>
   https://api.ClusterName.SubDomain.DomainName:6443
   ```

3. Log in to the container image registry with your access token (in this case, user ID and password):

   ```
   podman login -u kubeadmin -p $(oc whoami -t)
   image-registry.openshift-image-registry.svc:5000
   ```

4. Check all image streams in your registry by running the following command:

   ```
   oc get imagestreams  -A
   ```

   You should receive output similar to the output that is shown in Example 4-6.

   Example 4-6 Output of get imagestreams command

   ```
   NAMESPACE  NAME            IMAGE REPOSITORY
   TAGS     UPDATED
   default     php-73
   image-registry.openshift-image-registry.svc:5000/default/php-73                latest
   4 weeks ago
   openshift   cli
   image-registry.openshift-image-registry.svc:5000/openshift/cli                 latest
   8 days ago
   openshift   cli-artifacts
   image-registry.openshift-image-registry.svc:5000/openshift/cli-artifacts       latest
   11 days ago
   ```

```
openshift   installer
image-registry.openshift-image-registry.svc:5000/openshift/installer           latest
11 days ago
openshift   installer-artifacts
image-registry.openshift-image-registry.svc:5000/openshift/installer-artifacts  latest
8 days ago
openshift   must-gather
image-registry.openshift-image-registry.svc:5000/openshift/must-gather         latest
8 days ago
openshift   oauth-proxy
image-registry.openshift-image-registry.svc:5000/openshift/oauth-proxy         v4.4
8 days ago
openshift   tests
image-registry.openshift-image-registry.svc:5000/openshift/tests               latest
8 days ago
z-apps      ubuntu
image-registry.openshift-image-registry.svc:5000/z-apps/ubuntu                 latest
8 days ago
```

5. Optionally, check image streams by running the **curl** command, as shown in the following example:

```
# curl -k -u unused:$(oc whoami -t)
https://image-registry.openshift-image-registry.svc:5000/v2/_catalog?n=100
{"repositories":["default/php-73","openshift/cli","openshift/cli-artifacts","op
enshift/installer","openshift/installer-artifacts","openshift/must-gather","ope
nshift/oauth-proxy","openshift/tests","z-apps/ubuntu"]}
```

6. To pull an image from the Docker Hub into the local cache, run the following command:

```
$ podman pull s390x/ubuntu:latest
```

The output should look similar to the following example:

```
Trying to pull registry.access.redhat.com/s390x/ubuntu:latest...
  name unknown: Repo not found
Trying to pull docker.io/s390x/ubuntu:latest...
Getting image source signatures
Copying blob b3937ed3ecfe done
Copying blob 82ded071ccca done
Copying blob e3d3a3cd4bce done
Copying blob a2c8c4d2943d done
Copying config a2973813e5 done
Writing manifest to image destination
Storing signatures
a2973813e5bf0445eaf260dcaf4b393eaa156199106bdd4126804335a03100b8
```

7. Confirm that the image is available locally by running the following command:

```
\$ podman images
```

The output is similar to the following example:

```
REPOSITORY              TAG      IMAGE ID       CREATED      SIZE
docker.io/s390x/ubuntu  latest   a2973813e5bf   4 days ago   73.6 MB
```

8. Run the **podman tag** command to tag the new image by using the following format:

```
<registry_ip>:<port>/<project>/<image>
```

The project name (namespace) must appear in this pull specification for OpenShift Container Platform to correctly place and later access the image in the registry.

To tag the image, run the following command:

```
podman tag docker.io/s390x/ubuntu:latest
image-registry.openshift-image-registry.svc:5000/default/ubuntu:latest
```

Note: We used default as the `<project>` in our command.

9. Confirm that the new tag was created for the Ubuntu image by running the following command:

   ```
   $ podman images
   ```

   The expected output should be similar to the following example with your image ID and tag:

   ```
   REPOSITORY                                                         TAG
   IMAGE ID       CREATED      SIZE
   docker.io/s390x/ubuntu                                             latest
   a2973813e5bf   4 days ago   73.6 MB
   image-registry.openshift-image-registry.svc:5000/default/ubuntu   latest
   a2973813e5bf   4 days ago   73.6 MB
   ```

10. Push the new image into the private registry by running the following command:

    ```
    podman push
    image-registry.openshift-image-registry.svc:5000/default/ubuntu:latest
    ```

    Expected output should be similar to the following example:

    ```
    Getting image source signatures
    Copying blob 34f01642c668 done
    Copying blob 7e615b21c648 done
    Copying blob 90826e4d4013 done
    Copying blob 33083667ca19 done
    Copying config a2973813e5 done
    Writing manifest to image destination
    Storing signatures
    ```

11. List the images to confirm that the Ubuntu image was added into the default namespace by running the following command:

    ```
    $ curl -k -u unused:$(oc whoami -t)
    https://image-registry.openshift-image-registry.svc:5000/v2/_catalog?n=100
    {"repositories":["default/php-73","default/ubuntu","openshift/cli","openshift/c
    li-artifacts","openshift/installer","openshift/installer-artifacts","openshift/
    must-gather","openshift/oauth-proxy","openshift/tests","z-apps/ubuntu"]}
    [core@z-master01 ~]$
    ```

4.8 Adding disks

From time to time, you might find that you must add large disks. z/VM CMS does not support formatting disks with a large size. However, RHCOS includes a utility that is called ignition, which is used by RHCOS to manipulate disks during initial configuration. It completes common disk tasks, including partitioning disks, formatting partitions, writing files, and configuring users.

You can optionally reduce the time of the RHCOS installation by using the RHCOS ignition utility in the following steps to format the disks in advance:

1. Ensure the bastion node (RDBKBAS3) where the disk exists is logged off.
2. (Optional) If your z/VM LPAR uses IBM RACF®, run the following commands to allow the bastion node to link to the bootstrap IPL disk and grant privileges:

   ```
   # vmcp link rdbkbas3 100 f100 m
   RAC PERMIT  RDBKBAS3.100 CL(VMMDISK) ID(RH8SERV1) AC(CONTROL)
   ```

   If you do not grant the privileges, you receive the following error:

   ```
   RPIMGR032E YOU ARE NOT AUTHORIZED TO LINK TO RDBKBAS3.100
   HCPLNM298E RDBKBAS3 0100 not linked; request denied
   Error: non-zero CP response for command 'link rdbkbas3 100 f100 m': #298
   ```

3. Connect to your bastion node by way of SSH as the root user.
4. Run the **`cio_ignore`** command to remove f100 from the exclude device list, as shown here:

   ```
   cio_ignore -r f100
   ```

5. Run the **`chccwdev`** command to activate the F100 disk, as shown here:

   ```
   # chccwdev -e f100

   Setting device 0.0.f100 online

   Done
   ```

6. Run the **`dasdfmt`** command to format the F100 disk, as shown here:

   ```
   # /sbin/dasdfmt -y -b 4096 -m 500 /dev/disk/by-path/ccw-0.0.f100
   ```

   The output from this command is shown in Example 4-7.

   Example 4-7 Format the DASD

   ```
   Printing hashmark every 500 cylinders.
   ################################################################################
   ################################################################################
   ################################################################################
   ################################################################################
   ################################################################################
   ################################################################################
   ####################################################################
   Finished formatting the device.
   Rereading the partition table... ok
   ```

7. Create first partition by running the following command:

   ```
   # fdasd -a /dev/disk/by-path/ccw-0.0.f100
   ```

   You should receive output similar to the following example:

   ```
   reading volume label ..: VOL1
   reading vtoc ..........: ok

   auto-creating one partition for the whole disk...
   writing volume label...
   writing VTOC...
   rereading partition table...
   ```

8. Deactivate the disk by running the following command:

   ```
   # chccwdev -d f100
   Setting device 0.0.f100 offline
   Done
   ```

9. Detach the disk from bastion by running the following command:

   ```
   # vmcp det 100
   DASD 0100 DETACHED
   ```

10. Remove the RACF privileges that are assigned to the bastion node by running the following command on VM as the MAINT user ID or a privileged user:

    ```
    RAC PERMIT  RDBKBAS3.100 CL(VMMDISK) ID(RH8SERV1) AC(CONTROL) DELETE
    ```

Repeat these steps for each disk you need to format.

5

Updating versions

This chapter discusses the planning steps that are recommended before upgrading the Red Hat OpenShift cluster. The present environment is considered and what types of upgrade can be handled by the Red Hat OpenShift Container Platform update service.

It is important to update to the latest version of OpenShift to apply the latest enhancements and bug fixes, which help to improve user productivity and business responsiveness.

This chapter includes the following topics:

5.1 Overview

Periodically, you must update your OpenShift cluster with the latest updates to ensure that your system is secure and protected, improve performance, or fix bugs. These updates are released by the OpenShift Container Platform update service and affect the OpenShift Container Platform and Red Hat Enterprise Linux CoreOS (RHCOS).

You can update your cluster by using the OpenShift client (oc) or web console (`https://console-openshift-console.<cluster>.<base_domain>`).

Issues can be resolved by using one of the following methods:

- Keep your cluster environment up to date.
- Make sure that your command-line tools are up to date.

OpenShift Container Platform 4.4 offers the following update channels:

- Stable: This channel contains releases as soon as their errata are published. Releases are added to the stable channel after a delay of several hours to a day.
- Fast: This channel is updated with new versions as soon as Red Hat declares that the specific version as a general availability release.
- Candidate: Release candidates include all the features of the product, but are not supported. Use release candidate versions to test feature acceptance and assist in qualifying the next version of OpenShift Container Platform.

Note: For production clusters, you must subscribe to the stable-4.x channel.

This section helps system administrators analyze their entire OpenShift environment to ensure that there are no problems before an update.

Before beginning your update, it is important to perform these maintenance tasks to ensure that the system is ready and can be rolled back if issues occur. This checklist is meant to identify important, and often time-consuming, pre-update tasks that you can perform before the update to reduce risks and ensure a smooth update. Although these tasks are not mandatory, they are recommended for any update. It can help you to identify any problems with pods or nodes before you perform the update.

The following tasks are discussed in this section:

- “Reviewing hardware requirements” on page 77
- “Confirming OpenShift cluster version” on page 77
- “Reviewing status for cluster operators” on page 77
- “Verifying the status of your nodes” on page 78
- “Verifying the status of your pods” on page 79

If your cluster is healthy, you can proceed to “Updating your OpenShift cluster” on page 79 and perform the update by completing steps in that section.

5.1.1 Reviewing hardware requirements

Check whether the CPU and memory requirements for your control plane and compute nodes changed by completing the following steps:

1. See this Red Hat OpenShift web page.
2. Select the suitable OpenShift version (in our example, 4.4).
3. Review the minimum requirements.

5.1.2 Confirming OpenShift cluster version

Complete the following steps:

1. Log in to your cluster by running the **oc** command:

   ```
   oc login -u <userid> -p <password> https://api.<clusterid>.<domain>:6443
   ```

2. Run the following command to verify the current cluster version:

   ```
   oc get clusterversion
   ```

   The expected output should be similar to the following example:

   ```
   NAME      VERSION   AVAILABLE   PROGRESSING   SINCE   STATUS
   version   4.3.18 True        False         16h     Cluster version is 4.3.18
   ```

5.1.3 Reviewing status for cluster operators

Complete the following steps to ensure that you do not have any degraded or progressing operators in your cluster:

1. Run the following command:

   ```
   oc get clusteroperators
   ```

   The expected output is shown in Example 5-1.

   Example 5-1 Checking for degraded or progressing operators

   ```
   NAME                                       VERSION   AVAILABLE   PROGRESSING
   DEGRADED   SINCE
   authentication                             4.3.18    True        False
   False      99d
   cloud-credential                           4.3.18    True        False
   False      99d
   cluster-autoscaler                         4.3.18    True        False
   False      99d
   console                                    4.3.18    True        False
   False      57d
   dns                                        4.3.18    True        False
   False      58d
   image-registry                             4.3.18    True        False
   False      36d
   ingress                                    4.3.18    True        False
   False      47d
   insights                                   4.3.18    True        False
   False      99d
   kube-apiserver                             4.3.18    True        False
   False      99d
   ```

```
kube-controller-manager                    4.3.18    True        False
False      99d
kube-scheduler                             4.3.18    True        False
False      99d
machine-api                                4.3.18    True        False
False      99d
machine-config                             4.3.18    True        False
False      47d
marketplace                                4.3.18    True        False
False      47d
monitoring                                 4.3.18    True        False
False      35d
network                                    4.3.18    True        False
False      99d
node-tuning                                4.3.18    True        False
False      3d
openshift-apiserver                        4.3.18    True        False
False      47d
openshift-controller-manager               4.3.18    True        False
False      64d
openshift-samples                          4.3.18    True        False
False      58d
operator-lifecycle-manager                 4.3.18    True        False
False      99d
operator-lifecycle-manager-catalog         4.3.18    True        False
False      99d
operator-lifecycle-manager-packageserver   4.3.18    True        False
False      36h
service-ca                                 4.3.18    True        False
False      99d
service-catalog-apiserver                  4.3.18    True        False
False      99d
service-catalog-controller-manager         4.3.18    True        False
False      99d
storage                                    4.3.18    True        False
False      58d
```

If you find failed operators, we recommended that they are fixed before starting the update process.

5.1.4 Verifying the status of your nodes

Confirm that you have all nodes ready before the upgrade. Complete the following steps to verify the status of your control plane and compute nodes:

1. Query all nodes by running the following command:

   ```
   oc get nodes -o wide
   ```

   The expected output is shown in Example 5-2.

Example 5-2 Verification of node status

```
   NAME                     STATUS   ROLES    AGE   VERSION
   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE
   KERNEL-VERSION         CONTAINER-RUNTIME
```

```
   master1.ocp4.ibm.local   Ready    master   99d   v1.14.6-152-g117ba1f
   9.16.8.24     <none>        Red Hat Enterprise Linux CoreOS
   42s390x.81.20200406.0 (Ootpa)   4.18.0-147.el8.s390x
   cri-o://1.14.12-10.dev.rhaos4.2.git313d784.el8
   master2.ocp4.ibm.local   Ready    master   99d   v1.14.6-152-g117ba1f
   9.16.8.25     <none>        Red Hat Enterprise Linux CoreOS
   42s390x.81.20200406.0 (Ootpa)   4.18.0-147.el8.s390x
   cri-o://1.14.12-10.dev.rhaos4.2.git313d784.el8
   master3.ocp4.ibm.local   Ready    master   99d   v1.14.6-152-g117ba1f
   9.16.8.26     <none>        Red Hat Enterprise Linux CoreOS
   42s390x.81.20200406.0 (Ootpa)   4.18.0-147.el8.s390x
   cri-o://1.14.12-10.dev.rhaos4.2.git313d784.el8
   worker1.ocp4.ibm.local   Ready    worker   99d   v1.14.6-152-g117ba1f
   9.16.8.27     <none>        Red Hat Enterprise Linux CoreOS
   42s390x.81.20200406.0 (Ootpa)   4.18.0-147.el8.s390x
   cri-o://1.14.12-10.dev.rhaos4.2.git313d784.el8
worker2.ocp4.ibm.local   Ready    worker   42d   v1.14.6-152-g117ba1f
9.16.8.28     <none>        Red Hat Enterprise Linux CoreOS
42s390x.81.20200406.0 (Ootpa)   4.18.0-147.el8.s390x
cri-o://1.14.12-10.dev.rhaos4.2.git313d784.el8
```

2. Review the output and ensure that all nodes are healthy.

5.1.5 Verifying the status of your pods

Run the following command to confirm that all pods are running before the updates:

```
oc get pods -A | grep -v "Running\|Completed\|Terminated"
```

Note: You can redirect the output of this command to a log file appending `/tmp/all_pods_before_update.log` to the end of command.

The following output is expected:

```
NAMESPACE                              NAME
READY   STATUS      RESTARTS   AGE
```

The pre-check tasks are not mandatory, but it is a best practice to perform them for any update. It can help you to identify if problems with pods or nodes were caused by an update.

If your cluster is healthy, you can continue and perform the update as described in 5.2, “Updating your OpenShift cluster” on page 79.

5.2 Updating your OpenShift cluster

You can use the following options to update OpenShift Container Platform clusters:

- Graphical user interface by using the OpenShift web console
- Text interface by using the OpenShift client

Before you proceed with the upgrade, you can use a bash script that is called **`ocp4upc.sh`** to check available upgrade possibilities for your OpenShift cluster. This tool gives you an overview about upgrade paths by using stable and fast production-ready.

To install OpenShift Container Platform Upgrade Paths Checker (OCP4UPC), complete the following steps after you are connected to the bastion node and have root privileges:

1. Install all required packages by using the following command:

   ```
   yum install -y curl jq graphviz
   ```

2. Go to the following FTP directory:

   ```
   cd /var/ftp/pub
   ```

3. Download the OCP4UPC bash script by running the following command:

   ```
   curl -o ocp4upc.sh
   https://raw.githubusercontent.com/pamoedom/ocp4upc/master/ocp4upc.sh
   ```

   The expected output is similar to the following example:

   ```
     % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                    Dload  Upload   Total   Spent    Left  Speed
   100 13334  100 13334    0     0  30865      0 --:--:-- --:--:-- --:--:-- 30865
   ```

4. Update the file permission on the bash script by running the following command:

   ```
   chmod 700 ocp4upc.sh
   ```

5. Confirm that the file permissions are correct by running the following command:

   ```
   ls -la ocp4upc.sh
   ```

   The expected is similar to the following example:

   ```
   -rwx------. 1 root root 13334 Jul  1 07:09 ocp4upc.sh
   ```

6. Generate the graph for your current OpenShift 4 version (for example, 4.3.18):

   ```
   /var/ftp/pub/ocp4upc.sh 4.3.18 s390x
   ```

   Where:

   – 4.3.18 is your current OpenShift 4 version
   – s390x is the IBM Z platform

   The output of the command looks similar to the output that is shown in Example 5-3. Two files are created (highlighted in blue in Example 5-3).

   Example 5-3 Generating the graph data

   ```
   [INFO] Checking prerequisites (curl jq dot)... [OK]
   [INFO] Errata provided (4.x.z mode), targeting '4.4' channels for upgrade path
   generation.
   [INFO] Checking if '4.3.18' (s390x) is a valid release... [OK]
   [INFO] Result exported as 'stable-4.4_s390x_20200701.svg'
   [INFO] Result exported as 'fast-4.4_s390x_20200701.svg'
   ```

7. Open the graph by using an internet browser. Use the following URLs:

   ```
   ftp://129.40.23.109/pub/stable-4.4_s390x_20200701.svg'
   ftp://129.40.23.109/pub/fast-4.4_s390x_20200701.svg
   ```

   Where `129.40.23.109` the IP address of your Bastion node.

An example of the graph is shown in Figure 5-1.

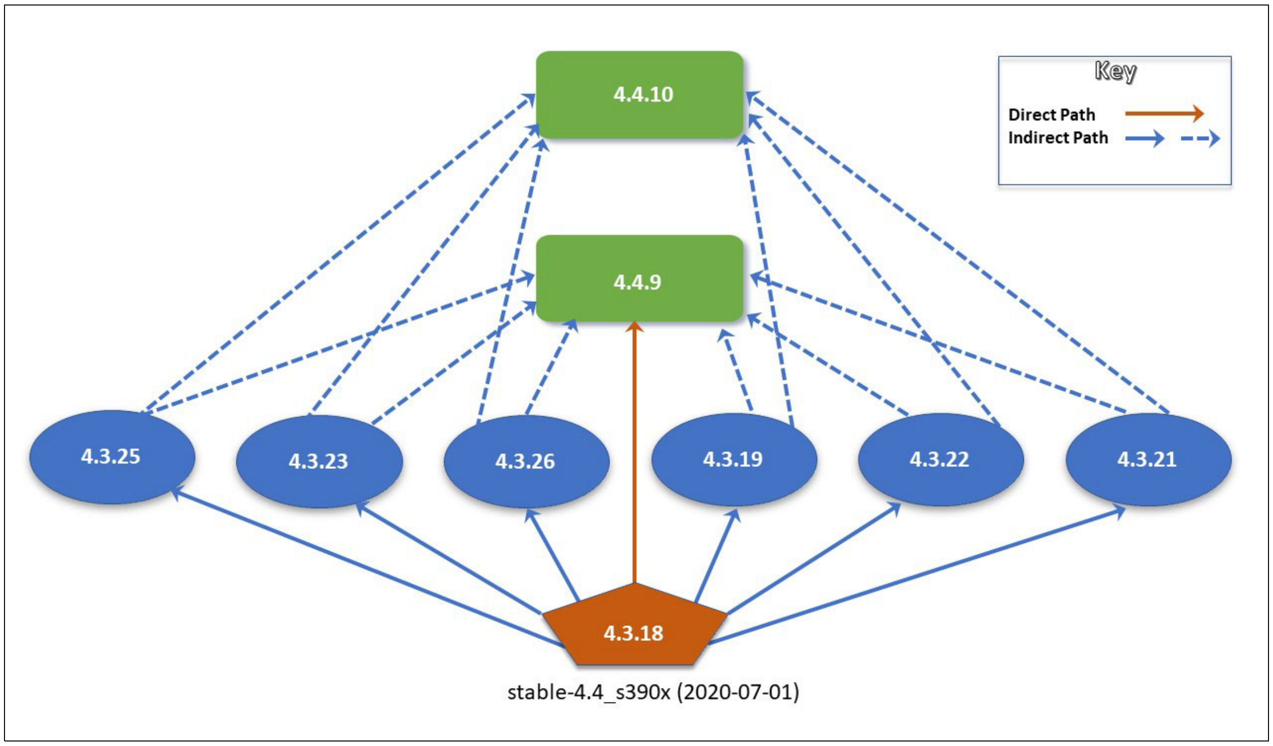


Figure 5-1 OCP4 Upgrade Paths Checker Output

In Figure 5-1, the upgrade is performed from 4.3.18 to 4.3.26 and then to 4.4.10. In our example, the following target versions are used:

– Minor target version: 4.3.26
– Major target version: 4.4.10

We chose the stable 4-x upgrade channel to perform an update by using these minor and major target versions.

After reviewing the possibilities, proceed with the upgrade.

Using the graphical interface

Complete the following steps to perform an upgrade by using the OpenShift web console.

These steps provide information to update your OpenShift Cluster from 4.3.y (for example, 4.3.18) to the latest version available on stable-4.3 channel that is 4.3.26:

1. Open the Web console (`https://console-openshift-console.<cluster>.<base_domain>`) by using an internet browser.
2. Log in as the kubeadmin user or as one of the members of the group whose members have cluster-admin privileges.
3. Select **Administration** → **Cluster Settings** → **Overview**.
4. Check the current version and the message `Update Available` is shown.
5. Keep the channel at `stable-4.3`.
6. Click **Update now**.
7. In the Update Cluster dialog, select as New Version 4.3.29, and click **Update**.

8. Wait until the update completes, which means that all the Cluster Operators are with status Available and with version 4.3.26.

If the update is successfully applied and the cluster is healthy and running on 4.3.26, you can repeat these steps. Under step 5 on page 81, select the **`stable-4.4`** channel (the target one) and update system to the latest version available in this channel.

5.2.1 Using the OpenShift client text interface

Complete the following steps to upgrade by using the OpenShift client (OC):

1. Log in to your cluster by running the **`oc`** command line interface (CLI) as follows:

   ```
   oc login -u <userid> -p <password> https://api.<clusterid>.<domain>:6443
   ```

   or

   ```
   export KUBECONFIG=/stage/auth/kubeconfig
   ```

Note: The latest s390x version of the **`oc`** command can be downloaded from this OpenShift web page.

2. To run an upgrade from previous minor versions, such as release 4.3.19 to 4.3.26 and applying asynchronous errata updates within a minor version, you can continue with the upgrade without changing the current 4.x channel. To do this, complete the following steps:

   a. Check the latest updates available to the cluster.

      ```
      oc adm upgrade --to-latest
      ```

   b. To monitor the upgrade process, run the following command:

      ```
      watch -n10 "oc get clusterversion && echo && oc get co && echo && oc get
      nodes -o wide"
      ```

      Wait until it is 100% complete and then press CTRL+C to exit.

   c. Continue to step 3 to upgrade the major target from 4.3.26 to 4.4.9.

3. Update the OpenShift channel from 4.3 to 4.4 (next version) by running the following command:

   ```
   oc patch clusterversion/version -p '{"spec":{"channel":"stable-4.4"}}'
   --type=merge
   ```

   Note: OpenShift uses the `.spec.channel` field to handle the updates.

4. Run the following command to update to the latest version for this channel:

   ```
   oc adm upgrade --to-latest
   ```

   You receive the following message:

   ```
   Updating to latest version 4.4.9
   ```

   To monitor the progress of the upgrade, run the following command:

   ```
   watch -n10 "oc get clusterversion && echo && oc get co && echo && oc get nodes
   -o wide"
   ```

   Wait until the process is complete and press CTRL+C to exit.

For more information, see “Related publications” on page 95.

6

Ansible and OpenShift

This chapter provides information about how to automate the steps to install required services, such as DNS, HAProxy, and FTP on the bastion node.

The topics that are covered in this chapter include installing the prerequisite packages, such as Python and Ansible, downloading and setting up Ansible roles, and configuring Ansible playbook variables for successful installations.

This chapter includes the following topics:

6.1 Ansible overview

Ansible is an engine that automates multiple tasks, such as cloud provisioning and multitier application deployment. Instead of writing many commands to install OpenShift, Ansible allows you to easily set up DNS, FTP, and HAProxy services on a bastion node.

Ansible is a configuration management and provisioning tool. Based on Python and YAML language, it is easy to learn and use, and includes an impressive catalog of pre-built content in Ansible Galaxy for provisioning and configuration of open source applications. Galaxy provides pre-packaged units of work that are known to Ansible as *roles*.

Ansible connects your nodes and issues programs, called *Ansible modules*, to them. It starts these modules over SSH and removes them when the execution is complete.

For more information, see this web page.

Use Ansible if you want to automate the manual steps you ran when setting up the bastion host.

Consider the following points regarding Ansible:

- Built on open source and sponsored by Red Hat.
- Is a configuration management and provisioning tool, similar to Chef and Puppet.
- Designed to automate multitier app deployments and provisioning in the cloud.
- Written in Python.
- Uses YAML syntax to describe automation tasks, which makes Ansible easy to learn and use.
- Uses SSH (no agents to install on remote systems).

An overview of the Ansible architecture is shown in Figure 6-1.

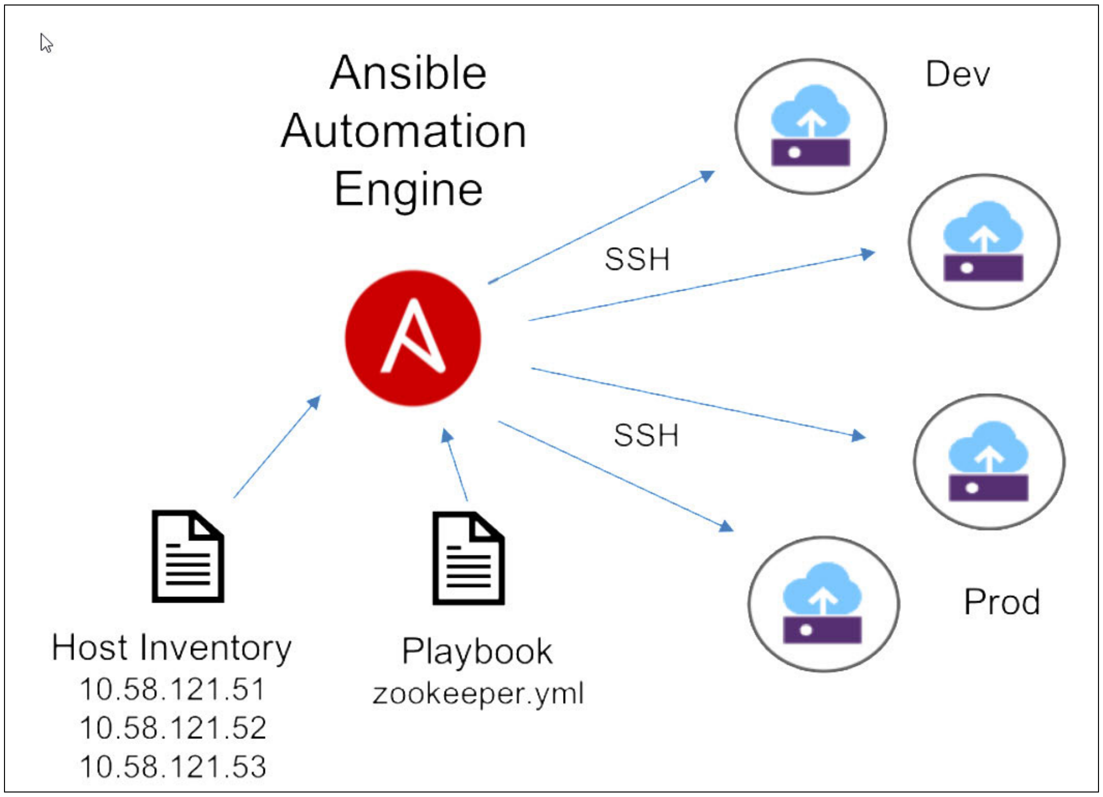


Figure 6-1 Ansible architecture

6.2 Installing OpenShift by using Ansible

In this section, we describe preparing an OpenShift environment. The host name for the bastion node is set in the Ansible inventory file, required packages are installed, and, among other things, properly configured.

6.2.1 Provisioning a Red Hat virtual machine

In this section, we guide you through the process to provision a Red Hat virtual machine (VM) on the bastion node.

This publication does *not* cover configuring a Red Hat VM; however, you can find more information at this web page.

Complete the following steps after you are connected to the bastion node and have root privileges:

1. Enable the EPEL repository.

   Many options are available to install Ansible. We used the Extra Packages for Enterprise Linux (EPEL) repository to install the Ansible package. We ran the following command:

   ```
   yum install https://dl.fedoraproject.org/pub/epel/epel-release-latest-$(rpm -E
   %rhel).noarch.rpm
   ```

Note: EPEL 8 s390x repository can be found at this web page.

2. Install the Ansible package by running the following command.

   ```
   yum -y install ansible
   ```

3. Create a user ID to be used with Ansible by running the following commands:

   ```
   useradd automation
   echo "automation:redhat" | chpasswd
   ```

4. Grant root privileges to the automation ID by running the following commands:

   ```
   cd /etc/sudoers.d
   echo '%automation ALL=(ALL) NOPASSWD:ALL' > automation
   ```

5. Switch to the automation ID by running the **`sudo`** command:

   ```
   sudo su - automation
   ```

6. Create a directory under the automation home directory (`/home/automation`) for the Ansible files:

   ```
   mkdir -p ocp
   ```

7. Go to the `ocp` folder:

   ```
   cd ocp
   ```

8. Create the following files with content that is shown:

   a. `ansible.cfg`:

      ```
      [defaults]
      inventory=/home/automation/ocp/inventory
      roles_path=/home/automation/ocp/roles
      remote_user=automation
      host_key_checking=false
      ```

   b. Inventory file (named `inventory`):

      ```
      [ocp]
      # Put the name of your bastion node here
      rdbkbas3.rdbk1.pok.ibm.local
      ```

9. Create the SSH keys for the automation ID:

   ```
   ssh-keygen -t rsa -N '' -f /home/automation/.ssh/id_rsa
   ```

10. Deploy the SSH key on the bastion node as an authorized key. Remember to update `rdbkbas2.rdbk1.pok.ibm.local` with your bastion node name and enter the password for the automation ID (in our example, **`redhat`**).

    ```
    ssh-copy-id -i /home/automation/.ssh/id_rsa.pub rdbkbas3.rdbk1.pok.ibm.local
    ```

    You should receive the following output:

    ```
    Number of key(s) added: 1
    ```

11. Test the access by using the following command:

    ```
    ssh rdbkbas3.rdbk1.pok.ibm.local date
    ```

    If it returns a date and time, the SSH configuration is correct.

12. Create a directory for the Ansible Galaxy role and browse to it:

    ```
    mkdir -p /home/automation/ocp/roles
    ```

    ```
    cd /home/automation/ocp/roles
    ```

13. Return to the ocp directory:

```
cd /home/automation/ocp/
```

14. Download the Ansible Galaxy role that is named `emarins.ocp` and put it into the `/home/automation/ocp/roles/` directory:

```
ansible-galaxy install emarins.ocp -p /home/automation/ocp/roles/
```

15. Copy the sample files to the ocp directory by using the following commands:

```
cp -a roles/emarins.ocp/globalvars.yml /home/automation/ocp/
cp -a roles/emarins.ocp/playbook_ocp.yml /home/automation/ocp/
```

16. Check the `globalvars.yml` file and update each variable to fit your network and cluster settings. The description of each tag is shown in Example 6-1.

Note: Update all values with the information for your OpenShift environment.

Example 6-1 globalvars.yml file

```
---
domain_name: pok.ibm.local       # OpenShift domain name
cluster_name: rdbk1          # OpenShift Cluster name
dns_network: 129.40.23.0/24  # DNS network where bootstrap, controller and worker nodes
will
domain_reverse: 23.40.129.in-addr.arpa    # Reverse DNS zone for your network. Ask
Network
defaultgw: 129.40.23.254    # Default gateway for the cluster network. Ask Network team
if you
netmask: 255.255.255.0      # Network subnet mask. Used for the PARM file. Ask Network
team
ipldisk: "0100"             # IPL disk address for the RHCOS instances. Disk for RHCOS
dns1: 129.40.106.1          # Primary DNS Server. Used in the DNS forwarder
/etc/named.conf
dns2: 129.40.106.2          # Secondary DNS Server. Used in the DNS forwarder
/etc/named.conf
vnic_addresses: "0.0.0640,0.0.0641,0.0.0642"  # Virtual Network Interface (VNIC). Ask
your
vnic_layer2: 1              # Layer 2 information. Ask network team if you do not know
how to get it
ocp_download_dir: /opt/downloads       # Where you need to put the instalation files
that were
rhcos_kernel: rhcos-4.4.9-s390x-installer-kernel-s390x  # RHCOS kernel file. Used to
build
rhcos_initrd: rhcos-4.4.9-s390x-installer-initramfs.s390x.img # RHCOS initrd file. Used
to build
rhcos_image: rhcos-4.4.9-s390x-dasd.s390x.raw.gz  # RHCOS image name. Used for the PARM
openshift_installer: openshift-install-linux.tar.gz  # OpenShift installer file name.
Used to install
openshift_client: openshift-client-linux.tar.gz      # OpenShift Client file name.  Used
to manage
worker_replicas: 2                       # Number of worker nodes. Recommended is two
worker nodes.
ocp_stage_dir: /stage                   # Temporary folder that will be used to install
OpenShift

all_nodes:               # Master, Worker, bootstrap and other nodes information.
  # Master Nodes
  - id: 1
    name: rdbko3m1       # Update with your value
    ip: 129.40.23.111     # Update with your value
    vmid: RDBKO3M1       # Update with your value
    role: control_node

  - id: 2
    name: rdbko3m2        # Update with your value
    ip: 129.40.23.112     # Update with your value
    vmid: RDBKO3M2       # Update with your value
    role: control_node

  - id: 3
    name: rdbko3m3        # Update with your value
    ip: 129.40.23.113     # Update with your value
    vmid: RDBKO3M3       # Update with your value
    role: control_node

  # Workers Nodes
  - id: 1
    name: rdbko3w1        # Update with your value
    ip: 129.40.23.114     # Update with your value
    vmid: RDBKO3W1       # Update with your value
    role: worker_node

  - id: 2
    name: rdbko3w2        # Update with your value
    ip: 129.40.23.115     # Update with your value
    vmid: RDBKO3W2       # Update with your value
    role: worker_node

  # Bootstrap
  - id: 0
    name: rdbko3b1      # Update with your value
    ip: 129.40.23.110     # Update with your value
    vmid: RDBKO3B1       # Update with your value
    role: bootstrap_node

  # Bastion
  - id: 0
    name: rdbkbas3       # Update with your value
    ip: 129.40.23.109     # Update with your value
    vmid: RDBKBAS3       # Update with your value
    role: bastion_node

  # etcd
  - id: 0
    name: etcd-0       # Do not change etcd-0 name
    ip: 129.40.23.111   # Update with the IP address of Master node 1
    role: etcd_node

  - id: 1
    name: etcd-1       # Do not change etcd-1 name
    ip: 129.40.23.112   # Update with the IP address of Master node 2
    role: etcd_node

  - id: 2
    name: etcd-2       # Do not change etcd-2 name
    ip: 129.40.23.113 # Update with the IP address of Master node 3
    role: etcd_node
```

17. Copy the SSH public key (`id_rsa`) to the ocp directory. It is used by the Ansible playbook:

```
cp /home/automation/.ssh/id_rsa.pub /home/automation/ocp/
```

18. Create the download directory and change the file permission to the automation ID:

```
sudo mkdir -p /opt/downloads
sudo chown automation /opt/downloads -R
```

19. Follow the instructions that are described 3.2, “Downloading OpenShift files” on page 32 to download required files and put them into the `/opt/downloads` directory.

20. Run the following command to set up required services on your bastion server:

```
ansible-playbook playbook_ocp.yml
```

Note: After your OpenShift cluster is operational, back up the `/stage` directory by running the **`cp -a /stage/ /stage.bkp`** command.

21. Confirm that the files that are shown in Example 6-2 were created in the `/var/ftp/pub` directory:

```
ls -l /var/ftp/pub
```

Example 6-2 Files created for required services

```
   total 1385308
   -rw-r--r--. 1 root root  49181833 Apr 27 21:44 INITRD.IMG
   -rw-r--r--. 1 root root   5120485 Apr 27 21:44 KERNEL.IMG
   -rw-r--r--. 1 root root        26 Jan 13 10:34 README
   -rw-r--r--. 1 root root       162 Apr 27 21:44 RHCOS.EXEC
   -rw-r--r--. 1 root root       389 Apr 28 08:38 RHCOS.PARM
   -rw-r--r--. 1 root root       385 Apr 28 07:54 RHCOS.PARM.ORI
   -rw-r--r--. 1 root root       382 Apr 28 07:54 bootstrap-0.parm
   -rw-r--r--. 1 root root    289428 Apr 28 08:39 bootstrap.ign
   -rw-r--r--. 1 root root       379 Apr 28 07:53 master-1.parm
   -rw-r--r--. 1 root root       379 Apr 28 07:53 master-2.parm
   -rw-r--r--. 1 root root       379 Apr 28 07:53 master-3.parm
   -rw-r--r--. 1 root root      1819 Apr 28 08:38 master.ign
   -rw-r--r--. 1 root root  24181127 Apr 27 21:46 openshift-client-linux.tar.gz
   -rw-r--r--. 1 root root  78883637 Apr 27 21:46 openshift-install-linux.tar.gz
   -rwxr-xr-x. 1 root root       272 Apr 27 22:13 punch-bootstrap-0.sh
   -rwxr-xr-x. 1 root root       269 Apr 27 22:13 punch-master-1.sh
   -rwxr-xr-x. 1 root root       269 Apr 27 22:13 punch-master-2.sh
   -rwxr-xr-x. 1 root root       269 Apr 27 22:13 punch-master-3.sh
   -rwxr-xr-x. 1 root root       269 Apr 27 22:13 punch-worker-1.sh
   -rwxr-xr-x. 1 root root       269 Apr 27 22:13 punch-worker-2.sh
   -rw-r--r--. 1 root root 630434101 Apr 27 20:58
   rhcos-4.2.18-s390x-metal-dasd.raw.gz
   -rw-r--r--. 1 root root 630368558 Apr 27 20:58
   rhcos-4.2.18-s390x-metal-zfcp.raw.gz
   -rw-r--r--. 1 root root       379 Apr 28 07:53 worker-1.parm
   -rw-r--r--. 1 root root       379 Apr 28 07:53 worker-2.parm
-rw-r--r--. 1 root root      1819 Apr 28 08:38 worker.ign
```

The Ansible playbook creates scripts to help you to punch the three required files to the reader of each z/VM guest machine.

Note: During the installation process, it is critical to monitor the HAProxy to ensure that cluster services are coming up (green state). We recommend that you use the HAProxy stats page that is enabled by way of the Ansible playbook. The address is `https://x.x.x.x:8404/stats`, where `x.x.x.x` is the IP address of your bastion server.

The default user ID is `admin` and the default password is `redhat`. If you want to change the user ID or password, change this information in the `/etc/haproxy/haproxy.cfg` file.

On the stats page, you might see the bootstrap down after the control nodes take control of the OpenShift Cluster. This outcome is expected.

Complete the following steps to send the files and start installing RHCOS on each node:

a. Load the module and activate the required devices:

```
/sbin/modprobe vmur
/usr/sbin/cio_ignore -r c-e
/usr/sbin/chccwdev -e c-e
```

b. Go to the `/var/ftp/pub/` directory:

```
cd /var/ftp/pub/
```

c. Upload files to the bootstrap server by running the command that is shown in Example 6-3. Use the `/var/ftp/pub/punch-bootstrap-0.sh` script.

Example 6-3 Upload files to bootstrap server

```
# /var/ftp/pub/punch-bootstrap-0.sh
Reader file with spoolid 1069 created and transferred to RDBKO3B1.
Reader file with spoolid 1073 created and transferred to RDBKO3B1.
Reader file with spoolid 1077 created and transferred to RDBKO3B1.
```

d. Upload files to controller node 1 by running the command that is shown in Example 6-4. Use the **`/var/ftp/pub/punch-master-1.sh`** script.

Example 6-4 Upload files to controller node 1

```
# /var/ftp/pub/punch-master-1.sh
Reader file with spoolid 1081 created and transferred to RDBKO3M1.
Reader file with spoolid 1085 created and transferred to RDBKO3M1.
Reader file with spoolid 1089 created and transferred to RDBKO3M1.
```

e. Upload files to controller node 2 by running the command that is shown in Example 6-5. Use the **`/var/ftp/pub/punch-master-2.sh`** script.

Example 6-5 Upload files to controller node 2

```
# /var/ftp/pub/punch-master-2.sh
Reader file with spoolid 1093 created and transferred to RDBKO3M2.
Reader file with spoolid 1097 created and transferred to RDBKO3M2.
Reader file with spoolid 1101 created and transferred to RDBKO3M2.
```

f. Upload files to controller node 3 by running the command that is shown in Example 6-6 on page 91. Use the **`/var/ftp/pub/punch-master-3.sh`** script.

Example 6-6 Upload files to controller node 3

```
# /var/ftp/pub/punch-master-3.sh
Reader file with spoolid 1105 created and transferred to RDBKO3M3.
Reader file with spoolid 1109 created and transferred to RDBKO3M3.
Reader file with spoolid 1113 created and transferred to RDBKO3M3.
```

g. Upload files to worker node 1 by running the command shown that is in Example 6-7. Use the **/var/ftp/pub/punch-worker-1.sh** script.

Example 6-7 Upload files to worker node 1

```
# /var/ftp/pub/punch-worker-1.sh
Reader file with spoolid 1117 created and transferred to RDBKO3W1.
Reader file with spoolid 1121 created and transferred to RDBKO3W1.
Reader file with spoolid 1125 created and transferred to RDBKO3W1.
```

h. Upload files to worker node 2 by running the command that is shown in Example 6-8. Use the **/var/ftp/pub/punch-worker-1.sh** script.

Example 6-8 Upload files to worker node 2

```
# /var/ftp/pub/punch-worker-2.sh
Reader file with spoolid 1129 created and transferred to RDBKO3W2.
Reader file with spoolid 1133 created and transferred to RDBKO3W2.
Reader file with spoolid 1137 created and transferred to RDBKO3W2.
```

22. Ensure that files were successfully uploaded on each z/VM guest. Because the bootstrap is the first server to be installed, log on to the z/VM guest virtual machine that was chosen for the bootstrap installation (RDBKO3B1). Use an x3270 or c3270 terminal emulator and complete the following steps:

a. Query the guest reader by running the following command:

```
CP QUERY RDR ALL
```

The output of the command is shown in Example 6-9.

Example 6-9 Output of the cp query rdr command

```
Ready;  T-0.01/0.01 09:23:51
00:
00:  CP QUERY RDR ALL
ORIGINID  FILE  CLASS   RECORDS CPY  HOLD  DATE  TIME      NAME    TYPE
RDBKBAS3  0079  A PUN   00064058 001  NONE  07/01 09:23:27  kernel  img
RDBKBAS3  0083  A PUN   00000006 001  NONE  07/01 09:23:32  generic parm
RDBKBAS3  0087  A PUN   00702849 001  NONE  07/01 09:23:41  initrd  img
```

b. If you notice extra files, you must purge them and move the kernel image to the first position and parm file to the second position in the reader queue. To purge the files, run the **purge** command with the spool ID 0NNN, as shown in Example 6-10. To reorder the files, run the **order** command, as shown in Example 6-10, where 0YYY is the spool ID.

Example 6-10 Purge and reorder files

```
purge rdr 0NNN
order rdr 0YYY
```

If you receive the NO RDR FILES message, it means that something went wrong and the files were not uploaded to z/VM. You must upload the files to that z/VM guest again by using the punch scripts that are listed in Example 6-3 on page 90 - Example 6-8.

23. To start the installation, run the following command:

```
CP IPL 00C
```

The installation starts and we see the output that is shown in Example 6-11.

Example 6-11 Output from IPL command

```
00:  CP IPL 00C
00:       NO FILES CHANGED
[     0.088788] Linux version 4.18.0-147.20.1.el8_1.s390x (mockbuild@s390-018bui
ld.eng.bos.redhat.com) (gcc version 8.3.1 20190507 (Red Hat 8.3.1-4) (GCC)) #1 S
MP Wed June 10 19:29:13 UTC2020
[    0.088790] setup: Linux is running as a z/VM guest operating system in 64-bi
t mode
[    0.088832] setup: The maximum memory size is 16384MB
```

24. After the installation completes, you see the Linux login, as shown in Example 6-12.

Example 6-12 Linux login prompt

```
Red Hat Enterprise Linux CoreOS 44.81.202006171550-0 (Ootpa) 4.4
SSH host key: SHA256:u5MCV8ejWu+9h4kosogWvDF8EbaMyOyrkEbJBdy86D8 (ED25519)
SSH host key: SHA256:kScOYNr5IhCjLJP8OridhEUOpyOcgFIXTMDvFUwDDT4 (ECDSA)
SSH host key: SHA256:GpyRQzFqXaxMlvCSKL//BQU+dTH9JEXddtb6WkBBlzo (RSA)
enc640: 129.40.23.110 fe80::1:2ff:fe00:1a
rdbko3b1 login: [175316.757707] SELinux: mount invalid.  Same superblock, 
differ
ent security settings for (dev mqueue, type mqueue)
```

Run the following commands to disconnect from console:

```
#CP SET RUN ON
#CP DISC HOLD
```

The installation of the bootstrap node is complete.

Repeat the steps for the others IDs, starting with controller node 1 and ending with worker node 2. You should start with the controller nodes and then go to the worker nodes. In our installation, we used the VMIDs that are listed in Table 6-1.

Table 6-1 VMIDs

Node	**VMID**
Controller Node 1	RDBKO3M1
Controller Node 2	RDBKO3M2
Controller Node 3	RDBKO3M3
Worker Node 1	RDBKO3W1
Worker Node 2	RDBKO3W2

You must repeat the procedure starting at step on page 89 first for the controller nodes ID and then for the worker nodes ID.

Note: If you do not see the worker nodes that are listed in the output of the **`oc get nodes`** command after the installation process completes, see “Certificate signing requests” on page 65.

It is recommended that you monitor the bootstrap installation by completing the following steps:

1. Perform the steps that are described in 3.6, “Post deployment activities” on page 46.
2. Perform the steps that are described in 3.7, “Configuring a persistent volume for use by the OpenShift cluster” on page 49 .
3. Perform the steps that are described in 3.8, “Installing an NFS server” on page 51.
4. Perform the steps that are described on 3.9, “Setting up an image registry” on page 59.
5. Access the console by following the instructions that are described in 3.10, “Accessing the console” on page 60.

Related publications

The publications that are listed in this section are considered particularly suitable for a more detailed discussion of the topics that are covered in this paper.

Online resources

The following websites are also relevant as further information sources about installing Red Hat OpenShift 4.4 on IBM Z:

- Installing a cluster on IBM Z and LinuxONE:

  https://docs.openshift.com/container-platform/4.4/installing/installing_ibm_z/installing-ibm-z.html
- How to install an NFS server;

  https://medium.com/faun/openshift-dynamic-nfs-persistent-volume-using-nfs-client-provisioner-fcbb8c9344e
- openshift-nfs-provisioner:

  https://github.ibm.com/Garrett-Lee-Woodworth/openshift-nfs-provisioner
- IBM Cloud Documentation:

  https://cloud.ibm.com/docs/
- Updating a cluster by using the CLI:

  https://docs.openshift.com/container-platform/4.4/updating/updating-cluster-cli.html?extIdCarryOver=true&sc_cid=701f2000001Css0AAC#update-upgrading-cli_updating-cluster-cli
- How to Upgrade OpenShift 4 between different minor versions via “oc” CLI:

  https://access.redhat.com/solutions/4606811
- Upgrade OpenShift 4 clusters with the CLI:

  https://rcarrata.com/openshift/upgrade-ocp4-from-the-cli/
- For more information about the OpenShift update process, see these websites:
  - https://docs.openshift.com/container-platform/4.4/updating/updating-cluster-cli.html?extIdCarryOver=true&sc_cid=701f2000001Css0AAC#update-upgrading-cli_updating-cluster-cli
  - https://access.redhat.com/solutions/4606811

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

REDP-5605-00

ISBN 0738459054