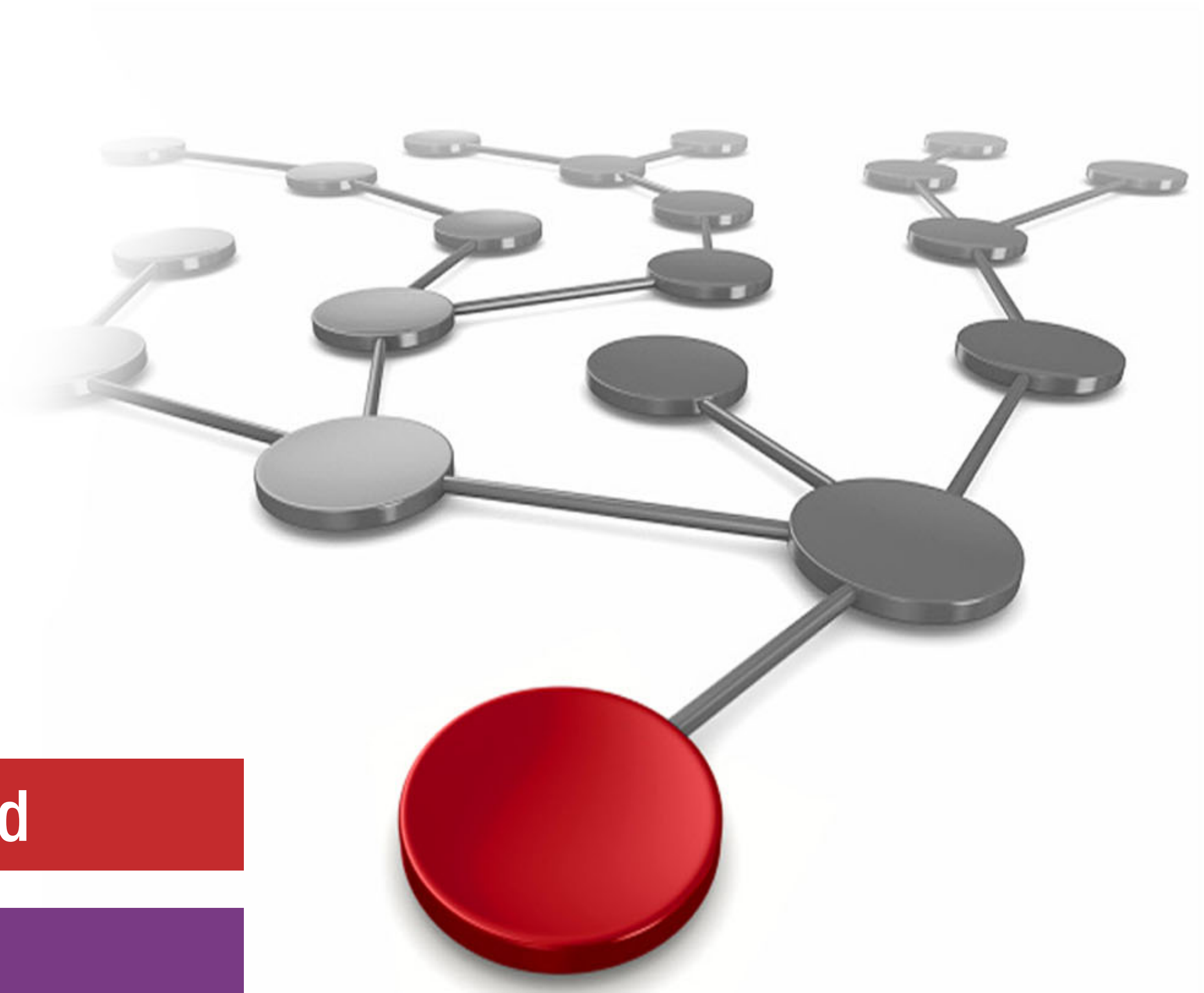
Redbooks

ibm.com/redbooks

Deploying a Database Instance in an IBM Cloud Private Cluster on IBM Z

Christian May



Cloud

Storage

IBM

Redpaper

IBM Redbooks

Deploying a Database Instance in an IBM Cloud Private Cluster on IBM Z

July 2019

REDP-5547-00

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

First Edition (July 2019)

This edition applies to Version 3, Release 1, Modification 2 of IBM Cloud™ Private and Version 5, Release 0, Modification 3 of IBM Spectrum Scale.

This document was created or updated on July 28, 2019.

Contents

Notices

Trademarks

Preface

This IBM® Redpaper™ publication shows you how to deploy a database instance within a container using an IBM Cloud™ Private cluster on IBM Z®. A preinstalled IBM Spectrum™ Scale 5.0.3 cluster file system provides back-end storage for the persistent volumes bound to the database.

A *container* is a standard unit of software that packages code and all its dependencies, so the application runs quickly and reliably from one computing environment to another. By default, containers are ephemeral. However, stateful applications, such as databases, require some type of persistent storage that can survive service restarts or container crashes.

IBM provides several products helping organizations build an environment on an IBM Z infrastructure to develop and manage containerized applications, including dynamic provisioning of persistent volumes.

As an example for a stateful application, this paper describes how to deploy the relational database MariaDB using a Helm chart. The IBM Spectrum Scale V5.0.3 cluster file system is providing back-end storage for the persistent volumes.

This document provides step-by-step guidance regarding how to install and configure the following components:

- IBM Cloud Private 3.1.2 (including Kubernetes)
- Docker 18.03.1-ce
- IBM Storage Enabler for Containers 2.0.0 and 2.1.0

This Redpaper demonstrates how we set up the example for a stateful application in our lab. The paper gives you insights about planning for your implementation. IBM Z server hardware, the IBM Z hypervisor z/VM®, and the IBM Spectrum Scale cluster file system are prerequisites to set up the example environment. The Redpaper is written with the assumption that you have familiarity with and basic knowledge of the software products used in setting up the environment.

The intended audience includes the following roles:

- Storage administrators
- IT/Cloud administrators
- Technologists
- IT specialists

Authors

This paper was produced by a team of specialists from around the world working with IBM Deutschland GmbH, Kelsterbach.

Christian May works in the IBM Spectrum Scale for Linux on IBM Z team in Kelsterbach, Germany. He holds a degree in Applied Physics. Christian has been with IBM for 25 years, and started as a mechanical engineer in the magnet head manufacturing line. He later moved to the TotalStorage Interoperabilty Center in Mainz, Germany, where he was running proofs-of-concept and test activities in the SAN Storage Disk area. This was followed by test activities with IBM Disk Storage products attached to IBM Z servers. In addition to his current test activities for IBM Spectrum Scale on IBM Z, he is a lab advocate for IBM Spectrum Scale on IBM Z customers.

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

Larry Coyne
IBM Redbooks, Tucson Center

Michael Diederich, Aaron Palazzolo, Ruben-Norbert Straus, Susanne Wintenberger
IBM Systems

Now you can become a published author, too

Comments welcome

Your comments are important to us.

We want our papers to be as helpful as possible. Send us your comments about this paper or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form:

  ibm.com/redbooks

- Send your comments in an email:

  redbooks@us.ibm.com

- Mail your comments:

  IBM Corporation, IBM Redbooks
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks

- Follow us on Twitter:

  http://twitter.com/ibmredbooks

- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- 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

From the dawn of computing, developers have employed several different methods of building and distributing software applications.[1] Early methods of development used dedicated hardware and OS that were installed on the actual hardware, without any portability. Any changes would also require backups and restores to cloned hardware, for regression testing. Those methods have progressively evolved to virtual machines and to the current use of micro services and containers.

Using containers may simplify the creation of highly distributed systems by enabling multiple applications, worker tasks, and other processes to run autonomously on a single physical machine or across multiple virtual machines. This allows the deployment of nodes to be performed as the resources become available or when more nodes are needed, enabling a platform as a service (PaaS)-style of deployment and scaling for systems, such as Apache Cassandra, MongoDB, and Riak.

Containers are isolated from each other and bundle their own application, tools, libraries and configuration files; they can communicate with each other through well-defined channels. All containers are run by a single operating-system kernel and are thus more lightweight than virtual machines. Containers are created from images that specify their precise contents. Images are often created by combining and modifying standard images downloaded from public repositories.

This chapter describes the software needed to deploy a database instance within a container on IBM Z®.

[1] Parts of this section are based on the IBM Redbooks® publication *IBM Cloud Private System Administrator's Guide*, SG24-8440 and the IBM Redpaper *IBM Spectrum Connect and IBM Storage Enabler for Containers: Practical Example with IBM FlashSystem A9000*, REDP-5470.

1.1 Docker

Docker[2] is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure, so that you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications. By taking advantage of Docker's methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.

Docker provides the ability to package and run an application in a loosely isolated environment called a container. The isolation and security enable you to run many containers simultaneously on a given host. Containers are lightweight because they don't need the extra load of a hypervisor but run directly within the host machine's kernel.

Therefore, you can run more containers on a given hardware combination than if you were using virtual machines. You can even run Docker containers within host machines that are actually virtual machines. See Figure 1-1.

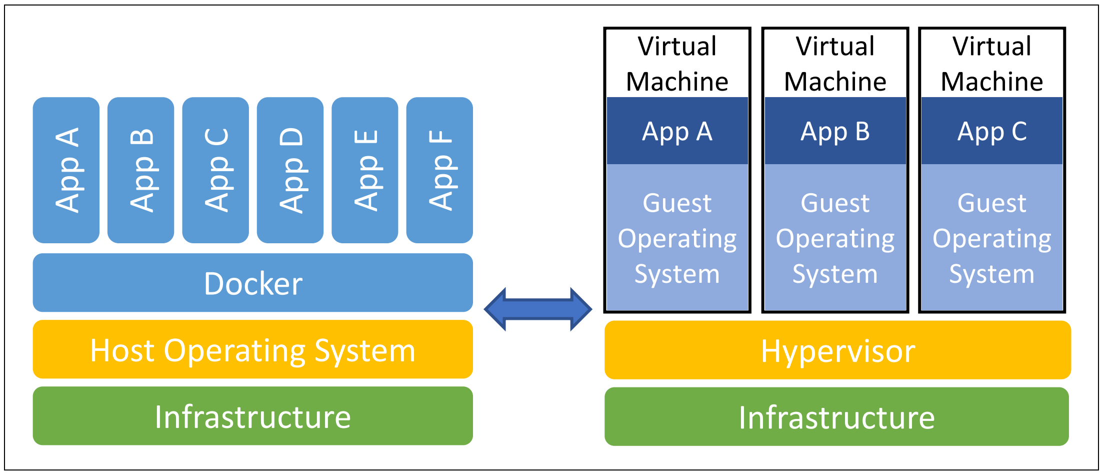


Figure 1-1 Docker versus virtual machines

Docker provides tooling and a platform to manage the lifecycle of your containers:

- Develop your application and its supporting components using containers.
- The container becomes the unit for distributing and testing your application.
- When you are ready, deploy your application into your production environment, as a container or an orchestrated service. This works the same whether your production environment is a local data center, a cloud provider, or a hybrid of the two.

For more information about Docker, see Get Started, Part 1: Orientation and setup.

1.2 Kubernetes

With Docker, development teams cease to become isolated, and can use resources and distribution, all within the corporate guidance or budget. Docker also enhances flexibility in application platform hosting, be it on-premises, in remote data centers, or in the cloud.

[2] The Docker section is based on the Docker Overview.

Although the container runtime APIs are well suited to managing individual containers, they are inadequate when it comes to managing applications that might comprise hundreds of containers spread across multiple hosts. Containers need to be managed and connected to the outside world for tasks (such as scheduling, load balancing, and distribution), and this is where a container orchestration tool (such as Kubernetes) comes into its own.

Kubernetes is an open source container orchestration platform, enabling large numbers of containers to work together in harmony, reducing operational burden. Refer to Figure 1-2. It helps with many activities, including the following tasks:

- Running containers across many different machines
- Scaling up or down by adding or removing containers when demand changes
- Keeping storage consistent with multiple instances of an application
- Distributing load between the containers
- Launching new containers on different machines if something fails

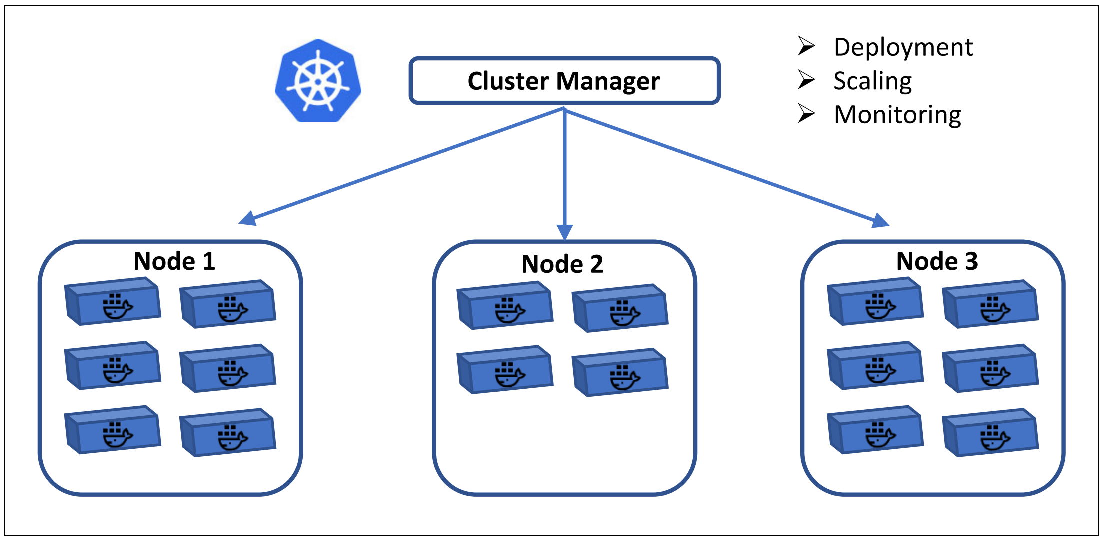


Figure 1-2 Kubernetes diagram as a popular orchestrator and how it works with containers

Read more about Kubernetes on the Kubernetes Basics website.

1.3 IBM Cloud Private

IBM Cloud Private[3] is an application platform for developing and managing containerized applications across hybrid cloud environments, on-premises and public clouds. It is an integrated environment that includes the container orchestrator Kubernetes, a private image registry, a management console, and monitoring frameworks.

IBM Cloud Private is a next-generation, pre-packaged, enterprise-class solution and platform for developing and managing containerized applications. This integrated environment can be deployed behind firewalls, and managed or controlled by whomever the enterprise determines. It is built on Kubernetes.

[3] This section is based on IBM Cloud Private overview.

Here are some of IBM Cloud Private version 3.1.2 features and functions:

- A unified installer

  Rapidly set up a Kubernetes-based cluster that contains master, worker, proxy, and optional management and Vulnerability Advisor nodes by using an Ansible-based installer. This Ansible-based installer is fast and simple to use. Run a few simple commands from a single boot node, and your cluster is up and running in a few minutes.

- IBM Cloud Private management console

  Manage, monitor, and troubleshoot your applications and cluster from a single, centralized, and secure management console.

- Kubernetes

  Kubernetes, an open-source system, enables you to run containerized applications in a clustered environment. It is a platform designed to completely manage the lifecycle of containerized applications and services using methods that provide predictability, scalability, and high availability.

- Private Docker image registry

  The private Docker registry integrates with the Docker registry V2 API to provide a local registry service that functions in the same way as the cloud-based registry service, Docker Hub. This local registry has all the same features as Docker Hub, but you can also restrict which users can view or pull images from this registry.

- Helm

  Helm, the Kubernetes native package management system, is used for application management inside an IBM Cloud Private cluster. The Helm GitHub community curates and continuously expands a set of tested and preconfigured Kubernetes applications.

  Helm charts describe even the most complex applications; provide repeatable application installation, and serve as a single point of authority. Helm charts are easy to update with in-place upgrades and custom hooks. Charts are also easy to version, share, and host on public or private servers. You can use helm rollback to roll back to an older version of a release with ease.

- Catalog

  IBM Cloud Private provides an easy-to-use, extend, and compose Catalog of IBM and third-party content. The Catalog provides a centralized location from which you can browse for and install packages in your cluster.

For a complete list of features for Linux on IBM Z and LinuxONE, see the Supported features website.

The IBM Cloud Private cluster on IBM Z and LinuxONE has four main classes of nodes: boot, master, worker, and proxy. However, Vulnerability Advisor (VA) node is not supported in release 3.1.2.

An example for a basic IBM Cloud Private cluster is shown in Figure 1-3.

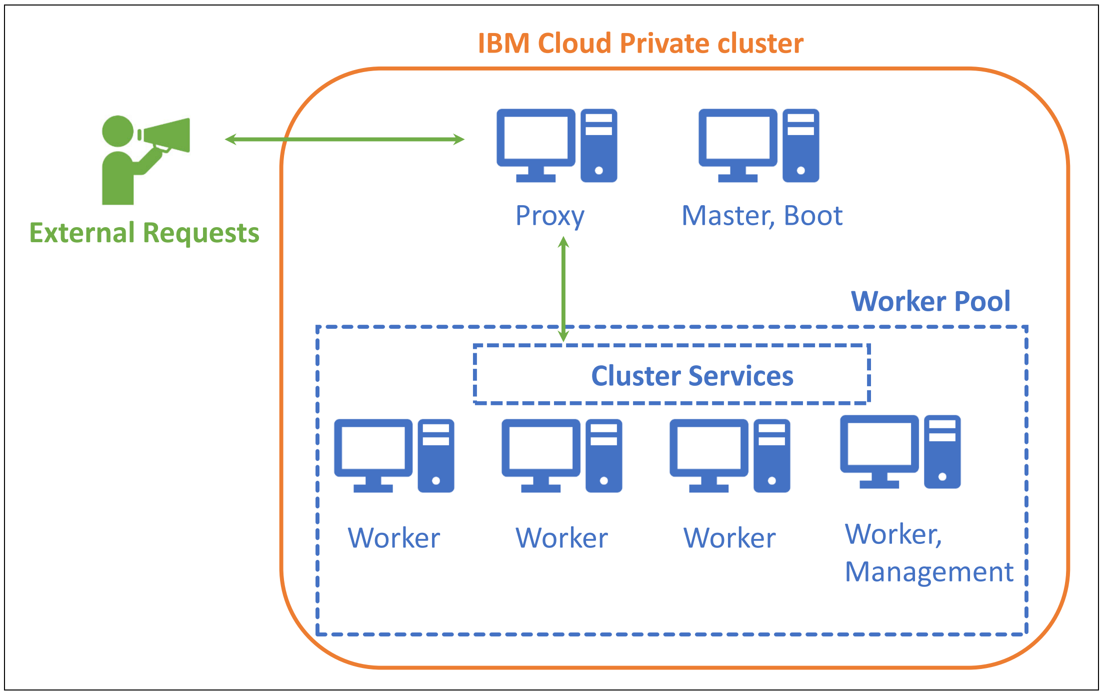


Figure 1-3 Basic IBM Cloud Private cluster with shared master and boot node

The following list describes the types of supported nodes:

- A boot or bootstrap node is used for running installation, configuration, node scaling, and cluster updates. Only one boot node is required for any cluster. You can use a single node for both master and boot.
- A master node provides management services and controls the worker nodes in a cluster. Master nodes host processes that are responsible for resource allocation, state maintenance, scheduling, and monitoring.
- A worker node is a node that provides a containerized environment for running tasks. As demands increase, more worker nodes can easily be added to your cluster to improve performance and efficiency. A cluster can contain any number of worker nodes, but a minimum of one worker node is required.
- A proxy node is a node that transmits external request to the services created inside your cluster.
- A management node is an optional node that only hosts management services, such as monitoring, metering, and logging. By configuring dedicated management nodes, you can prevent the master node from becoming overloaded. You can enable the management node only during IBM Cloud Private installation.

The cluster setup shown in Figure 1-3 is just an example. A cluster can also be set up with a single node for both master and boot. Even a single boot node for multiple clusters is possible. In such a case, the boot and master cannot be on a single node. Each cluster must have its master node. If you plan a high availability (HA) environment multiple master nodes can be set up.

For the best experience in using IBM Cloud Private, you must understand how Kubernetes, Docker, and Helm work. These open source components are fundamental to the IBM Cloud Private platform. You use Kubernetes deployments to place instances of applications, which are built into Helm charts that reference Docker images. The Helm charts contain the details about your application, and the Docker images contain all the software packages that your applications need to run.

For a list of Known issues and limitations see IBM Knowledge Center for Installing IBM Cloud Private.

1.4 IBM Spectrum Scale

IBM Spectrum Scale™ is a cluster file system that provides concurrent access to a single file system or set of file systems from multiple nodes. The nodes can be SAN-attached, network-attached, a mixture of SAN-attached and network-attached, or in a shared-nothing cluster configuration. IBM Storage Enabler for Containers allows IBM Spectrum Scale to be used as a source for persistent volumes intended for stateful application running in Kubernetes clusters.

1.5 IBM Storage Enabler for Containers

Because containers[4] are meant to be portable, containers initially restricted applications from persistently storing data inside the container itself. It is only later that containers that could keep persistent storage were developed. These are known as *stateful containers*. Support for stateful containers with IBM storage originated in an IBM research open source project called *Ubiquity*.

The Ubiquity project at IBM enables persistent storage for the Kubernetes and Docker container frameworks. It is a pluggable framework available for different storage systems. This framework essentially relies on the Kubernetes concept of *Persistent Volumes* (PV). A PV is a piece of storage in the cluster that has been provisioned by an administrator. It is a resource in the cluster just like a node is a cluster resource.

A PersistentVolumeClaim (PVC) is a request for storage by a user. It is similar to a pod. Pods consume node resources and PVCs consume PV resources. Pods can request specific levels of resources (CPU and memory). Claims can request specific size and access modes as specified by a Storage Class (see Figure 1-4 on page 7).

The Ubiquity project led to the development and availability of the *IBM Storage Enabler for Containers* solution. Using the IBM Storage Enabler for Containers solution allows several IBM block storage systems as well as the IBM Spectrum Scale cluster file system to be used as persistent storage for Kubernetes orchestrated container environments.

Currently, only one back end (block storage or IBM Spectrum Scale) can be configured on a single Kubernetes cluster via IBM Storage Enabler for Containers. IBM Storage Enabler for Containers uses Kubernetes Dynamic Volume Provisioner and FlexVolume driver for volume provisioning.

[4] Parts of this section are based on the Redpaper *IBM Spectrum Connect and IBM Storage Enabler for Containers: Practical Example with IBM FlashSystem A9000*, REDP-5470.

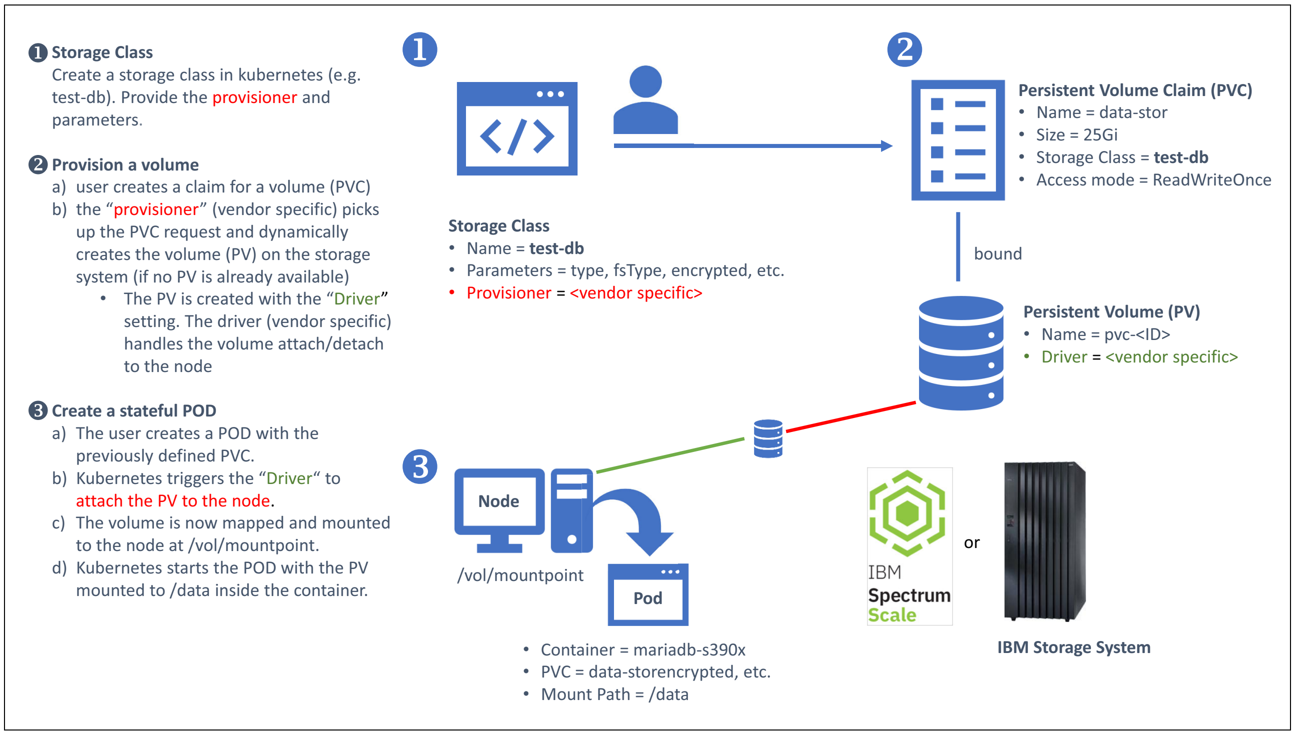

❶ Storage Class
Create a storage class in kubernetes (e.g. test-db). Provide the provisioner and parameters.
❷ Provision a volume
a) user creates a claim for a volume (PVC)
b) the "provisioner" (vendor specific) picks up the PVC request and dynamically creates the volume (PV) on the storage system (if no PV is already available)
• The PV is created with the "Driver" setting. The driver (vendor specific) handles the volume attach/detach to the node
❸ Create a stateful POD
a) The user creates a POD with the previously defined PVC.
b) Kubernetes triggers the "Driver" to attach the PV to the node.
c) The volume is now mapped and mounted to the node at /vol/mountpoint.
d) Kubernetes starts the POD with the PV mounted to /data inside the container.
1
2
3
Persistent Volume Claim (PVC)
• Name = data-stor
• Size = 25Gi
• Storage Class = test-db
• Access mode = ReadWriteOnce
Storage Class
• Name = test-db
• Parameters = type, fsType, encrypted, etc.
• Provisioner = <vendor specific>
bound
Persistent Volume (PV)
• Name = pvc-<ID>
• Driver = <vendor specific>
Node
/vol/mountpoint
Pod
IBM Spectrum Scale
or
IBM Storage System
• Container = mariadb-s390x
• PVC = data-storencrypted, etc.
• Mount Path = /data

Figure 1-4 Interaction of storage class, PVC, and PV, to provision a container and pod

2

Use Case: Deploying MariaDB on IBM Cloud Private with IBM Spectrum Scale storage on IBM Z

This example describes how to deploy MariaDB, an open source object-relational database system, into IBM Cloud Private, by using Helm charts and *persistent* volumes. Through the IBM Storage Enabler for Containers, Kubernetes persistent volumes (PVs) are provisioned from the IBM Spectrum Scale file system.

2.1 Example configuration

Figure 2-1 shows the configuration used for this document. In total, seven Red Hat Linux 7.6 nodes were installed inside an IBM z/VM® 6.4 hypervisor on an IBM Z14. The IBM Cloud Private 3.1.2 cluster consists of five nodes. One node was designated as boot and master node for running installation, configuration, node scaling, and cluster updates.

Another node was intended for providing management services and controls, and transmission of external requests to the services created inside the cluster (management and proxy node). Three worker nodes provided a containerized environment for running tasks.

Those three worker nodes were also part of the IBM Spectrum Scale 5.0.3 cluster. The IBM Spectrum Scale file system used for the persistent volumes must always be mounted on all the worker nodes. For performance reasons, the IBM Spectrum Scale NSD server node and the GUI server node were not part of the IBM Cloud Private cluster. A proper communication link between the IBM Spectrum Scale Management API Server (GUI) and the ICP cluster is mandatory.

Several direct access storage device (DASD) volumes form the basis for the network shared disks (NSD) and the IBM Spectrum Scale file system. The NSD clients and thus the ICP worker nodes access the back-end storage through the NSD server over the IP network.

Through the IBM Storage Enabler for Containers, the containerized database application MariaDB can access IBM Spectrum Scale to be used as a source for the persistent volumes intended for the stateful application.

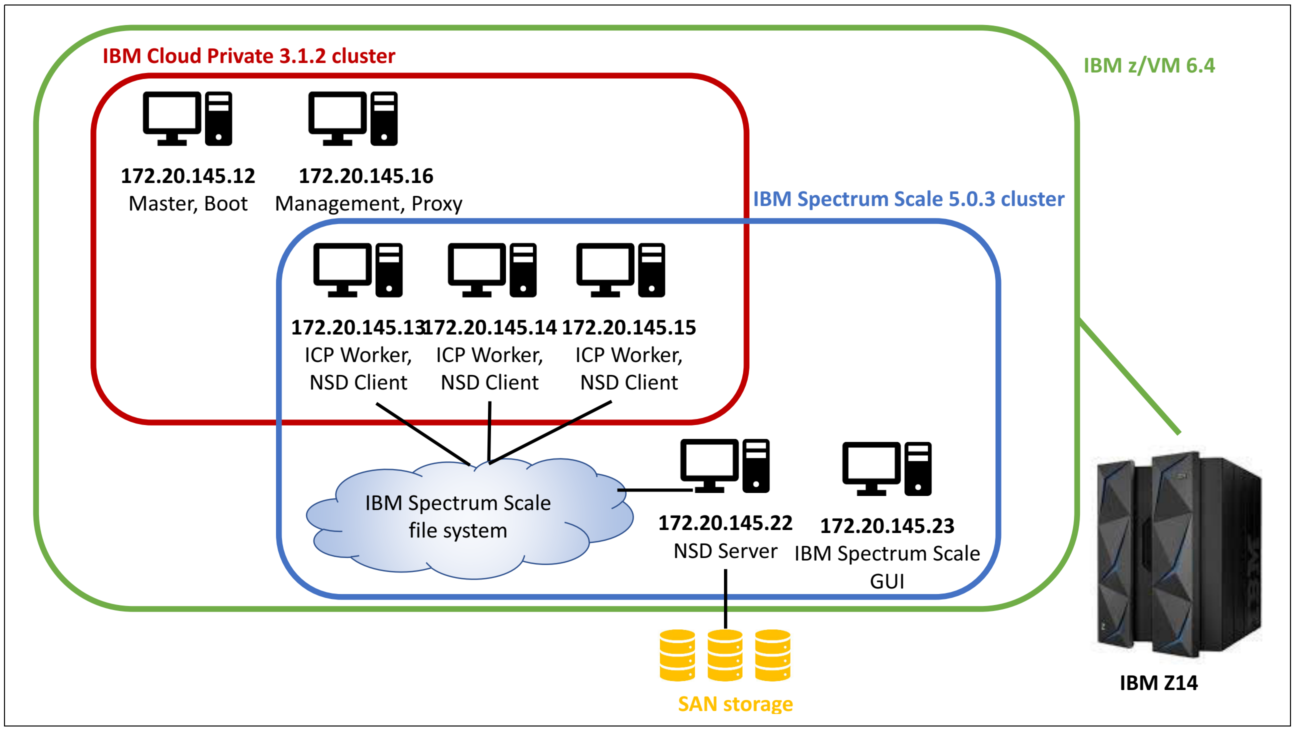


Figure 2-1 Setup for MariaDB use case; IBM Cloud Private and IBM Spectrum Scale cluster running in an IBM z/VM Hypervisor on an IBM Z14

2.1.1 General requirements

Note the following general requirements for each component:

- IBM Cloud Private 3.1.2
  - Hardware
  - Disk Space
  - Supported operating systems and platforms
  - Supported node types
  - Required ports

  Note: Avoid running an ICP proxy node on an IBM Spectrum Scale GUI server. Both node types use port 443 for communication, and can interfere with each other.

- IBM Spectrum Scale 5.0.3
  - Current requirements/limitations
  - Linux distributions supported by IBM Spectrum Scale
- Storage Enabler for Containers 2.0.0
  - Compatibility and requirements
- Storage Enabler for Containers 2.1.0
  - Compatibility and requirements

2.2 Docker Installation (on boot node only)

IBM Cloud Private requires Docker. You must manually install Docker on your boot node. The boot node is the node that is used for installation of your cluster. Usually, it is also your master node. For more information about the boot node, see Boot node.

You can either manually install Docker on the rest of your cluster nodes, or the installer can automatically install Docker on your correctly configured master, worker, proxy, and optional management nodes (which is the default setting). If you want the installer to set up Docker on your cluster nodes, you can set this configuration during the installation of your cluster. See Configuring cluster nodes for automatic Docker installation.

It is recommended to install the Docker package provided by IBM Cloud Private (which will be used in this example). For more information regarding Docker versions, see Supported Docker versions.

1. Configure `SELinux=disabled` in the `/etc/selinux/config` file, as shown in Example 2-1.

Example 2-1 SELinux config file

```
# cat /etc/selinux/config

# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=disabled
```

```
# SELINUXTYPE= can take one of three two values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are 
protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

2. Reboot your system. After reboot, confirm that the **sestatus** command returns `disabled`, as shown in Example 2-2.

Example 2-2 Verify SELinux status set to disabled

```
# sestatus
SELinux status:                 disabled
```

3. Ensure that all required ports are open, but not in use. No firewall rules must block these ports. For more information about the IBM Cloud Private required ports, see Required ports. For the sake of simplicity, shut down the firewall, as shown in Example 2-3.

Example 2-3 Stopping the firewall

```
# systemctl stop firewalld
```

4. Download the IBM Cloud Private Docker package for your boot/master node from the IBM Passport Advantage website.

5. Manually install Docker, as shown in Example 2-4.

Example 2-4 Installing Docker

```
# cd <docker_download_directory>
# chmod +x ./icp-docker-18.03.1_s390x.bin
# ./icp-docker-18.03.1_s390x.bin
```

6. To ensure that the Docker engine started run the following commands, as shown in Example 2-5.

Example 2-5 Enabling, starting, and getting the status for Docker

```
# systemctl enable docker
# systemctl start docker
# systemctl status docker
```

See Configure your Docker engine for more options to configure the Docker engine.

2.3 Set up the IBM Cloud Private installation environment

This section describes the steps to set up the IBM Cloud Private Installation environment:

1. Configure the `/etc/hosts` file on each node in your cluster:

   a. Add the IP addresses and host names for all of the nodes to the `/etc/hosts` file on each node.

   Important: Ensure that the host name is listed by the IP address for the local host. You cannot list the host name by the loopback address, `127.0.0.1`.

- Host names in the /etc/hosts file cannot contain uppercase letters.
- If your cluster contains a single node, you must list its IP address and host name.

b. Comment out the line of the file that begins with 127.0.1.1 and ::1 localhost.

Example 2-6 shows an /etc/hosts file for a cluster that contains a master node, a proxy node, and two worker nodes resembles the following code.

Example 2-6 An /etc/hosts file

```
127.0.0.1       localhost
 # 127.0.1.1     <host_name>
 # The following lines are desirable for IPv6 capable hosts
 #::1     localhost ip6-localhost ip6-loopback
 ff02::1 ip6-allnodes
 ff02::2 ip6-allrouters
 <master_node_IP_address> <master_node_host_name>
 <worker_node_1_IP_address> <worker_node_1_host_name>
 <worker_node_2_IP_address> <worker_node_2_IP_host_name>
 <proxy_node_IP_address> <proxy_node_host_name>
```

Note: While the IBM Cloud Private installation process is running, the /etc/hosts file on each of the cluster nodes is automatically updated to include an entry for clusterName.icp. This correlates to cluster_vip, unless cluster_vip is not set, in which case it correlates to cluster_lb_address. IBM Cloud Private uses the clustername.icp:8500/xx/xxx in the /etc/hosts file to pull the IBM Cloud Private registry Docker images.

2. Download the installation files from the IBM Passport Advantage® website. For the IBM Cloud Private for Linux on IBM Z and LinuxONE (s390x) cluster, download the ibm-cloud-private-s390x-3.1.2.tar.gz file.
3. Extract the images and load them into Docker, as shown in Example 2-7 (this can take a while).

Example 2-7 Extracting images to load them into Docker

```
# tar xf ibm-cloud-private-s390x-3.1.2.tar.gz -O | sudo docker load
```

4. Create an installation directory to store the IBM Cloud Private configuration files and change to that directory. For example, to store the configuration files in /opt/ibm-cloud-private-3.1.2, run the commands shown in Example 2-8.

Example 2-8 Creating an installation directory

```
# mkdir /opt/ibm-cloud-private-3.1.2
# cd /opt/ibm-cloud-private-3.1.2
```

5. Extract the configuration files from the installer image, as shown in Example 2-9.

Example 2-9 Extracting the configuration files

```
# docker run -v $(pwd):/data -e LICENSE=accept \
ibmcom/icp-inception-s390x:3.1.2-ee \
cp -r cluster /data
```

6. A cluster directory is created inside your installation directory. For example, if your installation directory is `/opt/ibm-cloud-private-3.1.2`, the `/opt/ibm-cloud-private-3.1.2/cluster` folder is created. For an overview of the cluster directory structure, see Cluster directory structure.
7. If is not already completed, create a secure connection from the boot node to all other nodes in your cluster. Therefore, set up SSH in your cluster. See Sharing SSH keys among cluster nodes.
8. As shown in Example 2-10, add the IP address of each node in the cluster to the `/<installation_directory>/cluster/hosts` file. See Setting the node roles in the hosts file. You can also define customized host groups. See Defining custom host groups.

Example 2-10 Host file example

```
# cat /opt/ibm-cloud-private-3.1.2/cluster/hosts
[master]
172.20.145.12
[worker]
172.20.145.13
172.20.145.14
172.20.145.15
[proxy]
172.20.145.16
[management]
172.20.145.16
```

9. If you use SSH keys to secure your cluster, in the `/<installation_directory>/cluster` folder, replace the `ssh_key` file with the private key file that is used to communicate with the other cluster nodes. See Sharing SSH keys among cluster nodes. Run the following command (Example 2-11).

Example 2-11 Sharing SSH keys

```
# cd /opt/ibm-cloud-private-3.1.2
# cp ~/.ssh/id_rsa ./cluster/ssh_key
```

10. Move the image files for your cluster to the `/<installation_directory>/cluster/images` folder, as shown in Example 2-12.

Example 2-12 Moving the image files

```
# cd /opt/ibm-cloud-private-3.1.2
# mkdir -p cluster/images
# mv /<path_to_installation_file>/ibm-cloud-private-s390x-3.1.2.tar.gz
cluster/images/
```

In this case, `<path_to_installation_file>` is the path to the images file.

2.4 Customize the Cluster

You can set a variety of optional cluster customizations that are available in the `/<installation_directory>/cluster/config.yaml` file, such as Kubernetes settings, network settings, proxy settings, and so on. See Customizing the cluster with the config.yaml file. For additional customizations, you can also review Customizing your installation.

Set up a default password in the `config.yaml` file (the **`default_admin_password`** parameter). Unless otherwise defined in the **`password_rules`** parameter, the **`default_admin_password`** must meet the default password enforcement rule `'^([a-zA-Z0-9\-]{32,})$'`: The password must be a minimum of 32 characters long, and can contain only lower and uppercase alphanumeric characters and dashes. You can also define a custom set of password rules (see Example 2-13).

Example 2-13 Setting a password rule and the default admin password

```
## Advanced Settings
default_admin_user: admin
# default_admin_password:
# ansible_user: <username>
# ansible_become: true
# ansible_become_password: <password>
password_rules:
- '^.{9,}'
- '^([a-zA-Z0-9\-]{9,})$'
default_admin_password: <your password that meets rules above>
#
```

2.5 Set up Docker for your cluster nodes

Cluster nodes are the master, worker, proxy, and management nodes. You need a version of Docker that is supported by IBM Cloud Private installed on your cluster nodes. See Supported Docker versions.

As mentioned in 2.2, “Docker Installation (on boot node only)” on page 11, it is also recommended to install the Docker package that comes with IBM Cloud Private on all other ICP cluster nodes. IBM Cloud Private can automatically install Docker on your cluster nodes during the installation (default setting). See Configuring cluster nodes for automatic Docker installation.

If you have a version of Docker that is supported by IBM Cloud Private already installed on your node, you need to change the **`install_docker`** parameter in the `/<`*`installation_directory`*`>/cluster/config.yaml` to `install_docker=false`. It is also possible to change the docker version and the docker environment (see Table 2-1 on page 16).

Table 2-1 Docker settings

Parameter	Description	Format	Default Value
`docker_version`	Specify the version of Docker that you want to install. A package for the required version must be available in the `/<installation_directory>/cluster/runtime-engine` directory.	string	18.03.1
`install_docker`	Allows the installer to automatically install Docker on your cluster nodes	true or false	true
`docker_env`	Sets the environment for Docker. For example, you might configure an `https_proxy` location if Docker runs behind a firewall. The environment location is stored in the `/etc/systemd/system/docker.service.d/docker` folder.	`"HTTP_PROXY=http://proxy-server:port/", "HTTPS_PROXY=http://proxy-server:port", "NO_PROXY=localhost,127.0.0.1,{{ cluster_CA_domain }}"`	None

Note: These configurations can be set only for the supplied IBM Cloud Private Docker packages.

2.6 Deploy the environment

To deploy the environment, complete the following steps:

1. Change to the cluster folder in your installation directory.

   ```
   # cd /<installation_directory>/cluster
   ```

2. By default, the command to deploy your environment is set to deploy 15 nodes at a time. If your cluster has more than 15 nodes, the deployment might take a longer time to finish. If you want to speed up the deployment, you can specify a higher number of nodes to be deployed at a time. Use the argument **`-f`** ***`<number of nodes to deploy>`*** with the command.

3. The `config.yaml` file contains all of the configuration settings that are needed to deploy your cluster. From the `config.yaml` file, you can customize your installation by using various parameters. Open the `/<installation_directory>/cluster/config.yaml` file and add or modify the parameters or values. See Customizing the cluster with the config.yaml file for more information.

   For IBM Cloud Private for Linux on IBM Z and LinuxONE, run the following command:

   ```
   # docker run --net=host -t -e LICENSE=accept -v "$(pwd)":/installer/cluster
   ibmcom/icp-inception-s390x:3.1.2-ee install
   ```

   a. If you encounter errors during deployment, run the deployment command again with **`-v`** to collect other error messages. If you continue to receive errors during the rerun, run the following command to collect the log files:

      ```
      # docker run --net=host -t -e LICENSE=accept -v "$(pwd)":/installer/cluster
      ibmcom/icp-inception-s390x:3.1.2-ee healthcheck
      ```

      The log files are located in the `/<installation_directory>/cluster/logs` directory.

b. If the installation succeeded, the access information for your cluster is displayed. The *<Cluster Master Host>*:*<Cluster Master API Port>* is defined in the Master endpoint. The Master and Proxy endpoints are the external endpoints that are used for access from outside the cluster. You define these endpoints in the `config.yaml` file during installation, as shown in Example 2-14.

Example 2-14 Successfully finished IBM Cloud Private installation

```
[...]
PLAY RECAP *****************************************************************
172.20.145.12     : ok=216  changed=142  unreachable=0    failed=0
172.20.145.13     : ok=108  changed=68   unreachable=0    failed=0
172.20.145.14     : ok=101  changed=61   unreachable=0    failed=0
172.20.145.15     : ok=101  changed=61   unreachable=0    failed=0
172.20.145.16     : ok=102  changed=62   unreachable=0    failed=0
localhost         : ok=263  changed=165  unreachable=0    failed=0
POST DEPLOY MESSAGE***********************************************

The Dashboard URL: https://172.20.145.12:8443, please use credentials in
config.yaml to login.
```

4. Access your cluster. From a web browser, browse to the URL for your cluster as shown in Figure 2-2. The username and password are defined in the `config.yaml` file.

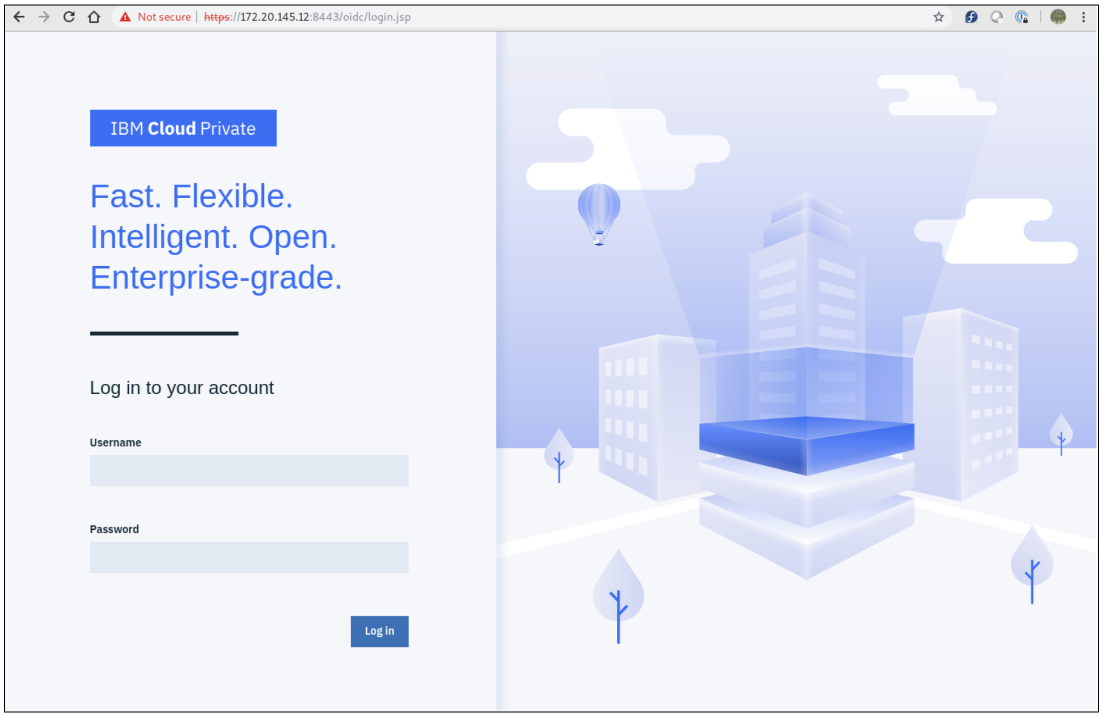


Figure 2-2 IBM Cloud Private login screen

5. IBM Cloud Private includes a CLI for managing your cluster and performing various operations. You can also use the Helm CLI, Kubernetes CLI, and other CLI tools with IBM Cloud Private. For example, log in to your cluster from the IBM Cloud Private CLI and query all nodes in the Kubernetes cluster (Example 2-15 on page 18):

```
cloudctl login -a https://<Cluster Master Host>:<Cluster Master API Port>
--skip-ssl-validation
```

The *<Cluster Master Host>*:*<Cluster Master API Port>* is defined in the Master endpoint.

Example 2-15 Login to IBM Cloud Private cluster and query nodes using Kubernetes CLI

```
# cloudctl login -a https://172.20.145.12:8443 --skip-ssl-validation

Username> admin

Password>
Authenticating...
OK

Targeted account mycluster Account (id-mycluster-account)

Select a namespace:
1. cert-manager
2. default
3. ibmcom
4. istio-system
5. kube-public
6. kube-system
7. platform
8. services
Enter a number> 2
Targeted namespace default

Configuring kubectl ...
Property "clusters.mycluster" unset.
Property "users.mycluster-user" unset.
Property "contexts.mycluster-context" unset.
Cluster "mycluster" set.
User "mycluster-user" set.
Context "mycluster-context" created.
Switched to context "mycluster-context".
OK

Configuring helm: /root/.helm
OK
# kubectl get nodes
NAME            STATUS   ROLES              AGE   VERSION
172.20.145.12   Ready    etcd,master        18h   v1.12.4+icp-ee
172.20.145.13   Ready    worker             18h   v1.12.4+icp-ee
172.20.145.14   Ready    worker             18h   v1.12.4+icp-ee
172.20.145.15   Ready    worker             18h   v1.12.4+icp-ee
172.20.145.16   Ready    management,proxy   18h   v1.12.4+icp-ee
```

For more information, see the CLI tools guide.

You can install other IBM software onto IBM Cloud Private to either extend the functionality of the IBM Cloud Private platform or add to the IBM Cloud Private Catalog. For more information about how to install other software from your bundle, see Installing bundled products.

2.7 Install IBM Spectrum Scale

IBM Spectrum Scale must be installed on all IBM Cloud Private worker nodes that access IBM Spectrum Scale file systems (see Figure 2-1 on page 10 as an example). Additional IBM Spectrum Scale nodes might exist outside of the Kubernetes cluster. For example, the IBM Spectrum Scale GUI and IBM Spectrum Scale I/O servers might exist outside of the Kubernetes cluster. In this case, the storage from these nodes might be made available directly to worker nodes that run IBM Spectrum Scale.

There are two methods available for installing IBM Spectrum Scale on Linux nodes (see Table 2-2).

Table 2-2 Methods available for installing IBM Spectrum Scale

Method	Description
Manual	You can install the IBM Spectrum Scale software packages using the `rpm` command (for SUSE Linux Enterprise Server and Red Hat Enterprise Linux) or the `dpkg` command (for Ubuntu Linux). For more information, see Manually installing the IBM Spectrum Scale software packages on Linux nodes.
Installation toolkit	On Red Hat Enterprise Linux 7.x, Ubuntu 16.04 and 18.04 LTS, and SUSE Linux Enterprise Server 12 and 15, you can install IBM Spectrum Scale and deploy protocols using the installation toolkit. For more information, see Installing IBM Spectrum Scale on Linux nodes with the installation toolkit. For information about limitations if you are using the installation toolkit, see Limitations of the installation toolkit.

Depending on the installation method and cluster configuration, additional tasks need to be performed. See Configuration tasks or Performing additional tasks using the installation toolkit.

IBM Spectrum Scale uses the network for communication between nodes in the Spectrum Scale cluster, as well as for accessing the Storage Enabler for Containers. For example, the Storage Enabler for Containers uses the RESTful API, running on the IBM Spectrum Scale GUI node.

Describing the installation and configuration of an IBM Spectrum Scale cluster is not part of this Redpaper but can be read in the whitepaper *Getting Started with IBM Spectrum Scale for Linux on IBM Z*.

2.8 Install IBM Storage Enabler for Containers

IBM Storage Enabler for Containers (SEC) allows IBM storage systems and IBM Spectrum Scale to be used as persistent volumes for stateful applications running in Kubernetes clusters or on IBM Cloud Private platform. It simplifies storage provisioning for containers by defining policies by SLA or by workload, enabling the containers to be used with stateful microservices, such as database applications.

IBM Storage Enabler for Containers uses Kubernetes dynamic provisioning for creating and deleting volumes on IBM storage systems. In addition, IBM Storage Enabler for Containers utilizes the full set of Kubernetes FlexVolume APIs for volume operations on a host. The operations include initiation, attachment/detachment, mounting/unmounting, and so on.

This installation package includes:

- A Storage Enabler for Containers server (ubiquity) for running Kubernetes Dynamic Provisioner and FlexVolume.
- A Storage Enabler for Containers database (ubiquity-db) for storing the persistent data for the Enabler for Container service.
- A Kubernetes Dynamic Provisioner (ubiquity-k8s-provisioner) for creating storage volumes on-demand, using Kubernetes storage classes based on IBM Spectrum Connect storage services or IBM Spectrum Scale storage classes.
- A Kubernetes FlexVolume DaemonSet (ubiquity-k8s-flex) for attaching/detaching and mounting/unmounting storage volumes into a pod within a Kubernetes node.

The IBM Storage Kubernetes FlexVolume driver is running on all master and worker nodes as a Kubernetes daemon set. In contrast, the IBM Storage Enabler for Containers server, database and provisioner are running as a Kubernetes deployment (see Figure 2-3, and also Example 2-22 on page 29).

A single IBM Storage Enabler for Containers instance can be installed per one Kubernetes cluster.

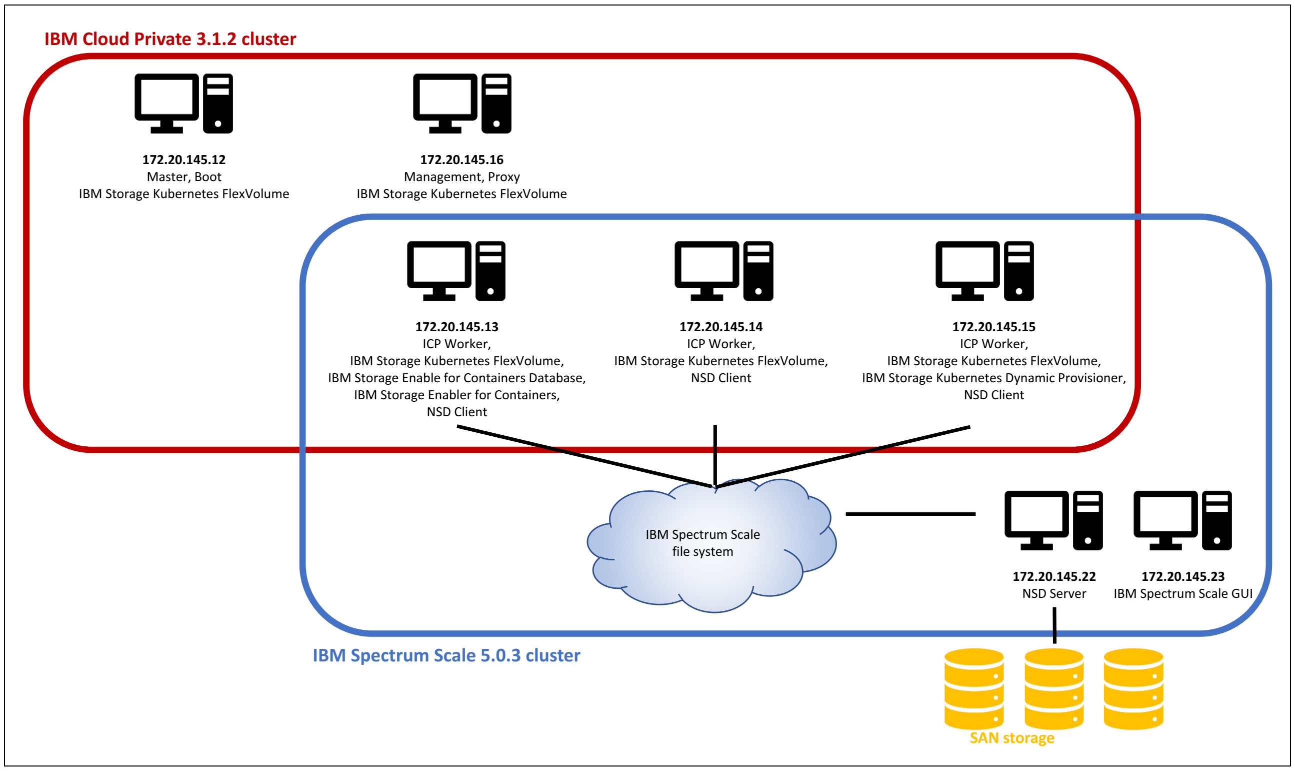

IBM Cloud Private 3.1.2 cluster
172.20.145.12
Master, Boot
IBM Storage Kubernetes FlexVolume
172.20.145.16
Management, Proxy
IBM Storage Kubernetes FlexVolume
172.20.145.13
ICP Worker,
IBM Storage Kubernetes FlexVolume,
IBM Storage Enable for Containers Database,
IBM Storage Enabler for Containers,
NSD Client
172.20.145.14
ICP Worker,
IBM Storage Kubernetes FlexVolume,
NSD Client
172.20.145.15
ICP Worker,
IBM Storage Kubernetes FlexVolume,
IBM Storage Kubernetes Dynamic Provisioner,
NSD Client
IBM Spectrum Scale
file system
172.20.145.22
NSD Server
172.20.145.23
IBM Spectrum Scale GUI
IBM Spectrum Scale 5.0.3 cluster
SAN storage

Figure 2-3 IBM Storage Enabler for Containers environment

The installation of the IBM Storage Enabler for Containers software can be done in two ways:

- Use the SEC 2.0.0 version if you prefer the CLI installation method.
- Use the SEC 2.1.0 version for a Helm chart installation.

In either case, before installing the IBM Storage Enabler for Containers the following conditions must be met, so complete the following steps:

1. Run the **`# mmlsmount all -L`** command to ensure that the IBM Spectrum Scale file systems are mounted before starting Kubernetes on the nodes. The command gives an output similar to Example 2-16.

   Example 2-16 The mmlsmount all -L output

   ```
   # mmlsmount all -L
   File system fs1 is mounted on 5 nodes:
     172.20.145.23   vlxssl12
     172.20.145.22   vlxssl11
     172.20.145.14   vlxssl03
     172.20.145.13   vlxssl02
     172.20.145.15   vlxssl04
   ```

2. Run the **`# mmlsfs fs1 -Q`** command shown in Example 2-17 to ensure that the quota is enabled on the file systems.

   Example 2-17 The mmlsfs <filesystem> output

   ```
   # mmlsfs fs1 -Q
   flag          value                    description
   ------------- ----------------------------------------------------
    -Q           user;group;fileset       Quotas accounting enabled
                 user;group;fileset       Quotas enforced
                 none                     Default quotas enabled
   ```

   If you fail to obtain this output, run the **`# mmchfs gpfs0 -Q yes`** command.

3. Run the following command to ensure that the GUI server is running and can communicate with the Kubernetes nodes (see Example 2-18):

   ```
   curl -k -u gui_admin_user:gui_adm_user_password -X GET --header
   'accept:application/json' 'https://gui_node:443/scalemgmt/v2/filesystems'
   ```

   Example 2-18 The curl command output

   ```
   # curl -k -u admin:pass4root -X GET --header 'accept:application/json'
   'https://vlxssl12.gpfs.boe:443/scalemgmt/v2/filesystems'
   {
     "filesystems" : [ {
       "name" : "fs1"
     } ],
     "status" : {
       "code" : 200,
       "message" : "The request finished successfully."
     }
   }#
   ```

4. Run the **`# mmchconfig enforceFilesetQuotaOnRoot=yes`** command to set the `enforceFilesetQuotaOnRoot` value to `yes`. This ensures that quotas are enforced for the PVC created with root user ID.

5. Run the `# mmlsnodeclass` command to ensure that the Kubernetes are not installed or configured on the nodes running the IBM Spectrum Scale GUI. This prevents port number conflicts and memory usage concerns. The command gives an output similar to that shown in Example 2-19. In this example, the name of the node running the IBM Spectrum Scale GUI is defined by the Node Class Name with a value of `GUI_MGMT_SERVERS`, and the hostname is `vlxssl12.gpfs.boe`.

   Example 2-19 The mmlsnodeclass output

   ```
   Node Class Name       Members
   -------------------------------------------------------------------------------
   GUI_MGMT_SERVERS      vlxssl12.gpfs.boe
   GUI_SERVERS           vlxssl12.gpfs.boe,vlxssl02.gpfs.boe,vlxssl03.gpfs.boe
                         vlxssl04.gpfs.boe,vlxssl11.gpfs.boe
   ```

   See Performing pre-installation tasks for more information.

6. IBM Storage Enabler for Containers uses SSL certificates for maintaining a secure communication link between the IBM Storage Enabler for Containers server, its database, the Dynamic Provisioner, the FlexVolume, and the IBM Spectrum Scale Management API (GUI) server.

   Two modes are supported when communicating with its components: the `require` mode and the `verify-full` mode:

   – The `require` mode

     This SSL mode is activated when no validation is required. The IBM Storage Enabler for Containers server generates self-signed certificates as required during run time. In this mode, no additional configuration is necessary.

   – The `verify-full` mode

     This SSL mode is activated when the user is expected to provide the relevant certificates. When enabled, this SSL mode requires additional configuration steps. Follow the steps described at Managing SSL certificates to manage the SSL certificates when using the `verify full` mode.

For this example, the `require` mode is used.

2.8.1 Install IBM Storage Enabler for Containers 2.0.0

IBM Storage Enabler for Containers 2.0.0 is available as a free software solution for IBM Spectrum Scale customers and is deployed using the CLI method. For the Helm chart method, skip to 2.8.2, “Install IBM Storage Enabler for Containers 2.1.0” on page 23.

1. Download the 2.0.0 version of the installer from IBM Fix Central.
2. Copy the installer to a local folder on a host that can manage the Kubernetes cluster, for example, the Master node.
3. Extract the installer file:

   ```
   # tar -xzvf installer-for-ibm-storage-enabler-for-containers-
   x.y.z-2.0.0.tar.gz
   ```

4. Change to the `installer` directory and enter all of the necessary information in `ubiquity_installer_scale.conf` according to your environment requirements (see Example 2-20). Replace the `VALUE` placeholders in the files with your values.

Example 2-20 Sample values of ubiquity_installer_scale.conf

```
# IP or FQDN of Spectrum Scale Management API server(GUI).
SPECTRUMSCALE_MANAGEMENT_IP_VALUE=172.20.145.23

# Default Filesystem for creating pvc
SPECTRUMSCALE_DEFAULT_FILESYSTEM_NAME_VALUE=fs1

# SSL verification mode. Allowed values: require (no validation is required)
and verify-full (user-provided certificates).
SSL_MODE_VALUE=require

# Parameters in scale-credentials-secret.yml that impact ubiquity
# Username and password defined for IBM Storage Enabler for Containers
interface in Spectrum Scale.
SPECTRUMSCALE_USERNAME_VALUE=<username>
SPECTRUMSCALE_PASSWORD_VALUE=<password>

# Storage Class name
STORAGE_CLASS_NAME_VALUE=gpfs
```

5. Apply the `ubiquity_installer_scale.conf` settings to the relevant `.yml` files of the installer. Run the following command:

   ```
   # ./ubiquity_installer.sh -s update-ymls -c ubiquity_installer_scale.conf
   ```

   Run the following command to start the installation:

   ```
   # ./ubiquity_installer.sh -s install
   ```

   It is recommended to remove the plain text password specified for `SPECTRUMSCALE_PASSWORD_VALUE` and `UBIQUITY_DB_PASSWORD_VALUE` in the `ubiquity_installer_scale.conf` file immediately after the installation.

6. When the installation is complete, verify the post-installation status of the IBM Storage Enabler for containers service. Verify that the database status is set to `bound`, and all other elements are available and running:

   ```
   # ./ubiquity_cli.sh -a status
   ```

7. Run the command to perform a "sanity" test that verifies the expected behavior of a pod creating a PVC:

   ```
   # ./ubiquity_cli.sh-a sanity
   ```

   When running this verification test, the following output is expected:

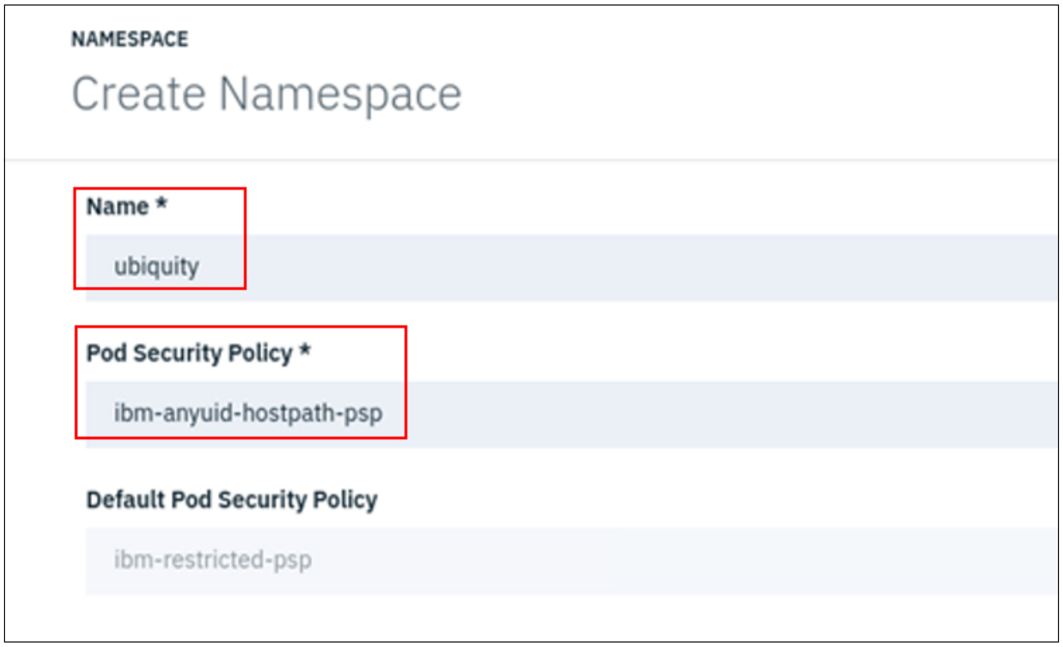
   ```
   "IBM Storage Enabler for Containers" sanity finished successfully.
   ```

2.8.2 Install IBM Storage Enabler for Containers 2.1.0

With version 2.1.0, the IBM Storage Enabler for Containers can be installed by using a Helm chart. For the CLI-based method, return to 2.8.1, "Install IBM Storage Enabler for Containers 2.0.0" on page 22.

The Helm chart is available through the IBM stable charts repository. You may need to sync the repositories. Before installing the Helm chart for Storage Enabler for Containers in conjunction with IBM Spectrum Scale, you need to ensure that the following conditions are met:

1. Storage Enable for Containers requires two secrets. One for its database and one for IBM Spectrum Scale. In order to create both secrets you first need to provide a namespace to be used for creating the secrets:
   a. In the ICP GUI, create a namespace (**Manage** →**Namespaces** →**Create Namespace**). See Figure 2-4.
   b. Provide a name and the pod security policy. The predefined pod security policy name is `ibm-anyuid-hostpath-psp`, and it has been verified for the IBM SEC Helm chart.


NAMESPACE

Create Namespace

Name *

ubiquity

Pod Security Policy *

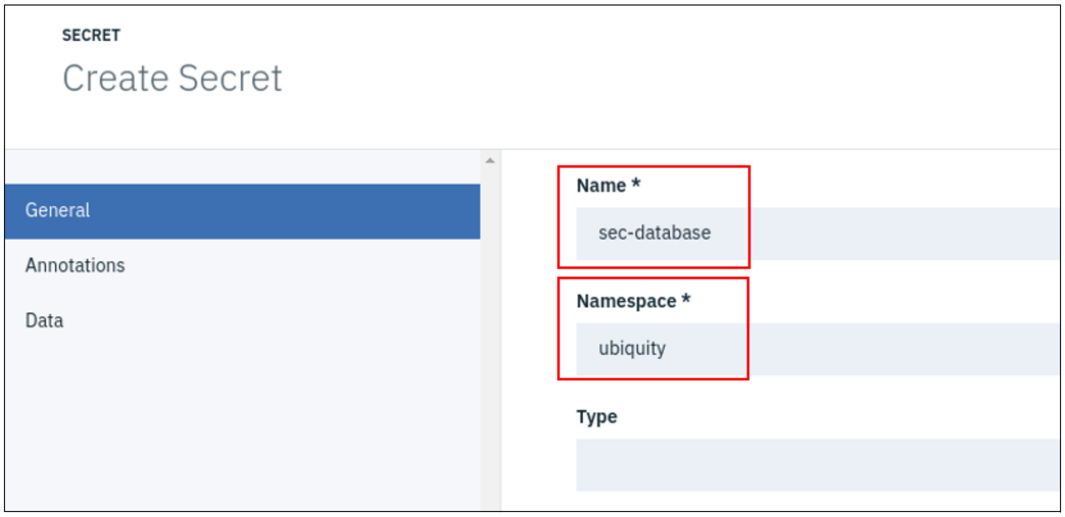
ibm-anyuid-hostpath-psp

Default Pod Security Policy

ibm-restricted-psp

Figure 2-4 Create Namespace

2. Now, create the two secrets using the IBM Cloud Private GUI (**Configuration** →**Secrets** →**Create Secrets**). See Figure 2-5 on page 25.
   a. For the SEC database, provide the following values in the General tab:
      - A name for the secret
      - The previously created namespace


SECRET

Create Secret

General

Annotations

Data

Name *: sec-database

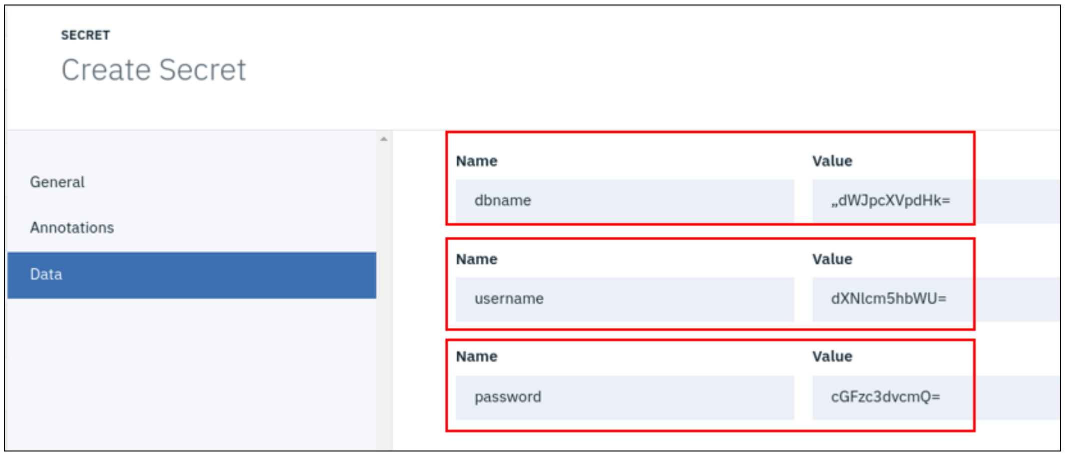
Namespace *: ubiquity

Type

Figure 2-5 Create Secret (General tab)

b. In the **Data** tab, Figure 2-6, provide:

- A dbname with the value "`dWJpcXVpdHk=`"
- A username with a username of your choice in Base64-encrypted format
- A password with a password of your choice in Base64-encrypted format


SECRET

Create Secret

General

Annotations

Data

Name	Value
dbname	„dWJpcXVpdHk=
username	dXNlcm5hbWU=
password	cGFzc3dvcmQ=

Figure 2-6 Create Secret (Data tab)

Encoding the dbname, username, and password can be done with the BASE64 Decode and Encode online tool.

c. Click **Create** to finish.

3. For the IBM Spectrum Scale management API (GUI) server secret, repeat the procedure and create another secret:

   a. In the **General** tab provide:

   - A name for the secret
   - The previously created namespace

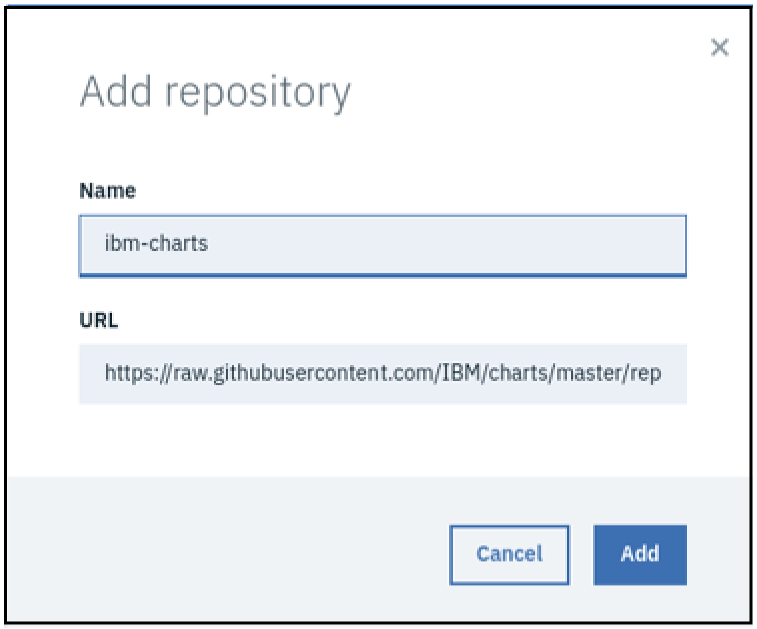
b. In the **Data** tab provide:

- Username. The value should be an IBM Spectrum Scale Management API (GUI) username in Base64-encrypted format.
- Password. The value should be an IBM Spectrum Scale Management API (GUI) password in Base64-encrypted format.

Ensure that all conditions are met (you have finished `Performing pre-installation tasks`) before you proceed with the installation.

4. If not already available, add the IBM Helm charts repository.

   In the IBM Cloud Private GUI, navigate to **Manage** →**Helm Repositories** →**Add repository** and add a name and the URL, as shown in Figure 2-7.



ibm-charts

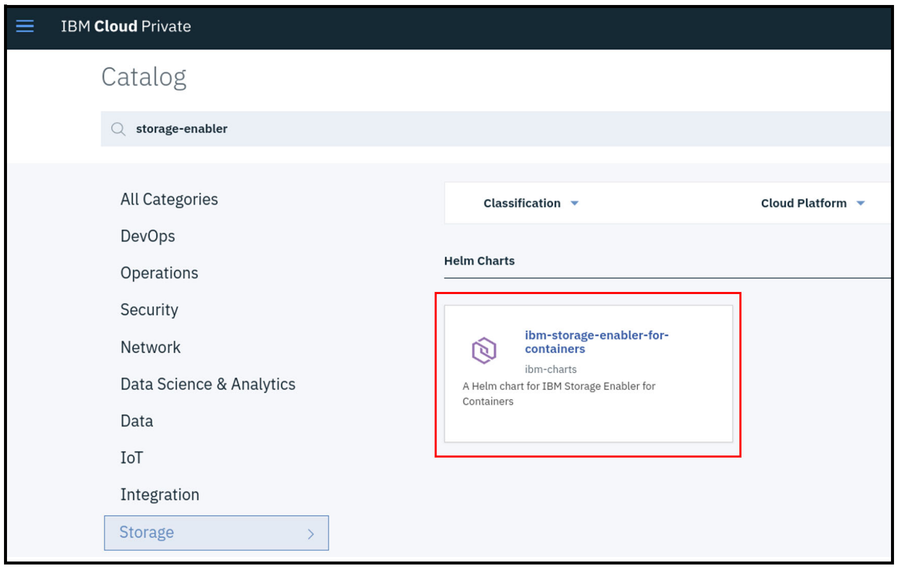
https://raw.githubusercontent.com/IBM/charts/master/rep

Figure 2-7 Add repository

5. Performing the IBM SEC 2.1.0 installation.

   During installation of the IBM Storage Enabler for Containers, the IBM Storage Kubernetes FlexVolume driver is automatically installed on all master and worker nodes in a Kubernetes cluster, using the ubiquity-k8s-flex DaemonSet.

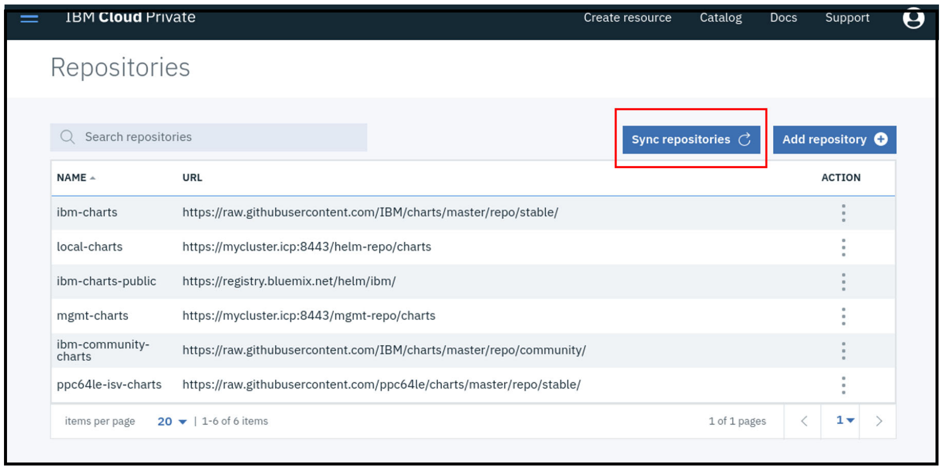
For further configuration of the chart and to start the installation, navigate to **Catalog** → **Storage** and search for the `ibm-storage-enabler-for-containers` Helm chart. See Figure 2-8.



A Helm chart for IBM Storage Enabler for Containers

Figure 2-8 IBM Storage Enabler for Containers Helm chart

6. If the `ibm-storage-enabler-for-containers` Helm chart is not available in the catalog, you need to sync the Helm repositories to get the latest charts. Navigate from the Menu list to **Manage** →**Helm Repositories** and sync the repositories. See Figure 2-9.

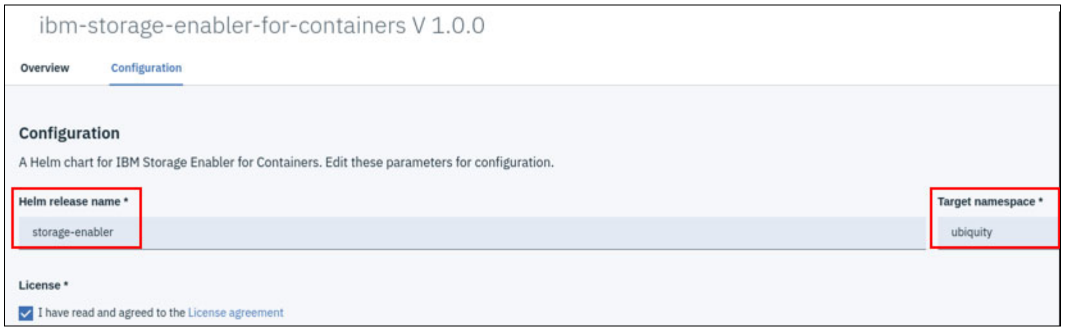


NAME	URL	ACTION
ibm-charts	https://raw.githubusercontent.com/IBM/charts/master/repo/stable/	⋮
local-charts	https://mycluster.icp:8443/helm-repo/charts	⋮
ibm-charts-public	https://registry.bluemix.net/helm/ibm/	⋮
mgmt-charts	https://mycluster.icp:8443/mgmt-repo/charts	⋮
ibm-community-charts	https://raw.githubusercontent.com/IBM/charts/master/repo/community/	⋮
ppc64le-isv-charts	https://raw.githubusercontent.com/ppc64le/charts/master/repo/stable/	⋮

items per page 20 | 1-6 of 6 items 1 of 1 pages 1

Figure 2-9 Sync Helm chart repositories

7. Open the chart and switch to the Configuration tab, set the values according to your needs. For this setup, as shown in Figure 2-10 on page 28, the following values were modified, the rest remained unchanged (default):
   – Helm release name
   – Target namespace (created in a previous step, the `PodSecurityPolicy` should be `ibm-anyuid-hostpath-psp`)


ibm-storage-enabler-for-containers V 1.0.0

Overview Configuration

Configuration

A Helm chart for IBM Storage Enabler for Containers. Edit these parameters for configuration.

Helm release name *: storage-enabler

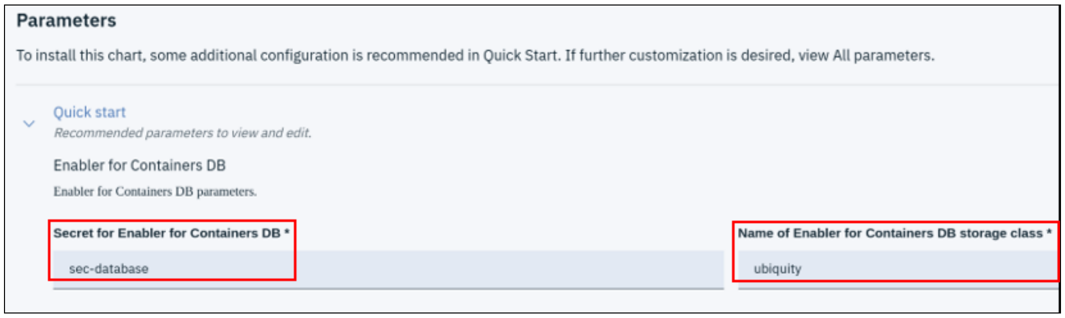
Target namespace *: ubiquity

License *

☑ I have read and agreed to the License agreement

Figure 2-10 IBM Storage Enabler for Containers Helm chart configuration 1

8. In the **Quick Start** section shown in Figure 2-11, edit the following fields:
   – Secret for Enabler for Containers DB
   – Name of Enabler for Containers DB storage class


Parameters

To install this chart, some additional configuration is recommended in Quick Start. If further customization is desired, view All parameters.

Quick start

Recommended parameters to view and edit.

Enabler for Containers DB

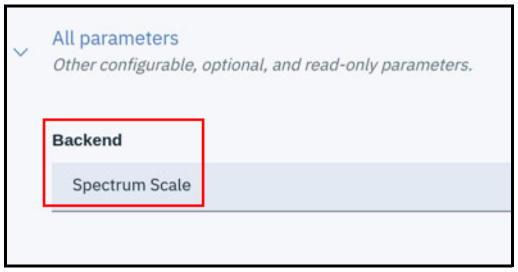
Enabler for Containers DB parameters.

Secret for Enabler for Containers DB *: sec-database

Name of Enabler for Containers DB storage class *: ubiquity

Figure 2-11 Enter secret for Enabler for Containers DB

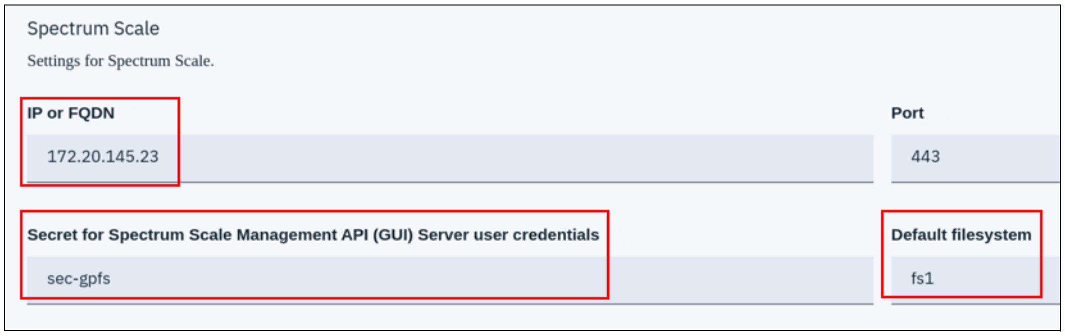
9. In the **All parameters** section, edit the fields for:
   - Backend (Figure 2-12)
   - IP or FQDN (Figure 2-13 on page 29)
   - Secret for Spectrum Scale Management API (Figure 2-14 on page 29)
   - Default filesystem (Figure 2-14 on page 29)
   - Secret for Enabler for Containers DB (Figure 2-14 on page 29)


All parameters

Other configurable, optional, and read-only parameters.

Backend: Spectrum Scale

Figure 2-12 IBM Storage Enabler for Containers Helm chart configuration 2


Spectrum Scale

Settings for Spectrum Scale.

IP or FQDN: 172.20.145.23

Port: 443

Secret for Spectrum Scale Management API (GUI) Server user credentials: sec-gpfs

Default filesystem: fs1

Figure 2-13 IBM Storage Enabler for Containers Helm chart configuration 3

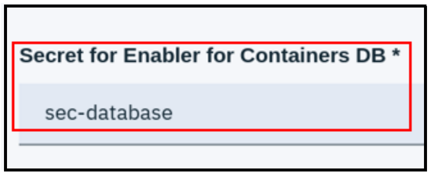

Secret for Enabler for Containers DB *: sec-database

Figure 2-14 IBM Storage Enabler for Containers Helm chart configuration 4

10. Click **Install** to start the installation.
11. When the installation is complete, ensure that the following steps are taken:
    a. Verify the post-installation status of the IBM Storage Enabler for Containers service. Run the following command:

    ```
    # helm status <release_name> --tls
    ```

    b. Check that the status of all components is error-free.

    Perform the sanity test, Figure 2-21, by running the following command:

    ```
    # helm test <release_name> --tls
    ```

    Example 2-21 Command output for sanity check

    ```
    # helm test storage-enabler --tls

    RUNNING: storage-enabler-sanity-test
    PASSED: storage-enabler-sanity-test
    ```

    c. (optional) Verify the environment using the Kubernetes CLI as shown in Example 2-22. All IBM Cloud Private cluster nodes should display the status `Ready`, pods should be in the state `Running`, and all deployments `Available`. For all PVs and PVCs, the status should be `Bound`.

    Example 2-22 Verifying IBM Storage Enabler for Containers environment using Kubernetes CLI

    ```
    # kubectl get nodes
    NAME            STATUS    ROLES              AGE    VERSION
    172.20.145.12   Ready     etcd,master        10d    v1.12.4+icp-ee
    172.20.145.13   Ready     worker             10d    v1.12.4+icp-ee
    172.20.145.14   Ready     worker             10d    v1.12.4+icp-ee
    172.20.145.15   Ready     worker             10d    v1.12.4+icp-ee
    172.20.145.16   Ready     management,proxy   10d    v1.12.4+icp-ee

    # kubectl get pods
    ```

```
NAME                                     READY   STATUS      RESTARTS
AGE   IP             NODE           NOMINATED NODE
ubiquity-6cb649f567-f6r6x                1/1     Running     0
20h   10.1.197.150   172.20.145.13   <none>
ubiquity-db-5b759d6868-f76v6             1/1     Running     0
20h   10.1.197.151   172.20.145.13   <none>
ubiquity-k8s-flex-9l844                  2/2     Running     1
20h   10.1.225.111   172.20.145.16   <none>
ubiquity-k8s-flex-bknkf                  2/2     Running     1
20h   10.1.197.149   172.20.145.13   <none>
ubiquity-k8s-flex-g6nt7                  2/2     Running     1
20h   10.1.32.38     172.20.145.15   <none>
ubiquity-k8s-flex-s282x                  2/2     Running     0
20h   10.1.221.69    172.20.145.14   <none>
ubiquity-k8s-flex-tvt6z                  2/2     Running     1
20h   10.1.85.211    172.20.145.12   <none>
ubiquity-k8s-provisioner-646ff59fc4-4qfvl   1/1     Running     0
20h   10.1.32.39     172.20.145.15   <none>

# kubectl get deployment
NAME       DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
ubiquity     1         1         1            1         20h
ubiquity-db  1         1         1            1         20h
ubiquity-k8s 1         1         1            1         20h
-provisioner

# kubectl get pvc
NAME             STATUS   VOLUME           CAPACITY   ACCESS MODES
STORAGECLASS   AGE
ibm-ubiquity-db   Bound    ibm-ubiquity-db   20Gi       RWO
ubiquity       20h

# kubectl get pv
NAME                             CAPACITY   ACCESS MODES   RECLAIM POLICY
STATUS   CLAIM                                    STORAGECLASS
REASON   AGE
helm-repo-pv                     5Gi        RWO            Delete
Bound   kube-system/helm-repo-pvc             helm-repo-storage
10d
ibm-ubiquity-db                  20Gi       RWO            Delete
Bound   ubiquity/ibm-ubiquity-db              ubiquity
20h
image-manager-172.20.145.12      20Gi       RWO            Retain
Bound  kube-system/image-manager-image-manager-0  image-manager-storage
10d
logging-datanode-172.20.145.16   20Gi       RWO            Retain
Bound    kube-system/data-logging-elk-data-0
logging-storage-datanode           10d
mariadb-172.20.145.12            10Gi       RWO            Retain
Bound   kube-system/mysqldata-mariadb-0        mariadb-storage
10d
mgmt-repo-pv                     5Gi        RWO            Delete
Bound   kube-system/mgmt-repo-pvc             mgmt-repo-storage
10d
```

```
mongodb-172.20.145.12          20Gi       RWO            Retain
Bound   kube-system/mongodbdir-icp-mongodb-0     mongodb-storage
10d
```

For more information regarding the installation of the IBM Storage Enabler for Containers can be found on the IBM Spectrum Scale deployment website.

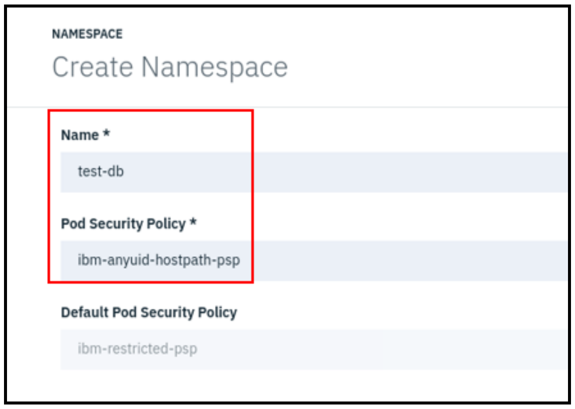
2.9 Deploying MariaDB instance

MariaDB is an open source, object-relational database system, used by notable users include Wikipedia, WordPress.com, and Google. In this example MariaDB is deployed into IBM Cloud Private by using a Helm chart and persistent volumes.

The persistent volumes are based on IBM Spectrum Scale filesets. Those filesets are created automatically during the Helm chart installation.

Kubernetes namespaces help different projects, teams, or customers to share a Kubernetes cluster. Although it is not necessary to create a namespace for the mariadb deployment in this small cluster, the `test-db` namespace will be set up for demonstration purposes.

1. Create namespace `test-db`. **Navigate to Manage →Namespaces**. See Figure 2-15.



NAMESPACE

Create Namespace

Name *

test-db

Pod Security Policy *

ibm-anyuid-hostpath-psp

Default Pod Security Policy

ibm-restricted-psp

Figure 2-15   Create namespace for Mariadb

Ensure the Pod Security Policy is set to `ibm-anyuid-hostpath-psp`. This policy allows running pods as any user with any fsgroups / supplemental group IDs and all volumes.

A description of some security requirements of Helm charts available on IBM Cloud can be found on the Chart Security Summary.

2. Create a YAML that contains a storage class definition, as shown in Example 2-23.

Example 2-23   Storage class yml-file named "test-db"

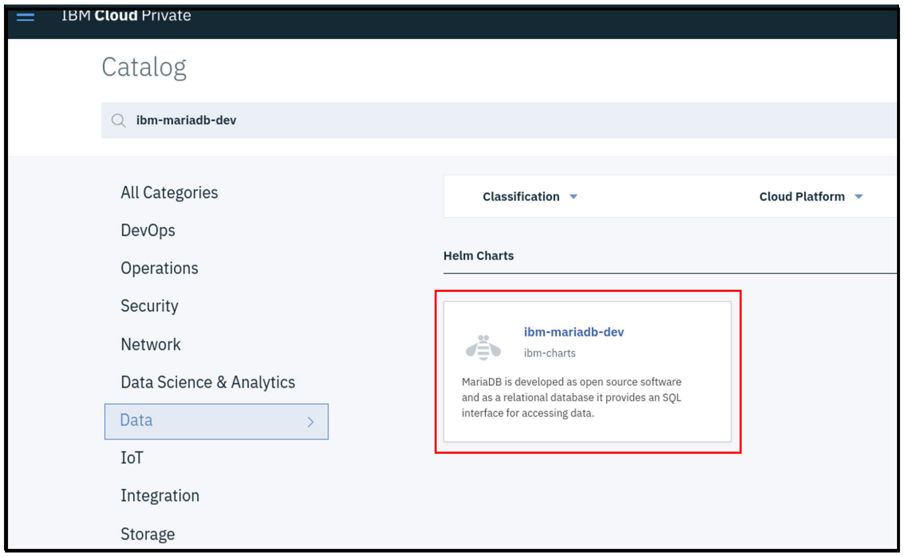
```
# cat testdb-storage-class.yml
kind: StorageClass
apiVersion: storage.k8s.io/v1beta1
metadata:
  name: "test-db"
  labels:
    product: ibm-storage-enabler-for-containers
provisioner: "ubiquity/flex"
parameters:
  backend: "spectrum-scale"
  filesystem: "fs1"
  fileset-type: "dependent"
  type: fileset
```

3. Create the storage class as shown in Example 2-24.

Example 2-24   Create storage class

```
# kubectl create -f testdb-storage-class.yml
storageclass.storage.k8s.io/test-db created
```

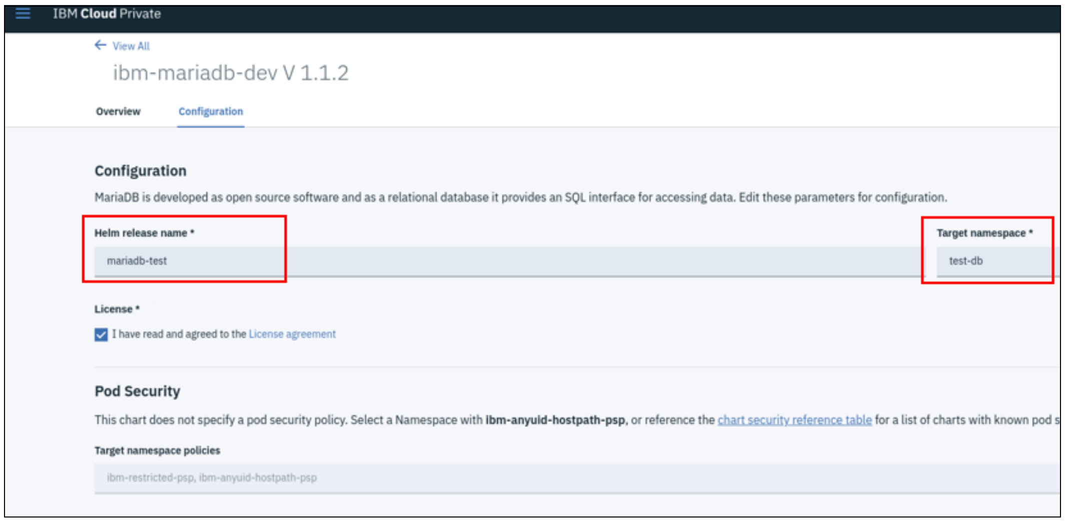
4. From the **Catalog**, search for the `ibm-mariadb-dev` Helm chart shown in Figure 2-16.



MariaDB is developed as open source software and as a relational database it provides an SQL interface for accessing data.

Figure 2-16   IBM MariaDB Helm chart

5. In the **Configuration** tab, Figure 2-17, enter a Helm release name and select the previously created namespace for `Target namespace`. Verify that the `ibm-anyuid-hostpath-psp` policy is available for the target namespace policy.



Configuration

MariaDB is developed as open source software and as a relational database it provides an SQL interface for accessing data. Edit these parameters for configuration.

Helm release name * mariadb-test

Target namespace * test-db

License *

✓ I have read and agreed to the License agreement

Pod Security

This chart does not specify a pod security policy. Select a Namespace with **ibm-anyuid-hostpath-psp**, or reference the chart security reference table for a list of charts with known pod s

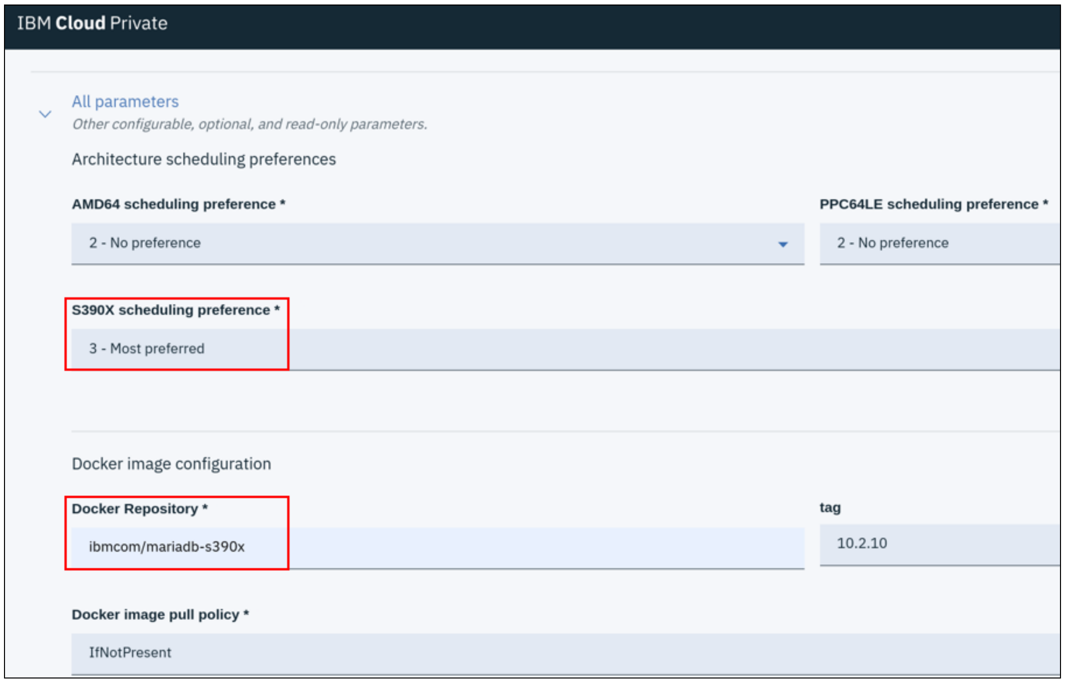
Target namespace policies

ibm-restricted-psp, ibm-anyuid-hostpath-psp

Figure 2-17   IBM MariaDB Helm chart configuration 1

6. In the **All parameters** section, Figure 2-18 and Figure 2-19, edit the fields for:
   – S390x scheduling preference →**Most preferred**
   – Docker repository →**ibmcom/mariadb-s390x**
   – Use dynamic provisioning for persistent volumes →put on check mark
   – Existing storage class name →**test-db** (previously created)

   All other parameters remain unchanged (default settings).


IBM Cloud Private

All parameters
Other configurable, optional, and read-only parameters.

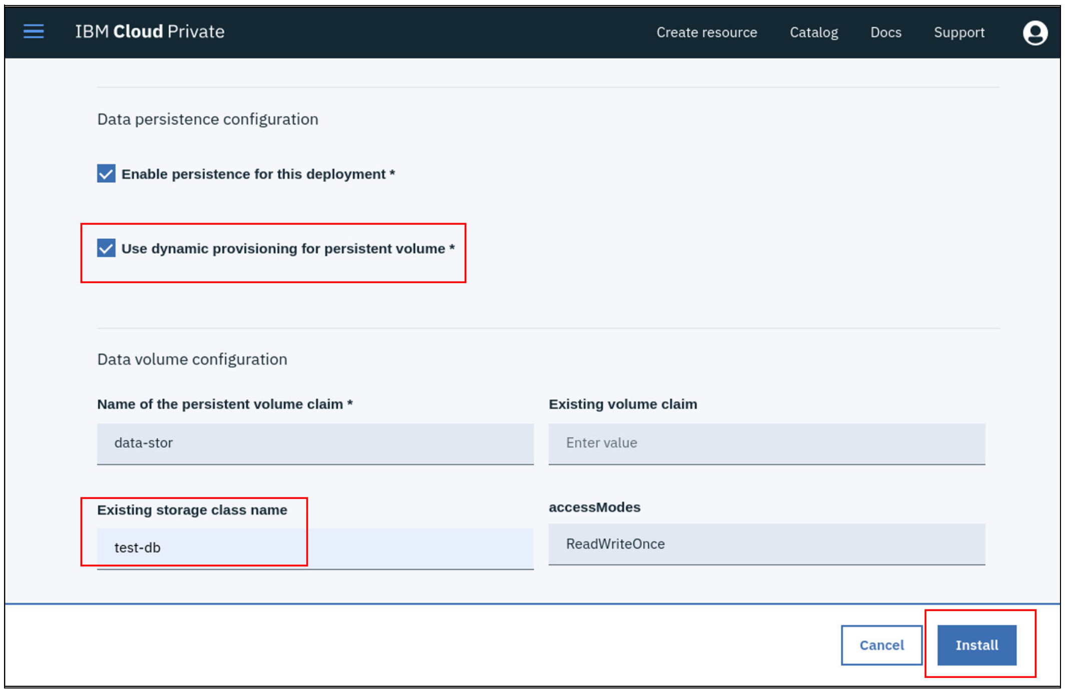
Architecture scheduling preferences

AMD64 scheduling preference *	2 - No preference
PPC64LE scheduling preference *	2 - No preference
S390X scheduling preference *	3 - Most preferred

Docker image configuration

Docker Repository *	ibmcom/mariadb-s390x
tag	10.2.10
Docker image pull policy *	IfNotPresent

Figure 2-18 IBM MariaDB Helm chart configuration 2



Data persistence configuration

✓ Enable persistence for this deployment *

✓ Use dynamic provisioning for persistent volume *

Data volume configuration

Name of the persistent volume claim *	data-stor
Existing volume claim	Enter value
Existing storage class name	test-db
accessModes	ReadWriteOnce

Figure 2-19 IBM MariaDB Helm chart configuration 3

7. Click **Install** as shown in Figure 2-19 on page 33, and after a couple seconds the following screen, Figure 2-20, should display.

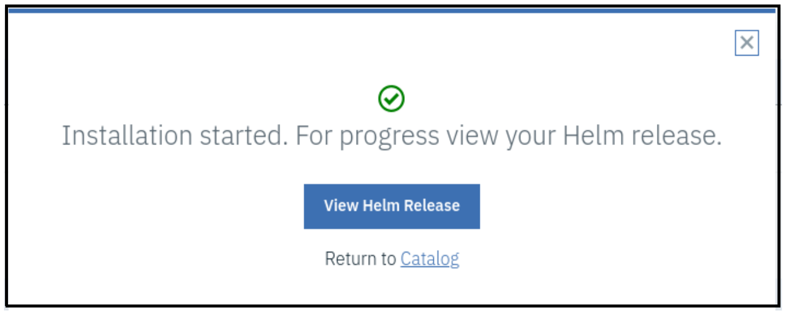

Installation started. For progress view your Helm release.

Figure 2-20 Helm chart installation in progress

8. After a couple minutes the status of pvc, pod, and deployment can be verified using the GUI or Kubernetes CLI (see Example 2-25).

Example 2-25 Verification of database deployment using Kubernetes CLI

```
# kubectl get deployment -n test-db
NAME                          DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
mariadb-test-ibm-mariadb-dev  1         1         1            1           157m

# kubectl get pod -n test-db
NAME                                        READY   STATUS    RESTARTS   AGE
mariadb-test-ibm-mariadb-dev-76cdf75bd-4zsp6   1/1     Running   0          157m

# kubectl get pvc -n test-db
NAME                                  STATUS   VOLUME
CAPACITY   ACCESS MODES   STORAGECLASS   AGE
mariadb-test-ibm-mariadb-dev-data-stor   Bound
pvc-8cbf8e49-667c-11e9-b543-029106000059   25Gi  RWO            test-db        157m

# kubectl get pv | grep "NAME\|test-db"
NAME                                      CAPACITY   ACCESS MODES   RECLAIM POLICY
STATUS   CLAIM                                          STORAGECLASS
REASON   AGE
pvc-8cbf8e49-667c-11e9-b543-029106000059   25Gi       RWO            Delete
Bound    test-db/mariadb-test-ibm-mariadb-dev-data-stor   test-db
4h25m
```

9. Verify database instance

   MariaDB can be accessed via port 3306 on the following DNS name from within the `mariadb-test-ibm-mariadb-dev.test-db.svc.cluster.local` cluster.

   Example 2-26 shows how to connect to the database using the DNS name from above.

Example 2-26 Determine the database password and connect to the database

```
# MARIADB_PASSWORD=$(kubectl get secret --namespace test-db
mariadb-test-ibm-mariadb-dev -o jsonpath="{.data.password}" | base64 --decode;
echo)
# kubectl run mariadb-test-ibm-mariadb-dev-client \
--rm --tty -i --image ibmcom/mariadb-s390x:10.2.19 \
--env "MARIADB_PASSWORD=$MARIADB_PASSWORD" --command \
-- mysql -u 'testuser' -p$MARIADB_PASSWORD \
-h mariadb-test-ibm-mariadb-dev.test-db.svc.cluster.local test
```

10. After a couple of seconds a command prompt should show up. If you don’t see a command prompt, try pressing enter. For verification, create a simple table, add some records, and run a query (see Example 2-27).

Example 2-27 Simple database query

```
MariaDB [test]> CREATE DATABASE somedata;
Query OK, 1 row affected (0.01 sec)

MariaDB [test]> USE somedata;
Database changed
MariaDB [somedata]> CREATE TABLE datahere ( name_id int(5) NOT NULL
AUTO_INCREMENT, namedata varchar(50) DEFAULT NULL, PRIMARY KEY(name_id) );
Query OK, 0 rows affected (0.15 sec)

MariaDB [somedata]> INSERT INTO datahere ( namedata ) VALUES ( "Tom" );
Query OK, 1 row affected (0.01 sec)

MariaDB [somedata]> INSERT INTO datahere ( namedata ) VALUES ( "Jerry" );
Query OK, 1 row affected (0.00 sec)

MariaDB [somedata]>  SELECT * FROM datahere;
+---------+----------+
| name_id | namedata |
+---------+----------+
|       1 | Tom      |
|       2 | Jerry    |
+---------+----------+
2 rows in set (0.00 sec)

MariaDB [somedata]> quit
Bye
Session ended, resume using 'kubectl attach
mariadb-test-ibm-mariadb-dev-client-78746cf664-dms4c -c
mariadb-test-ibm-mariadb-dev-client -i -t' command when the pod is running
deployment.apps "mariadb-test-ibm-mariadb-dev-client" deleted
```

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this paper.

IBM Redbooks

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

- *IBM Spectrum Connect and IBM Storage Enabler for Containers*, REDP-5470

You can search for, view, download or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, at the following website:

ibm.com/redbooks

Online resources

These websites are also relevant as further information sources:

- IBM Cloud Private v3.2.0 documentation

  https://www.ibm.com/support/knowledgecenter/SSBS6K_3.2.0/kc_welcome_containers.html
- IBM Spectrum Scale 5.0.3

  https://www.ibm.com/support/knowledgecenter/en/STXKQY_5.0.3/ibmspectrumscale503_welcome.html
- IBM Storage Enabler for Containers

  https://www.ibm.com/support/knowledgecenter/en/SS6JWS_3.5.0/UG/sc_ug_ch_containers.html
- Installing IBM Cloud Private for Linux on IBM Z and LinuxONE

  https://www.ibm.com/support/knowledgecenter/en/SSBS6K_3.1.2/installing/install_s390x.html

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

REDP-5547-00

ISBN 0738457833