Redbooks
ibm.com/redbooks

IBM Spectrum Connect and IBM Storage Enabler for Containers

Practical Example with IBM FlashSystem A9000

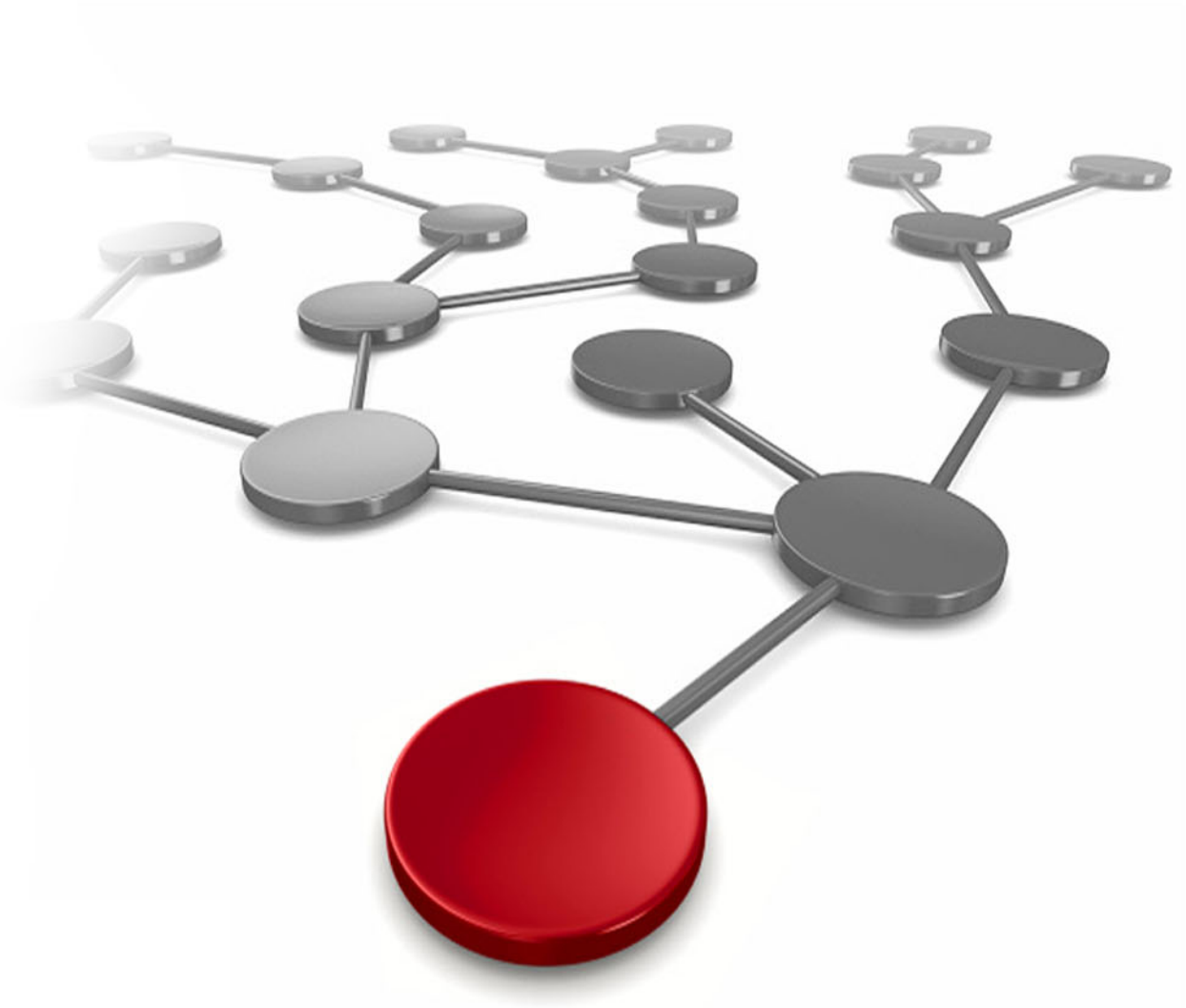
Markus Oscheka

Bert Dufrasne

Andrew Greenfield

Axel Westphal

Josh Blumert



IBM

Redpaper

International Technical Support Organization

IBM Spectrum Connect and IBM Storage Enabler for Containers

April 2018

REDP-5470-00

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

First Edition (April 2018)

This edition applies to IBM Spectrum Connect Version 3.4 with IBM Storage Enabler for Containers Version 1.0.

This document was created or updated on April 10, 2018.

Contents

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, MD-NC119, Armonk, NY 10504-1785, US

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at “Copyright and trademark information” at http://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks or registered trademarks of International Business Machines Corporation, and might also be trademarks or registered trademarks in other countries.

AIX®
DS8000®
HyperSwap®
IBM®
IBM FlashSystem®
IBM Spectrum™
IBM Spectrum Accelerate™
IBM Spectrum Control™
IBM Spectrum Virtualize™
MicroLatency®
Real-time Compression™
Redbooks®
Redpaper™
Redbooks (logo) ®
Storwize®
System Storage®
XIV®

The following terms are trademarks of other companies:

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Preface

This IBM® Redpaper™ publication provides an overview of containers and their framework. Container technology enables prepackaged and pre-configured software with the elements that are needed to run in any environment.

Because they are meant to be portable, containers normally restrict applications from storing data on external storage. To overcome this limitation, IBM has developed a solution to provide persistent storage for containers on IBM storage systems, known as the IBM Storage Enabler for Containers. The Enabler tightly integrates with IBM Spectrum™ Connect (formerly IBM Spectrum Control™ Base Edition).

IBM Storage Enabler for Containers extends IBM Spectrum Connect v 3.4 to Kubernetes orchestrated container environments.

The paper focuses on containers implementation, management, and control by using IBM Spectrum Connect and IBM Storage Enabler for Containers plug-in, with IBM FlashSystem® A9000 or A9000R.

Authors

This paper was produced by a team of specialists from around the world:

Markus Oscheka is an IT Specialist for Proof of Concepts and Benchmarks in the Disk Solution Europe team in Kelsterbach, Germany. His areas of expertise include setup and demonstration of IBM System Storage® solutions in various environments including IBM AIX®, Linux, Windows, VMware ESXi and Solaris. He has performed many proof of concepts and benchmarks for the IBM DS8000®, Spectrum Accelerate and the Spectrum Virtualize family of products. He is also a storage technical advisor. He has written in various IBM Redbooks® publications and spoken at several System Technical Universities. He holds a degree in Electrical Engineering.

Bert Dufrasne is an IBM Certified Consulting IT Specialist and Project Leader for IBM System Storage disk products at the International Technical Support Organization (ITSO), San Jose Center. He has worked at IBM in various IT areas. He has authored many IBM Redbooks publications, and has also developed and taught technical workshops. Before Bert joined the ITSO, he worked for IBM Global Services as an Application Architect. He holds a Master's degree in Electrical Engineering.

Andrew Greenfield is an IBM Global XIV® and Flash Solution Engineer who is based in Phoenix, Arizona. He holds numerous technical certifications from Cisco, Microsoft, and IBM. Andrew brings over 24 years of data center experience inside the Fortune 100 to the team. He graduated magna cum laude, honors, from the University of Michigan, Ann Arbor. Andrew has also written for and contributed to several IBM Redbooks publications.

Axel Westphal is working as certified IT Specialist at the IBM EMEA Storage Competence Center (ESCC) in Kelsterbach, Germany. He joined IBM in 1996, working for Global Services as a systems engineer. His areas of expertise include setup and demonstration of IBM storage products and solutions in various environments. He has written several storage white papers and co-authored several IBM Redbooks publications about DS8000, Spectrum Virtualize, and Spectrum Accelerate.

Josh Blumert is a member of the IBM Storage Solutions Engineering team. He has been with IBM for 16 years as a server specialist for IBM System x focusing on Linux, VMware, and Windows systems. Josh ran the IBM Solution center for financial services in New York City before joining the storage team, where he continues as a server and application expert for distributed systems. Before joining IBM, Josh was a computational engineer for Silicon Graphics covering high-performance computing. He holds an B.S. in Physics with a focus in computer science from Rensselaer Polytechnic Institute in New York.

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

Shay Berman
Dima Isayev
Ran Harel
Samuel Krikler
Marjorie Schejter
Harald Seipp
Yossi Siles
IBM

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

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 found at:

  ibm.com/redbooks

- Send your comments in an email to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  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 to containers

This chapter provides an overview of the history and current direction in the use of containers and their framework. Containers take virtualization of the operating system (OS) and applications, along with their development, to a new level.

Containers are well suited for both on-premises and cloud-based applications. Although containers and virtual machines (VMs) have similar resource isolation and allocation benefits, they function differently because containers virtualize the OS itself, instead of hardware. By doing so, containers are far more portable and efficient.

The focus of this chapter is on Docker's implementation of containers along with the various provisioning tools, and also orchestration software such as Kubernetes.

This introduction is solely meant to set the context for the IBM Storage Enabler for Containers that allows IBM storage systems to be used as block storage devices for Kubernetes container orchestrator.

This chapter includes the following sections:

- Context and history
- Storage containers

1.1 Context and history

From the dawn of computing, developers have employed several different methods of building and distributing software applications.

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.

The first container concept started with UNIX's *chroot* in 1979. There were several key evolutionary variants such as FreeBSD jails in 1998, SWsoft's (later Parallels') Virtuozzo in 2001 (and the associated OpenVZ in 2005), Solaris' Zone in 2005, and the workload partitions (WPARs) on AIX and also HP-UX Containers in 2007.

Solomon Hykes started Docker in France as an internal project within dotCloud, a Platform-as-a-Service (PaaS) company, with initial contributions by other dotCloud engineers. Docker represents an evolution of dotCloud's proprietary technology, which is, itself, built on earlier open-source projects such as Cloudlets. The software debuted to the public at PyCon in 2013.

Docker is a software technology providing containers, promoted by the company of the same name, Docker, Inc. Docker provides an additional layer of abstraction and automation of operating-system-level virtualization on Windows and Linux.

Docker was released as open source in March 2013. As of October 24, 2015, the project had over 25,600 GitHub stars (making it the 20th most-starred GitHub project), over 6,800 forks, and nearly 1,100 contributors.

Docker is available for both Linux and Windows-based applications. The current environment of Docker is quite popular across the IT industry.

1.1.1 Docker architecture and components

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, remote data centers, or the cloud.

Docker is a client/server architecture. The Docker server (Docker Daemon) is responsible for building, running, and distributing containers. The Docker client is the command-line interface that allows you to communicates to the Docker server (Docker Daemon) using a REST API.

Here are major elements of the Docker environment:

- Registry: This is the cold storage of container images. A Registry stores container images. A registry can be hosted on-premises or made available on a public cloud. Several software vendors make their registry public, allowing developers to acquire container images for applications or services from those vendors for use in their own environment.
- Images: Images are templates with information and all the dependencies for creating containers.
- Container: This is the active, running instance of an image.
- Volumes: These are the "data" part of a container, initialized when a container is created. Volumes allow you to persist and share a container's data.

1.1.2 Container benefits

One of the easiest ways to think about Docker containers is they provide a VM without the overhead of a VM. This section compares the major benefits of containers versus virtual machines.

A closer look at an individual container shows that it is a lightweight, stand-alone portable, executable, software package (application) that includes everything needed to run it, including the binary runtime code, system tools and libraries, and settings. The biggest benefit of the containers is that the application will always run the same, regardless of the underlying environment.

As shown in Figure 1-1, containers have far less overhead in both physical storage and processor footprint, which gives them better performance. This lighter footprint enables containers to boot and restart in seconds, compared to several minutes for VMs. Also, there is no extra overhead of a hypervisor and guest OS, which further enables containers to consume far less CPU and memory than a similar VM environment.

This reduced footprint also gives containers a portability advantage: VMs are usually in the gigabytes range, whereas containers are in the megabytes. A key portability concern with VMs is that they are limited to a particular Hypervisor and set of specific hardware-emulation instructions, whereas containers do not have such limitations. Lastly, remember that each VM requires a fully functional operating system, with extra management for each one of them, whereas none of this is needed for containers.

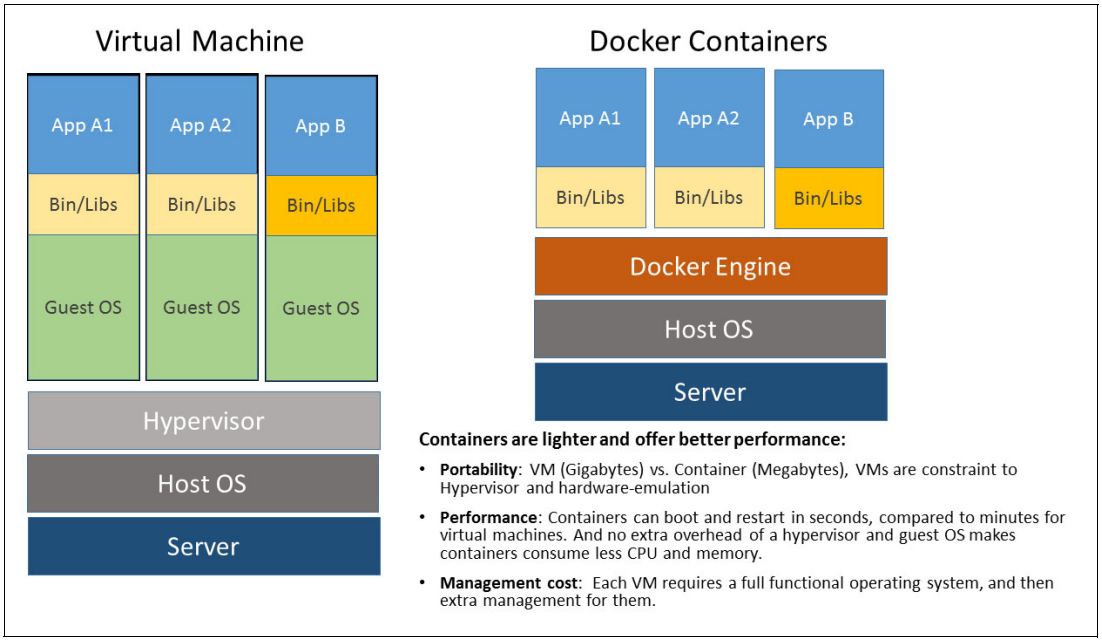


Figure 1-1 Containers have less overhead than VMs

The portability benefit is not to be underestimated, as containers truly reduce, if not eliminate, many concerns about compatibility across different platforms.

These advantages make containers far better suited than VMs for developers, batch computing, and lightweight PaaS environments.

Developers can “build once and then run anywhere.” Containers provide a true clean, safe, portable runtime environment for their applications without any worries about missing dependencies, packages, and other challenges during subsequent deployments.

As shown in Figure 1-1 on page 3, Docker allows each application to run in its own isolated container, while still allowing some specific runtime libraries and other dependencies required by a particular application to be shared, all without requiring the large footprint of a VM. Docker also share libraries, which ensures efficiencies for testing, integration, and packaging.

Administrators and developers can both benefit from containers' "configure once, then run anything" feature, alongside the entire application lifecycle, making it far more efficient, consistent, and repeatable, while eliminating inconsistencies between development, test, production, and customer environments.

Containers significantly improve the speed and reliability of a continuous deployment model. They also reduce and eliminate many of the deployment and portability issues normally associated with VMs. In summary, containers provide the following benefits:

- Loosely coupled
- Easier to scale development
- Improved fault isolation
- Each service can be developed and deployed independently
- Eliminates any long-term commitment to a technology stack
- Ability to move across Public, Private, and Hybrid Clouds in a seamless manner
- Seamless Development to Production, allowing you to use the exact same code in Development and Production
- Easier management of all apps, containers, images, users, and groups
- Supports Integrated security using RBAC for LDAP/Active Directory and TLS security
- Ability to establish trusted signing of containers, ensuring validity.

Figure 1-2 summarizes the characteristics of containers in comparison to virtual machines and bare-metal servers.

	Containers	**Virtual Machine**	**Bare Metal Server**
Platform	Operating system on Virtual Machine, Intel x86, IBM Power+ , Or IBM Z Mainframe	Hypervisor on Bare Metal Server	–
Performance	Average	Average	Fastest
Time to Provision	Seconds	Minutes	Hours
Isolation Enforcement for Tenants	Kernel of the OS	Hypervisor	Physical
Flexibility to Configure or Reconfigure	Most/fastest	Medium	Low/tedious
Host Density	Most Dense	Average	None
Granularity	Atomic and smallest	Average	Large

Figure 1-2 Comparison between Docker, Virtualization, and Bare Metal

1.1.3 Managing containers with Kubernetes

As previously described, a container is a group of processes or services run in complete isolation with only its needed runtime binary files. A container has methods to limit resource consumption, and uses namespaces for complete isolation without the need for a full guest OS storage image or performance footprint.

As part of learning about containers, it is also important to understand container management. The lead orchestrator is an open source project called Kubernetes. This configuration is shown in Figure 1-3.

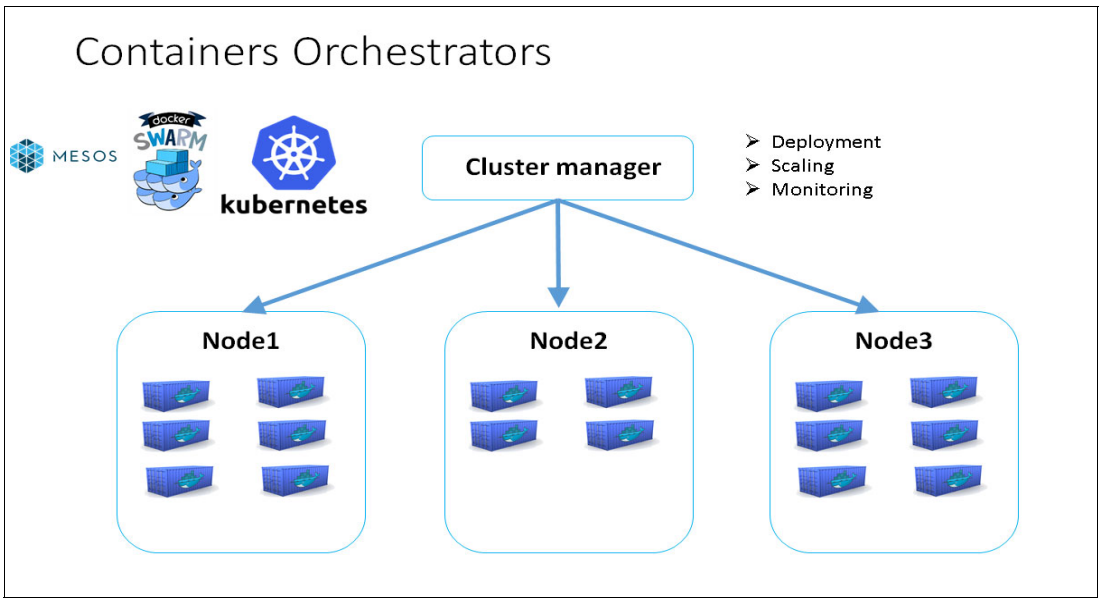


Figure 1-3 Kubernetes diagram as a popular orchestrator and how it works with containers

Kubernetes, commonly referred to as *K8s*, is an open source system for automating deployment, scaling, and management of containerized applications, which was originally designed by Google and donated to the Cloud Native Computing Foundation. It supports a range of container engines, including Docker.

You can see how Docker pods can be orchestrated inside a Kubernetes framework, as shown in Figure 1-4 on page 6. A pod is the designation for the simplest unit that can be created and deployed in Kubernetes. A pod encapsulates an application container, storage resources, a unique network IP, and options that govern how the container should run.

Kubernetes performs various tasks automatically, such as starting or restarting containers, scaling the number of replicas of a given application, mounting storage systems, and more. The Kubernetes framework, illustrated in Figure 1-4 on page 6 consists of a collection of processes that run on nodes (servers or VMs) in your cluster.

- The Kubernetes Master is a collection of three processes that run on a single node in your cluster, which is designated as the master node. Note that in production environment and for obvious High Availability reasons, multiple Master nodes will be deployed. The following are the processes involved:
  - API server (kube-apiserver): This process is the main management point of the entire cluster. It allows developers to configure many of Kubernetes' workloads and orchestrates the various components of the cluster.

- controller-manager (kube-controller-manager): This process regulates the state of the cluster and performs routine tasks.
- Scheduler (kube-scheduler): This process assigns and regulates workloads on cluster nodes, within available resources.
- Etcd: This is the configuration store. It stores configuration data that can be used by each of the nodes in the cluster.

► Each individual non-master node in the cluster runs two processes:

- kubelet, which is a service to manage pods that have been assigned to its node and to communicate with the Kubernetes Master.
- kube-proxy, which is a network proxy that reflects Kubernetes networking services on each node.

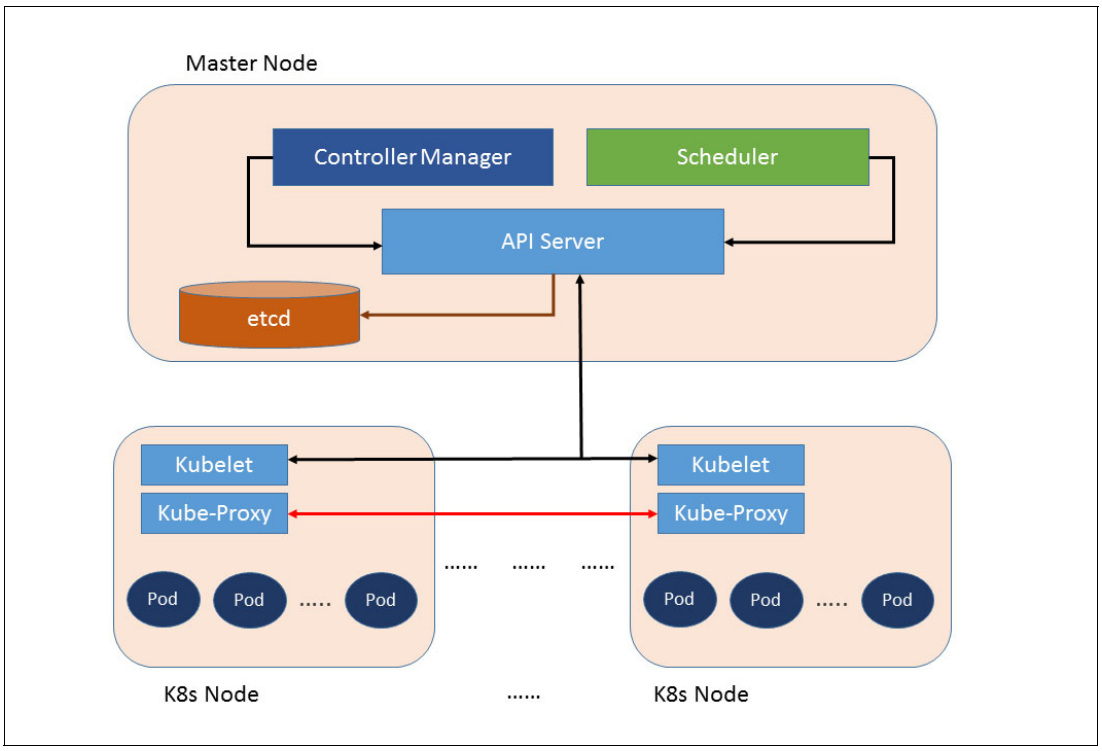


Figure 1-4 Kubernetes framework

The Kubernetes nodes include pods. A pod is a group of one or more containers that are always collocated, coscheduled, and run inside a shared context.

Data inside a container is not persistent. When the container crashes or is terminated, kubelet will normally restart it. However, the prior data is lost. To solve this issue, Kubernetes provides a Persistent Volume (PV) concept, which is also part of a Persistent Volume Claim (PVC), which is explored in 1.2, “Storage containers” on page 7.

1.2 Storage containers

Because they 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. This process is shown in Figure 1-5

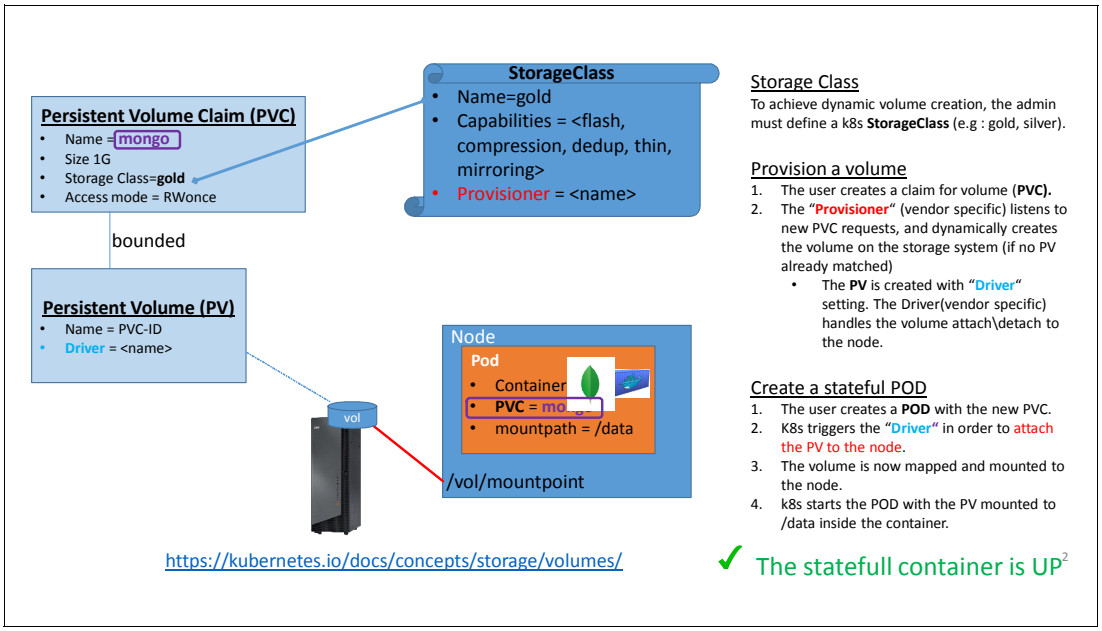


Figure 1-5 How PV and PVC work with a Storage Class to provision a container and POD

1.2.1 IBM Storage Enabler for Containers

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 to be used as persistent storage for Kubernetes orchestrated container environments. IBM Storage Enabler for Containers uses Kubernetes Dynamic Volume Provisioner and FlexVolume driver for volume provisioning.

As shown in Figure 1-6, IBM is using Spectrum Connect as the unifying framework to provide a single control pane across the diverse storage portfolio that is offered by IBM.

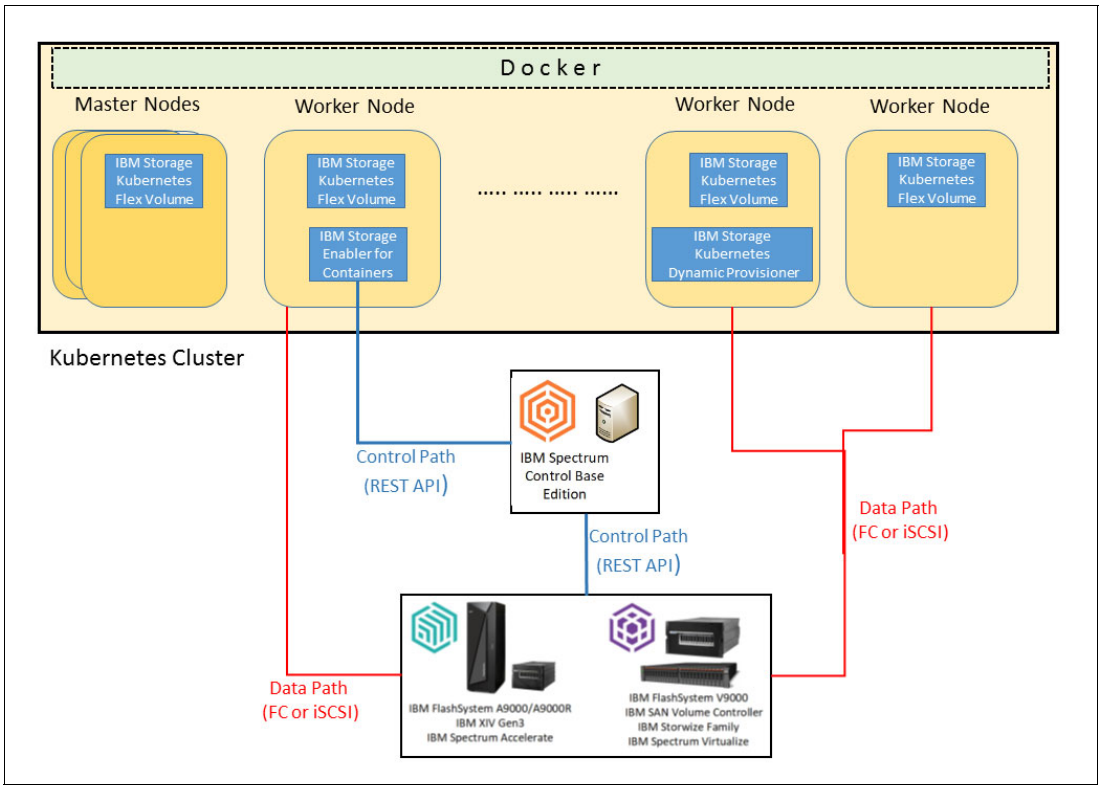


Figure 1-6 IBM Storage systems integration in a Kubernetes orchestrated container environment

IBM Storage Enabler for Containers communicates with the IBM storage systems through IBM Spectrum Connect. The Spectrum Connect storage administrator creates a storage profile and makes it available for Kubernetes Dynamic Provisioner and FlexVolume, automating IBM storage provisioning for Kubernetes persistent volumes. This feature provides these benefits:

- The Volume Provisioner allows storage volumes to be created on-demand, using Kubernetes storage classes as templates based on Spectrum Control Base storage services. This feature provides abstraction for the underlying storage platform, eliminating the need for cluster administrators to pre-provision storage.
- The FlexVolume enables automatic mounting of storage volumes into a pod within a Kubernetes node.

The following IBM block storage systems are available:

- The IBM Spectrum Accelerate™ family products such as IBM FlashSystem A9000 and A9000R
- IBM XIV
- IBM Spectrum Virtualize™
- The IBM Storwize® family of products such as IBM FlashSystem V9000, IBM SAN Volume Controller, and other members of the IBM Storwize Family

2

Using IBM Storage Enabler for Containers in Kubernetes

This chapter shows how to use the IBM Storage Enabler for Containers to provision storage for stateful containers running in Kubernetes, in IBM Spectrum Connect v3.4.

Although the chapter presents a practical approach and illustrations with IBM FlashSystem A9000 as the back-end storage system, the IBM Spectrum Accelerate family and IBM Spectrum Virtualize family systems are also supported by IBM Storage Enabler for Containers. For more information about the supported IBM storage systems, see the IBM Spectrum Connect release notes.

This chapter includes the following sections:

- IBM Spectrum Connect overview
- IBM Spectrum Connect and IBM FlashSystem A9000 or A9000R configuration
- Docker installation and configuration
- Kubernetes installation and configuration
- IBM Storage Enabler for Containers installation and configuration
- IBM Storage Enabler for Containers provisioning tasks

2.1 IBM Spectrum Connect overview

IBM Spectrum Connect is a centralized server system that consolidates a range of storage provisioning, automation, and monitoring solutions through a unified server platform. It provides a single-server, back-end location and enables centralized management of storage resources for different virtualization and cloud platforms.

IBM Spectrum Connect facilitates the integration of IBM storage systems resources by taking advantage of features provided by independent software vendor (ISV) products and solutions. It provides a foundation for integration with IBM systems and ISV solutions.

IBM Spectrum Connect version 3.4 supports different IBM storage systems, including IBM DS8000 family, IBM FlashSystem A9000, A9000R, and V9000; IBM Spectrum Accelerate and XIV; and IBM Spectrum Virtualize. For a complete list of the supported IBM Storage Systems and respective microcode levels, refer to IBM Knowledge Center for IBM Spectrum Connect.

IBM Storage Enabler for Containers v1.0 extends IBM Spectrum Connect v3.4 to Kubernetes orchestrated container environments. IBM Spectrum Connect's policy-based automated storage provisioning can now be applied to container environments, VMware environments, and mixed environments.

Note: This paper and the illustrations provided in this chapter focus on the IBM FlashSystem A9000 or A9000R as the back-end storage system.

Figure 2-1 shows a conceptual diagram of the consolidation that IBM Spectrum Connect enables. This is a conceptual diagram, with VMware vSphere Web Client and vRealize, Microsoft PowerShell (supported only with Spectrum Virtualize), and containers, with Kubernetes on Docker as currently supported target environments. Other environments that are denoted as Other ISVs can be supported in the same framework.

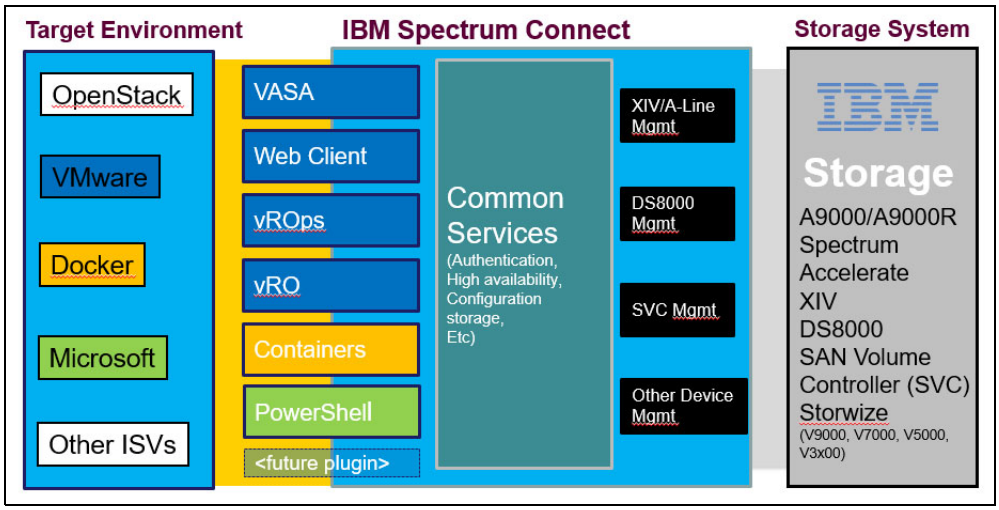


Figure 2-1 IBM Spectrum Connect concept

2.1.1 IBM Spectrum Connect management

IBM Spectrum Connect can be managed through a standard web browser with a graphical user interface (GUI), or through a terminal and command-line interface (CLI). Spectrum Connect can either run stand-alone or integrated with Hyper-Scale Manager.

IBM Spectrum Connect runs as a host application under Linux with minimum requirements and a straightforward installation and configuration process. For more information, see 2.2, “IBM Spectrum Connect and IBM FlashSystem A9000 or A9000R configuration”.

For more information:

For more information about compatibility and the requirements of the IBM Spectrum Connect, see IBM Knowledge Center.

For more information about extracting and installing the IBM Spectrum Connect software package, see IBM Knowledge Center.

2.2 IBM Spectrum Connect and IBM FlashSystem A9000 or A9000R configuration

The necessary steps for configuration are summarized below:

1. Log in to Spectrum Connect and start the Initial Setup Wizard. It includes the following steps, as detailed in 2.2.1, “Log in and initial Setup Wizard” on page 12:

   Note: The initial setup step is only required for the first time use after a fresh installation of Spectrum Connect.

   a. Define the IBM Spectrum Connect (SC) fully qualified domain name and high availability group.
   b. Generate a server certificate.
   c. Set up storage credentials.

2. Set up a domain with user and host on A9000 using Hyper-Scale Manager (HSM). See 2.2.2, “FlashSystem A9000 or A9000R domain configuration” on page 15 for details.

   Note: Setting up domains is only necessary if you either want to use multi tenancy for your containers or if a domain already existed on your FlashSystem A9000.

3. Configure Spectrum Connect for Containers. It includes the following steps, as detailed in 2.2.3, “Spectrum Connect configuration for containers” on page 20:

   a. Add the IBM storage system as a storage array.
   b. Create a Storage Space.
   c. Create a Storage Service.
   d. Assign Resource to Storage Service.
   e. Add the container server.
   f. Delegate Storage Service to container server.

2.2.1 Log in and initial Setup Wizard

Log in to the IBM Spectrum Control web interface:

`https://IBM_Spectrum_Control_IP_address:8440`

Enter the default login credentials of user `admin` and password `admin1!`, and click **Login**, as shown in Figure 2-2.

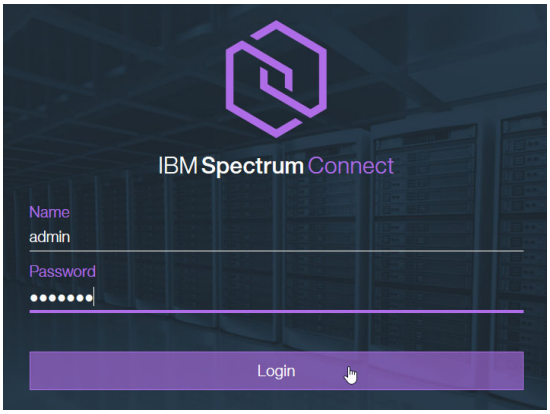


Figure 2-2 Log in to SC user interface

The IBM Spectrum Connect GUI consists of these four panes:

Interfaces	Integration with vCenter, vRO, Windows, and Docker servers
Spaces/Services	Handling storage spaces and services
Storage Systems	Management of storage systems and storage resources
Monitoring	Integration with vROps server

After a successful login, the Spaces/Services and Storage Systems panes are displayed. In the upper left and upper right corners are the **Navigation** arrows (< >) to navigate to the Applications and Monitoring panes.

If the IBM Spectrum Connect server is newly installed, following the first logon, an Initial Setup Wizard is started. Complete the following steps:

1. Complete the General settings form with the following parameters and click **Next**, as shown in Figure 2-3 on page 13:

FQDN	Fully qualified domain name of the IBM Spectrum Control server
HA GROUP	High availability group containing this IBM Spectrum Control server

   The first defined IBM Spectrum Connect server within a HA group is the active server. The second one is the standby.

   > **Note:** The HA Group feature is only used with VMware vSphere Virtual Volume (VVol). Although it is not relevant for containers, a default HA group is created automatically during Spectrum Connect installation and a group name is assigned automatically by Spectrum Connect.

   This feature is only used with VMware vSphere Virtual Volume (VVol). Although it is not relevant for containers, a default HA group is created automatically during Spectrum Connect installation and a group name is assigned automatically by Spectrum Connect.

For more information about the high availability feature, see the *IBM Spectrum Connect User Guide*, which is provided when you download IBM Spectrum Connect from IBM Fix Central.

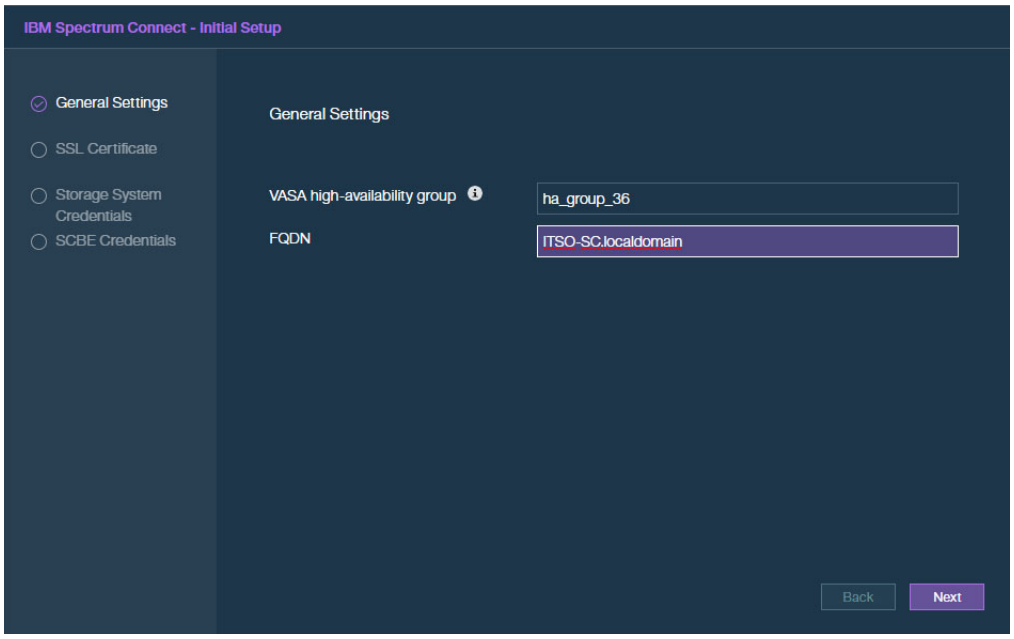


Figure 2-3 IBM Spectrum Connect Initial Setup: General Settings

2. A certificate normally already exists and is valid from the date of your configuration. Regardless, you must regenerate the certificate so that the FDQN previously specified is integrated in the certificate. Complete the fields on the form and click **Next**, as shown in Figure 2-4.

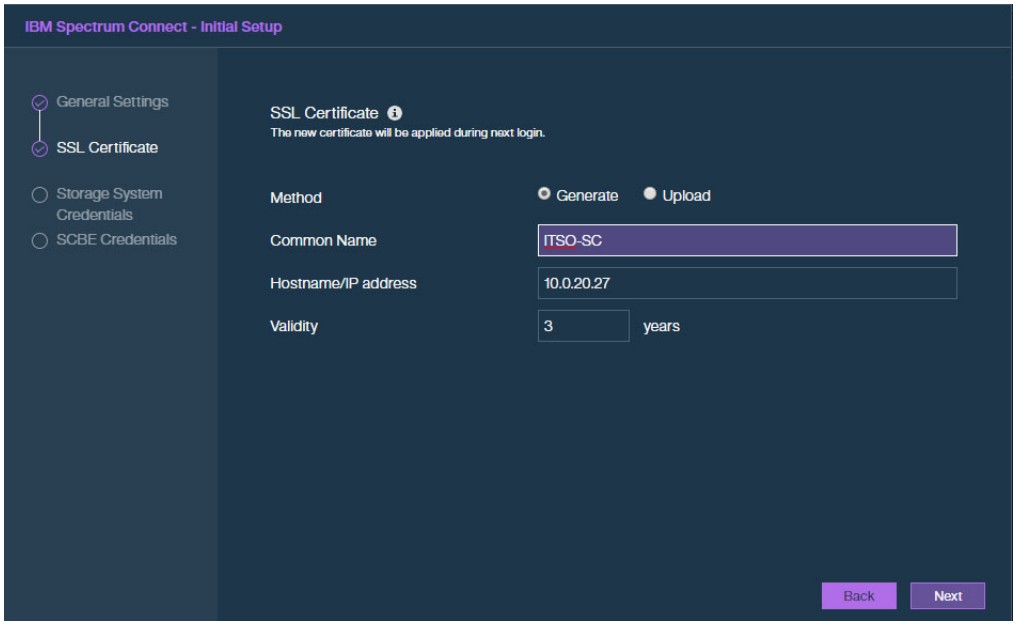


Figure 2-4 IBM Spectrum Connect Initial Setup: SSL Certificate

3. Set the storage credentials, which must be common to all of the storage devices that are connected to your IBM Spectrum Connect server and must have been previously created on IBM Spectrum Accelerate family systems. Click **Next**, as shown in Figure 2-5.

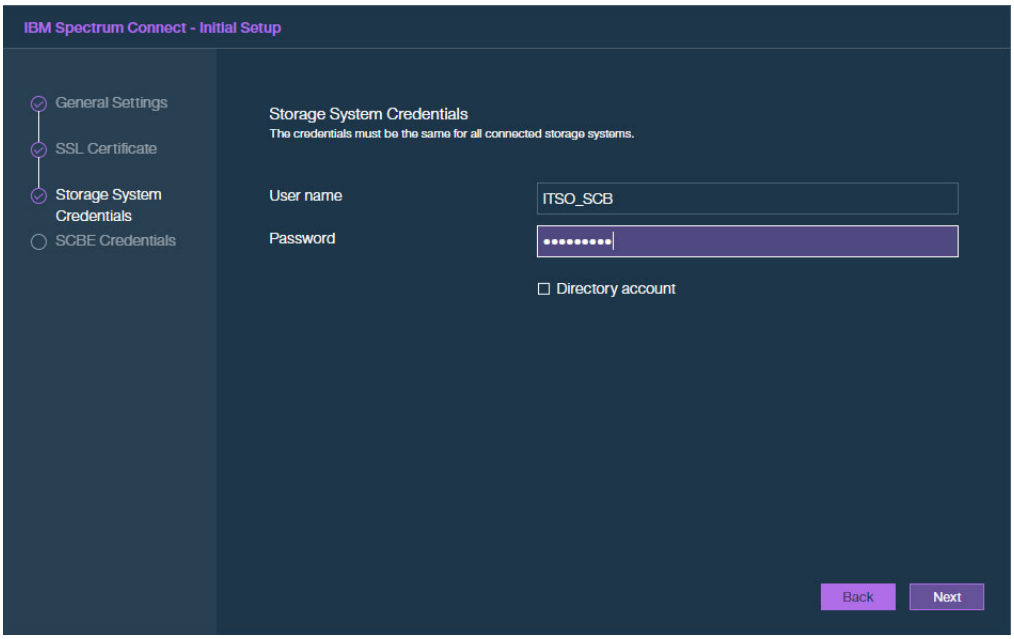


Figure 2-5 IBM Spectrum Connect Initial Setup: Storage System Credentials

Important information: The storage credentials must be common to all the storage devices that will be connected to your IBM Spectrum Connect server.

4. Change the password for the *admin* user and click **Finish**, as shown in Figure 2-6.

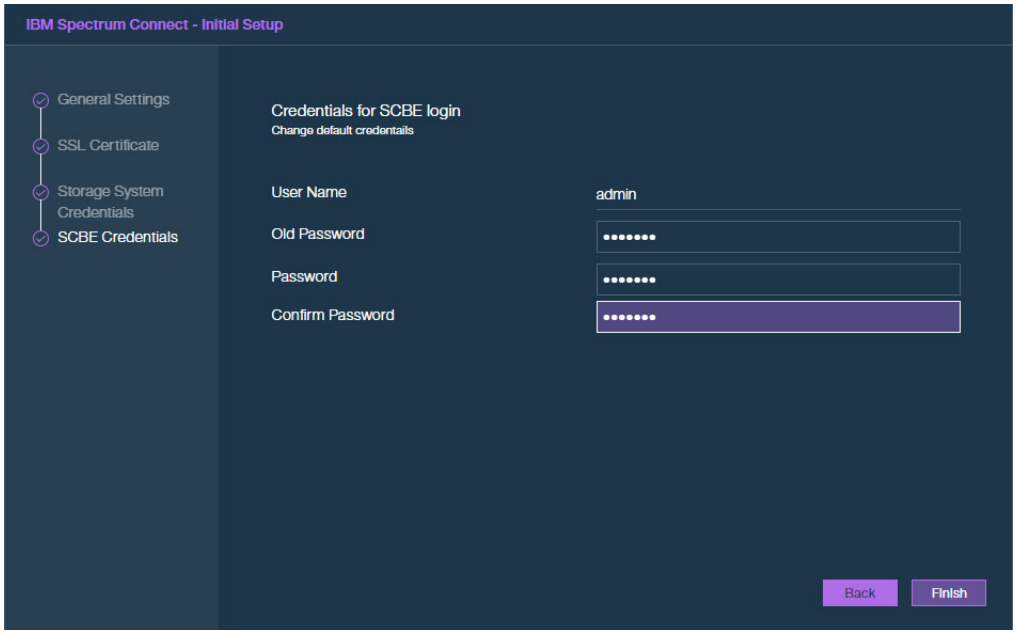


Figure 2-6 IBM Spectrum Connect Initial Setup: Change admin password

2.2.2 FlashSystem A9000 or A9000R domain configuration

As previously note (see Note box on page 11), defining domains is not required for containers. However, if domains were already defined for other reasons, then you must create a new domain for containers. On the FlashSystem A9000 used for our experimentation, at least one domain was already set, so we had to use domains.

With IBM FlashSystem A9000 and A9000R, multi tenancy is realized through the concept of *domains*. A domain is a logical partition in the system (more than one can exist in a system) with its own administrators and resources. This partitioning enables the secure isolation of users and resources among domains.

Domains enable the ability to control what object a user can perform actions on. A user is able to see and act upon resources only within the domains that they are associated with. When a domain user connects to a system by using either the GUI or CLI, they can access only the resources within their domains.

IBM FlashSystem A9000 and A9000R use a role-based access control (RBAC) model to control what actions that a user can perform.

Domains can be associated with the following objects:

- Users and user groups
- Storage pools and all of their content, which can be associated with only a single domain
- Hosts and clusters
- Remote mirror targets

Important: The implementation of domains occurs at the management and virtualization level. All physical system resources (IBM MicroLatency® modules, CPUs, memory, and interfaces) are shared among domains.

Domain administrators cannot modify physical system resources and they cannot access resources that belong to another domain, including the “global domain” resources of the system. However, they can be notified of events that relate to physical system attributes because these events might affect the objects within their domain. For example, a domain administrator can be alerted upon the failure of a MicroLatency module or the disconnection of an interface link.

For more details about multi tenancy and domains, see *IBM Hyper-Scale Manager for IBM Spectrum Accelerate Family: IBM XIV, IBM FlashSystem A9000 and A9000R, and IBM Spectrum Accelerate*, SG24-8376.

Domain definition

It is a simple process to create a domain with the HSM GUI on FlashSystem A9000 and A9000R:

1. Open Hyper-Scale Manager (`https://<Hyper-Scale Manager server IP address>:8443`) and select **New** from the top navigation pane to open the **CREATE NEW** menu with a listing of all of the objects that can be created within this menu.

2. By clicking **Domain**, as shown in Figure 2-7, a new wizard opens with all of the attribute fields for the domain.

Figure 2-7 Create a domain

3. Complete the required information for creating a domain, as shown in Figure 2-8.

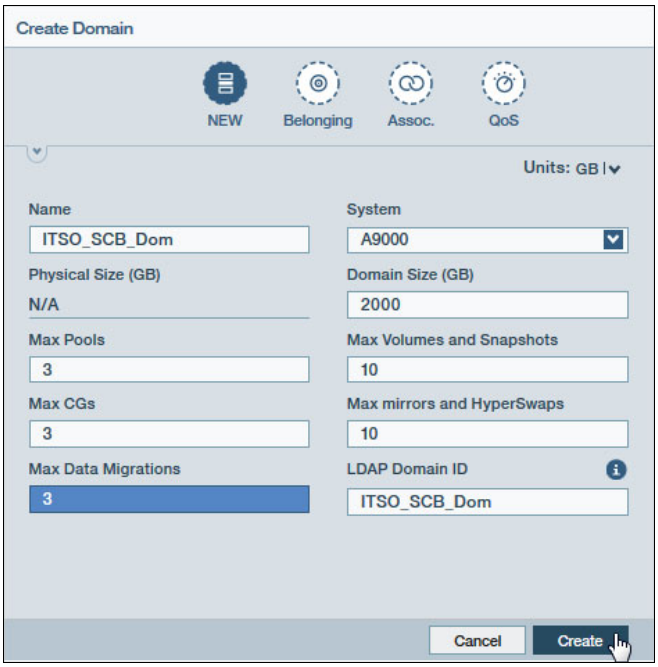


Figure 2-8 New domain attributes

In this example, we created a domain with a maximum capacity of 2000 GB on an IBM FlashSystem A9000 system.

The fields that are shown in Figure 2-8 on page 16 are described in detail in Table 2-2.

Table 2-1 Parameters and description for capabilities

Field name	Description
Name	The logical name of the domain. It must be unique within a system.
System	IBM XIV Gen3, IBM FlashSystem A9000 and A9000R, or IBM Spectrum Accelerate software where the domain will be defined.
Size (GB)	The capacity that is given to the domain that can be used to create pools within the domain, the minimum capacity is 2000 GB for IBM FlashSystem A9000 and A9000R.
Max Pools	The maximum number of pools that can be configured within this domain.
Max Volumes and Snapshots	The maximum number of volumes and snapshots that can be defined within this domain.
Max CGs	The maximum number of consistency groups that are allowed within this domain.
Max mirrors and IBM HyperSwap®	The maximum number of remote mirrors and HyperSwap operations that are allowed within this domain.
Max Data Migrations	The maximum number of active data migrations that can be defined within this domain at any time.
LDAP Domain ID	The ID attribute of the domain to use when you configure multitenancy with Lightweight Directory Access Protocol (LDAP) authentication. The default is set to the name of the domain.

Host definition

To define the Kubernetes cluster hosts on FlashSystem A9000 (in our case just one host), complete these steps:

1. From the main GUI Dashboard, click **New** → **Host** from the menu (Figure 2-9).

Figure 2-9 Create New Host menu

2. The new Host view is displayed. The Add Host setting window opens (Figure 2-10) where you provide necessary informations. Name of Host must be the same as the output of command **hostname** on Linux host shows.

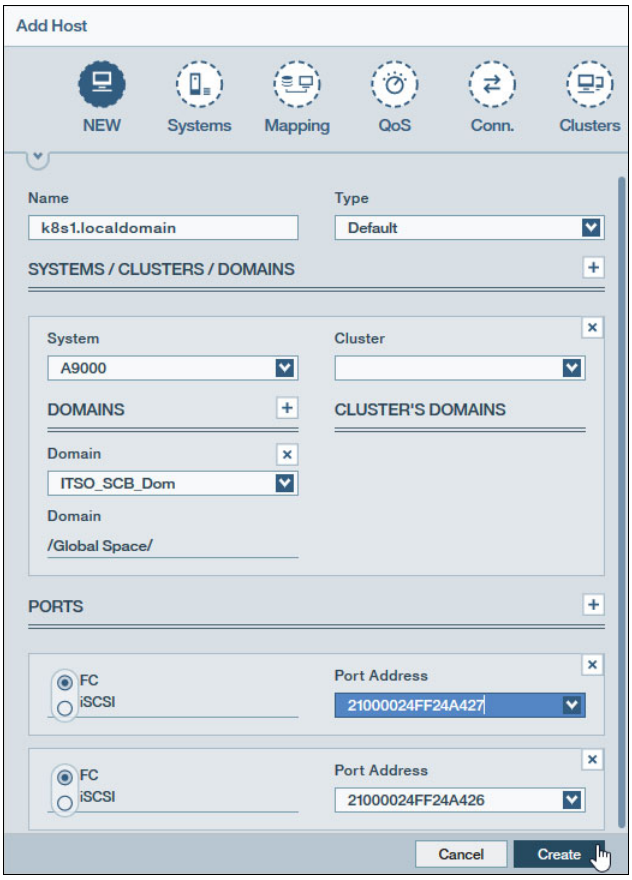


Figure 2-10 Add Host details

Enter the required information:

- Specify a host name (required).
- Select the type. In this example, **Default** is selected. For hosts, like Linux, the **Default** option is correct.
- Select whether you want an iSCSI or Fibre Channel (FC) connection. Host access to LUNs is granted depending on the host adapter ID. For an FC connection, the host adapter ID is the FC HBA WWPN. For an iSCSI connection, the host adapter ID is the host IQN. To add a WWPN or an IQN to a host definition, click the plus sign (**+**) icon to the right of the PORTS heading. Ports can be added in any order.
- Specify to which IBM FlashSystem A9000 or IBM FlashSystem A9000R in your inventory you want to attach the host. Alternatively, you can also add the host to a cluster. A cluster can be created from this view.
- To add a domain, click the plus sign (**+**) icon to the right of the DOMAINS heading. Select the domain.

For more information about host attachment, see *IBM FlashSystem A9000, IBM FlashSystem A9000R, and IBM XIV Storage System: Host Attachment and Interoperability*, SG24-8368.

User definition

The user definition process requires that you first log on to IBM Hyper-Scale Manager with storage administrator access rights (`storageadmin` role). If you are accessing the system for the first time, use the predefined user *admin* (the default password is `adminadmin`).

Complete these steps:

1. In the main GUI window, click **NEW+** in the upper-right corner of the window. The **CREATE NEW** menu opens, as shown in Figure 2-11.

Figure 2-11 Create New menu

2. Click **User**. The Add User window opens. The name, system, and password are required fields. The default category for a new user is Storage Administrator, which is the required one. Select the domain. Click **Create** to add the user (Figure 2-12).

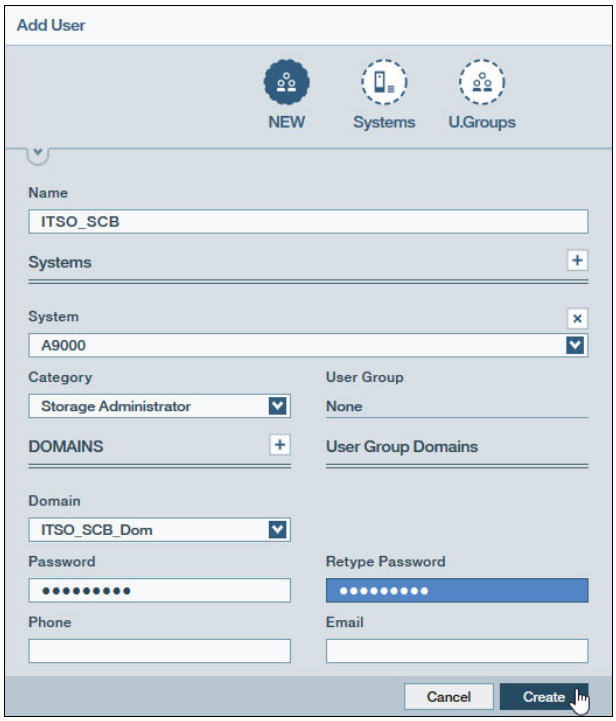


Figure 2-12 Add user window

For more details about user and user groups, see *IBM Hyper-Scale Manager for IBM Spectrum Accelerate Family: IBM XIV, IBM FlashSystem A9000 and A9000R, and IBM Spectrum Accelerate*, SG24-8376, and *IBM FlashSystem A9000 and IBM FlashSystem A9000R Architecture and Implementation*, SG24-8345.

2.2.3 Spectrum Connect configuration for containers

To configure Spectrum Connect for Containers, the following steps are necessary:

1. Add the IBM storage system as a storage array.
2. Create a Storage Space.
3. Create a Storage Service.
4. Assign Resource to Storage Service.
5. Add the container server.
6. Delegate Storage Service to container server.

Detailed tasks are as follows:

1. Open Spectrum Connect and add the FlashSystem A9000 or A9000R system, by completing the following steps:

   a. Click the plus sign (**+**) icon next to the Storage Systems pane to add the storage system to IBM Spectrum Control, as shown in Figure 2-13. If the system was defined before, proceed to step 2 on page 21.

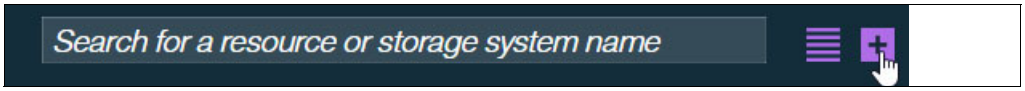


Figure 2-13 Open the Add New Array form

   b. Complete the Add New IBM Storage System form, specifying the IBM storage system Internet Protocol (IP) address or host name, as shown in Figure 2-14. Then, click **Add**.

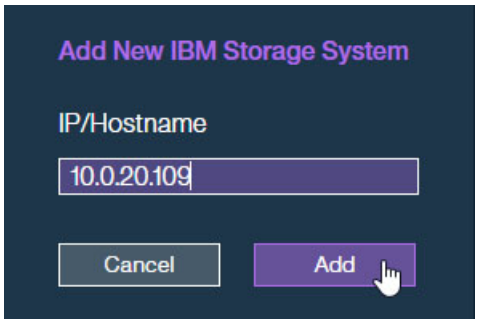


Figure 2-14 Add an IBM storage system as a storage array

c. The storage systems that are added to IBM Spectrum Connect are displayed in the Storage System pane that is shown in Figure 2-15.

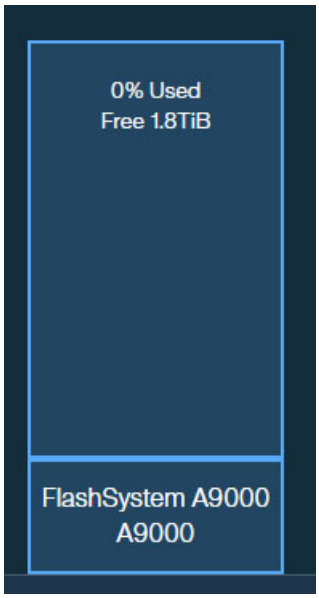


Figure 2-15 FlashSystem A9000 successfully added to the IBM Spectrum Connect server

2. Within the IBM Spectrum Connect server, virtual storage is defined with a *storage service* and a *storage space*.

   A storage service is a combination of assigned storage resources (pools) and user-defined policies (capabilities). The storage resources that are assigned to the service can reside on any storage system, as illustrated below. The policies are additional capabilities, or storage requirements for the service. For more information, see Table 2-2 on page 22.

   A storage space is logical grouping of several storage services. Usually, a single space is assigned to a specific organization (storage tenant).

   Complete the following steps to define a storage space:

   a. Select **Default_Space** from the drop-down box and select **Add New Storage Space** to add a storage space to IBM Spectrum Control, as shown in Figure 2-16.

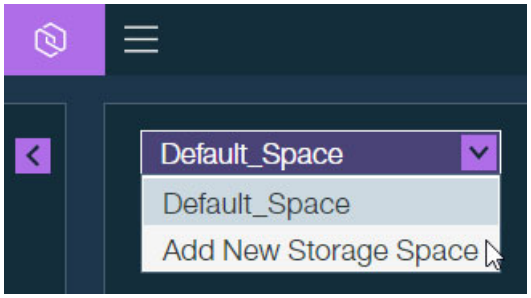


Figure 2-16 Add New Storage Space

b. Provide a meaningful name for your new storage space and click **Apply**, as shown in Figure 2-17.

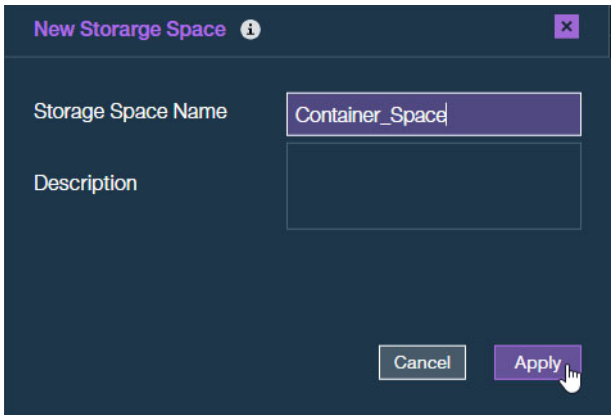


Figure 2-17 Add a storage space

c. Now that storage space is defined, a storage service must be configured. A storage service is the combination of storage resources and associated storage characteristics, such as encryption and thin provisioning (see Table 2-2 on page 22).

> **Important:** To link a Spectrum Connect storage service later to a Kubernetes storage class, make sure that the value of the `STORAGE_CLASS_PROFILE_VALUE` parameter in the `ubiquity_installer.conf` file is the same as the service name, as described in Example 2-23 on page 42.

3. To add a service to the newly created storage space, complete the following steps:

a. From the newly created storage space, click the **Storage Services +** icon next to the **Default_Space** drop down box.

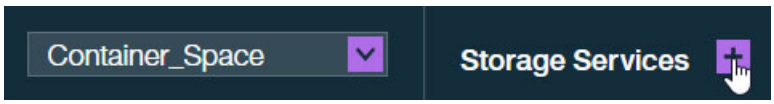


Figure 2-18 Add new storage service

b. When the New Service form opens, complete it as shown in Figure 2-19 on page 23. Specify features that are fulfilled by this service according to your needs. It is not necessary to select a capability, so the default capabilities will be used.

Table 2-2 explains the parameters and description for capabilities.

Table 2-2 Parameters and description for capabilities

Parameters	Description
Encryption	Enables encryption for the service. If enabled, you can attach only encrypted storage resources to the service.
Flash	Enables utilization of a storage resource on a flash-based storage resource, which can be one of the following storage systems: IBM FlashSystem 900, IBM FlashSystem V9000, or any of the Storwize family systems.

Parameters	Description
Space Efficiency	Enables storage space efficiency features for the service. When selected, you can configure the service to be attached to a thick- or thin-provisioned storage resource.
QoS	Enables the use of the Quality of Service (QoS) feature for the service. QoS is applicable to volumes (Max Independent Performance) or storage resources (Max Shared Performance), setting the IOPS and bandwidth limits within the following ranges: ► IOPS: 0 - 100000 ► Bandwidth (BW): 0 - 10000 MBps
Availability	Enables the use of IBM HA technology for highly available storage deployments of IBM SAN Volume controller (Stretched Cluster).
Data Reduction	Enables the use of IBM Real-time Compression™ for XIV, Spectrum Accelerate, and Spectrum Virtualize systems.
VVol Service	Enables virtual volume functionality for the service. The virtual volume functionality is supported by the IBM XIV (11.5.1 or later) and storage systems that run IBM Spectrum Virtualize software (7.6 or later). An XIV VVol-enabled service does not support IBM Real-time Compression.

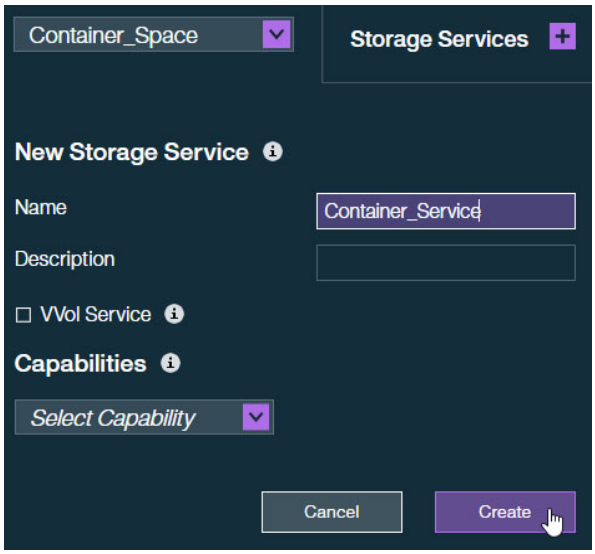


Figure 2-19 Add a service

c. Click **Add** to confirm the warning, as shown in Figure 2-20.

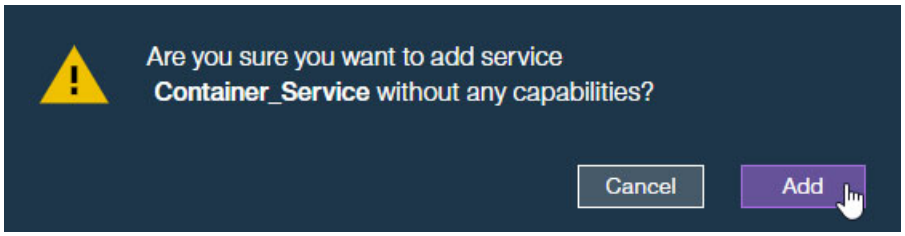


Figure 2-20 Warning: Service without capability

4. To add a storage resource for this service, complete the following steps:
   a. Right-click the service and select **Manage Resources**, as shown in Figure 2-21.

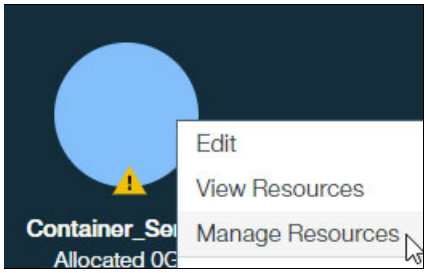


Figure 2-21 Service Manage Resources

   b. Click the **+** icon to add a resource, as shown in Figure 2-22. A resource can only belong to one service.

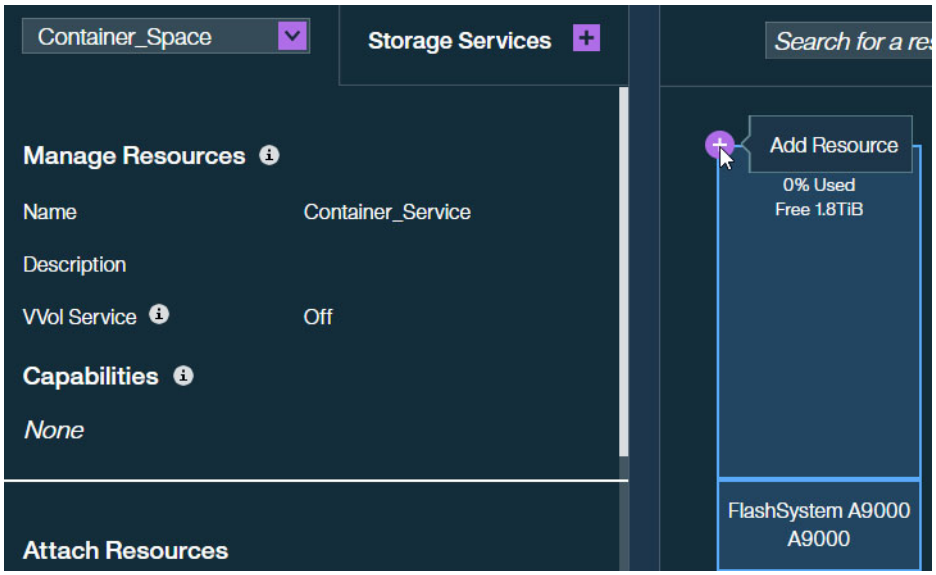


Figure 2-22 Service Add Resource

c. Enter the appropriate details to add a new storage resource and click **Add**, as shown in Figure 2-23. Those credentials are used by the IBM Storage Enabler for Containers to gain access to Spectrum Connect.

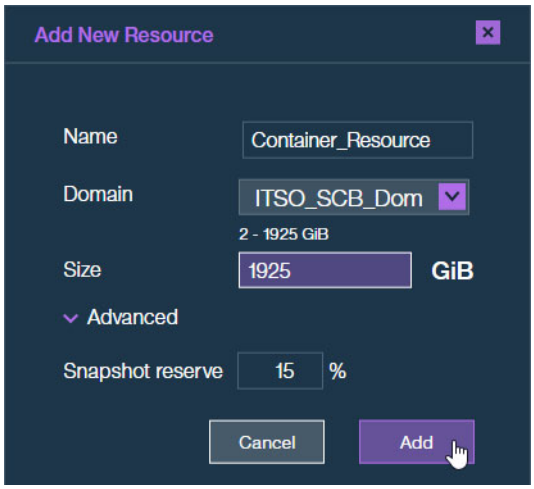


Figure 2-23 Service Add New Resource

d. Confirm creation and attachment of the resource by clicking **Ok**, as shown in Figure 2-24.

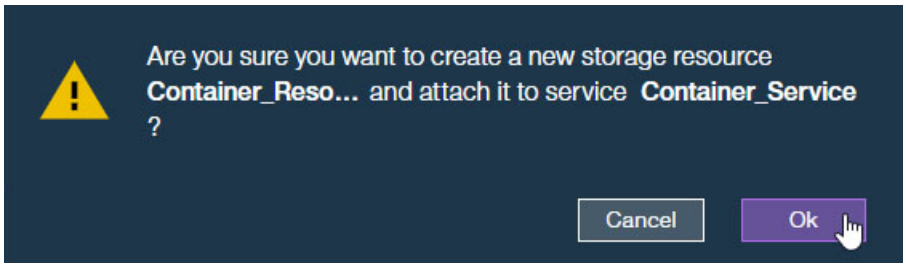


Figure 2-24 Resource confirmation

5. To add an IBM Storage Enabler for Containers interface, complete the following steps:

a. Click the left-arrow navigation pointer, as shown in Figure 2-25.

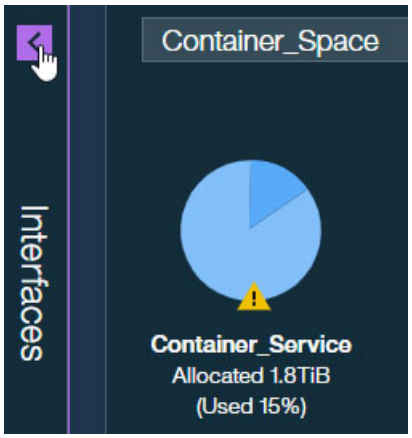


Figure 2-25 Interface pane

b. In the Interfaces pane, select **Add Interface** → **Enabler for Containers**, as shown in Figure 2-26.

Figure 2-26 Add a container to IBM Spectrum Connect

c. Enter the credentials, which will be used by Enable for Containers to connect to Spectrum Connect, as shown in Figure 2-27, and then click **Apply**.

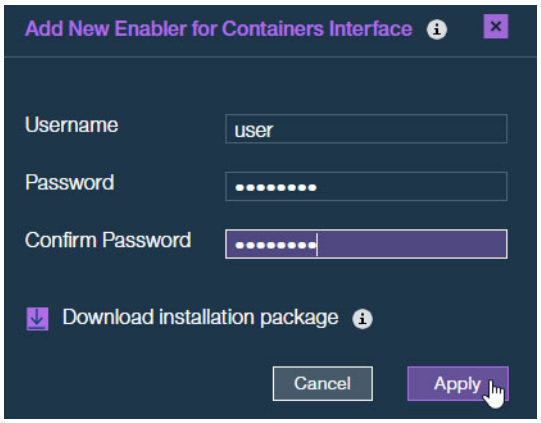


Figure 2-27 Add a container server

d. Delegate the service to the container by clicking the container and then the **Delegate** icon, as shown in Figure 2-28.

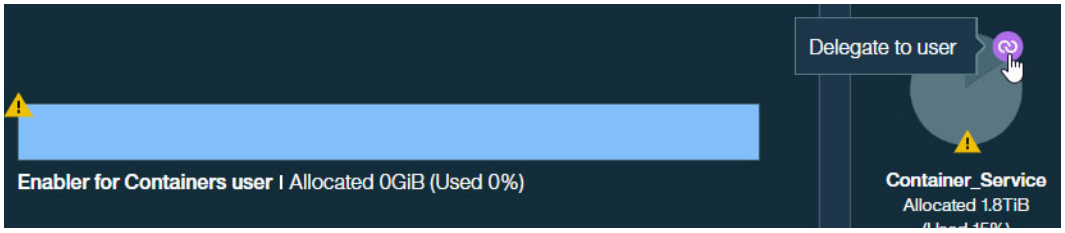


Figure 2-28 Delegate service to container

e. Click **Ok** to confirm, as shown in Figure 2-29.

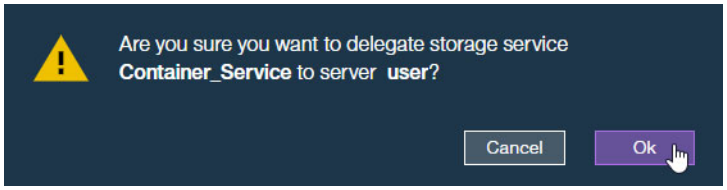


Figure 2-29 Delegate service to container confirmation

2.3 Docker installation and configuration

Docker Community Edition (CE) is ideal to get started with Docker and experimenting with container-based apps. Docker CE provides an installer for a simple and quick installation.

For the following examples, we used Docker CE on RedHat Enterprise Linux.

> **Note:** For support and in production environments, you must use Docker Enterprise Edition (EE).

Proceed as follows:

1. Download Docker CE, available at the Docker website.
2. Run the **yum install** command, as shown in the truncated output in Example 2-1.

Example 2-1 Install Docker CE

```
[root@k8s1 ~]# yum install -y docker-ce-17.03.2.ce-1.el7.centos.x86_64.rpm
docker-ce-selinux-17.03.2.ce-1.el7.centos.noarch.rpm
Loaded plugins: product-id, search-disabled-repos, subscription-manager
This system is not registered to Red Hat Subscription Management. You can use
subscription-manager to register.
Examining docker-ce-17.03.2.ce-1.el7.centos.x86_64.rpm:
docker-ce-17.03.2.ce-1.el7.centos.x86_64
Marking docker-ce-17.03.2.ce-1.el7.centos.x86_64.rpm to be installed
Examining docker-ce-selinux-17.03.2.ce-1.el7.centos.noarch.rpm:
docker-ce-selinux-17.03.2.ce-1.el7.centos.noarch
Marking docker-ce-selinux-17.03.2.ce-1.el7.centos.noarch.rpm to be installed
Resolving Dependencies
--> Running transaction check
---> Package docker-ce.x86_64 0:17.03.2.ce-1.el7.centos will be installed
--> Processing Dependency: libcgroup for package:
docker-ce-17.03.2.ce-1.el7.centos.x86_64
rhel-dvd
| 4.1 kB  00:00:00
(1/2): rhel-dvd/group_gz
| 136 kB  00:00:00
(2/2): rhel-dvd/primary_db
| 3.9 MB  00:00:06
--> Processing Dependency: libltdl.so.7()(64bit) for package:
docker-ce-17.03.2.ce-1.el7.centos.x86_64
--> Processing Dependency: libseccomp.so.2()(64bit) for package:
docker-ce-17.03.2.ce-1.el7.centos.x86_64
---> Package docker-ce-selinux.noarch 0:17.03.2.ce-1.el7.centos will be
installed
--> Processing Dependency: policycoreutils-python for package:
docker-ce-selinux-17.03.2.ce-1.el7.centos.noarch
--> Running transaction check
---> Package libcgroup.x86_64 0:0.41-11.el7 will be installed
---> Package libseccomp.x86_64 0:2.3.1-2.el7 will be installed
---> Package libtool-ltdl.x86_64 0:2.4.2-21.el7_2 will be installed
---> Package policycoreutils-python.x86_64 0:2.5-8.el7 will be installed
--> Processing Dependency: audit-libs-python >= 2.1.3-4 for package:
policycoreutils-python-2.5-8.el7.x86_64
```

```
--> Processing Dependency: libsemanage-python >= 2.5-4 for package:
policycoreutils-python-2.5-8.el7.x86_64
--> Processing Dependency: setools-libs >= 3.3.8-1 for package:
policycoreutils-python-2.5-8.el7.x86_64
--> Processing Dependency: checkpolicy for package:
policycoreutils-python-2.5-8.el7.x86_64
--> Processing Dependency: libapol.so.4(VERS_4.0)(64bit) for package:
policycoreutils-python-2.5-8.el7.x86_64
--> Processing Dependency: libqpol.so.1(VERS_1.2)(64bit) for package:
policycoreutils-python-2.5-8.el7.x86_64
--> Processing Dependency: libqpol.so.1(VERS_1.4)(64bit) for package:
policycoreutils-python-2.5-8.el7.x86_64
--> Processing Dependency: python-IPy for package:
policycoreutils-python-2.5-8.el7.x86_64
--> Processing Dependency: libapol.so.4()(64bit) for package:
policycoreutils-python-2.5-8.el7.x86_64
--> Processing Dependency: libqpol.so.1()(64bit) for package:
policycoreutils-python-2.5-8.el7.x86_64
--> Running transaction check
---> Package audit-libs-python.x86_64 0:2.6.5-3.el7 will be installed
---> Package checkpolicy.x86_64 0:2.5-4.el7 will be installed
---> Package libsemanage-python.x86_64 0:2.5-4.el7 will be installed
---> Package python-IPy.noarch 0:0.75-6.el7 will be installed
---> Package setools-libs.x86_64 0:3.3.8-1.1.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
==============================
 Package                       Arch              Version
Repository                                                                Size
================================================================================
============================
Installing:
 docker-ce                                  x86_64
17.03.2.ce-1.el7.centos
/docker-ce-17.03.2.ce-1.el7.centos.x86_64                                 65 M
 docker-ce-selinux                          noarch
17.03.2.ce-1.el7.centos
/docker-ce-selinux-17.03.2.ce-1.el7.centos.noarch                         43 k
Installing for dependencies:
 audit-libs-python                          x86_64
2.6.5-3.el7                      rhel-dvd
70 k
 checkpolicy                   x86_64            2.5-4.el7
rhel-dvd                                                                 290 k
 libcgroup                                  x86_64
0.41-11.el7                      rhel-dvd
65 k
 libseccomp                                 x86_64
2.3.1-2.el7                      rhel-dvd
56 k
 libsemanage-python              x86_64            2.5-4.el7
rhel-dvd                                                                 103 k
```

```
 libtool-ltdl                              x86_64
2.4.2-21.el7_2                   rhel-dvd
49 k
 policycoreutils-python           x86_64           2.5-8.el7
rhel-dvd                                                          444 k
 python-IPy                    noarch            0.75-6.el7
rhel-dvd                                                           32 k
 setools-libs                              x86_64
3.3.8-1.1.el7                    rhel-dvd
610 k

Transaction Summary
================================================================================
================================================================================
================================
Install  2 Packages (+9 Dependent packages)

Total size: 66 M
Total download size: 1.7 M
Installed size: 70 M
Downloading packages:
.................................................................
Installed:
docker-ce.x86_64 0:17.03.2.ce-1.el7.centos
docker-ce-selinux.noarch 0:17.03.2.ce-1.el7.centos

Dependency Installed:
  audit-libs-python.x86_64 0:2.6.5-3.el7 checkpolicy.x86_64 0:2.5-4.el7
libcgroup.x86_64 0:0.41-11.el7 libseccomp.x86_64 0:2.3.1-2.el7
libsemanage-python.x86_64 0:2.5-4.el7
  libtool-ltdl.x86_64 0:2.4.2-21.el7_2   policycoreutils-python.x86_64
0:2.5-8.el7 python-IPy.noarch 0:0.75-6.el7 setools-libs.x86_64 0:3.3.8-1.1.el7

Complete!
```

3. Enable Docker service to start after reboot and start it directly, as shown in Example 2-2.

Example 2-2 Enable Docker automatic start after reboot

```
[root@k8s1 ~]# systemctl enable docker && systemctl start docker
Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service
to /usr/lib/systemd/system/docker.service.
```

For a complete list of Docker CLI commands, see Use the Docker command line on the Docker website.

2.4 Kubernetes installation and configuration

Kubernetes (k8s) is an open source platform that orchestrates Linux container frameworks like Docker. It consists of one or more master nodes and several worker nodes. We introduced it in “Managing containers with Kubernetes” on page 5. The architecture of Kubernetes is shown in Figure 2-30.

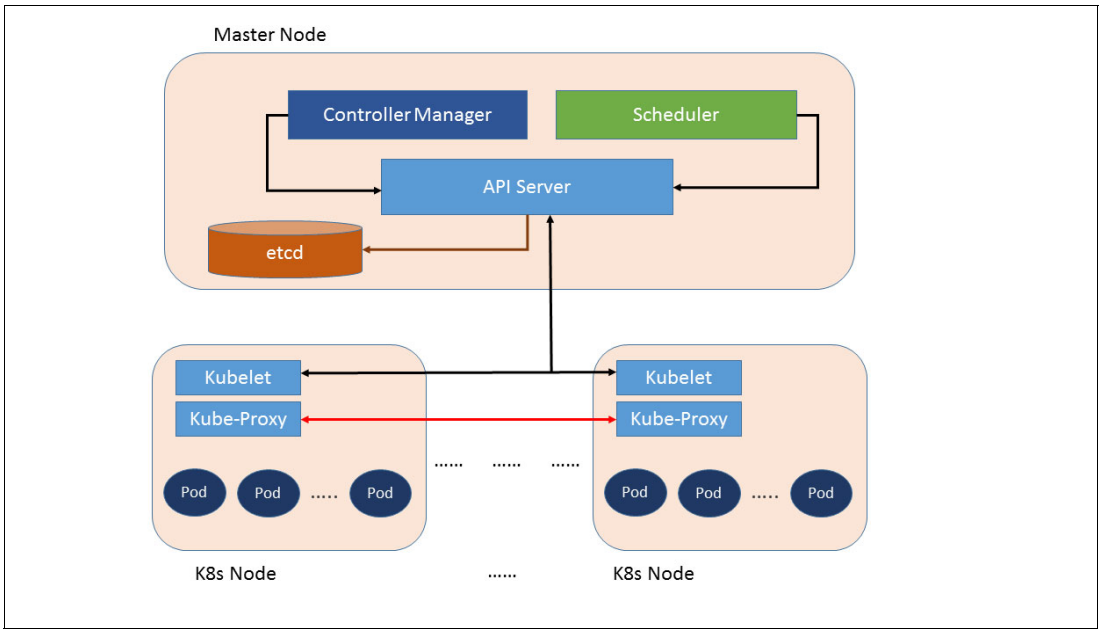


Figure 2-30 Kubernetes architecture

The master node can serve as worker node as well. A tool called minkube runs a single node k8s cluster for development and test reasons. In the following examples, for brevity a single node cluster with v1.7.11 was used.

More details about Kubernetes can be found at the Kubernetes website.

Because the systems used for our experimentation were behind a firewall in a protected lab environment, the k8s node in our example has no access to the internet. We chose to document the installation as such because production servers and systems are often behind firewalls in real environments. In this case, you need an additional VM with Docker and internet access to download the Kubernetes packages and all necessary Docker images. For more information, see 2.5, “IBM Storage Enabler for Containers installation and configuration” on page 38.

To install and configure Kubernetes, complete these steps:

1. Create a repository on the additional VM with **`vi /etc/yum.repos.d/kubernetes.repo`** and enter the necessary information as shown in Example 2-3.

Example 2-3 Kubernetes repository

```
[kubernetes]
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86_64
enabled=1
gpgcheck=1
```

```
repo_gpgcheck=1
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg
https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
```

2. Download the k8s packages by using the **yum** command, as illustrated in Example 2-4.

Example 2-4 yum download

```
[root@QW2 ~]# yum install --downloadonly --downloaddir=/root kubelet-1.7.11
kubeadm-1.7.11 kubectl-1.7.11
Loaded plugins: product-id, search-disabled-repos, subscription-manager
This system is not registered to Red Hat Subscription Management. You can use
subscription-manager to register.
Resolving Dependencies
--> Running transaction check
---> Package kubeadm.x86_64 0:1.7.11-0 will be installed
--> Processing Dependency: kubernetes-cni for package: kubeadm-1.7.11-0.x86_64
---> Package kubectl.x86_64 0:1.7.11-0 will be installed
---> Package kubelet.x86_64 0:1.7.11-0 will be installed
--> Running transaction check
---> Package kubernetes-cni.x86_64 0:0.5.1-1 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
================================================================================
==========
 Package                                    Arch
Version                                  Repository
Size
================================================================================
================================================================================
==========
Installing:
 kubeadm                                    x86_64
1.7.11-0                                kubernetes
8.6 M
 kubectl                                    x86_64
1.7.11-0                                kubernetes
8.9 M
 kubelet                                    x86_64
1.7.11-0                                kubernetes
17 M
Installing for dependencies:
 kubernetes-cni                             x86_64
0.5.1-1                                 kubernetes
7.4 M

Transaction Summary
================================================================================
================================================================================
==========
Install  3 Packages (+1 Dependent package)

Total size: 41 M
```

```
Total download size: 34 M
Installed size: 222 M
Background downloading packages, then exiting:
(1/3):
b10ac8cf7ee52d4f4144b523b1f33061b7429bea3fcf9e863261423de090804a-kubeadm-1.7.11
-0.x86_64.rpm                                              | 8.6 MB  00:00:01
(2/3):
7da91dc73456ad61f773fdd893a96b22b9537da395658b29c2cbda376f701f50-kubectl-1.7.11
-0.x86_64.rpm                                              | 8.9 MB  00:00:02
(3/3):
57fc3fb190b0f538f1f6b109f0b23f3456bc48aae3e2ceac5041c6438aeb6a50-kubelet-1.7.11
-0.x86_64.rpm                                              |  17 MB  00:00:01
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------
Total
12 MB/s |  34 MB  00:00:02
exiting because "Download Only" specified
```

3. Transfer the packages to the k8s host and use the **yum** command to install them as displayed in Example 2-5.

Example 2-5 yum installation

```
[root@k8s1 ~]# yum install -y
b10ac8cf7ee52d4f4144b523b1f33061b7429bea3fcf9e863261423de090804a-kubeadm-1.7.11
-0.x86_64.rpm
7da91dc73456ad61f773fdd893a96b22b9537da395658b29c2cbda376f701f50-kubectl-1.7.11
-0.x86_64.rpm
57fc3fb190b0f538f1f6b109f0b23f3456bc48aae3e2ceac5041c6438aeb6a50-kubelet-1.7.11
-0.x86_64.rpm
79f9ba89dbe7000e7dfeda9b119f711bb626fe2c2d56abeb35141142cda00342-kubernetes-cni
-0.5.1-1.x86_64.rpm
Loaded plugins: product-id, search-disabled-repos, subscription-manager
This system is not registered to Red Hat Subscription Management. You can use
subscription-manager to register.
Examining
b10ac8cf7ee52d4f4144b523b1f33061b7429bea3fcf9e863261423de090804a-kubeadm-1.7.11
-0.x86_64.rpm: kubeadm-1.7.11-0.x86_64
Marking
b10ac8cf7ee52d4f4144b523b1f33061b7429bea3fcf9e863261423de090804a-kubeadm-1.7.11
-0.x86_64.rpm to be installed
Examining
7da91dc73456ad61f773fdd893a96b22b9537da395658b29c2cbda376f701f50-kubectl-1.7.11
-0.x86_64.rpm: kubectl-1.7.11-0.x86_64
Marking
7da91dc73456ad61f773fdd893a96b22b9537da395658b29c2cbda376f701f50-kubectl-1.7.11
-0.x86_64.rpm to be installed
Examining
57fc3fb190b0f538f1f6b109f0b23f3456bc48aae3e2ceac5041c6438aeb6a50-kubelet-1.7.11
-0.x86_64.rpm: kubelet-1.7.11-0.x86_64
Marking
57fc3fb190b0f538f1f6b109f0b23f3456bc48aae3e2ceac5041c6438aeb6a50-kubelet-1.7.11
-0.x86_64.rpm to be installed
```

```
Examining
79f9ba89dbe7000e7dfeda9b119f711bb626fe2c2d56abeb35141142cda00342-kubernetes-cni
-0.5.1-1.x86_64.rpm: kubernetes-cni-0.5.1-1.x86_64
Marking
79f9ba89dbe7000e7dfeda9b119f711bb626fe2c2d56abeb35141142cda00342-kubernetes-cni
-0.5.1-1.x86_64.rpm to be installed
Resolving Dependencies
--> Running transaction check
---> Package kubeadm.x86_64 0:1.7.11-0 will be installed
---> Package kubectl.x86_64 0:1.7.11-0 will be installed
---> Package kubelet.x86_64 0:1.7.11-0 will be installed
--> Processing Dependency: socat for package: kubelet-1.7.11-0.x86_64
---> Package kubernetes-cni.x86_64 0:0.5.1-1 will be installed
--> Running transaction check
---> Package socat.x86_64 0:1.7.2.2-5.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
================================================================================
================================
 Package                    Arch               Version
Repository
Size
================================================================================
================================================================================
================================
Installing:
 kubeadm                    x86_64             1.7.11-0
/b10ac8cf7ee52d4f4144b523b1f33061b7429bea3fcf9e863261423de090804a-kubeadm-1.7.1
1-0.x86_64                    49 M
 kubectl                    x86_64             1.7.11-0
/7da91dc73456ad61f773fdd893a96b22b9537da395658b29c2cbda376f701f50-kubectl-1.7.1
1-0.x86_64                    49 M
 kubelet                    x86_64             1.7.11-0
/57fc3fb190b0f538f1f6b109f0b23f3456bc48aae3e2ceac5041c6438aeb6a50-kubelet-1.7.1
1-0.x86_64                    96 M
 kubernetes-cni             x86_64             0.5.1-1
/79f9ba89dbe7000e7dfeda9b119f711bb626fe2c2d56abeb35141142cda00342-kubernetes-cn
i-0.5.1-1.x86_64              28 M
Installing for dependencies:
 socat                      x86_64             1.7.2.2-5.el7
rhel-dvd
255 k

Transaction Summary
================================================================================
================================================================================
================================
Install  4 Packages (+1 Dependent package)

Total size: 222 M
Total download size: 255 k
Installed size: 223 M
```

```
..................................................
Installed:
  kubeadm.x86_64 0:1.7.11-0              kubectl.x86_64 0:1.7.11-0
kubelet.x86_64 0:1.7.11-0                    kubernetes-cni.x86_64 0:0.5.1-1

Dependency Installed:
  socat.x86_64 0:1.7.2.2-5.el7

Complete!
```

4. Pull all the necessary docker images (for Kubernetes with pod networks) on the VM with internet access. The commands are shown in Example 2-6. The following steps that involve pulling, saving, and loading the images are not necessary if the k8s nodes have internet access. It is an offline installation and configuration.

Example 2-6 Kubernetes image pull

```
[root@QW2 ~]# docker pull gcr.io/google_containers/pause-amd64:3.0
[root@QW2 ~]# docker pull gcr.io/google_containers/etcd-amd64:3.0.17
[root@QW2 ~]# docker pull
gcr.io/google_containers/kube-controller-manager-amd64:v1.7.11
[root@QW2 ~]# docker pull gcr.io/google_containers/kube-scheduler-amd64:v1.7.11
[root@QW2 ~]# docker pull gcr.io/google_containers/kube-apiserver-amd64:v1.7.11
[root@QW2 ~]# docker pull gcr.io/google_containers/kube-proxy-amd64:v1.7.11
[root@QW2 ~]# docker pull quay.io/calico/node:v2.6.2
[root@QW2 ~]# docker pull quay.io/coreos/flannel:v0.9.1
[root@QW2 ~]# docker pull quay.io/calico/cni:v1.11.0
[root@QW2 ~]# docker pull
gcr.io/google_containers/k8s-dns-kube-dns-amd64:1.14.5
[root@QW2 ~]# docker pull
gcr.io/google_containers/k8s-dns-dnsmasq-nanny-amd64:1.14.5
[root@QW2 ~]# docker pull gcr.io/google_containers/k8s-dns-sidecar-amd64:1.14.5
```

5. Save all the images as tar files as shown in Example 2-7.

Example 2-7 Docker save

```
[root@QW2 ~]# docker save gcr.io/google_containers/etcd-amd64:3.0.17 -o
/tmp/etcd-amd64.tar
[root@QW2 ~]# docker save gcr.io/google_containers/pause-amd64:3.0 -o
/tmp/pause-amd64.tar
[root@QW2 ~]# docker save
gcr.io/google_containers/kube-controller-manager-amd64:v1.7.11 -o
/tmp/kube-controller-manager-amd64-v1711.tar
[root@QW2 ~]# docker save gcr.io/google_containers/kube-scheduler-amd64:v1.7.11
-o /tmp/kube-scheduler-amd64-v1711.tar
[root@QW2 ~]# docker save gcr.io/google_containers/kube-apiserver-amd64:v1.7.11
-o /tmp/kube-apiserver-amd64-v1711.tar
[root@QW2 ~]# docker save gcr.io/google_containers/kube-proxy-amd64:v1.7.11 -o
/tmp/kube-proxy-amd64-v1711.tar
[root@QW2 ~]# docker save quay.io/calico/node:v2.6.2 -o /tmp/node.tar
[root@QW2 ~]# docker save quay.io/coreos/flannel:v0.9.1 -o /tmp/flannel.tar
[root@QW2 ~]# docker save quay.io/calico/cni:v1.11.0 -o /tmp/cni.tar
[root@QW2 ~]# docker save
gcr.io/google_containers/k8s-dns-kube-dns-amd64:1.14.5 -o
/tmp/k8s-dns-kube-dns-amd64.tar
```

```
[root@QW2 ~]# docker save
gcr.io/google_containers/k8s-dns-dnsmasq-nanny-amd64:1.14.5 -o
/tmp/k8s-dns-dnsmasq-nanny-amd64.tar
[root@QW2 ~]# docker save gcr.io/google_containers/k8s-dns-sidecar-amd64:1.14.5
-o /tmp/k8s-dns-sidecar-amd64.tar
```

6. Transfer the tar packages to the Kubernetes node and load the images into docker with **docker** commands, as illustrated in Example 2-8.

Example 2-8 docker load

```
[root@k8s1 ~]# docker load -i kube-controller-manager-amd64-v1711.tar
[root@k8s1 ~]# docker load -i kube-scheduler-amd64-v1711.tar
[root@k8s1 ~]# docker load -i kube-apiserver-amd64-v1711.tar
[root@k8s1 ~]# docker load -i pause-amd64.tar
[root@k8s1 ~]# docker load -i etcd-amd64.tar
[root@k8s1 ~]# docker load -i kube-proxy-amd64-v1711.tar
[root@k8s1 ~]# docker load -i node.tar
[root@k8s1 ~]# docker load -i cni.tar
[root@k8s1 ~]# docker load -i flannel.tar
[root@k8s1 ~]# docker load -i k8s-dns-kube-dns-amd64.tar
[root@k8s1 ~]# docker load -i k8s-dns-dnsmasq-nanny-amd64.tar
[root@k8s1 ~]# docker load -i k8s-dns-sidecar-amd64.tar
```

7. Disable `selinux` with **`setenforce 0`** and open the firewall ports, as displayed in Example 2-9.

Example 2-9 setenforce and firewall-cmd

```
[root@k8s1 ~]# setenforce 0
[root@k8s1 ~]# firewall-cmd --permanent --add-port=6443/tcp
success
[root@k8s1 ~]# firewall-cmd --permanent --add-port=10250/tcp
success
[root@k8s1 ~]# firewall-cmd --reload
Success
```

8. Run **`kubeadm init`** to install the cluster database and control plane components, as shown in Example 2-10.

Example 2-10 kubeadm init

```
[root@k8s1 ~]# kubeadm init --kubernetes-version=v1.7.11
--pod-network-cidr=10.244.0.0/16 --apiserver-advertise-address 10.0.20.29
[kubeadm] WARNING: kubeadm is in beta, please do not use it for production
clusters.
[init] Using Kubernetes version: v1.7.11
[init] Using Authorization modes: [Node RBAC]
[preflight] Running pre-flight checks
[preflight] WARNING: docker version is greater than the most recently validated
version. Docker version: 17.03.2-ce. Max validated version: 1.12
[preflight] WARNING: hostname "k8s1.localdomain" could not be reached
[preflight] WARNING: hostname "k8s1.localdomain" lookup k8s1.localdomain on
[::1]:53: read udp [::1]:53482->[::1]:53: read: connection refused
[preflight] WARNING: firewalld is active, please ensure ports [6443 10250] are
open or your cluster may not function correctly
[preflight] Starting the kubelet service
```

```
[kubeadm] WARNING: starting in 1.8, tokens expire after 24 hours by default (if
you require a non-expiring token use --token-ttl 0)
[certificates] Generated CA certificate and key.
[certificates] Generated API server certificate and key.
[certificates] API Server serving cert is signed for DNS names
[k8s1.localdomain kubernetes kubernetes.default kubernetes.default.svc
kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 10.0.20.29]
[certificates] Generated API server kubelet client certificate and key.
[certificates] Generated service account token signing key and public key.
[certificates] Generated front-proxy CA certificate and key.
[certificates] Generated front-proxy client certificate and key.
[certificates] Valid certificates and keys now exist in "/etc/kubernetes/pki"
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/admin.conf"
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/kubelet.conf"
[kubeconfig] Wrote KubeConfig file to disk:
"/etc/kubernetes/controller-manager.conf"
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/scheduler.conf"
[apiclient] Created API client, waiting for the control plane to become ready
[apiclient] All control plane components are healthy after 27.001037 seconds
[token] Using token: 61585a.feb02a24df5afd44
[apiconfig] Created RBAC rules
[addons] Applied essential addon: kube-proxy
[addons] Applied essential addon: kube-dns

Your Kubernetes master has initialized successfully!

To start using your cluster, you need to run (as a regular user):

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  http://kubernetes.io/docs/admin/addons/

You can now join any number of machines by running the following on each node
as root:

  kubeadm join --token 61585a.feb02a24df5afd44 10.0.20.29:6443
```

9. To make `kubectl` work, run the commands shown in Example 2-11.

Example 2-11 Configuration for kubectl

```
[root@k8s1 ~]# mkdir -p $HOME/.kube
[root@k8s1 ~]# cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
[root@k8s1 ~]# chown $(id -u):$(id -g) $HOME/.kube/config
[root@k8s1 ~]# export KUBECONFIG=/etc/kubernetes/admin.conf
```

10. To use canal as pod network download the following two yaml files: `rbac.yaml` and `canal.yaml`.

11. Apply both yaml files with **kubectl** commands, as displayed in Example 2-12.

Example 2-12 kubectl apply

```
[root@k8s1 ~]# kubectl apply -f rbac.yaml
clusterrole "calico" created
clusterrole "flannel" created
clusterrolebinding "canal-flannel" created
clusterrolebinding "canal-calico" created
[root@k8s1 ~]# kubectl apply -f canal.yaml
configmap "canal-config" created
daemonset "canal" created
customresourcedefinition "globalfelixconfigs.crd.projectcalico.org" created
customresourcedefinition "globalbgpconfigs.crd.projectcalico.org" created
customresourcedefinition "ippools.crd.projectcalico.org" created
customresourcedefinition "globalnetworkpolicies.crd.projectcalico.org" created
serviceaccount "canal" created
```

12. To untaint the master to be a worker node and to join the cluster, use the commands shown in Example 2-13.

Example 2-13 kubectl taint and kubeadm join

```
[root@k8s1 ~]# kubectl taint nodes --all node-role.kubernetes.io/master-
node "k8s1.localdomain" untainted
[root@k8s1 ~]# kubeadm join --skip-preflight-checks --token
61585a.feb02a24df5afd44 10.0.20.29:6443
[kubeadm] WARNING: kubeadm is in beta, please do not use it for production
clusters.
[preflight] Skipping pre-flight checks
[discovery] Trying to connect to API Server "10.0.20.29:6443"
[discovery] Created cluster-info discovery client, requesting info from
"https://10.0.20.29:6443"
[discovery] Cluster info signature and contents are valid, will use API Server
"https://10.0.20.29:6443"
[discovery] Successfully established connection with API Server
"10.0.20.29:6443"
[bootstrap] Detected server version: v1.7.11
[bootstrap] The server supports the Certificates API
(certificates.k8s.io/v1beta1)
[csr] Created API client to obtain unique certificate for this node, generating
keys and certificate signing request
[csr] Received signed certificate from the API server, generating KubeConfig...
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/kubelet.conf"

Node join complete:
* Certificate signing request sent to master and response
  received.
* Kubelet informed of new secure connection details.

Run 'kubectl get nodes' on the master to see this machine join.
```

13. Run **kubectl get nodes** to see the node in state Ready, as illustrated in Example 2-14.

Example 2-14 kubectl get nodes

```
[root@k8s1 ~]# kubectl get nodes
NAME                STATUS    AGE       VERSION
k8s1.localdomain    Ready     37m       v1.7.11
```

2.5 IBM Storage Enabler for Containers installation and configuration

IBM Storage Enabler for Containers allows IBM storage systems to be used as block storage devices for Kubernetes container orchestrator.

As explained in 1.2.1, “IBM Storage Enabler for Containers” on page 7, IBM Storage Enabler for Containers is based on an open source IBM project, Ubiquity, that integrated it with IBM Spectrum Connect. Through IBM Storage Enabler for Containers, storage volumes are provisioned from an external IBM storage to container frameworks, allowing the use of the containers with stateful microservices such as database applications (MongoDB, PostgreSQL, and so on). Storage provisioning can be fully automated with additional support of cluster orchestration systems to automatically deploy, scale, and manage containerized applications.

IBM Storage Enabler for Containers uses Kubernetes FlexVolume driver for volume provisioning. It supports the list of volume commands defined in the FlexVolume API.

For more information about IBM Storage Enabler for Containers see the IBM Spectrum Control Base Edition User Guide.

To install and configure IBM Storage Enabler for Containers, complete these steps:

1. Pull all the necessary docker images (for IBM Storage Enabler for Containers) on the VM with internet access. The commands are shown in Example 2-15.

> **Note:** The following steps with pulling, saving, and loading the images are not necessary if the k8s node has internet access. Because the systems used for our experimentation are behind a firewall in protected lab environment, we must do an offline installation and configuration. You might be facing the same situation with systems in your IT center.
>
> If your systems have internet access, you can proceed to 2.5.1, “Start here if your system has internet access” on page 41.

Example 2-15 IBM Storage Enabler image pull

```
[root@QW2 ~]# docker pull ibmcom/ibm-storage-enabler-for-containers:1.0.0
[root@QW2 ~]# docker pull ibmcom/ibm-storage-enabler-for-containers-db:1.0.0
[root@QW2 ~]# docker pull
ibmcom/ibm-storage-dynamic-provisioner-for-kubernetes:1.0.0
[root@QW2 ~]# docker pull ibmcom/ibm-storage-flex-volume-for-kubernetes:1.0.0
```

2. Save all the images as tar files, as shown in Example 2-16.

Example 2-16 IBM Storage Enabler docker save

```
[root@QW2 ~]# docker save ibmcom/ibm-storage-flex-volume-for-kubernetes:1.0.0
-o /tmp/ibm-storage-flex-volume-for-kubernetes.tar
[root@QW2 ~]# docker save ibmcom/ibm-storage-enabler-for-containers-db:1.0.0 -o
/tmp/ibm-storage-enabler-for-containers-db.tar
[root@QW2 ~]# docker save
ibmcom/ibm-storage-dynamic-provisioner-for-kubernetes:1.0.0 -o
/tmp/ibm-storage-dynamic-provisioner-for-kubernetes.tar
[root@QW2 ~]# docker save ibmcom/ibm-storage-enabler-for-containers:1.0.0 -o
/tmp/ibm-storage-enabler-for-containers.tar
```

3. Transfer the tar packages to the Kubernetes node and load the images into docker with **docker** commands, as illustrated in Example 2-17.

Example 2-17 IBM Storage Enabler docker load

```
[root@k8s1 ~]# docker load -i ibm-storage-enabler-for-containers.tar
[root@k8s1 ~]# docker load -i
ibm-storage-dynamic-provisioner-for-kubernetes.tar
[root@k8s1 ~]# docker load -i ibm-storage-enabler-for-containers-db.tar
[root@k8s1 ~]# docker load -i ibm-storage-flex-volume-for-kubernetes.tar
```

4. Install the `sg3_utils` and `device-mapper-multipath` packages for Fibre Channel attachment, if not already installed before, as illustrated in Example 2-18.

Example 2-18 yum install sg3_utils and device-mapper-multipath

```
[root@k8s1 ~]# yum -y install sg3_utils
Loaded plugins: product-id, search-disabled-repos, subscription-manager
This system is not registered to Red Hat Subscription Management. You can use
subscription-manager to register.
Resolving Dependencies
--> Running transaction check
---> Package sg3_utils.x86_64 0:1.37-9.el7 will be installed
--> Processing Dependency: sg3_utils-libs = 1.37-9.el7 for package:
sg3_utils-1.37-9.el7.x86_64
--> Processing Dependency: libsgutils2.so.2()(64bit) for package:
sg3_utils-1.37-9.el7.x86_64
--> Running transaction check
---> Package sg3_utils-libs.x86_64 0:1.37-9.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

.............................................................
Installed:
  sg3_utils.x86_64 0:1.37-9.el7

Dependency Installed:
  sg3_utils-libs.x86_64 0:1.37-9.el7

Complete!

[root@k8s1 ~]# yum install -y device-mapper-multipath
```

```
Loaded plugins: product-id, search-disabled-repos, subscription-manager
This system is not registered to Red Hat Subscription Management. You can use
subscription-manager to register.
Package device-mapper-multipath-0.4.9-99.el7.x86_64 already installed and
latest version
Nothing to do
```

5. Copy the `multipath.conf` file, load the module `dm-multipath`, and start the multipath daemon, as displayed in Example 2-19.

Example 2-19 cp multipath.conf, modprobe dm-multipath, and start multipathd

```
[root@k8s1 ~]# cp /usr/share/doc/device-mapper-multipath-0.4.9/multipath.conf
/etc/multipath.conf
[root@k8s1 ~]# modprobe dm-multipath
[root@k8s1 ~]# systemctl start multipathd
```

6. Edit `/etc/systemd/system/kubelet.service.d/10-kubeadm.conf` and add the option `--enable-controller-attach-detach=true`, as shown in Example 2-20.

Example 2-20 cat 10-kubeadm.conf

```
[root@k8s1 ~]# cat /etc/systemd/system/kubelet.service.d/10-kubeadm.conf
[Service]
Environment="KUBELET_KUBECONFIG_ARGS=--kubeconfig=/etc/kubernetes/kubelet.conf
--require-kubeconfig=true --enable-controller-attach-detach=true"
Environment="KUBELET_SYSTEM_PODS_ARGS=--pod-manifest-path=/etc/kubernetes/manif
ests --allow-privileged=true"
Environment="KUBELET_NETWORK_ARGS=--network-plugin=cni
--cni-conf-dir=/etc/cni/net.d --cni-bin-dir=/opt/cni/bin"
Environment="KUBELET_DNS_ARGS=--cluster-dns=10.96.0.10
--cluster-domain=cluster.local"
Environment="KUBELET_AUTHZ_ARGS=--authorization-mode=Webhook
--client-ca-file=/etc/kubernetes/pki/ca.crt"
Environment="KUBELET_CADVISOR_ARGS=--cadvisor-port=0"
Environment="KUBELET_CGROUP_ARGS=--cgroup-driver=cgroupfs"
ExecStart=
ExecStart=/usr/bin/kubelet $KUBELET_KUBECONFIG_ARGS $KUBELET_SYSTEM_PODS_ARGS
$KUBELET_NETWORK_ARGS $KUBELET_DNS_ARGS $KUBELET_AUTHZ_ARGS
$KUBELET_CADVISOR_ARGS $KUBELET_CGROUP_ARGS $KUBELET_EXTRA_ARGS
```

7. Enable port `9999` for IBM Storage Enabler for Containers at the firewall, as illustrated in Example 2-21.

Example 2-21 firewall-cmd for Storage Enabler

```
[root@k8s1 ~]# firewall-cmd --permanent --add-port=9999/tcp
success
[root@k8s1 ~]# firewall-cmd --reload
Success
```

2.5.1 Start here if your system has internet access

To install and configure IBM Storage Enabler for Containers, continue with these steps:

8. From IBM Fix Central, download the `installer-for-ibm-storage-enabler-for-containers-1.0.0-<version>.tar.gz` file.
9. Extract the file by using the **tar** command, as shown in Example 2-22.

Example 2-22 tar IBM Storage Enabler for Containers

```
[root@k8s1 ~]# tar -xzvf
installer-for-ibm-storage-enabler-for-containers-1.0.0-185.tar.gz
installer-for-ibm-storage-enabler-for-containers-1.0.0-185/
installer-for-ibm-storage-enabler-for-containers-1.0.0-185/ubiquity_lib.sh
installer-for-ibm-storage-enabler-for-containers-1.0.0-185/scbe-credentials-sec
ret.yml
installer-for-ibm-storage-enabler-for-containers-1.0.0-185/ubiquity_installer.c
onf
installer-for-ibm-storage-enabler-for-containers-1.0.0-185/LICENSE
installer-for-ibm-storage-enabler-for-containers-1.0.0-185/ubiquity-db-credenti
als-secret.yml
installer-for-ibm-storage-enabler-for-containers-1.0.0-185/yamls/
installer-for-ibm-storage-enabler-for-containers-1.0.0-185/yamls/ubiquity-k8s-f
lex-daemonset.yml
installer-for-ibm-storage-enabler-for-containers-1.0.0-185/yamls/sanity_yamls/
installer-for-ibm-storage-enabler-for-containers-1.0.0-185/yamls/sanity_yamls/s
anity-deployment.yml
installer-for-ibm-storage-enabler-for-containers-1.0.0-185/yamls/sanity_yamls/s
anity-pod.yml
installer-for-ibm-storage-enabler-for-containers-1.0.0-185/yamls/sanity_yamls/s
anity-pvc.yml
installer-for-ibm-storage-enabler-for-containers-1.0.0-185/yamls/ubiquity-names
pace.yml
installer-for-ibm-storage-enabler-for-containers-1.0.0-185/yamls/storage-class.
yml
installer-for-ibm-storage-enabler-for-containers-1.0.0-185/yamls/ubiquity-db-pv
c.yml
installer-for-ibm-storage-enabler-for-containers-1.0.0-185/yamls/ubiquity-db-se
rvice.yml
installer-for-ibm-storage-enabler-for-containers-1.0.0-185/yamls/ubiquity-deplo
yment.yml
installer-for-ibm-storage-enabler-for-containers-1.0.0-185/yamls/ubiquity-k8s-p
rovisioner-deployment.yml
installer-for-ibm-storage-enabler-for-containers-1.0.0-185/yamls/templates/
installer-for-ibm-storage-enabler-for-containers-1.0.0-185/yamls/templates/pvc-
template.yml
installer-for-ibm-storage-enabler-for-containers-1.0.0-185/yamls/templates/stor
age-class-template.yml
installer-for-ibm-storage-enabler-for-containers-1.0.0-185/yamls/ubiquity-servi
ce.yml
installer-for-ibm-storage-enabler-for-containers-1.0.0-185/yamls/ubiquity-db-de
ployment.yml
installer-for-ibm-storage-enabler-for-containers-1.0.0-185/ubiquity_installer.s
h
installer-for-ibm-storage-enabler-for-containers-1.0.0-185/ubiquity_uninstall.s
h
```

```
installer-for-ibm-storage-enabler-for-containers-1.0.0-185/ubiquity_cli.sh
installer-for-ibm-storage-enabler-for-containers-1.0.0-185/ubiquity-configmap.y
ml
```

10. Change to the installer directory and enter all the necessary information in `ubiquity_installer.conf`, as shown in Example 2-23. `SCBE_USERNAME_VALUE`, `SCBE_PASSWORD_VALUE`, `SCBE_DEFAULT_SERVICE_VALUE`, and `STORAGE_CLASS_PROFILE_VALUE` must be the same values as defined during Spectrum Connect configuration in 2.2.3, “Spectrum Connect configuration for containers” on page 20.

Example 2-23 Change directory and edit ubiquity_installer.conf

```
[root@k8s1 ~]# cd installer-for-ibm-storage-enabler-for-containers-1.0.0-185
[root@k8s1 installer-for-ibm-storage-enabler-for-containers-1.0.0-185]# cat
ubiquity_installer.conf
# ----------------------------------------------------
# Description:
# This is a configuration file for the ubiquity_installer.sh script.
# When run (# ubiquity_installer.sh -s update-ymls -c <conf file>),
# this script replaces the relevant values in all the yaml files of the
installer.
#
# Attention:
# 1. Replace the "VALUE"s placeholder below with the relevant value for your
environment.
# 2. Any change required after running this installer script must be performed
manually in the corresponding *.yml file itself.
# ----------------------------------------------------

# The Docker images of IBM Storage Enabler for Containers.
UBIQUITY_IMAGE=ibmcom/ibm-storage-enabler-for-containers:1.0.0
UBIQUITY_DB_IMAGE=ibmcom/ibm-storage-enabler-for-containers-db:1.0.0
UBIQUITY_K8S_PROVISIONER_IMAGE=ibmcom/ibm-storage-dynamic-provisioner-for-kuber
netes:1.0.0
UBIQUITY_K8S_FLEX_IMAGE=ibmcom/ibm-storage-flex-volume-for-kubernetes:1.0.0

# Parameters in ubiquity-configmap.yml that impact on ubiquity deployment
#-----------------------------------------------------------------
# IP or FQDN of SCBE server.
SCBE_MANAGEMENT_IP_VALUE=10.0.20.27

# Communication port of SCBE server.
SCBE_MANAGEMENT_PORT_VALUE=8440

# Default SCBE storage service to be used, if not specified by the storage
class.
SCBE_DEFAULT_SERVICE_VALUE=Container_Service

# A prefix for any new volume created on the storage system.
UBIQUITY_INSTANCE_NAME_VALUE=k8s1

# The fstype of a new volume if not specified by the user in the storage class.
# File system type. Allowed values: ext4 or xfs.
DEFAULT_FSTYPE_VALUE=ext4
```

```
# The default volume size (in GB) if not specified by the user when creating a
new volume.
DEFAULT_VOLUME_SIZE_VALUE=5


# Parameter in ubiquity-configmap.yml that impact on "ubiquity-k8s-flex"
daemonset
#-----------------------------------------------------------------
# Allowed values: true or false. Set to true if the nodes have FC connectivity.
SKIP_RESCAN_ISCSI_VALUE=true


# Parameters in ubiquity-configmap.yml that impact on "ubiquity" and
"ubiquity-k8s-provisioner" deployments, And "ubiquity-k8s-flex" daemonset
#-----------------------------------------------------------------
# Log level. Allowed values: debug, info, error.
LOG_LEVEL_VALUE=info

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


# Parameters in scbe-credentials-secret.yml that impact ubiquity and
ubiquity-k8s-provisioner deployments, And ubiquity-k8s-flex daemonset
#-----------------------------------------------------------------
# Username and password defined for IBM Storage Enabler for Containers
interface in SCBE.
SCBE_USERNAME_VALUE=user
SCBE_PASSWORD_VALUE=password


# Parameters in ubiquity-db-credentials-secret.yml that impact ubiquity and
ubiquity-db deployments
#-----------------------------------------------------------------
# Username and password for the deployment of ubiquity-db database. Note : Do
not use the "postgres" username, because it already exists.
UBIQUITY_DB_USERNAME_VALUE=ubiquity
UBIQUITY_DB_PASSWORD_VALUE=ubiquity


# Parameters to create the first Storage Class that also be used by ubiquity
for ibm-ubiquity-db PVC.
# The parameters in the following files: yamls/storage-class.yml,
ubiquiyt-db-pvc.yml, sanity_yamls/sanity-pvc.yml
#-----------------------------------------------------------------
# Storage Class name
STORAGE_CLASS_NAME_VALUE=a9000

# Storage Class profile parameter should point to the SCBE storage service name
STORAGE_CLASS_PROFILE_VALUE=Container_Service

# Storage Class file-system type, Allowed values: ext4 or xfs.
STORAGE_CLASS_FSTYPE_VALUE=ext4
```

11. Start the IBM Storage Enabler for Containers installation by using the **./ubiquity_installer.sh** command, as illustrated in Example 2-24.

Example 2-24 ./ubiquity_installer.sh

```
[root@k8s1 installer-for-ibm-storage-enabler-for-containers-1.0.0-185]#
./ubiquity_installer.sh -s install -k ~/.kube/config
Executing STEP [install]...
Starting installation  "IBM Storage Enabler for Containers"...
Installing on the namespace  [ubiquity].
namespace "ubiquity" created
service "ubiquity" created
service "ubiquity-db" created
Update the UBIQUITY-IP-ADDRESS: 10.109.227.200 in the file
[./yamls/../ubiquity-configmap.yml]
configmap "ubiquity-configmap" created
secret "scbe-credentials" created
secret "ubiquity-db-credentials" created
deployment "ubiquity" created
Waiting for deployment [ubiquity] to be in Running state...
deployment [ubiquity] reached the expected generation [1]
deployment [ubiquity] available replicas are [0] while expected value is [1],
sleeping [5 sec] before retrying to check [0/20]
deployment [ubiquity] available replicas are [0] while expected value is [1],
sleeping [5 sec] before retrying to check [1/20]
deployment [ubiquity] reached the expected replicas [1]
Creating k8s-config for ubiquity-k8s-provisioner from file
[/root/.kube/config].
configmap "k8s-config" created
deployment "ubiquity-k8s-provisioner" created
Waiting for deployment [ubiquity-k8s-provisioner] to be in Running state...
deployment [ubiquity-k8s-provisioner] reached the expected generation [1]
deployment [ubiquity-k8s-provisioner] available replicas are [0] while expected
value is [1], sleeping [5 sec] before retrying to check [0/20]
deployment [ubiquity-k8s-provisioner] reached the expected replicas [1]
storageclass "a9000" created
persistentvolumeclaim "ibm-ubiquity-db" created
Waiting for ibm-ubiquity-db PVC to be created
pvc [ibm-ubiquity-db] status is [Pending] while expected status is [Bound].
sleeping [5 sec] before retrying to check [0/60]
pvc [ibm-ubiquity-db] status [Bound] as expected (after 1/60 tries)
Waiting for ibm-ubiquity-db PV to be created
pv [ibm-ubiquity-db] status [Bound] as expected (after 0/20 tries)
Deploying FlexVolume driver as a daemonset on all nodes and masters. The
daemonset will use the ubiquity service IP.
daemonset "ubiquity-k8s-flex" created
Waiting for daemonset [ubiquity-k8s-flex] to be in Running state...
daemonset [ubiquity-k8s-flex] reached the expected generation [1]
daemonset [ubiquity-k8s-flex] available pods are [0] while expected quantity is
[1], sleeping [3 sec] before retrying to check [0/20]
daemonset [ubiquity-k8s-flex] available pods are [0] while expected quantity is
[1], sleeping [3 sec] before retrying to check [1/20]
daemonset [ubiquity-k8s-flex] reached the expected available pods [1]

"IBM Storage Enabler for Containers" Installation finished, but the deployment
is not ready yet.
```

```
  Perform the following:
     (1) Manually restart the kubelet service on all Kubernetes nodes to reload
the new FlexVolume driver.
     (2) Deploy ubiquity-db by $> ./ubiquity_installer.sh -s create-ubiquity-db
-n ubiquity
     Note : View status by $> ./ubiquity_cli.sh -a status -n ubiquity
```

12. Restart the kubelet service, as displayed in Example 2-25.

Example 2-25 Restart kubelet service

```
[root@k8s1 installer-for-ibm-storage-enabler-for-containers-1.0.0-185]# service
kubelet restart
Redirecting to /bin/systemctl restart  kubelet.service
```

13. Deploy the ubiquity database by using the **./ubiquity_installer.sh -s create-ubiquity-db** command, as shown in Example 2-26.

Example 2-26 Deploy Ubiquity DB

```
[root@k8s1 installer-for-ibm-storage-enabler-for-containers-1.0.0-185]#
./ubiquity_installer.sh -s create-ubiquity-db
Executing STEP [create-ubiquity-db]...
Creating ubiquity-db deployment... (Assuming that the IBM Storage Kubernetes
FlexVolume(ubiquity-k8s-flex) plugin is already loaded on all the nodes)
deployment "ubiquity-db" created
Waiting for deployment [ubiquity-db] to be created...
Waiting for deployment [ubiquity-db] to be in Running state...
deployment [ubiquity-db] reached the expected generation [1]
deployment [ubiquity-db] available replicas are [0] while expected value is
[1], sleeping [5 sec] before retrying to check [0/50]
deployment [ubiquity-db] available replicas are [0] while expected value is
[1], sleeping [5 sec] before retrying to check [1/50]
deployment [ubiquity-db] available replicas are [0] while expected value is
[1], sleeping [5 sec] before retrying to check [2/50]
deployment [ubiquity-db] available replicas are [0] while expected value is
[1], sleeping [5 sec] before retrying to check [3/50]
deployment [ubiquity-db] reached the expected replicas [1]

"IBM Storage Enabler for Containers" installation finished successfully in the
Kubernetes cluster.
           - Get status      $> ./ubiquity_cli.sh -a status -n ubiquity
           - Run sanity test $> ./ubiquity_cli.sh -a sanity -n ubiquity
```

The Ubiquity database deployment creates a 20 GB volume on FlashSystem A9000 and maps it to the Kubernetes host `k8s1.localdomain`, as depicted in Figure 2-31.

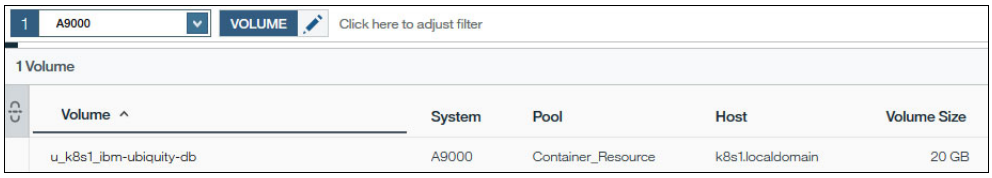


Figure 2-31 Ubiquity DB volume

14. You can check the status of IBM Storage Enabler for Containers by using **./ubiquity_cli.sh** command, as shown in Example 2-27.

Example 2-27 Ubiquity status

```
[root@k8s1 installer-for-ibm-storage-enabler-for-containers-1.0.0-185]#
./ubiquity_cli.sh -a status -n ubiquity
Working in namespace [ubiquity].
kubectl get storageclass | egrep "ubiquity|^NAME"
----------------------------------------------------------------------
NAME      TYPE
a9000     ubiquity/flex

kubectl get --namespace ubiquity secret/ubiquity-db-credentials
secret/scbe-credentials cm/k8s-config cm/ubiquity-configmap pv/ibm-ubiquity-db
pvc/ibm-ubiquity-db svc/ubiquity svc/ubiquity-db daemonset/ubiquity-k8s-flex
deploy/ubiquity deploy/ubiquity-db deploy/ubiquity-k8s-provisioner
----------------------------------------------------------------------
NAME                              TYPE      DATA      AGE
secrets/ubiquity-db-credentials   Opaque    3         10m
secrets/scbe-credentials          Opaque    2         10m

NAME                    DATA      AGE
cm/k8s-config           1         10m
cm/ubiquity-configmap   10        10m

NAME            CAPACITY  ACCESSMODES  RECLAIMPOLICY  STATUS   CLAIM
STORAGECLASS   REASON    AGE
pv/ibm-ubiquity-db   20Gi       RWO           Delete          Bound
ubiquity/ibm-ubiquity-db   a9000                    10m

NAME                  STATUS    VOLUME            CAPACITY   ACCESSMODES
STORAGECLASS   AGE
pvc/ibm-ubiquity-db   Bound    ibm-ubiquity-db   20Gi      RWO         a9000
10m
NAME              CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
svc/ubiquity      10.109.227.200   <none>        9999/TCP   10m
svc/ubiquity-db   10.110.73.113    <none>        5432/TCP   10m

NAME                   DESIRED   CURRENT   READY     UP-TO-DATE   AVAILABLE
NODE-SELECTOR   AGE
ds/ubiquity-k8s-flex   1         1         1         1            1
<none>          9m

NAME                              DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
deploy/ubiquity                   1         1         1            1           10m
deploy/ubiquity-db                1         1         1            1           6m
deploy/ubiquity-k8s-provisioner   1         1         1            1           10m
kubectl get --namespace ubiquity pod | egrep "^ubiquity|^NAME"
----------------------------------------------------------------------
NAME                                        READY     STATUS    RESTARTS   AGE
ubiquity-1594673819-4zbhl                   1/1       Running   0          10m
ubiquity-db-1704881346-9rvxc                1/1       Running   0          6m
ubiquity-k8s-flex-n4msk                     1/1       Running   0          9m
ubiquity-k8s-provisioner-2800068825-481zq   1/1       Running   0          10m
```

2.6 IBM Storage Enabler for Containers provisioning tasks

You can use IBM Storage Enabler for Containers together with IBM Storage Kubernetes Dynamic Provisioner and the IBM Storage Kubernetes FlexVolume to run stateful containers with a storage volume provisioned from an external IBM storage system.

To be able to run a stateful container, a storage class is needed, which was already defined before the deployment of the IBM Storage Enabler for Containers in 2.1, “IBM Spectrum Connect overview” on page 10. For additional storage classes, see the Spectrum Connect User Guide.

The storage classes can be listed by using **`kubectl get storageclass`** command, as shown in Example 2-28.

Example 2-28 kubectl get storageclass

```
[root@k8s1 installer-for-ibm-storage-enabler-for-containers-1.0.0-185]# kubectl
get storageclass a9000
NAME      TYPE
a9000     ubiquity/flex
```

Creating a PersistentVolumeClaim (PVC) `pvc1` that uses the storage class `a9000` and a volume size of 10 GB by using the **`kubectl create`** command is illustrated in Example 2-29.

Example 2-29 pvc1.yml and kubectl create

```
[root@k8s1 installer-for-ibm-storage-enabler-for-containers-1.0.0-185]# cat pvc1.yml
# This is an IBM Storage Enabler for Containers PVC template.
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1
  # NOTE : Uses annotations storage-class only in k8s 1.5 version.
  # ------
  #annotations:
  #  volume.beta.kubernetes.io/storage-class: "STORAGE_CLASS_NAME"
# NOTE : User label only if you want to set the PV name (the default is PV=PVC-ID)
  # ------
  #labels:
  #  pv-name: "ibm-ubiquity-db"   # Ubiquity provisioner will create a PV with
ibm-ubiquity-db instead of PVC-ID.
spec:
  # NOTE : Use storageClassName only in k8s version 1.6+. For lower versions uses
volume.beta.kubernetes.io/storage-class
  storageClassName: a9000
accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
[root@k8s1 installer-for-ibm-storage-enabler-for-containers-1.0.0-185]# kubectl create -f
pvc1.yml
persistentvolumeclaim "pvc1" created
```

A volume with 10 GB size is created on FlashSystem A9000, as depicted in Figure 2-32.

Volume	System	Pool	Volume Size	Host
u_k8s1_ibm-ubiquity-db	A9000	Container_Resource	20 GB	k8s1.local
u_k8s1_pvc-494d76e8-e803-11e7-825f-00215eda6778	A9000	Container_Resource	10 GB	0

Figure 2-32 Volume pvc1

To create a pod named `pod1` with a persistent volume `vol1`, as shown in Example 2-30, the IBM Storage Kubernetes FlexVolume driver executes the following tasks:

- Attach the volume to the host
- Rescan and discover the multipath device of the new volume
- Create XFS or EXT4 file system on the device (if the file system does not exist on the volume)
- Mount the new multipath device on `/ubiquity/[WWN of the volume]`
- Create a symbolic link from `/var/lib/kubelet/pods/[Pod ID]/volumes/ibm~ubiquity-k8s-flex/[PVC ID]` to `/ubiquity/[WWN of the volume]`

Example 2-30 pod1.yml and kubectl create pod

```
[root@k8s1 installer-for-ibm-storage-enabler-for-containers-1.0.0-185]# cat
pod1.yml
kind: Pod
apiVersion: v1
metadata:
  name: pod1
spec:
  containers:
  - name: container1
    image: alpine:latest
    command: [ "/bin/sh", "-c", "--" ]
    args: [ "while true; do sleep 30; done;" ]
    volumeMounts:
    - name: vol1
      mountPath: "/data"
  restartPolicy: "Never"
  volumes:
  - name: vol1
    persistentVolumeClaim:
      claimName: pvc1
[root@k8s1 installer-for-ibm-storage-enabler-for-containers-1.0.0-185]# kubectl
create -f pod1.yml
pod "pod1" created
```

The volume is mapped to the host as depicted in Figure 2-33.

Volume	System	Pool	Volume Size	Host
u_k8s1_ibm-ubiquity-db	A9000	Container_Resource	20 GB	k8s1.localdomain
u_k8s1_pvc-494d76e8-e803-11e7-825f-00215eda6778	A9000	Container_Resource	10 GB	k8s1.localdomai

Figure 2-33 Volume pvc1 mapped

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 FlashSystem A9000 and IBM FlashSystem A9000R Architecture and Implementation*, SG24-8345
- *IBM Hyper-Scale Manager for IBM Spectrum Accelerate Family: IBM XIV, IBM FlashSystem A9000 and A9000R, and IBM Spectrum Accelerate*, SG24-8376
- *IBM FlashSystem A9000, IBM FlashSystem A9000R, and IBM XIV Storage System: Host Attachment and Interoperability*, SG24-8368
- *IBM XIV and VMware Synergy with IBM Spectrum Control Base Edition*, REDP-5131

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

Other publications

These publications are also relevant as further information sources:

- *Hyper-Scale Manager 5.3 REST API Specification*, SC27-6440-01
- *Hyper-Scale Manager 5.3 User Guide*, SC27-8560
- *IBM FlashSystem A9000 Models 9836-415, 9838-415, 9836-425, and 9838-425 Deployment Guide,* GC27-8564
- *IBM FlashSystem A9000R Models 9835-415, 9837-415, 9835-425, and 9837-425 Deployment Guide,* GC27-8565
- *IBM Spectrum Control Base Edition Version 3.3.0 User Guide,* SC27-5999-21

Online resources

These websites are also relevant as further information sources:

- IBM Hyper-Scale Manager at IBM Knowledge Center:

  https://www.ibm.com/support/knowledgecenter/SSUMNQ_5.3.0
- IBM FlashSystem A9000 product page:

  http://www.ibm.com/systems/storage/flash/a9000

- IBM FlashSystem A9000R product page:
  http://www.ibm.com/systems/storage/flash/a9000r
- IBM Fix Central:
  http://www.ibm.com/support/fixcentral/

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

REDP-5470-00

ISBN 0738456721

Printed in U.S.A.