Redbooks
ibm.com/redbooks

IBM Spectrum Scale CSI Driver for Container Persistent Storage

Abhishek Jain

Andrew Beattie

Daniel de Souza Casali

Deepak Ghuge

Harald Seipp

Kedar Karmarkar

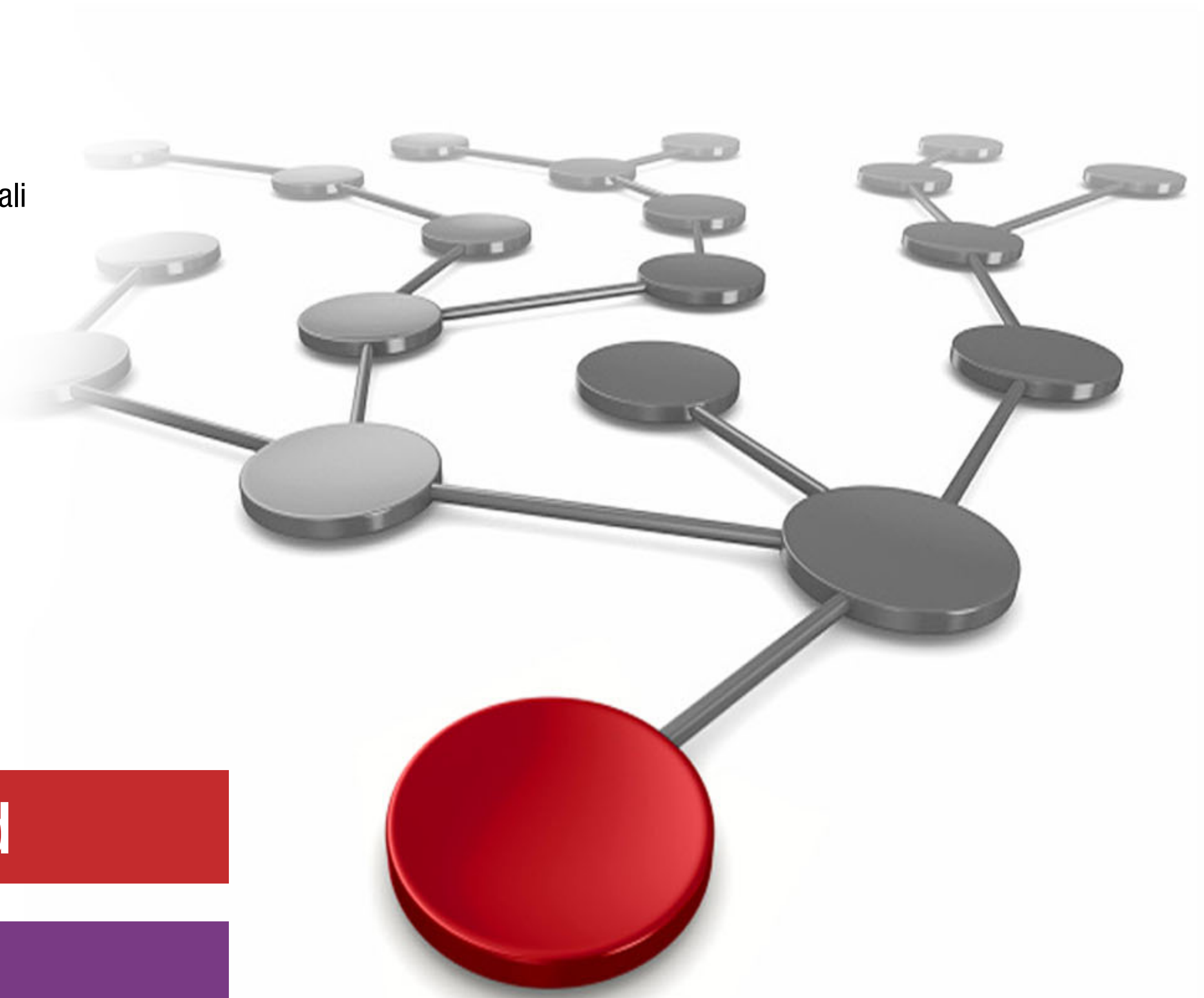
Muthu Muthiah

Pravin P. Kudav

Sandeep R. Patil

Smita Raut

Yadavendra Yadav



Cloud

Storage

IBM

Redpaper

IBM Redbooks

IBM Spectrum Scale CSI Driver for Container Persistent Storage

April 2020

REDP-5589-00

Note: Before using this information and the product it supports, read the information in "Notices" on page ix.

First Edition (April 2020)

This edition applies to Version 5, Release 0, Modification 4 of IBM Spectrum Scale.

Contents

Notices . . . vii
Trademarks . . . viii

Preface . . . ix
Authors . . . ix
Now you can become a published author, too . . . xi
Comments welcome . . . xii
Stay connected to IBM Redbooks . . . xii

Chapter 1. IBM Spectrum Scale and Containers Introduction . . . 1
1.1 Abstract . . . 2
1.2 Assumptions . . . 2
1.3 Key concepts and terminology . . . 3
  1.3.1 IBM Spectrum Scale . . . 3
  1.3.2 Container runtime . . . 3
  1.3.3 Container Orchestration System . . . 4
1.4 Introduction to persistent storage for containers Flex volumes . . . 4
  1.4.1 Static provisioning . . . 4
  1.4.2 Dynamic provisioning . . . 4
  1.4.3 Container Storage Interface (CSI) . . . 5
  1.4.4 Advantages of using IBM Spectrum Scale storage for containers . . . 5

Chapter 2. Architecture of IBM Spectrum Scale CSI Driver . . . 7
2.1 CSI Component Overview . . . 8
2.2 IBM Spectrum Scale CSI driver architecture . . . 8

Chapter 3. Solutions and Use Cases . . . 11
3.1 Multiple containers accessing the same data . . . 12
3.2 Multi-protocol access . . . 13
3.3 Multi-tenancy use case . . . 14
3.4 Remote file system access use cases . . . 16
  3.4.1 Separation of IBM Spectrum Scale clients and NSD servers . . . 16
3.5 Kubernetes multi-cluster use cases . . . 17
3.6 CSI driver for multisite use cases (AFM use cases) . . . 18
3.7 Compressed Volumes use case . . . 20

Chapter 4. Planning for IBM Spectrum Scale CSI driver deployment . . . 23
4.1 Deployment on Kubernetes . . . 24
  4.1.1 Kubernetes deployment mechanism . . . 24
  4.1.2 Container runtime . . . 25
  4.1.3 Network plug-in . . . 25
4.2 Deployment on OpenShift . . . 25
4.3 IBM Spectrum Scale deployment . . . 27
  4.3.1 IBM Spectrum Scale deployment models . . . 27
  4.3.2 Network considerations . . . 27
  4.3.3 Primary File System . . . 27
4.4 What type of volumes do I need . . . 28
4.5 Limitations of IBM Spectrum Scale CSI driver . . . 28

Chapter 5. Deployment and Administration 29
5.1 IBM Spectrum Scale CSI Driver configuration 30
5.2 Deployment of IBM Spectrum Scale CSI Driver 30
5.2.1 Using Operator Lifecycle Manager 30
5.2.2 Using Operator with command line interface (CLI) 31
5.3 Volume provisioning 33
5.3.1 Static provisioning 33
5.3.2 Dynamic provisioning 34
5.4 Using the volume in a container 35
5.5 Releasing the volume on pod deletion 36
5.6 Node Selector 37
5.7 Kubernetes to IBM Spectrum Scale Cluster node mapping 37
5.8 Upgrading IBM Spectrum Scale Container Storage Interface driver. 37

Chapter 6. Security considerations. 39
6.1 Secure administration and deployment. 40
6.1.1 Network deployment and firewall configuration. 40
6.1.2 Secure communication with IBM Spectrum Scale GUI server 40
6.1.3 SELinux considerations 41
6.1.4 Configuring UID/GID ownership to ensure Persistent Volume access for non-root containers 41
6.1.5 Privileged CSI PODs. 45
6.2 Secure data for containers 45
6.2.1 Secure data at rest for container data. 45
6.2.2 Secure data in transit 46

Chapter 7. Problem determination and troubleshooting. 49
7.1 How to collect debug data for IBM Spectrum Scale CSI driver. 50
7.2 Understanding log files 51
7.2.1 Checking the state of Kubernetes cluster 51
7.2.2 Checking for issues during driver initialization 52
7.2.3 Checking for issues during provisioning of volumes 52
7.2.4 Checking for issues during attaching of volumes 52

Chapter 8. Migration from SEC to CSI driver 53
8.1 Migration from one community Kubernetes version using SEC to a community Kubernetes version using CSI 54
8.2 Migration scenario from IBM Cloud Private using SEC to OpenShift 4.2 or community Kubernetes using CSI. 54
8.3 Migration from OpenShift 3.x using SEC to OpenShift 4.2 using CSI. 54

Chapter 9. Support 55
9.1 Support Matrix. 56
9.2 Tested Configurations. 56
9.3 Performance 56
9.3.1 I/O from container 57
9.3.2 Volume provisioning 57
9.3.3 PVC deletion 57

Appendix A. Implementation details for use cases 59
A.1 Multiple cluster use cases. 60
A.2 Compression use case 61
A.3 Encryption use case 63

Related publications 67
IBM Redbooks 67
Other publications 67
IBM Spectrum Scale resources from IBM.com 67
Online resources 69
Help from IBM 71

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®
- IBM®
- IBM Cloud™
- IBM Garage™
- IBM Spectrum®
- Redbooks®
- Redbooks (logo) ®
- Storwize®

The following terms are trademarks of other companies:

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

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

Ansible, Ceph, OpenShift, Red Hat, are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.

VMware, VMware vSphere, and the VMware logo are registered trademarks or trademarks of VMware, Inc. or its subsidiaries in the United States and/or other jurisdictions.

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

Preface

IBM® Spectrum Scale is a proven, scalable, high-performance data and file management solution. It provides world-class storage management with extreme scalability, flash accelerated performance, automatic policy-based storage that has tiers of flash through disk to tape. It also provides support for various protocols, such as NFS, SMB, Object, HDFS, and iSCSI. Containers can leverage the performance, information lifecycle management (ILM), scalability, and multisite data management to give the full flexibility on storage as they experience on the runtime.

Container adoption is increasing in all industries, and they sprawl across multiple nodes on a cluster. The effective management of containers is necessary because their number will probably reach a far greater number than virtual machines today. Kubernetes is the standard container management platform currently being used.

Data management is of ultimate importance, and often is forgotten because the first workloads containerized are ephemeral. For data management, many drivers with different specifications were available. A specification named *Container Storage Interface* (CSI) was created and is now adopted by all major Container Orchestrator Systems available.

Although other container orchestration systems exist, Kubernetes became the standard framework for container management. It is a very flexible open source platform used as the base for most cloud providers and software companies' container orchestration systems.

Red Hat OpenShift is one of the most reliable enterprise-grade container orchestration systems based on Kubernetes, designed and optimized to easily deploy web applications and services. OpenShift enables developers to focus on the code, while the platform takes care of all of the complex IT operations and processes.

This IBM Redbooks® publication describes how the CSI Driver for IBM file storage enables IBM Spectrum® Scale to be used as persistent storage for stateful applications running in Kubernetes clusters. Through the Container Storage Interface Driver for IBM file storage, Kubernetes persistent volumes (PVs) can be provisioned from IBM Spectrum Scale. Therefore, the containers can be used with stateful microservices, such as database applications (MongoDB, PostgreSQL, and so on).

Authors

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

Abhishek Jain is a Master Inventor and Software Engineer with IBM Systems in Pune (India). He holds a Bachelor of Technology degree in Computer Engineering. His current interests are in hybrid multicloud, container technology solutions. Abhishek has more than 7 years of experience working on various scale-out file systems and block/file/object storage topologies. He started working in his current role as a Test Lead Engineer for IBM Spectrum Scale CSI Driver & Cloud enablement deliveries. You can reach him at abjain39@in.ibm.com.

Andrew Beattie is the File and Object Storage Technical Sales Lead, for IBM Australia. Joining IBM in 2010, He has held both sales and technical roles with a focus on storage and virtual infrastructure solutions. Currently focused on delivering data management solutions for AI/ML, Big Data, and Analytics projects with a focus on high performance and scale-out capacity requirements. Andrew has 25 years of experience in infrastructure solution design and implementation.

Daniel de Souza Casali is a Thought Leader Specialist working at the WW IBM Systems Red Hat Synergy team as a Technical SME. Daniel has more than 15 years of experience with Enterprise Unix Systems, and more than 10 years on multi-tenant/customer computing power sharing (before it was called *the Cloud*). He worked with OpenStack, and today his main focus is Kubernetes Infrastructure and Red Hat OpenShift.

Deepak Ghuge is a Master Inventor and an Advisory Software Engineer with IBM. He has been working with IBM for Eleven years on various products, such as IBM Spectrum Scale, Scale Out Network Attached Storage, and IBM Storwize® V7000 Unified. Currently he is part of a development team responsible for IBM Spectrum Scale CSI Driver. Previously, Deepak has worked on IBM Storage Enabler for Containers (Ubiquity), OpenStack Keystone, Object Storage Security, and Authentication. Deepak holds a Bachelor of Technology degree in Computer Engineering from the College of Engineering, Pune.

Harald Seipp is a Senior Technical Staff Member with IBM Systems in Germany. He is the founder and Technical Leader of the Center of Excellence for Cloud Storage as part of the EMEA Storage Competence Center. He is providing guidance to worldwide IBM teams across organizations, and works with customers and IBM Business Partners across EMEA to create and implement complex storage cloud architectures. His more than 25 years of technology experience includes previous job roles as Software Developer, Software Development Leader, Lead Developer, and Architect for successful software products, and co-inventor of an IBM storage product. He holds various patents on storage and networking technology.

Kedar Karmarkar is a Solution Architect and consulting IT specialist with the IBM Spectrum Scale development team. Kedar is part of the IBM Spectrum Scale Client adoption team and was the IBM Storwize V7000 Unified Level 3 support lead in his earlier role at IBM. Kedar has over 20 years of infrastructure software and storage development experience in management and architect roles. He has led development of network-attached storage (NAS), block-level virtualization, replication, systems, and storage management products. Kedar has a Bachelor of Engineering (Computer Science) degree from the University of Pune, India.

Muthu Muthiah is a Software Architect with IBM Systems Group's Software Defined Storage organization. His 20 years of industry experience has been focused in the areas of Storage, Systems, and Platform Management. His current area of focus is enabling IBM Spectrum Scale, a highly parallel distributed file system on the cloud. In addition, he also serves as the architect responsible for the installation, deployment, and initial configuration of IBM Spectrum Scale.

Pravin P. Kudav works as an IBM Storage Solution Architect with IBM Systems Labs. He is working with various leading clients at IBM India. Pravin has over 15 years of solution architecture and design experience with diversified storage infrastructures. Pravin's current interest is in Software Defined Storage Solutions using IBM Spectrum Scale. Pravin holds a Master in Computer Applications degree from Madurai Kamaraj University.

Sandeep R. Patil is a Senior Technical Staff Member who works as a Storage Architect with IBM System Labs. He has over 19 years of product architecture and design experience. Sandeep is an IBM Master Inventor, and a member of the IBM Academy of Technology. Sandeep holds a Bachelor of Engineering (Computer Science) degree from the University of Pune, India. He is recognized and listed by Wikipedia in the World Wide Prolific Inventors list.

Smita Raut is a Senior Software Engineer with IBM Storage Labs in Pune, India. She works with the IBM Spectrum Scale development team as the architect for persistent storage for containers. In her nine years with IBM, she has lead the development on various projects, including Object protocol for IBM Spectrum Scale and enablement of IBM Spectrum Scale on public cloud. She is an active technical blogger and has published several blogs on object protocol and container storage interface drivers.

Yadavendra Yadav is a Software Architect for IBM Systems Development Labs. He has 16 years of industry experience in Linux Kernel, File System, Device Drivers, Containers (Docker/Kubernetes), CSI and Flex storage plug-ins for Kubernetes, and OpenStack Swift. In the past 3 years at IBM, he has worked on developing Flex and CSI plug-in for Kubernetes.

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

Larry Coyne
IBM Redbooks, Tucson Center

Piyush Chaudhary
Sanjay Gandhi
Ted Hoover
Shailesh Jeurkar
Aaron Palazollo
Kumaran Rajaram
IBM Systems

Special thanks to Tomonori Kubota

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 IBM Redbooks 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:

`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:

  `ibm.com/redbooks`

- Send your comments in an email:

  `redbooks@us.ibm.com`

- Mail your comments:

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

Stay connected to IBM Redbooks

- Find us on Facebook:

  `http://www.facebook.com/IBMRedbooks`

- Follow us on Twitter:

  `http://twitter.com/ibmredbooks`

- Look for us on LinkedIn:

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

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

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

- Stay current on recent Redbooks publications with RSS Feeds:

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

1

IBM Spectrum Scale and Containers Introduction

IBM Spectrum Scale is a proven, scalable, high-performance data and file management solution. It provides world-class storage management with extreme scalability, flash accelerated performance, automatic policy-based storage that has tiers of flash through disk to tape, and support for various protocols, such as NFS, SMB, Object, HDFS, and iSCSI. Containers can leverage the performance, Information Lifecycle Management (ILM), and Scalability and multisite data management to give the same full flexibility on storage as they experience on the runtime.

This chapter covers the following topics:

- 1.1, “Abstract” on page 2
- 1.2, “Assumptions” on page 2
- 1.3, “Key concepts and terminology” on page 3
- 1.4, “Introduction to persistent storage for containers Flex volumes” on page 4

This understanding will help for subsequent chapters about planning, use cases, and troubleshooting. Though not mandatory, it is recommended for all readers.

1.1 Abstract

IBM Spectrum Scale is a cluster file system that provides concurrent access to a single file system or set of file systems from multiple nodes. The nodes can be SAN attached, network attached, a mixture of SAN attached and network attached, or in a shared nothing cluster configuration. This enables high-performance access to this common set of data to support a scale-out solution or to provide a high-availability platform.

IBM Spectrum Scale has many features beyond common data access, including data replication, policy-based storage management, and multisite operations. You can create a cluster of IBM AIX® nodes, Linux nodes, Windows server nodes, or a mix of all three. IBM Spectrum Scale can run on virtualized instances providing common data access in environments, leverage logical partitioning, or other hypervisors. Multiple IBM Spectrum Scale clusters can share data within a location or across wide area network (WAN) connections.

Containers adoption is increasing in all industries, and containers sprawl across multiple nodes on a cluster. The effective management of containers is necessary because they will probably reach far greater numbers than virtual machines today. Kubernetes is the standard container management platform being used. Data management is of ultimate importance and often is forgotten because the first workloads containerized are ephemeral.

For data management, many drivers with different specifications were available. A specification named *Container Storage Interface* (CSI) was created and is now adopted by all major Container Orchestrator Systems available.

Although other container orchestration systems exist, Kubernetes became the standard framework for container management. It is a very flexible open source platform used as base for most cloud providers and software companies container orchestration system.

Red Hat OpenShift is one of the most reliable enterprise-grade container orchestration systems based on Kubernetes, designed and optimized to easily deploy web applications and services. OpenShift enables developers to focus on the code, while the platform takes care of all of the complex IT operations and processes.

The CSI Driver for IBM file storage enables IBM Spectrum Scale to be used as persistent storage for stateful application running in Kubernetes clusters. Through the Container Storage Interface Driver for IBM file storage, Kubernetes persistent volumes (PVs) can be provisioned from IBM Spectrum Scale. Therefore, the containers can be used with stateful microservices, such as database applications (MongoDB, PostgreSQL, and so on).

1.2 Assumptions

This IBM Redpaper publication assumes that you are familiar with basic IBM Spectrum Scale knowledge. If you need more information while reading, you can see IBM Knowledge Center for IBM Spectrum Scale:

https://www.ibm.com/support/knowledgecenter/STXKQY_5.0.4/ibmspectrumscale504_welcome.html

We also assume you have Kubernetes and OpenShift knowledge. To learn more about Kubernetes, see the following website:

https://kubernetes.io/docs/home

To learn about Red Hat OpenShift, see the following websites:

https://learn.openshift.com
https://docs.openshift.com

1.3 Key concepts and terminology

The information in this section can help you remember some of the content that you will need to follow the examples in this document. If you are familiar with any of these topics, skip to the next section.

1.3.1 IBM Spectrum Scale

Here are the major topics about IBM Spectrum Scale that will be used in this publication. We will give you links to the documentation where you can deepen your skills if you feel that it is necessary at any time.

- File sets

  https://www.ibm.com/support/knowledgecenter/en/STXKQY_5.0.4/com.ibm.spectrum.scale.v5r04.doc/bl1adv_filesets.htm

- Active File Management

  https://www.ibm.com/support/knowledgecenter/en/STXKQY_5.0.4/com.ibm.spectrum.scale.v5r04.doc/bl1ins_activefilemanagement.htm

- Multi-cluster and Remote Mounts

  https://www.ibm.com/support/knowledgecenter/en/STXKQY_5.0.4/com.ibm.spectrum.scale.v5r04.doc/bl1adv_admmcch.htm

- IBM Spectrum Scale on AWS

  https://www.ibm.com/support/knowledgecenter/en/STXKQY_AWS_SHR/com.ibm.spectrum.scale.aws.v5r03.doc/bl1cld_aws_introtoaws.htm

- IBM Spectrum Scale GUI

  https://www.ibm.com/support/knowledgecenter/en/STXKQY_5.0.4/com.ibm.spectrum.scale.v5r04.doc/bl1ins_introtogui.htm

- IBM Spectrum Scale Developer Edition

  https://www.ibm.com/account/reg/us-en/signup?formid=urx-41728

- IBM Spectrum Scale REST API

  https://www.ibm.com/support/knowledgecenter/en/STXKQY_5.0.4/com.ibm.spectrum.scale.v5r04.doc/bl1adm_restapi_main.htm

If you need a different version, there is a selector that can take you to the one. that you want Remember that CSI is supported from version 5.0.4.1 on.

1.3.2 Container runtime

A container runtime environment is a logical grouping of libraries that are referenced on the lifecycle of containers. It is responsible for the container while it is running (like listing running instances), the management of the container (start/stop), and their image management (pull/push/load/save). A great explanation this topic can on the following website:

https://www.ianlewis.org/en/container-runtimes-part-1-introduction-container-r

1.3.3 Container Orchestration System

Containers are the best way to run and maintain microservices architecture-oriented applications, that creates a better environment for continuous improvement, because each part of the application can be independently upgraded and deployed. This also means that many containers will be used to support a single application. The components must be able to communicate securely across many nodes while blocking other access from containers that should not reach the provided services.

Kubernetes Although other container orchestration systems exist, Kubernetes became the standard framework for container management. It is a very flexible, open source platform used as the base for most cloud providers and software companies' container orchestration systems:

https://kubernetes.io

OpenShift Red Hat OpenShift is one of the most reliable enterprise-grade container orchestration systems, designed and optimized to easily deploy web applications and services. Categorized as a cloud development Platform as a Service (PaaS), OpenShift enables developers to focus on code while it runs all of the complex IT operations and processes:

https://learn.openshift.com

1.4 Introduction to persistent storage for containers Flex volumes

Persistent Volume (PV) is a unit of storage in the cluster that has been provisioned by an administrator or dynamically provisioned via a storage driver or plug-in. A Persistent Volume Claim (PVC) is a request for storage by a user (for example, creating a pod). The PV is used by the PVC.

1.4.1 Static provisioning

A cluster administrator creates several Persistent Volumes up front. PVs carry the details of the real storage that is available for use by cluster users. This causes the administrator to know and set the storage requirements up front. This is useful when there is existing data on IBM Spectrum Scale cluster that needs to be provisioned as persistent volume for containers.

1.4.2 Dynamic provisioning

Dynamic volume provisioning enables storage volumes to be created on-demand. Without dynamic provisioning, cluster administrators must manually make calls to their cloud or storage provider to create new storage volumes, and then create Persistent Volumes. The dynamic provisioning feature eliminates the need for cluster administrators to pre-provision storage. Instead, it automatically provisions storage when it is requested by users.

The implementation of dynamic volume provisioning is based on Storage Class objects. A cluster administrator can define as many Storage Class objects as needed, each specifying a volume plug-in (also known as a *provisioner*) that provisions a volume and the set of parameters to pass to that provisioner when provisioning. A cluster administrator can define and expose multiple types of storage (from the same or different storage systems) within a cluster, each with a custom set of parameters.

1.4.3 Container Storage Interface (CSI)

CSI is the result of a collaborative initiative to unify the storage interface of Container Orchestrator Systems (COS), such as Kubernetes, Mesos, Docker Swarm, Cloud Foundry, and so on, combined with storage vendors (such as IBM, Ceph, Portworx, and NetApp). A single CSI implementation for a storage vendor should work with all COS. CSI defines a specification that Storage vendors implement in a CSI driver. CSI is based on the general-purpose Remote Procedure Calls (gRPC) framework. It provides extended volume management functions, such as snapshots, clones, and volume expansion.

1.4.4 Advantages of using IBM Spectrum Scale storage for containers

There are many advantages to using IBM Spectrum Scale storage for Containers:

- IBM Spectrum Scale is used on high-performance computing and delivers great performance.
- It is the provisioner that provides the most flexible way to provision storage:
  - All servers can have direct access to the physical disk, or just designated ones as it best fits the purpose.
  - The disks can be attached using different networks, such as FC, Ethernet, or InfiniBand.
  - Can be provisioned on pre-defined Arrays.
  - Can use your own storage (including cloud storage).
  - Can be created as a Shared Nothing Cluster with two or three copies for resiliency and data availability.

2

Architecture of IBM Spectrum Scale CSI Driver

This chapter describes the architecture of IBM Spectrum Scale CSI diver. It describes various components of the driver and interaction between them. It also covers the architecture of IBM Spectrum Scale CSI operator that is used for driver deployment.

This chapter will help with the understanding of subsequent chapters like “Solutions and Use Cases”, ““Planning for IBM Spectrum Scale CSI driver deployment” and “Deployment and Administration”. Though not mandatory, this chapter is recommended for all users.

This chapter covers the following topics:

- 2.1, “CSI Component Overview” on page 8
- 2.2, “IBM Spectrum Scale CSI driver architecture” on page 8

2.1 CSI Component Overview

The Container Storage Interface (CSI) is a standard for exposing arbitrary block and file storage systems to containerized workloads on Container Orchestration Systems (COs), such as Kubernetes.

Kubernetes supports a number of sidecar containers, which provide the logic to watch the Kubernetes API and trigger the appropriate operations against the *CSI Volume Driver* Container.

For more details about the Kubernetes CSI reference architecture, see the following website:

https://kubernetes-csi.github.io/docs/sidecar-containers.html

The CSI Specification references two types of Storage Plug-ins:

Node Plugin:	A gRPC endpoint serving CSI RPCs that *must* be run on the Node whereupon an SP-provisioned volume will be published
Controller Plugin:	A gRPC endpoint serving CSI RPCs that *can* be run anywhere

https://github.com/container-storage-interface/spec/blob/master/spec.md

2.2 IBM Spectrum Scale CSI driver architecture

This section provides a component level overview of the CSI driver. It describes the deployment model of CSI driver with OpenShift/Kubernetes cluster and an IBM Spectrum Scale cluster. Figure 2-1 illustrates the deployment architecture of various CSI driver and IBM Spectrum Scale components, and the interaction between the driver and the IBM Spectrum Scale cluster.

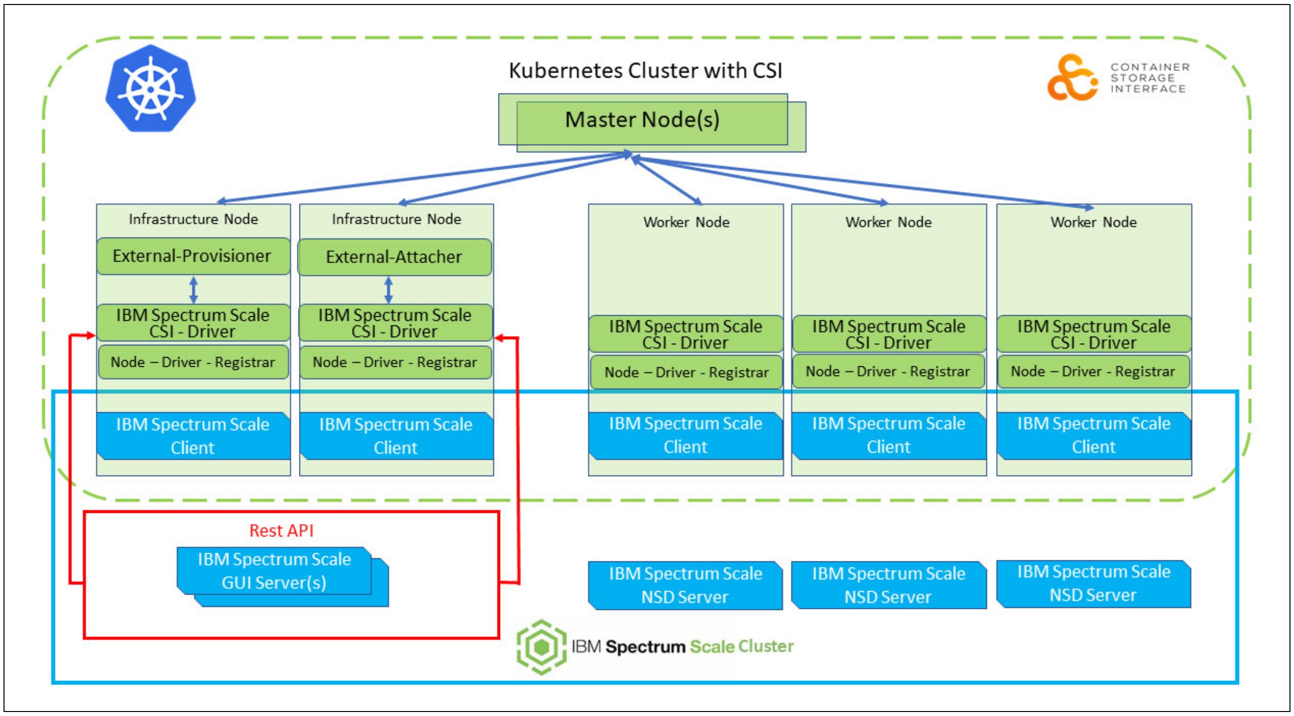

Kubernetes Cluster with CSI
CONTAINER STORAGE INTERFACE
Master Node(s)
Infrastructure Node
External-Provisioner
IBM Spectrum Scale CSI - Driver
Node – Driver - Registrar
IBM Spectrum Scale Client
Infrastructure Node
External-Attacher
IBM Spectrum Scale CSI - Driver
Node – Driver - Registrar
IBM Spectrum Scale Client
Worker Node
IBM Spectrum Scale CSI - Driver
Node – Driver - Registrar
IBM Spectrum Scale Client
Worker Node
IBM Spectrum Scale CSI - Driver
Node – Driver - Registrar
IBM Spectrum Scale Client
Worker Node
IBM Spectrum Scale CSI - Driver
Node – Driver - Registrar
IBM Spectrum Scale Client
Rest API
IBM Spectrum Scale GUI Server(s)
IBM Spectrum Scale NSD Server
IBM Spectrum Scale NSD Server
IBM Spectrum Scale NSD Server
IBM Spectrum Scale Cluster

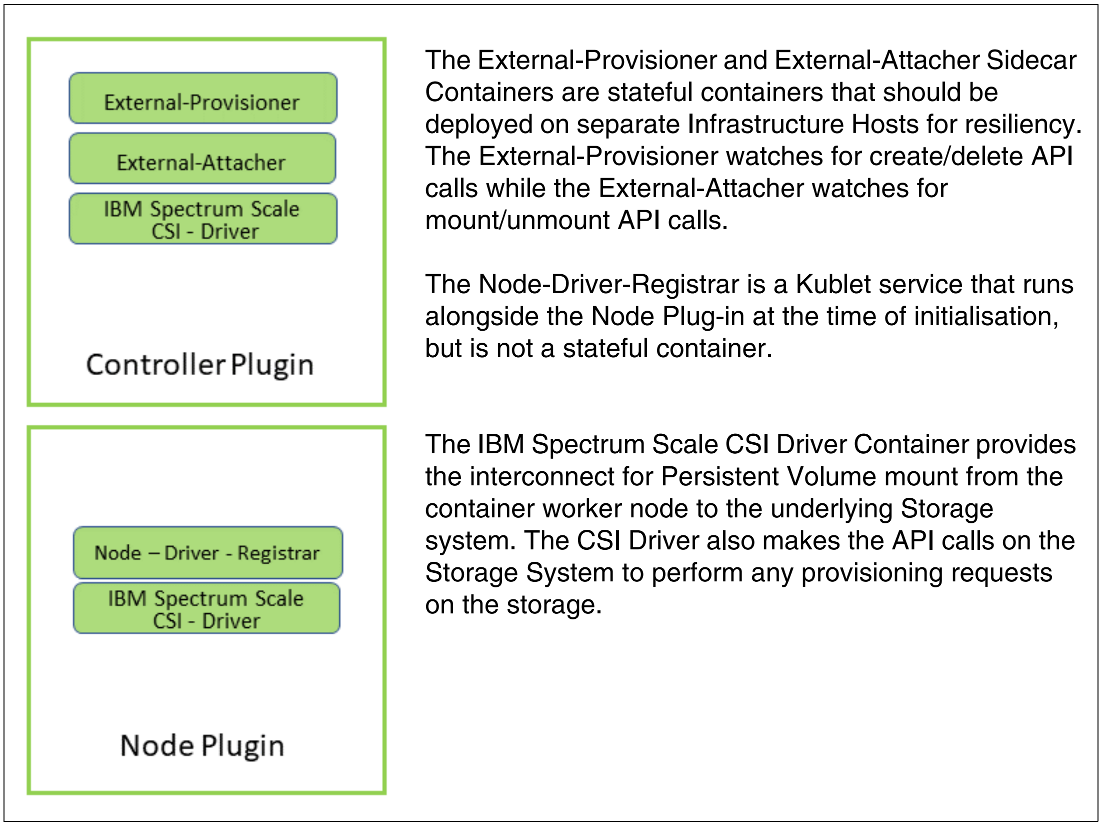
Figure 2-1 Block Diagram of IBM Spectrum Scale CSI Driver with Kubernetes Cluster

The IBM Spectrum Scale Container Storage Interface (CSI) driver enables IBM Spectrum Scale to be used as persistent storage for stateful applications running in Kubernetes clusters. Through this CSI Driver, Kubernetes persistent volumes (PVs) can be provisioned from IBM Spectrum Scale.

The IBM Spectrum Scale CSI driver supports the following features:

- **Operator-based deployment:** Availability of an operator to deploy the CSI driver
- **Static provisioning:** Ability to use existing directories as persistent volumes
- **Lightweight dynamic provisioning:** Ability to create directory-based volumes dynamically
- **File Set-based dynamic provisioning:** Ability to create file set-based volumes dynamically
- **Multiple file systems support:** Volumes can be created across multiple file systems
- **Remote mount support:** Volumes can be created on a remotely mounted file system

IBM implements the CSI Specification of Storage Plug-ins in the following manner shown in Figure 2-2.


External-Provisioner
External-Attacher
IBM Spectrum Scale CSI - Driver
Controller Plugin
Node – Driver - Registrar
IBM Spectrum Scale CSI - Driver
Node Plugin

The External-Provisioner and External-Attacher Sidecar Containers are stateful containers that should be deployed on separate Infrastructure Hosts for resiliency. The External-Provisioner watches for create/delete API calls while the External-Attacher watches for mount/unmount API calls.

The Node-Driver-Registrar is a Kublet service that runs alongside the Node Plug-in at the time of initialisation, but is not a stateful container.

The IBM Spectrum Scale CSI Driver Container provides the interconnect for Persistent Volume mount from the container worker node to the underlying Storage system. The CSI Driver also makes the API calls on the Storage System to perform any provisioning requests on the storage.

Figure 2-2 IBMs implementation of the CSI specification of storage plug-ins

IBM Spectrum Scale CSI Operator architecture

IBM Spectrum Scale CSI Operator is an Ansible-based operator for deployment and management of IBM Spectrum Scale CSI driver, Figure 2-3. This section describes the architecture of the CSI operator.

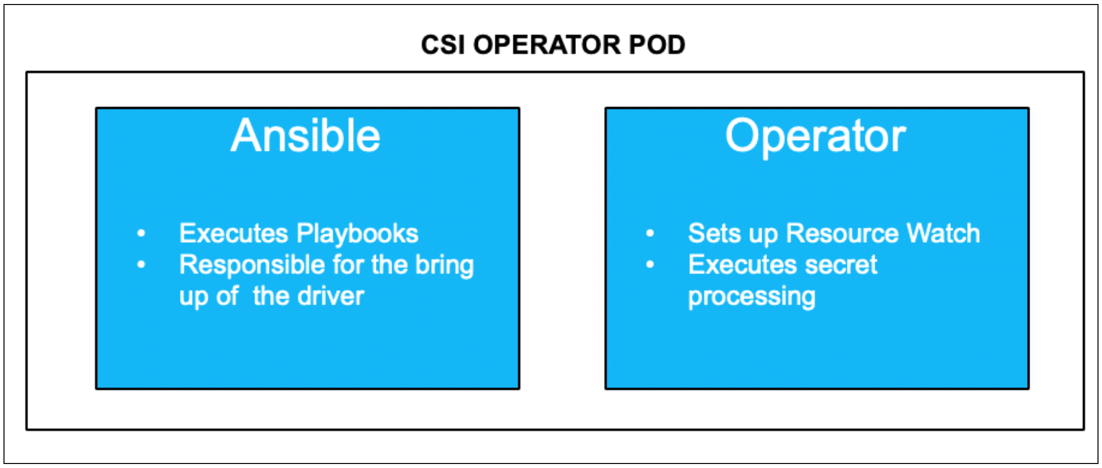


Figure 2-3 Component diagram of IBM Spectrum Scale CSI Operator

Operator is deployed as a Kubernetes deployment object. It consists of a POD with two containers:

- Ansible

  Ansible container is responsible for running playbooks for CSI driver management and deployment. This is responsible for bringing up the CSI driver.

- Operator

  Operator container watches for changes in CSI driver resources, such as **`daemonset`** and **`statefulset`**, and takes appropriate action if any change is detected. Operator container also handles changes in GUI secret that can typically happen after GUI password change or expiry. For more details, go to the following link:

  `https://github.com/IBM/ibm-spectrum-scale-csi-operator`

For more details about provisioning storage within a container environment, see Chapter 5, “Deployment and Administration” on page 29.

3

Solutions and Use Cases

IBM Spectrum Scale along with the IBM Spectrum Scale CSI driver offers unique capabilities to support solutions and use cases for the containerized applications landscape.

Moving to containerized applications is one of the foundations to be able to exploit the benefits of Multicloud environments. Modern container-based cloud environments provide:

- **Improved agility** through more efficient resource utilization and faster workload deployment times
- **Improved application elasticity** by leveraging the container environment scheduling and auto-scaling capabilities
- **Improved security** through service isolation and multi-tenancy
- **Efficient reuse of applications and services** in a DevOps environment
- **Simplified and accelerated application deployment** through ready-to-use CI/CD solution stacks

When considering application Lift and Shift, refactoring, or a green-field approach to the transformation of applications into a container-based microservice architecture, IBM recommends the IBM Garage™ Method approach, details of which can be found at the following website:

`https://www.ibm.com/garage/method`

The IBM Garage method is IBM’s approach to enable business, development, and operations to continuously design, deliver, and validate new solutions leveraging cloud technologies.

Furthermore, it is recommended to consider a Cloud Adoption Briefing:

`https://www.ibm.com/account/reg/us-en/signup?formid=urx-34892`

Or use the Cloud Transformation Advisor software at to get application transformation advise:

`https://www.ibm.com/garage/method/practices/learn/ibm-transformation-advisor`

This chapter describes various use cases for applications that are already containerized. Understanding these use cases will help for subsequent chapters of IBM Spectrum Scale CSI driver planning, deployment, and usage. Though not mandatory, it is recommended for all readers.

This chapter covers the following topics:

- 3.1, "Multiple containers accessing the same data"
- 3.2, "Multi-protocol access" on page 13
- 3.3, "Multi-tenancy use case" on page 14
- 3.4, "Remote file system access use cases" on page 16
- 3.5, "Kubernetes multi-cluster use cases" on page 17
- 3.6, "CSI driver for multisite use cases (AFM use cases)" on page 18
- 3.7, "Compressed Volumes use case" on page 20

3.1 Multiple containers accessing the same data

Microservices applications that require persistent data to be shared across multiple services have the choice to follow two different approaches.

With the independent reader/writer approach, each service reads and writes independent fragments of the application data. Data distribution and synchronization is done on the application level. An example for this pattern is an application that replicates data across multiple service instances to provide high availability or load balancing. The MongoDB NoSQL databases horizontal sharding uses this pattern to distribute workload across multiple server instances.

A different approach is reading and writing the same data by leveraging a shared storage solution across the different service instance hosts. This approach would be used if maximum data reuse is required while the data availability is handled by the underlying storage solution. IBM Spectrum Scale as a high performance, very resilient, and highly available storage solution strongly supports this approach.

Figure 3-1 on page 13 shows a Kubernetes cluster with an exemplary microservices application consisting of two containers (Microservice A and Microservice B). Both containers read and write to common data that is exposed by the IBM Spectrum Scale file system through the CSI driver to Microservice A and Microservice B.

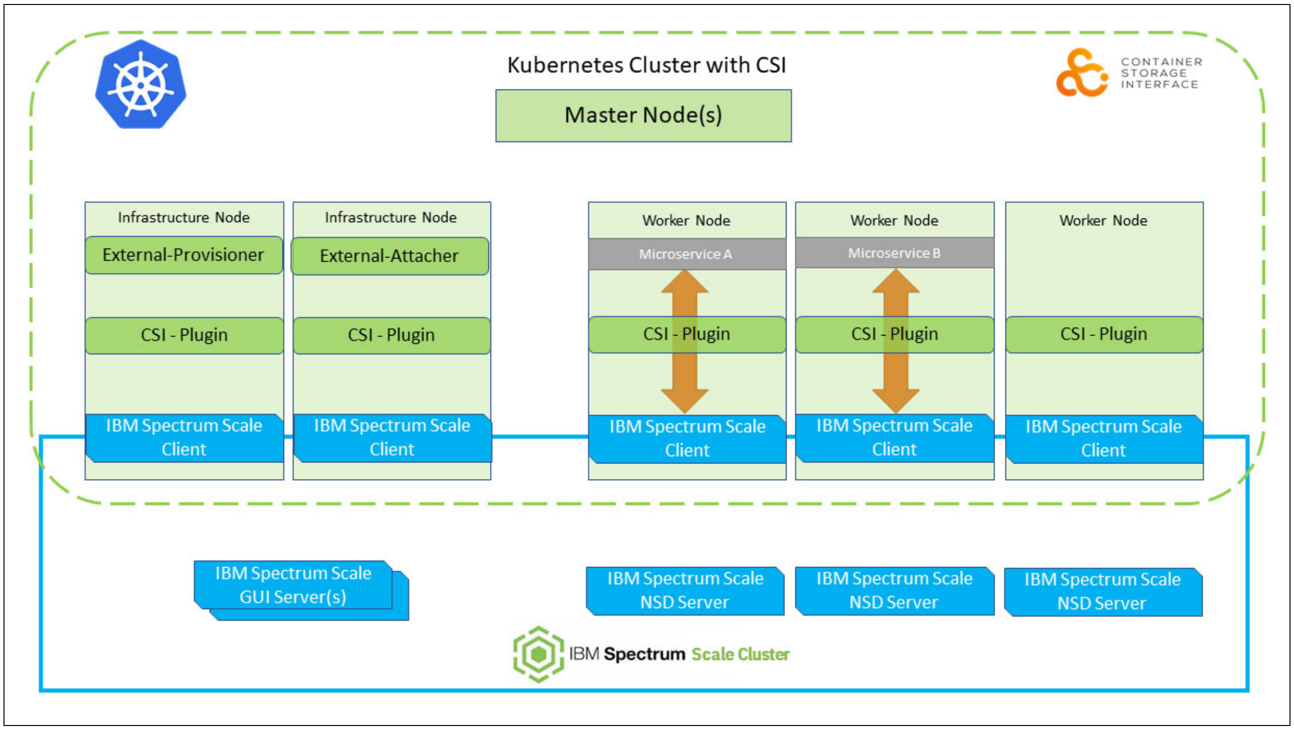

Kubernetes Cluster with CSI
Master Node(s)
CONTAINER STORAGE INTERFACE
Infrastructure Node
External-Provisioner
Infrastructure Node
External-Attacher
Worker Node
Microservice A
Worker Node
Microservice B
Worker Node
CSI - Plugin
IBM Spectrum Scale Client
IBM Spectrum Scale GUI Server(s)
IBM Spectrum Scale NSD Server
IBM Spectrum Scale Cluster

Figure 3-1 Basic multi reader/writer microservices use case

The backing image store for a container registry (like Red Hat Quay) in a highly available setup is a practical example for this use case. Multiple container registry service instances need access to shared container images that are stored on a shared file system. If one instance should fail, the remaining instances still have access to the container image data.

Because container images might have a substantial size (up to 100s of MB), storing them only once using a shared IBM Spectrum Scale file system is very cost-efficient. When the container registry is configured to provide active-active HA for scalability, the simultaneous access to images requires a high-performing storage backend, such as IBM Spectrum Scale.

This capability can be achieved by using `ReadWriteMany` access mode defined in persistent volume claim configuration, which will enable multiple pod to consume one persistent storage at the same time. For details about using `ReadWriteMany` access mode during volume creation, see Chapter 5, “Deployment and Administration” on page 29.

3.2 Multi-protocol access

Research institutes use data collection scientific instruments that rely on NAS protocols, such as NFS and SMB, to deliver the data they generate.

Automotive companies participating in the area of advanced driver-assistance systems (ADAS) use similar devices to ingest the data collected within the cars through NAS protocols to centralized storage systems within their corporate network. When using IBM Spectrum Scale for these use cases, data can be ingested through the IBM Spectrum Scale CES protocol services for NFS and SMB, and then be used and analyzed by containerized services running in a Kubernetes cluster.

This capability of consuming NFS data from Kubernetes/OpenShift container can be achieved by using the static provisioning feature of IBM Spectrum Scale CSI driver, where data written by NAS applications should be statically provisioned as volumes to be made accessible to containers.

Such a scenario is depicted in Figure 3-2, where the IBM Spectrum Scale cluster has additional IBM Spectrum Scale Cluster Export Service (Protocol) nodes outside the Kubernetes cluster that could be used for high-performance data ingest. The data is then analyzed using big data analytics (BDA) or artificial intelligence (AI) Machine Learning/Deep Learning (ML/DL) applications, which are commonly deployed in a containerized way. After analysis, the results are written to some persistent storage as well, but because this is usually a different volume than the one used for ingest, this is not shown in Figure 3-2.

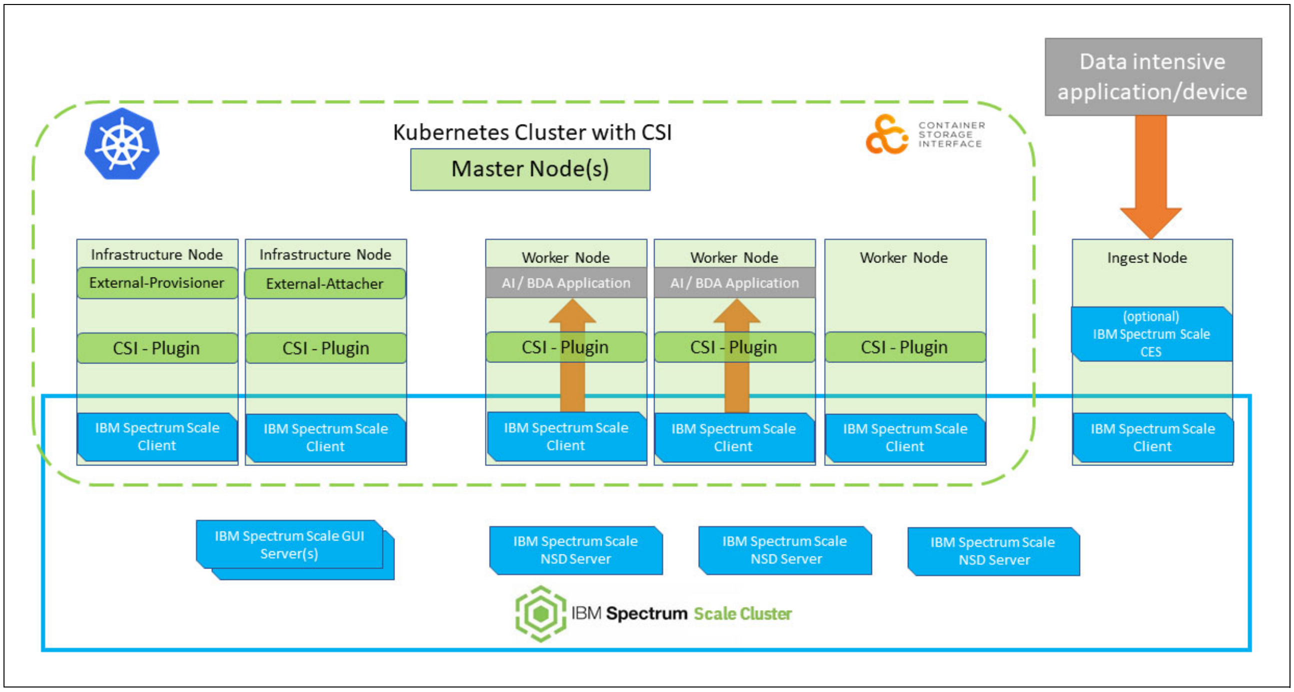

Data intensive application/device
Kubernetes Cluster with CSI
CONTAINER STORAGE INTERFACE
Master Node(s)
Infrastructure Node
External-Provisioner
CSI - Plugin
IBM Spectrum Scale Client
Infrastructure Node
External-Attacher
CSI - Plugin
IBM Spectrum Scale Client
Worker Node
AI / BDA Application
CSI - Plugin
IBM Spectrum Scale Client
Worker Node
AI / BDA Application
CSI - Plugin
IBM Spectrum Scale Client
Worker Node
CSI - Plugin
IBM Spectrum Scale Client
Ingest Node
(optional) IBM Spectrum Scale CES
IBM Spectrum Scale Client
IBM Spectrum Scale GUI Server(s)
IBM Spectrum Scale NSD Server
IBM Spectrum Scale NSD Server
IBM Spectrum Scale NSD Server
IBM Spectrum Scale Cluster

Figure 3-2 Multi-protocol use case

3.3 Multi-tenancy use case

Multilevel department organization where microservices applications require persistent data to be stored and shared across multiple departments, data can be ingested through applications to IBM Spectrum Scale file system and can be reused for reporting and archive purposes.

Universities participating in multiple research projects and researchers produce data during their research, or need access to the existing data, which is backed by IBM Spectrum Scale file system. The data can be consumed using IBM Spectrum Scale CSI driver in microservices applications on container orchestrator platforms, such as Kubernetes, OpenShift, and so on.

Such a scenario is depicted in Figure 3-3, where the University has multiple colleges and the colleges have multiple schools, such as school of engineering, school of management, and so on. In each school, researchers might be working on multiple research projects. Today, the container orchestrator platform environment can be used to run microservice applications in the university, and data can be ingested through container instances to IBM Spectrum Scale file system.

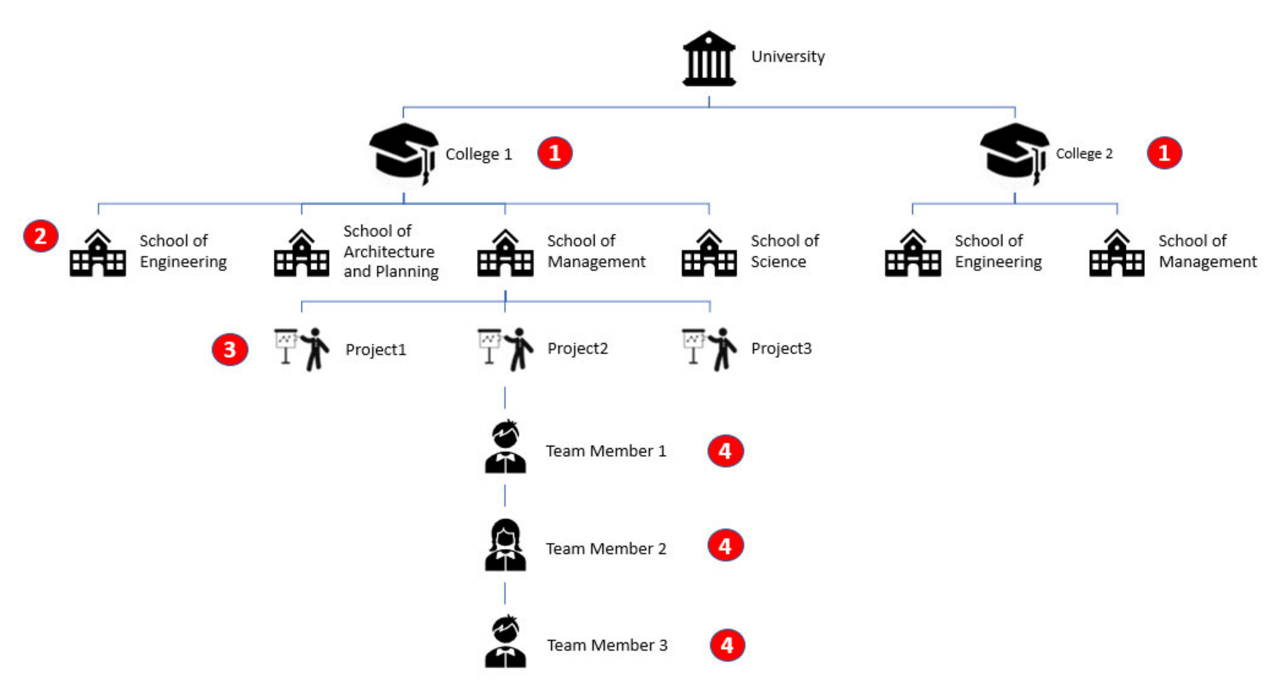


Figure 3-3 University use case with multiple colleges with multiple schools

In Figure 3-3, consider that a university has one IBM Spectrum Scale cluster and that the admin can create multiple file systems underneath:

- Each college can be assigned with one file system.
- Each school can be assigned with one independent file set based on storage requirements (multiple file set-based policies can be applied to this file set based on the requirement, and a snapshot can also be taken at the file set level).
- Each research project can be assigned with one dependent file set, where the parent file set will be the school's independent file set under which the project is running.
- Each research team member can be assigned with a lightweight directory inside the dependent file set. The Kubernetes/OpenShift admin can create users per storageclass for each research project.

For detailed information to achieve this file set/lightweight directory based provisioning in Kubernetes/OpenShift container orchestrator environment, Refer to Chapter 5, "Deployment and Administration" on page 29.

3.4 Remote file system access use cases

IBM Spectrum Scale provides a way to access a file system from another IBM Spectrum Scale cluster. Such remote file system access is described in IBM Knowledge Center for IBM Spectrum Scale in the section related to Accessing a remote GPFS file system.

Some of the common use cases when remote file system access is used are:

- Separation of IBM Spectrum Scale clients and IBM Spectrum Scale NSD servers for administration purposes
- Network isolation of multiple clients within a cluster
- Configuring different protocol authentication mechanisms for different protocol (CES) clusters

The same use cases also apply to the Kubernetes and OpenShift environments. For these environments, other than setting up remote file system access, additional configuration is required during deployment and configuration. This section describes the use case, deployment, and configuration considerations when using remote file system access.

3.4.1 Separation of IBM Spectrum Scale clients and NSD servers

You can set up a different IBM Spectrum Scale cluster for clients that are part of a Kubernetes cluster to isolate their administration from IBM Spectrum Scale NSD server administration.

Figure 3-4 shows the architecture when IBM Spectrum Scale clients in a Kubernetes cluster are configured in a separate IBM Spectrum Scale cluster.

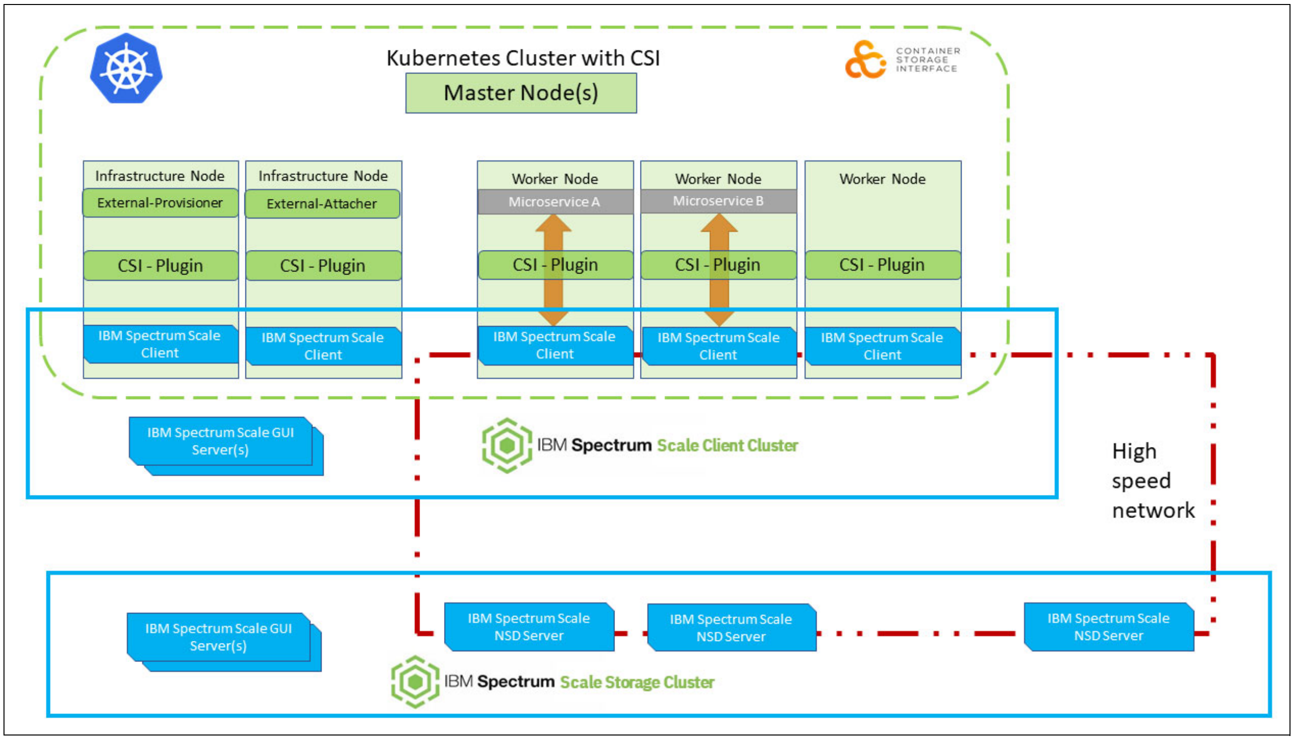

Kubernetes Cluster with CSI
CONTAINER STORAGE INTERFACE
Master Node(s)
Infrastructure Node
External-Provisioner
Infrastructure Node
External-Attacher
Worker Node
Microservice A
Worker Node
Microservice B
Worker Node
CSI - Plugin
IBM Spectrum Scale Client
IBM Spectrum Scale GUI Server(s)
IBM Spectrum Scale Client Cluster
High speed network
IBM Spectrum Scale NSD Server
IBM Spectrum Scale Storage Cluster

Figure 3-4 Remote file system access: IBM Spectrum Scale configuration

The configuration in Figure 3-4 on page 16 shows setting up two different IBM Spectrum Scale clusters as follows:

- IBM Spectrum Scale Storage cluster: You can configure NSD servers and GUI server as part of this cluster (referred as remote cluster in subsequent paragraphs). The remote GUI servers are used by IBM Spectrum Scale CSI driver to create dynamic persistent volumes that are mapped to new dependent or independent filesets.
- IBM Spectrum Scale Client cluster: In this cluster, you can configure all Scale client nodes that are part of Kubernetes cluster including Scale clients on Kubernetes infrastructure nodes and worker nodes. You also need to configure local IBM Spectrum Scale GUI servers as part of this cluster that are used to create static or dynamic light-weight volumes based on directories.
- Optionally, you can configure all the Scale clients on Kubernetes worker nodes and Scale NSD servers as part of the same high-speed network for exchanging data or daemon traffic while keeping administration isolated within each cluster.

The configuration for this environment involves the following steps:

1. Enable and mount the IBM Spectrum Scale file system in the IBM Spectrum Scale Storage cluster to be accessible via IBM Spectrum Scale client nodes cluster. See IBM Knowledge Center for IBM Spectrum Scale Accessing a remote GPFS file system section for instructions on setting up the remote file system access.
2. Configure a remotely mounted file system as local storage along with a remote GUI server and local GUI server.
3. Configure PV using the remotely mounted storage as local storage.

For details about configuring a remote cluster with IBM Spectrum Scale CSI driver, see Chapter 5, “Deployment and Administration” on page 29.

3.5 Kubernetes multi-cluster use cases

Companies need to operate multiple Kubernetes/OpenShift clusters depending on the workload and service levels required for the applications. In some cases, it is required to share the same set of data not only within the single cluster but also across the multi-cluster. IBM Spectrum Scale provides the scalability and flexibility in performance and capacity with the single storage name space and that enables users to centralize their data management and avoid the silo of a data store.

These use cases of the data sharing with multi-cluster are typical:

AI and Big Data	There are several applications running in AI and Big Data pipeline including “Data Injection”, “Data Preparation/Processing”, “Training,” and “Inference”. In that case, one Kubernetes/OpenShift cluster is being used by many applications that generate a large amount of data and another Kubernetes/OpenShift cluster is being used to run analytics over that data. In order to support the whole pipeline, there is a need for sharing the data between the two Kubernetes/OpenShift clusters. Rather than creating multiple copies of data, IBM Spectrum Scale CSI Driver provides a better approach to share data between two or more clusters.
High Availability	Users prepare multi-cluster and run application pods on them to ensure high availability. For instance, users prepare two Kubernetes/OpenShift clusters, then run the production pods on one cluster and prepare a hot/cold standby pod on another cluster.

In that case, it is required to share the same set of data between production and the hot/cold standby pod. IBM Spectrum Scale CSI driver has functions to create the Persistent Volume from the existing directories, and that enables users to easily share the data across multi-cluster.

The configuration in Figure 3-5 describes the data sharing model between two Kubernetes clusters. Users can share existing directories with the Static provisioning volume. The access control can be managed by applying the accessMode for the Persistent Volume.

IBM Spectrum Scale supports the following accessMode:

ReadWriteOnce (RWO)	Single node can perform read and write.
ReadWriteMany (RWM)	Multiple nodes can perform read and write.

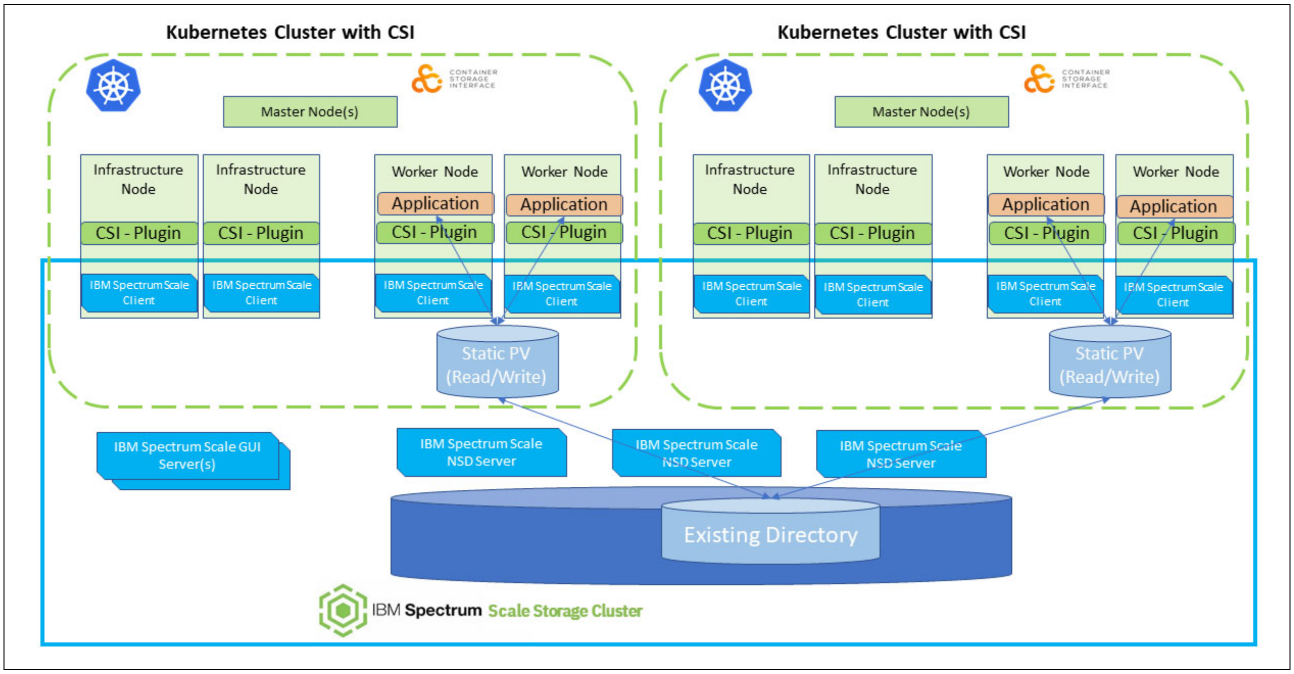

Kubernetes Cluster with CSI
Kubernetes Cluster with CSI
CONTAINER STORAGE INTERFACE
Master Node(s)
Infrastructure Node
Worker Node
Application
CSI - Plugin
IBM Spectrum Scale Client
Static PV (Read/Write)
IBM Spectrum Scale GUI Server(s)
IBM Spectrum Scale NSD Server
Existing Directory
IBM Spectrum Scale Storage Cluster

Figure 3-5 Data sharing model between two Kubernetes clusters

Note: When sharing the volume created by the Dynamic provisioning, the path of the Dynamic provisioning volume will be automatically deleted at the Persistent Volume Claim deletion because the reclaim policy of the Dynamic provisioning volume is “Delete” by default. It is recommended to use the Static volume provisioning with the existing directories and use “Retain” reclaim policy.

3.6 CSI driver for multisite use cases (AFM use cases)

IBM Spectrum Scale has an asynchronous caching mechanism, Active File Management, that can be used to implement multisite use cases. This can be achieved even on a public cloud setup where and if AFM, GUI, and CSI driver are supported, enabling hybrid cloud.

Note: As of the writing of this Redpaper publication, IBM Spectrum Scale on AWS 1.2.0 does not enable GUI by default so it would not be supported in this version. In a future version when all of the needed features are supported, the multisite feature will enable hybrid multi-cloud out-of-the-box.

Static provisioning on directories that reside inside of an AFM file set should be used to provide persistent volumes, ensuring that data will be used as is and not deleted after the pod is deleted. Make sure that you are familiar with all of the concepts and limitations of AFM, because they will all apply to the multisite use case. See the following documentation:

`https://www.ibm.com/support/knowledgecenter/en/STXKQY_5.0.4/com.ibm.spectrum.scale.v5r04.doc/bl1ins_activefilemanagement.htm`

Depending on the use case, a different AFM mode can be used to cache data across sites. The two use cases we show here are the Independent Writer (IW) and the Single Writer (SW) with Read Only (RO) modes. On our example, we use a MongoDB stateful pod that uses the static provisioned volume on all clusters.

The implementation of the Independent Writer AFM mode is done updating all the caches sites with the latest metadata and data pointers found in the home cluster. All the writes done on the file set are asynchronously sent to the home and no locking is done across caches.

Because of this, the user implementation of the use case must ensure correct usage not to cause unintended data corruption. Therefore, the MongoDB can never be up on more than one site at a time. In this use case, the only automation needed is the periodic prefetch of the data to ensure minimum data transfer is needed in case the activation of the cache side is performed. Figure 3-6 depicts the implementation of the case.

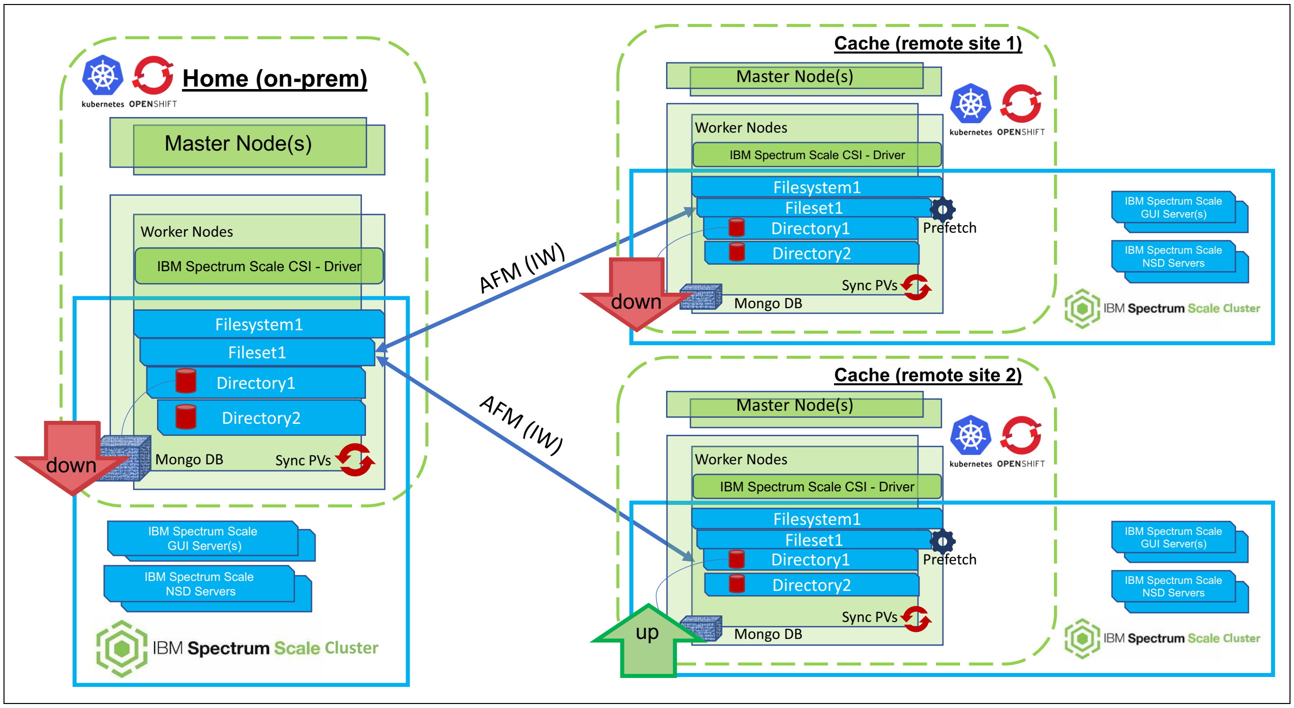


Figure 3-6 Multisite use case with AFM independent writer

Implementing an Independent writer architecture also implies that from time to time the cache must go back to the home and see if any data has changed, and if so, replicate any metadata updates that are needed. Be aware that this can add latency because the check from each cache needs to be performed on home regularly.

To avoid the downsides of the Independent Writer on a multi-cloud case, one might use the Read only and Single Writer modes. The changes on the mode on each cache need to be automated and write locking will be enforced on all RO caches. The client must ensure all caches that are not active are Read Only when implementing this use case to avoid unwanted writes corrupting data. If the workload is running on home, all caches must be RO. If it is running on one of the caches, only the one that is running the workload must be a writer, other sites must be RO. This scenario is shown in Figure 3-7.

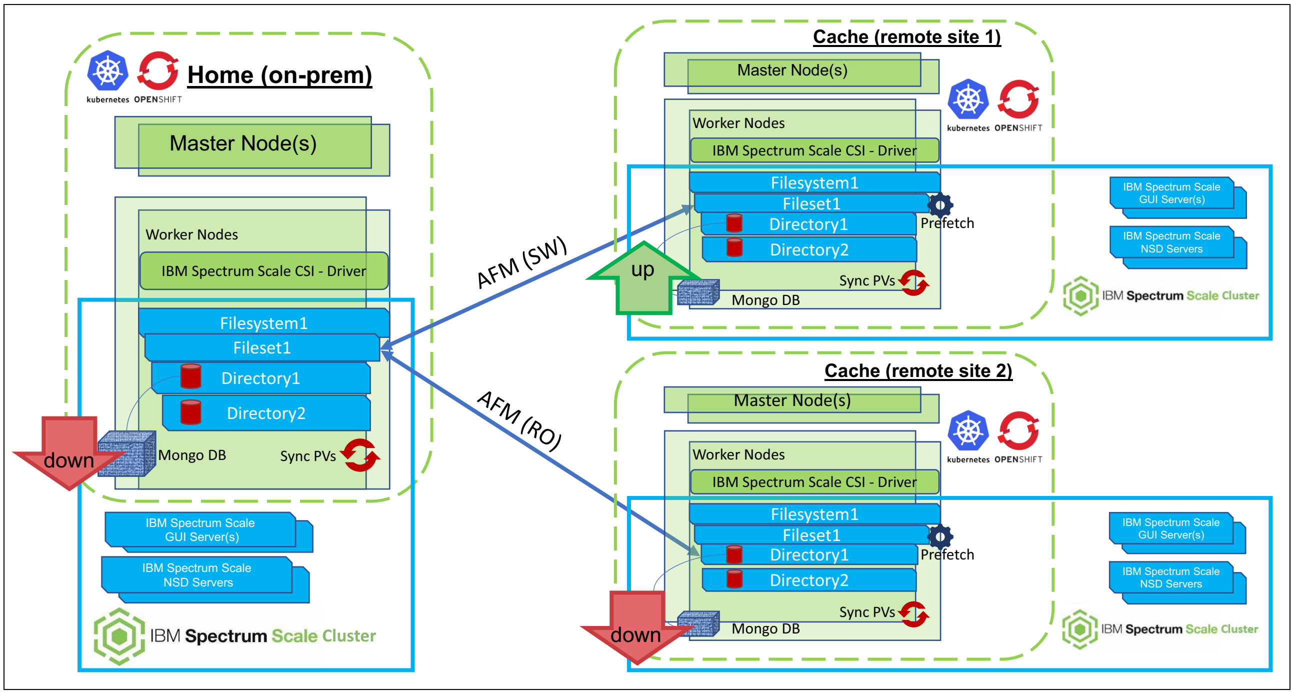


Figure 3-7 Multisite use case with AFM Single Writer

Although this mode will not have the data change verification delay like the Independent Writer mode, it will require an automation to change the mode from SW to RO and vice-versa when migrating the workload across clusters.

3.7 Compressed Volumes use case

This section describes how one could leverage IBM Spectrum Scale's file compression feature in order to achieve volume compression for persistent volumes for containers.

IBM Spectrum Scale Compression

File compression is an important technology for storage products. Compression improved overall storage space efficiency. It also reduces I/O bandwidth consumption resulting in reduced load on the storage back-end. Further, caching compressed data on the client increases the apparent cache size. IBM Spectrum Scale 5.0.0 supports the LZ4 compression algorithm. LZ4 is a much faster compression algorithm, with decompression speed up to 5 times better than zlib.

When to use file compression

ZLIB is intended primarily for cold objects and files. It favors saving space over read-access speed. Compressing other types of data can result in performance degradation. LZ4 is intended primarily for active data and favors read-access speed over maximized space saving.

Sample use case

Due to legal and compliance requirements, applications need to maintain data/logs for a predefined period. If such applications are deployed in a container environment that uses Persistent Storage volume for storing its data/logs, one can use the compression functionality available with IBM Spectrum Scale for compressing the files in the persistent volume. Usually logs have a good compression ratio.

To use IBM Spectrum Scale functionality with IBM Spectrum scale CSI driver, complete the following steps:

1. Create PV/PVC.
2. Write a compression policy based on your requirements.
3. Apply the compression policy based on your requirement.

For detailed steps and an example, see A.2, “Compression use case” on page 61. See the **`mmapplypolicy`** command and file compression policy for more details:

https://www.ibm.com/support/knowledgecenter/en/STXKQY_5.0.4/com.ibm.spectrum.scale.v5r04.doc/bl1adm_mmapplypolicy.htm

https://www.ibm.com/support/knowledgecenter/en/STXKQY_5.0.4/com.ibm.spectrum.scale.v5r04.doc/bl1adm_compression.htm

4

Planning for IBM Spectrum Scale CSI driver deployment

There are several components that must be installed and setup before deploying and using IBM Spectrum Scale CSI driver. An understanding of various deployment mechanisms and topologies of these components will help meet the pre-requisites for IBM Spectrum Scale CSI driver.

In this chapter we cover the following topics:

- 4.1, “Deployment on Kubernetes” on page 24
- 4.2, “Deployment on OpenShift” on page 25
- 4.3, “IBM Spectrum Scale deployment” on page 27
- 4.4, “What type of volumes do I need” on page 28
- 4.5, “Limitations of IBM Spectrum Scale CSI driver” on page 28

This chapter is recommended for all administrators.

4.1 Deployment on Kubernetes

This section describes the IBM Spectrum Scale components that need to be installed on each of the Kubernetes cluster nodes:

- **Worker nodes:** You need to install IBM Spectrum Scale client and IBM Spectrum Scale CSI driver on the Worker nodes on which you need access to persistent volumes.
- **Infrastructure nodes:** You need to install IBM Spectrum Scale client and IBM Spectrum Scale CSI driver on infrastructure nodes, such as Attacher node or Provisioner node. The infrastructure node must be able to communicate with IBM Spectrum Scale GUI server in the IBM Spectrum Scale cluster.
- **Master node:** Do not include this node as part of IBM Spectrum Scale cluster.

See Figure 4-1 for typical deployment on a Kubernetes cluster. the following are recommendations for IBM Spectrum Scale cluster nodes:

- GUI nodes, protocol nodes, and AFM nodes should be configured outside of the Kubernetes cluster.
- NSD nodes are to be outside of Kubernetes cluster but can be included in Kubernetes cluster for Shared Nothing Cluster (SNC) setup.
- IBM Spectrum Scale GUI node is a prerequisite for the IBM Spectrum Scale CSI driver. The IBM Spectrum CSI Driver uses the GUI REST API feature to perform all of its operations. Currently, CSI Driver can be configured to use only a single GUI node in IBM Spectrum Scale Cluster.

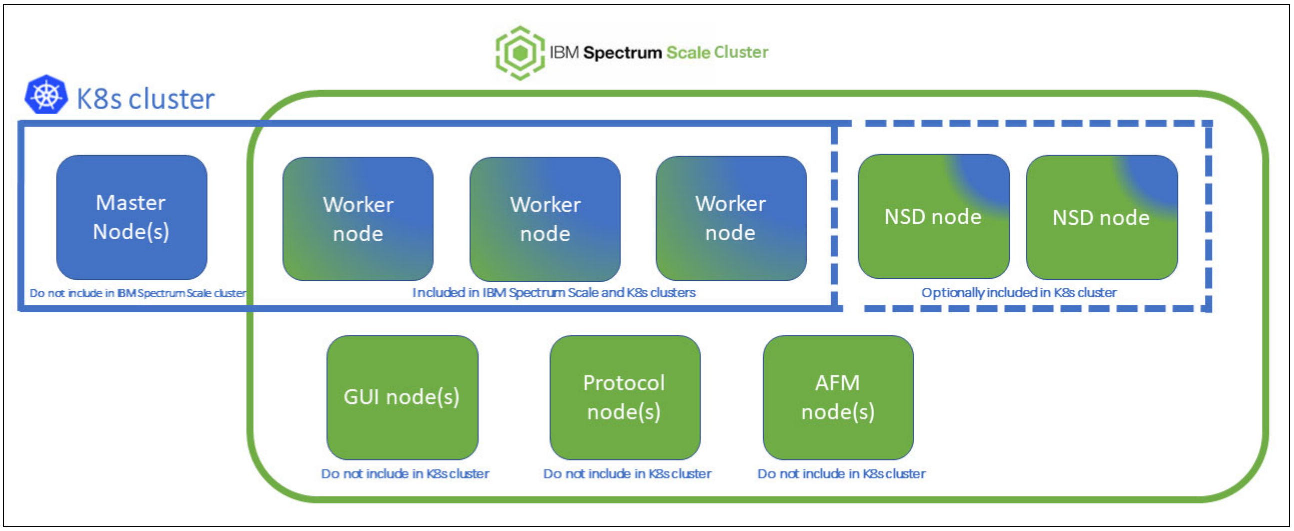

IBM Spectrum Scale Cluster
K8s cluster
Master Node(s)
Do not include in IBM Spectrum Scale cluster
Worker node
Worker node
Worker node
Included in IBM Spectrum Scale and K8s clusters
NSD node
NSD node
Optionally included in K8s cluster
GUI node(s)
Do not include in K8s cluster
Protocol node(s)
Do not include in K8s cluster
AFM node(s)
Do not include in K8s cluster

Figure 4-1 Typical deployment on a Kubernetes cluster

See IBM Knowledge Center section IBM Spectrum Scale Container Storage Interface Driver for additional deployment considerations.

4.1.1 Kubernetes deployment mechanism

Kubernetes can be deployed with various tools described in this section and the majority used for Kubernetes deployment methods along with components such as container runtime and network plug-ins.

IBM Spectrum Scale CSI driver is tested with Kubeadm, Kubespray, and Minikube. You can get more information about each of these methods using the following links:

Kubeadm: `https://kubernetes.io/docs/reference/setup-tools/kubeadm/kubeadm`

Kubespray: `https://kubernetes.io/docs/setup/production-environment/tools/kubespray`

Minikube: `https://kubernetes.io/docs/tutorials/hello-minikube`

4.1.2 Container runtime

A container runtime is software that executes containers and manages container images on a node. There are many container runtimes available. However, currently the most widely used container runtime is Docker. IBM Spectrum Scale CSI driver is tested with Docker and CRI-O. You can use the following links for more information on the container runtime:

Docker: `https://docs.docker.com/engine/docker-overview`

CRI-O: `https://cri-o.io`

4.1.3 Network plug-in

Container networking supports many network plug-ins currently available. Typically, Kubernetes use Calico and OpenShift uses OVS. The IBM Spectrum Scale CSI driver has been tested with Calico and OVS.

4.2 Deployment on OpenShift

OpenShift 4 introduces a new way of installing the platform that is automated, reliable, and repeatable. Based on the Kubernetes Cluster-API SIG, Red Hat has developed an OpenShift installer for full stack automated deployments. With this release, the installer not only installs OpenShift, but it installs (and manages) the entire infrastructure as well, from DNS all the way down the stack to the VM. This provides a fully integrated system that can resize automatically with the needs of your workload. Currently, full stack automated deployment is supported on AWS. Currently, guides for pre-existing infrastructure installs exist for AWS, VMWare vSphere, and bare metal.

IBM Spectrum Scale CSI is tested with Kubernetes Standard Deployment, OpenShift with Bare Metal, VMware vSphere, and AWS.

See Figure 4-2 for the IBM Spectrum Scale CSI driver deployment architecture on OpenShift.

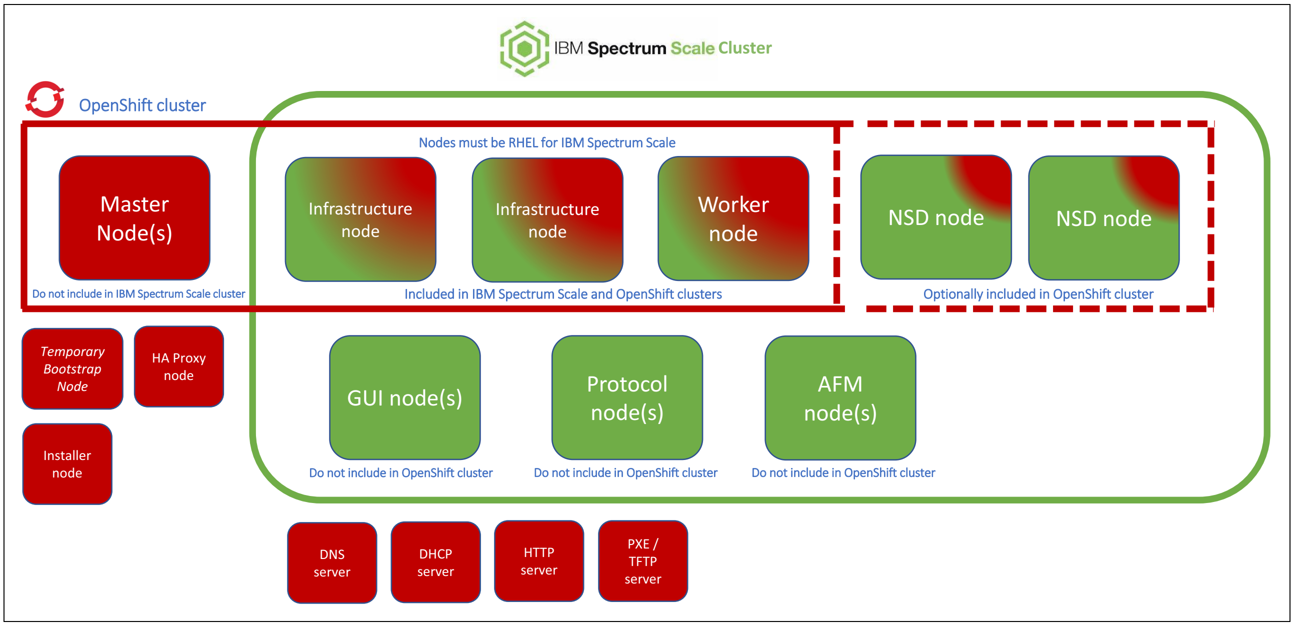

IBM Spectrum Scale Cluster
OpenShift cluster
Nodes must be RHEL for IBM Spectrum Scale
Master Node(s)
Infrastructure node
Infrastructure node
Worker node
NSD node
NSD node
Do not include in IBM Spectrum Scale cluster
Included in IBM Spectrum Scale and OpenShift clusters
Optionally included in OpenShift cluster
Temporary Bootstrap Node
HA Proxy node
Installer node
GUI node(s)
Protocol node(s)
AFM node(s)
Do not include in OpenShift cluster
Do not include in OpenShift cluster
Do not include in OpenShift cluster
DNS server
DHCP server
HTTP server
PXE / TFTP server

Figure 4-2 IBM Spectrum Scale CSI driver deployment on OpenShift

OpenShift on Bare Metal

OpenShift Bare Metal requires a minimum of one bootstrap machine; three control planes, or *master*, machines; and at least two computes, or *worker*, machines. IBM Spectrum Scale is deployed on worker nodes.

For more detail about an OpenShift Bare Metal install, see the following link:

https://docs.openshift.com/container-platform/4.2/installing/installing_bare_metal/installing-bare-metal.html#installation-requirements-user-infra_installing-bare-metal

The OpenShift deployment master nodes are on CoreOS and worker nodes are on RHEL. IBM Spectrum Scale and CSI are deployed on a worker node.

OpenShift on vSphere

Red Hat and VMware are collaborated to integrate OpenShift Container Platform and Funware's software-defined data center (SDDC) infrastructure stack. VMware and Red Hat both embrace Kubernetes as the core platform supporting their modern applications. IBM Spectrum Scales deployed on worker nodes.

For more Information about installing OpenShift on vSphere, see the following link:

https://docs.openshift.com/container-platform/4.2/installing/installing_vsphere/installing-vsphere.html

4.3 IBM Spectrum Scale deployment

In order for the CSI driver to function optimally, it is important to pay attention to some of the deployment aspects of IBM Spectrum Scale.

4.3.1 IBM Spectrum Scale deployment models

The IBM Spectrum Scale CSI driver can be used with IBM Spectrum Scale cluster that's configured to use Shared storage or NSD Server model. The IBM Spectrum Scale CSI driver can also be used in IBM Spectrum Scale configurations that use ESS storage; but Kubernetes, OpenShift, and CSI components cannot be installed on ESS I/O nodes.

4.3.2 Network considerations

The following network guidelines need to be considered during IBM Spectrum Scale CSI driver planning and implementation:

- The infrastructure node must be able to communicate with IBM Spectrum Scale GUI server in the Scale cluster
- If you are using storage from remote IBM Spectrum Scale cluster, the infrastructure nodes must be able to communicate with GUI servers in local IBM Spectrum Scale cluster and remote IBM Spectrum Scale cluster
- In case of higher performance requirement, you can configure the Kubernetes worker nodes and NSD server nodes in the Storage cluster on a high speed network.

4.3.3 Primary File System

During IBM Spectrum Scale CSI driver initialization one cluster must be designated as the *primary cluster*, and one file system within that cluster must be designated as the *primary file system* in the CSI driver configuration.

The CSI driver can be configured to use multiple IBM Spectrum Scale clusters in a remotely mounted cluster relationship but the primary cluster is the one whose nodes are part of the Kubernetes cluster. There can only be one cluster that can be used as a primary cluster.

The primary file system is a file system within the primary cluster that is used for storing links to all PVCs. This file system must be created by the IBM Spectrum Scale administrator before initializing the CSI driver. It must be mounted on all required worker nodes and the GUI node. A *primary file set* is created by the CSI driver within this file system if it doesn't exist. When PVCs are provisioned, softlinks to the corresponding directories/file sets are created inside this file set.

Other file systems can be created in IBM Spectrum Scale cluster anytime before or after CSI driver initialization, and can be used in the `storageclass` parameter for PVC provisioning. See Chapter 5, "Deployment and Administration" on page 29 for further details on configuration.

4.4 What type of volumes do I need

IBM Spectrum Scale CSI driver supports following three types of volumes

- Lightweight volumes
  - A new directory is created for every new volume provisioned dynamically using lightweight volume storageclass
  - Suitable for creating a large number of volumes (>10000)
  - PVC size cannot be imposed on storage due to lack of directory level quota support
- Independent file set based volumes
  - A new independent file set is created for every new volume provisioned dynamically using independent file set storageclass
  - Suitable when the number of volumes required is less than 1000
  - PVC size can be honored on storage by means of file set quota
  - IBM Spectrum Scale administrator can leverage management benefits of file sets on these volumes, such as snapshots and backup
- Dependent file set based volumes
  - A new dependent file set is created for every new volume provisioned dynamically using dependent file set storageclass
  - Suitable when number of volumes required is less than 10000
  - PVC size can be honored on storage by means of file set quota

There are two ways in which volumes can be provisioned

- Dynamic provisioning: The resource is created on IBM Spectrum Scale file system dynamically during PVC creation.
- Static provisioning: The resource is pre-existing on IBM Spectrum Scale file system and can be used to define a PV.

4.5 Limitations of IBM Spectrum Scale CSI driver

For limitations of IBM Spectrum Scale CSI driver see the following IBM Knowledge Center document:

https://www.ibm.com/support/knowledgecenter/STXKQY_5.0.4/com.ibm.spectrum.scale.csi.v5r04.doc/bl1csi_limitations.html

5

Deployment and Administration

The CSI drivers of IBM Spectrum Scale provide multiple provisioning models: *static provisioning*, *lightweight dynamic provisioning*, and *file set based dynamic provisioning*. Users can choose the provisioning model depending on the use cases and the workload of applications.

This chapter covers the following topics:

- 5.1, “IBM Spectrum Scale CSI Driver configuration” on page 30
- 5.2, “Deployment of IBM Spectrum Scale CSI Driver” on page 30
- 5.3, “Volume provisioning” on page 33
- 5.4, “Using the volume in a container” on page 35
- 5.5, “Releasing the volume on pod deletion” on page 36
- 5.6, “Node Selector” on page 37
- 5.7, “Kubernetes to IBM Spectrum Scale Cluster node mapping” on page 37
- 5.8, “Upgrading IBM Spectrum Scale Container Storage Interface driver” on page 37

This understanding will help for configuring and deploying the IBM Spectrum Scale CSI driver on the Kubernetes and OpenShift platform for consuming persistent storage for containers. This will also help in provisioning the volume to ingest the data on the IBM Spectrum Scale file system.

5.1 IBM Spectrum Scale CSI Driver configuration

The Container Storage Interface (CSI) Driver for IBM Spectrum Scale storage systems enables container orchestrators such as Kubernetes to manage the life cycle of persistent storage. There is the official operator to deploy and manage the IBM Spectrum Scale storage CSI driver.

When using the IBM Spectrum Scale file system as the back-end storage for persistent volumes, it is required to run the CSI drivers on the specific nodes of IBM Spectrum Scale clusters. The node types and the required conditions of IBM Spectrum Scale and Kubernetes are described in Table 5-1.

Table 5-1 Node type required conditions of IBM Spectrum Scale and Kubernetes

Node Type	IBM Spectrum Scale	Kubernetes
Master Node(s)	Not required	Required
Worker Node(s)	Required	Required
IBM Spectrum Scale GUI node(s)	Required	Not required
IBM Spectrum Scale NSD node(s)	Required	Not required

Supported container platforms:

- Supported OS: Red Hat 7.5, 7.6, 7.7
- IBM Spectrum Scale version: 5.0.4.1+
- Kubernetes version: 1.13 to 1.17
- OpenShift version: 4.2

5.2 Deployment of IBM Spectrum Scale CSI Driver

There are two methods to deploy and initialize IBM Spectrum Scale CSI driver:

- Using Operator Lifecycle Manager (OLM)
- Using Operator with command-line interface

5.2.1 Using Operator Lifecycle Manager

OLM runs by default on Red Hat OpenShift Container Platform. Therefore, it is a preferred method to use OLM on Red Hat OpenShift platform. For more information, see Understand OLM. The following steps describe how to deploy IBM Spectrum Scale CSI driver using Operator Lifecycle Manager (OLM).

1. For Kubernetes, because OLM is not installed by default, install OLM:
   `https://github.com/operator-framework/operator-lifecycle-manager/blob/master/doc/install/install.md`
2. Create the Operator from the Red Hat OpenShift console, as follows:
   a. From the left panel, click **Operators** → **OperatorHub** page on Red Hat OpenShift Container Platform management portal.
   b. From the Project drop-down list, select the project or create a new project by clicking **Create Project**.

c. Select “IBM Spectrum Scale CSI Plugin Operator” from the storage section in the operator menu and click **install**.

d. On the Operator install page, select a namespace from the available options, select the approval strategy (automatic or manual), and click Subscribe. The Installed Operators page appears, where IBM Spectrum Scale CSI Plugin Operator is listed as successfully installed.

e. On the Installed Operators page, click **IBM Spectrum Scale CSI Plugin Operator**, select the IBM CSI Spectrum Scale Driver tab, and click **Create CSIScaleOperator**. The Create CSIScaleOperator page displays.

f. On this page, an editor displays, where you can update the manifest file according to your environment.

5.2.2 Using Operator with command line interface (CLI)

Installing IBM Spectrum Scale Container Storage Interface Driver using Operator involves the following phases:

1. Deploy the Operator on your cluster
2. Use the Operator for deploying IBM Spectrum Scale Container Storage Interface Driver

Phase 1: Deploying the Operator

To deploy Operator on your cluster, complete the following steps:

1. Create a namespace as shown in Example 5-1.

Example 5-1 Create a namespace

```
kubectl create namespace ibm-spectrum-scale-csi-driver
```

2. Create the Operator as shown in Example 5-2.

Example 5-2 Create the Operator

```
kubectl create -f https://raw.githubusercontent.com/IBM/ibm-
spectrum-scale-csi/v1.0.1/generated/installer/ibm-spectrum-scale-csi-operator.yaml
```

3. Verify that the Operator is deployed, and the Operator pod is in the `Running` state, as shown in Example 5-3.

Example 5-3 Verify the Operator is deployed

```
# kubectl get pods -n ibm-spectrum-scale-csi-driver
NAME                                             READY   STATUS    RESTARTS   AGE
ibm-spectrum-scale-csi-operator-6d4bd865f6-m297v 2/2     Running   0          25s
```

Phase 2: Deploying IBM Spectrum Scale Container Storage Interface Driver

Now that the Operator is up and running, you must access the Operator’s API and request a deployment using CSIScaleOperator Custom Resource. Follow these steps:

1. Create a **secret** with IBM Spectrum Scale GUI server’s credentials in the `ibm-spectrum-scale-csi-driver` namespace. A secret is needed to store credentials to connect to IBM Spectrum Scale REST API server. Secrets are defined in a data field with base64 encoded values in a JSON file. The GUI user must have *csiadmin* role.

A sample manifest file for the GUI **secret** is shown in Example 5-4.

Example 5-4 Sample manifest file for GUI Secret

```
apiVersion: v1
kind: Secret
metadata:
  name: [secret_name]
  labels:
    product: ibm-spectrum-scale-csi
data:
  username: [base64_username]
  password: [base64_password]
```

To create the **secret**, issue the following command:

```
kubectl apply -f secrets.yaml -n ibm-spectrum-scale-csi-driver
```

> **Note:** If the `secureSslMode=true` is specified, a configmap also needs to be created in the `ibm-spectrum-scale-driver` namespace with the certificate resource name mentioned in the `ibm-spectrum-scale-csi-operator-cr.yaml` file. See IBM Knowledge Center for IBM Spectrum Scale Container Storage Interface driver configurations for Certificate:
>
> `https://www.ibm.com/support/knowledgecenter/en/STXKQY_5.0.4/com.ibm.spectrum.scale.csi.v5r04.doc/bl1csi_config_csi.html`

2. Download the CSIScaleOperator Custom Resource file from GitHub:

   ```
   curl -O
   https://raw.githubusercontent.com/IBM/ibm-spectrum-scale-csi/v1.0.1/operator/deploy/crds/ibm-spectrum-scale-csi-operator-cr.yaml
   ```

3. To deploy IBM Spectrum Scale Container Storage Interface Driver, configure the `ibm-spectrum-scale-csi-operator-cr.yaml` file to suit your requirements and issue this command:

   ```
   kubectl apply -f ibm-spectrum-scale-csi-operator-cr.yaml
   ```

   For more information, see IBM Knowledge Center IBM Spectrum Scale Container Storage Interface driver configurations for Operator:

   `https://www.ibm.com/support/knowledgecenter/en/STXKQY_5.0.4/com.ibm.spectrum.scale.csi.v5r04.doc/bl1csi_config_csi.html`

4. Verify that the IBM Spectrum Scale Container Storage Interface Driver is installed, Operator and driver resources are ready, and pods are in the `Running` state:

Example 5-5 Verify that the IBM Spectrum Scale Container Storage Interface Driver is installed

```
# kubectl get pods -n ibm-spectrum-scale-csi-driver
NAME                                              READY   STATUS    RESTARTS   AGE
ibm-spectrum-scale-csi-attacher-0                 1/1     Running   0          5m54s
ibm-spectrum-scale-csi-fpf7w                      2/2     Running   0          5m52s
ibm-spectrum-scale-csi-operator-6d4bd865f6-m297v  2/2     Running   0          34m
ibm-spectrum-scale-csi-pfl2k                      2/2     Running   0          5m52s
ibm-spectrum-scale-csi-provisioner-0              1/1     Running   0          5m53s
ibm-spectrum-scale-csi-vqpk6                      2/2     Running   0          5m52s
```

5.3 Volume provisioning

To create and deploy the Persistent Volume (PV) on IBM Spectrum Scale, IBM Spectrum Scale Container Storage Interface (CSI) driver supports the following features:

- Static provisioning: Ability to use existing directories as persistent volumes
- Dynamic provisioning:
  - Lightweight dynamic provisioning: Ability to create directory-based volumes dynamically
  - File Set-based dynamic provisioning: Ability to create file set-based volumes dynamically

5.3.1 Static provisioning

Users might need to share the data with traditional applications and containerized applications, so static provisioning enables the users to make the data in existing directories available for the applications running on different platforms. In static provisioning, users need to define and create the PV with the existing directories manually. The sample definition of a persistent volume for static provisioning is described in Example 5-6.

Example 5-6 Sample PV definition of static provisioning

```
# cat static-pv-data.yaml
# -- static-pv-data.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
        name: static-pv-data
spec:
   capacity:
     storage: 10Gi
   accessModes:
     - ReadWriteMany
   csi:
     driver: ibm-spectrum-scale-csi
     volumeHandle: <clusterID>;<filesystem_uuid>;path=<path_to_dir>
```

Create a PVC (Example 5-7) to bind to the PV created in Example 5-6.

Example 5-7 Sample PVC definition for claiming an existing PV

```
# cat pvc.yaml
# -- pvc.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: scale-static-pvc
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
```

This PVC will be bound to an available PV with storage equal to or greater than what is specified in the `pvc.yaml` file.

5.3.2 Dynamic provisioning

Dynamic provisioning is used to dynamically provision the storage back-end volume based on the Storage Class. The Storage Class defines what type of back-end volume should be created by dynamic provisioning. IBM Spectrum Scale CSI Driver supports creation of directory based (also known as lightweight volumes) and file set based (independent as well as dependent) volumes.

The following parameters are supported by IBM Spectrum Scale CSI Driver storageclass:

volBackendFs	File system on which the volume should be created. This is a mandatory parameter.
clusterId	Cluster ID on which the volume should be created.
volDirBasePath	Base directory path relative to the file system mount point under which directory based volumes should be created. If specified, the storageclass is used for directory based (Lightweight) volume creation. If not specified, storageclass creates file set based volumes.
uid	UID with which the volume should be created. Optional.
gid	GID with which the volume should be created. Optional.
filesetType	Type of file set. Valid values are `independent` or `dependent`. Default is `independent`.
parentFileset	Specifies the parent file set under which dependent file set should be created. Required only if `filesetType` is specified as `dependent`.
inodeLimit	Inode limit for file set based volumes. If not specified, default IBM Spectrum Scale inode limit of 1 million is used.

Lightweight dynamic provisioning

When using the lightweight dynamic provisioning, users need to define the storage class with `volDirBasedPath` options. The PV will be created under the `volDirBasedPath` dynamically when the PVC is applied. The sample definition of storage class for lightweight dynamic provisioning is shown in Example 5-8.

Example 5-8 Sample storageclass definition of lightweight dynamic provisioning

```
# cat storageclasslw.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
   name: csi-spectrum-scale-lt
   provisioner: spectrumscale.csi.ibm.com
parameters:
   volBackendFs: "scalefs1"
   volDirBasePath: "pvfileset/lwdir"
reclaimPolicy: Delete
```

File set-based dynamic provisioning

The file set-based dynamic provisioning enables users to manage the volume with smaller granularity compared to the lightweight dynamic provisioning, as shown Example 5-9.

Example 5-9 Sample storageclass definition of independent file set-based dynamic provisioning

```
# cat storageclassfileset.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
   name: csi-spectrum-scale-file set
   provisioner: spectrumscale.csi.ibm.com
parameters:
   volBackendFs: "scalefs1"
   clusterId: "15635445795430606940"
reclaimPolicy: Delete
```

Creating a PVC

Create a persistent volume claim (PVC) using the defined storageclass. A sample manifest file is shown in Example 5-10.

Example 5-10 Sample PVC definition for using a storageclass for dynamic provisioning

```
# cat pvc.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: scale-fset-pvc
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: [name_of_your_storageclass]
```

5.4 Using the volume in a container

After the PVC is created in the Kubernetes/OpenShift clusters, it can attach PVC in a container and read/write files in the volumes. In this section, the steps to attach the PVC on the Pod and use the volume in a container are described.

Note: The following steps describe the Pod creation with the PVC created by either Static or Dynamic provisioning.

1. Define the Pod with PVC to be attached. It is required to specify the information of PVC in the “volumes” section. Example 5-11.

Example 5-11 Define the Pod with PVC to be attached

```
# cat static_pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: csi-scale-demo-pod
  labels:
    app: nginx
spec:
  containers:
   - name: web-server
     image: nginx
     volumeMounts:
       - name: mypvc
         mountPath: /usr/share/nginx/html/scale
     ports:
     - containerPort: 80
  volumes:
   - name: mypvc
     persistentVolumeClaim:
       claimName: scale-static-pvc
       readOnly: false
```

2. After the status of Pod becomes `Running`, log in to the `csi-scale-demo-pod` and check the attached volume.

```
# kubectl exec -it csi-scale-demo-pod -- /bin/bash
```

3. Now, it is possible to see the files stored in an existing directory, as shown in Example 5-12.

Example 5-12 List files from an existing directory on IBM Spectrum Scale inside the container

```
root@csi-scale-fsetdemo-pod:/# ls -l /usr/share/nginx/html/scale
total 102400
-rw-r--r-- 1 root root 10485760 Oct 14 11:12 testfile1
-rw-r--r-- 1 root root 10485760 Oct 14 11:12 testfile10
-rw-r--r-- 1 root root 10485760 Oct 14 11:12 testfile2
-rw-r--r-- 1 root root 10485760 Oct 14 11:12 testfile3
-rw-r--r-- 1 root root 10485760 Oct 14 11:12 testfile4
```

5.5 Releasing the volume on pod deletion

Storage resource can be released when users are done with the pod and the volume. Releasing the storage resources is managed by the reclaim policy defined in a PVC. IBM Spectrum Scale supports the following reclaim policies:

Delete When the PVC is deleted, the PV will be deleted along with underlying storage resource such as a file set or directory. Note that “delete” reclaim policy is not supported in case of statically provisioned PVCs.

Retain When the PVC is deleted, the PV is retained. Users could release the storage resource deleting the PV manually if required.

5.6 Node Selector

The Node Selector feature of Kubernetes is used to control the nodes on which pods should be scheduled. By default, IBM Spectrum Scale Container Storage Interface driver gets deployed on all worker nodes. Node Selector controls on which Kubernetes worker nodes IBM Spectrum Scale Container Storage Interface driver components should be installed. It helps in cases where new worker nodes are added to a Kubernetes cluster that does not have IBM Spectrum Scale installed. It also helps in ensuring that StatefulSets are running on the desired nodes.

For more information, see the following site:

https://www.ibm.com/support/knowledgecenter/STXKQY_5.0.4/com.ibm.spectrum.scale.csi.v5r04.doc/bl1csi_config_kubernet_SS_mapping_procedure_final.html

5.7 Kubernetes to IBM Spectrum Scale Cluster node mapping

In some environments, Kubernetes node names might be different from the IBM Spectrum Scale node names. This results in failure during mounting of pods. Kubernetes node to IBM Spectrum Scale node mapping must be configured to address this condition. The mapping can be defined in the Operator configuration.

For more information, see the following site:

https://www.ibm.com/support/knowledgecenter/STXKQY_5.0.4/com.ibm.spectrum.scale.csi.v5r04.doc/bl1csi_config_kubernet_SS_mapping_procedure_final.html

5.8 Upgrading IBM Spectrum Scale Container Storage Interface driver

Upgrading IBM Spectrum Scale Container Storage Interface driver steps, follow the instructions mentioned on the following site:

https://www.ibm.com/support/knowledgecenter/STXKQY_5.0.4/com.ibm.spectrum.scale.csi.v5r04.doc/bl1csi_instal_upgrade.html

6

Security considerations

Ensuring an appropriate security architecture is an essential step while planning an enterprise container environment.

In this chapter we will cover security topics related to IBM Spectrum Scale CSI:

- 6.1, “Secure administration and deployment” on page 40
- 6.2, “Secure data for containers” on page 45

Understanding the security-related aspects will help for IBM Spectrum Scale CSI driver planning, deployment, and usage. This chapter is recommended for architects planning the environment and security administrators.

6.1 Secure administration and deployment

Secure administration is one of the important aspects for any deployment. Based on IBM Spectrum Scale security features for administration, IBM Spectrum Scale CSI ensures the administration and deployment are secure. In this section, we describe the different aspects that one is required to consider for this purpose.

6.1.1 Network deployment and firewall configuration

From a security perspective it is recommended to use separate cluster/data and admin networks. The cluster network is used for the IBM Spectrum Scale cluster metadata and data traffic while the admin network is used for administrator access to the IBM Spectrum Scale GUI and CLI as well as for REST API communication.

All Kubernetes worker nodes need access to the IBM Spectrum Scale cluster network. The Kubernetes Infrastructure nodes need access to the IBM Spectrum Scale admin network (as they are expected to interact with IBM Spectrum Scale GUI nodes). One will need to likewise set the firewalls based on the deployment network topology. see the “Firewall recommendations for internal communication among nodes” in IBM Knowledge Center for more details:

https://www.ibm.com/support/knowledgecenter/en/STXKQY_5.0.4/com.ibm.spectrum.scale.v5r04.doc/bl1adm_guihaconfig.htm

> **Note:** For the firewall requirements from Kubernetes deployment, see the Kubernetes distribution documentation. For example, for OpenShift see the OpenShift documentation:
>
> https://docs.openshift.com/container-platform/4.2/installing/install_config/configuring-firewall.html

6.1.2 Secure communication with IBM Spectrum Scale GUI server

The IBM Spectrum Scale CSI driver communicates with IBM Spectrum Scale through the REST API provided by the IBM Spectrum Scale GUI node. If Kubernetes node labeling in combination with a nodeSelector is used to place the CSI controller plug-in (provisioner and attacher containers) onto the Kubernetes cluster infrastructure nodes, only the infrastructure nodes need access to the IBM Spectrum Scale GUI node using the admin network. To ensure the proper firewall rules are set, see the “Firewall recommendations for IBM Spectrum Scale GUI” entry in IBM Knowledge Center:

https://www.ibm.com/support/knowledgecenter/STXKQY_5.0.4/com.ibm.spectrum.scale.v5r04.doc/bl1ins_quickrefforgui.htm

In Figure 6-1 on page 41 there are three security parameters that are required for secure communication between the CSI plugin and the GUI server:

Secure SSL Mode	It specifies whether certificate validation should be done. If secure SSL Mode is `true`, providing of GUI CA certificate becomes mandatory. If `false`, the certificate validation is ignored.
Certificate	This is a CA certificate for GUI server. It is required if SSL mode is `true`.
User credential	In Plug-in configuration, we provide a Kubernetes secret object which contains the IBM Spectrum Scale GUI “username” and “Password” as base64 encoded values.

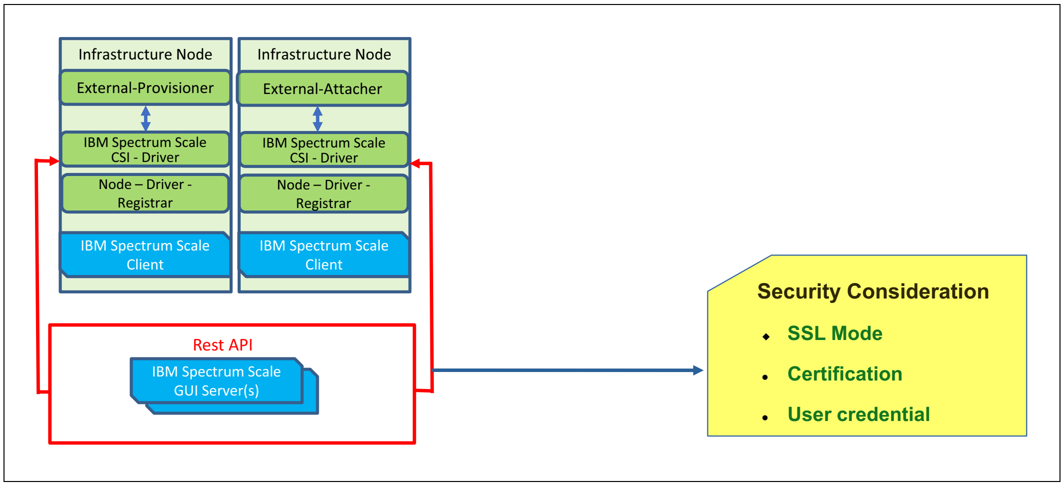

Infrastructure Node
External-Provisioner
IBM Spectrum Scale CSI - Driver
Node – Driver - Registrar
IBM Spectrum Scale Client
Infrastructure Node
External-Attacher
IBM Spectrum Scale CSI - Driver
Node – Driver - Registrar
IBM Spectrum Scale Client
Rest API
IBM Spectrum Scale GUI Server(s)
Security Consideration
SSL Mode
Certification
User credential

Figure 6-1 Security parameters for secure communication between CSI plugin and GUI server

These parameters are specified as part of operator specification. For more details about how to set up these security parameters, see IBM Knowledge Center:

`https://www.ibm.com/support/knowledgecenter/STXKQY_5.0.4/com.ibm.spectrum.scale.csi.v5r04.doc/bl1csi_instal_prereq.html`

6.1.3 SELinux considerations

Typically, SELinux prerequisites on the nodes are driven by the Kubernetes distribution. As an example, OpenShift deployment the worker nodes in the cluster are required to have SELinux enabled in enforcing mode with targeted policies while the control plane nodes runs on RHCOS (RHCOS is the immutable container host version of Red Hat Enterprise Linux (RHEL) and features a RHEL kernel with SELinux enabled by default).

On SELinux enabled systems, file system objects will have SELinux context as *user:role:type:level*. For access decisions governed by SELinux policies, mainly the `type` label is used and currently IBM Spectrum Scale file system is unlabeled. So, in order to have access to a IBM Spectrum Scale File Set or directory inside a container, a Container Orchestrator like OpenShift potentially might be required to perform relabeling. This setup enables file objects to be labelled with a label accessible to the domain in which the container process is running.

Note: The relabeling will not disrupt the regular data access. The GPFS daemon and CES services are running in an unconfined domain so they will have access to the file set or directory which are relabelled by the container orchestrator.

6.1.4 Configuring UID/GID ownership to ensure Persistent Volume access for non-root containers

For accessing a Persistent Volume (PV) inside a container where the main process is running as non-root user, proper ownership needs to be set for a PV and POD security context needs to be properly set.

For changing ownership of a PV, we can use storage class and set UID/GID of a PV. When a PV is created, the ownership of a volume will be dynamically set based on storage class parameters.

Setting of PV ownership using storage class is applicable for dynamic provisioning. For static provisioning, the admin must take care of setting permissions and ownership on the PV. In case UID/GID is not provided in the storage class, by default the file set/directory created for PV will have owner as `root`.

In Example 6-1 we have set the UID as 4000 and GID as 5555 in a storage class `csi-spectrum-scale-file set-uid`. This storage class will be used while creating a PV dynamically.

Example 6-1 Sample storageclass with volume ownership

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
   name: csi-spectrum-scale-file set-uid
provisioner: spectrumscale.csi.ibm.com
parameters:
    volBackendFs: "fs1"
    clusterId: "17934173018790868908"
    gid: 5555
    uid: 4000
reclaimPolicy: Delete
```

Using the storage class `csi-spectrum-scale-file set-uid` shown in Example 6-1, we create a PVC. For this example, the PVC name is `scale-fset-pvc-sample`.

After the PVC is created it will bind to a PV. On the IBM Spectrum Scale side, the PV will appear as a file set with permission 771 and ownership 4000:5555. This means that any process with uid of 4000 or which belongs to group 5555 will have access to the PV.

For the case in Example 6-2, we have created a POD with a security context such that it will run as user 4001 (which is not a member of group 5555). Because UID 4001 doesn't have access to PV, we will get a Permission Denied error.

Example 6-2 Definition of POD which will run with UID 4001

```
# cat podfileset-uid.yaml
kind: Pod
apiVersion: v1
metadata:
  name: sanity-pod          # Pod name
spec:
  nodeSelector:
    spectrumscalenode1: "yes"
  securityContext:
    runAsUser: 4001
  containers:
  - name: container1  # Container name
    image: alpine:latest
    command: [ "/bin/sh", "-c", "--" ]
    args: [ "while true; do sleep 30; done;" ]
    volumeMounts:
      - name: vol1
```

```
        mountPath: "/data"  # Where to mount the vol1(pvc1)
  restartPolicy: "Never"
  volumes:
    - name: vol1
      persistentVolumeClaim:
        claimName: scale-fset-pvc-sample
```

Using the following command, we can see that UID 4001 is unable to create a file in a PV that is mounted at `/data` inside a container:

```
# oc create -f podfileset-uid.yamlpod/sanity-pod created
```

Using the command in Example 6-3, we can see that UID 4001 is unable to create a file in a PV that is mounted at `/data` inside a container.

Example 6-3 Example of write from a container on a PVC with different UID

```
# oc exec -it sanity-pod -- sh -c "id && echo "hello" > /data/file"
uid=4001(4001) gid=0(root)
sh: can't create /data/file: Permission denied
command terminated with exit code 1
```

In Example 6-4, we have created a POD with a security context such that it will run as user 4000. Because UID 4000 has access to the PV, the write will succeed.

Example 6-4 Definition of POD which will run with UID 4000

```
# cat podfileset-uid.yaml
kind: Pod
apiVersion: v1
metadata:
  name: sanity-pod          # Pod name
spec:
  nodeSelector:
    spectrumscalenode1: "yes"
  securityContext:
    runAsUser: 4000
  containers:
  - name: container1  # Container name
    image: alpine:latest
    command: [ "/bin/sh", "-c", "--" ]
    args: [ "while true; do sleep 30; done;" ]
    volumeMounts:
      - name: vol1
        mountPath: "/data"  # Where to mount the vol1(pvc1)
  restartPolicy: "Never"
  volumes:
    - name: vol1
      persistentVolumeClaim:
        claimName: scale-fset-pvc-sample
```

The following command will create a POD sanity-pod running with UID 4000:

```
# oc create -f podfileset-uid.yaml
pod/sanity-pod created
```

Because POD is running with UID 4000, it will be able to write to a PV, because the owner of the PV is set to 4000:

```
# oc exec -it sanity-pod -- sh -c "id && echo "hello" > /data/file"
uid=4000(4000) gid=0(root)
#
```

In Example 6-5, we have created a security context such that POD becomes part of supplemental group 5555. Therefore, the container process will have a secondary group 5555. Because the 5555 group has access to the PV, the write will succeed.

Example 6-5 Definition of POD with supplemental group

```
# cat podfileset-uid.yaml
kind: Pod
apiVersion: v1
metadata:
  name: sanity-pod          # Pod name
spec:
  nodeSelector:
    spectrumscalenode1: "yes"
  securityContext:
    runAsUser: 4001
    supplementalGroups: [5555]
  containers:
  - name: container1  # Container name
    image: alpine:latest
    command: [ "/bin/sh", "-c", "--" ]
    args: [ "while true; do sleep 30; done;" ]
    volumeMounts:
      - name: vol1
        mountPath: "/data"  # Where to mount the vol1(pvc1)
  restartPolicy: "Never"
  volumes:
    - name: vol1
      persistentVolumeClaim:
        claimName: scale-fset-pvc-sample
```

The following command in will create a POD sanity-pod which will be part of supplementary group 5555:

```
# oc create -f podfileset-uid.yaml
pod/sanity-pod created
```

Because group 5555 has write permission on the PV, the write operation will succeed. We can also see in Example 6-6 that POD is made part of supplemental group 5555.

Example 6-6 Example of write from a container with required supplemental group

```
# oc exec -it sanity-pod -- sh -c "id && echo "hello" > /data/file1"
uid=4001(4001) gid=0(root) groups=5555
#
```

6.1.5 Privileged CSI PODs

Currently in IBM Spectrum Scale CSI Driver, the attacher and provisioner Statefulset run Pods in privileged mode on OpenShift. Also, the Daemonset Pod for driver registrar runs in privileged mode; however, the Daemonset Pod for the CSI plugin runs in non-privileged mode.

On a Kubernetes cluster, all CSI driver pods run in non-privileged mode.

6.2 Secure data for containers

It is important for workloads adopting to containerization to ensure that the solution provides end-to-end security. From the data point of view, it is vital to have the capabilities for secure data at rest and secure data in transit capabilities, especially if the workloads deal with sensitive data or are under regulatory compliances. In this section, we will go through the security features available with IBM Spectrum Scale that can be leveraged for securing data for containers.

6.2.1 Secure data at rest for container data

In real world deployments there will be a need for some containers to encrypt the data that they write to the attached persistent volumes, where other containers might not have that need. This requires the flexibility for the underlying storage to have policies to enable that functionality. Additionally, some compliance-driven workloads or microservices running inside the container might require FIPS compliant secure data at rest.

IBM Spectrum Scale supports FIPS compliant secure data at rest, which enables granular encryption of data when written to the file system depending upon the policy. It supports standard key managers, such as IBM SKLM and Vormetric, required for managing the keys used for encryption. The encryption policies enable administrators to control what needs to be encrypted and what is not required to be encrypted.

This IBM Spectrum Scale feature easily integrates to help secure the data at rest for containers where the persistent volume attached to the containers is nothing but either a file set or a directory on IBM Spectrum Scale file system. As a hypothetical example, suppose we need to run a workload inside a container associated with a POD that creates two sets of files

- *.doc
- *.txt

There is a need that all `*.doc` files are encrypted when a given container writes them to the attached volume, but the `.txt` files remain unencrypted. Following are sample pseudo steps which one can use to achieve this requirement using static provisioning:

1. Configure IBM Spectrum Scale with file system encryption using a supported key manager. For details, see IBM Knowledge Center for IBM Spectrum Scale regarding Preparation for encryption:

   `https://www.ibm.com/support/knowledgecenter/STXKQY_5.0.4/com.ibm.spectrum.scale.v5r04.doc/bl1adv_encryption_prep.htm`

2. Create an independent file set called FilesetA.

3. Set the required encryption polices on FilesetA as shown in Example 6-7.

Example 6-7 Example of encryption policy rules

```
RULE 'MYENCRULE1' ENCRYPTION 'E1' IS ALGO 'DEFAULTNISTSP800131A'
KEYS('1:RKM_1', '2:RKM_2')

RULE 'Encrypt files with extension doc with rule E1' SET ENCRYPTION 'E1' FOR
file set('FilesetA') WHERE NAME LIKE '%.doc'
```

4. Create a PV by providing the FilesetA details.
5. Bind a PVC to the created PV.
6. Attach the POD to the above created PVC ensuring the specific container in the POD is associated with that PVC.

This will meet the previously described need for granular secure data at rest required for a given workload. To make such a requirement more generic, one can create storage classes associated with independent File sets that are configured with IBM Spectrum Scale encryption policies. These storage classes can then be used to dynamically provision persistent volumes associated with that encryption policy. In this case, the files adhering to the encryption policy will be encrypted before they are stored to the disk.

6.2.2 Secure data in transit

When volumes are provisioned to containers/POD via external file storage, a question that comes up is if the data in transit between the attached volume and back-end storage is secured or not. In an IBM Spectrum Scale setup, the volumes attached to containers/POD are local bind mounted from the underlying IBM Spectrum Scale client running on the host. Therefore, the data being written on the volume by the container/POD translates locally and not “over the wire”. Now, the IBM Spectrum Scale client further stripes and stores this data on the disks via the IBM Spectrum Scale network shared disk (NSD) servers.

There might be a need to encrypt all of the communication between the IBM Spectrum Scale clients and servers, which will include the data from the containers/POD. To cater to this requirement, IBM Spectrum Scale supports secure data over wire. IBM Spectrum Scale cluster configuration has different security modes that can be changed using the **`mmchconfig cipherlist=security_mode`** command. When the security mode is set to one of the supported ciphers, all of the inter-node communication within the IBM Spectrum Scale cluster will be encrypted using the mentioned cipher.

Example: `mmchconfig cipherList=AES256-SHA256`

Note: The default mode of cipherList is `AUTHONLY`, which ensures that the inter-node traffic is authenticated. This caters to requirements for most deployments. Note that changing the cipherList to one of the supported encryption ciphers might impact the performance of the workload.

For more information, see IBM Knowledge Center for IBM Spectrum Scale in the Security mode section:

https://www.ibm.com/support/knowledgecenter/en/STXKQY_5.0.4/com.ibm.spectrum.scale.v5r04.doc/bl1adm_securitymode.htm

In case of remote cluster mounts, IBM Spectrum Scale supports secure data communication between clusters using **`mmauth`** commands. For more information see the “Managing a Remote GPFS file system” topic in IBM Knowledge Center for IBM Spectrum Scale:

https://www.ibm.com/support/knowledgecenter/en/STXKQY_5.0.4/com.ibm.spectrum.scale.v5r04.doc/bl1adv_admrmsec.htm

By securing inter-node IBM Spectrum Scale cluster communication, one can ensure secure data in transit for containerized workloads.

7

Problem determination and troubleshooting

The state of IBM Spectrum Scale CSI driver and its behavior depends on various Kubernetes IBM Spectrum Scale components working together. Problem determination in this area involves collecting debug data from these various components and using the same for troubleshooting.

In this chapter, we cover the following topics:

- 7.1, “How to collect debug data for IBM Spectrum Scale CSI driver” on page 50
- 7.2, “Understanding log files” on page 51

Specifically, we provide the following information:

- The process of log data collection
- How to use collected data for debugging issues
- Sample error scenarios during deployment and configuration
- Sample runtime issues

7.1 How to collect debug data for IBM Spectrum Scale CSI driver

IBM Spectrum Scale CSI driver provides a tool (`spectrum-scale-driver-snap.sh`) to collect the driver debug data. This tool gathers the state of required Kubernetes resources, such as nodes, pods, service accounts, and so on, and collects statefulset and daemonset logs from all nodes. It collects definition of resources in the given namespace with the label `product=ibm-spectrum-scale-csi`. The collected logs are stored in a given output directory.

Here is debug data collection usage format and parameters:

```
spectrum-scale-driver-snap.sh [-n namespace] [-o output-dir] [-h]
-n: Debug data for CSI resources under this namespace will be collected.
    If not specified, default namespace is used. The tool returns error
    if CSI is not running under the given namespace.
-o: Output directory where debug data will be stored. If not specified,
    the debug data is stored in current directory.
-h: Prints the usage
```

Download the tool from the following site:

https://raw.githubusercontent.com/IBM/ibm-spectrum-scale-csi/v1.0.1/tools/spectrum-scale-driver-snap.sh

Sample output of the command is shown in Example 7-1.

Example 7-1 Sample output from the execution of CSI driver debug data collection tool

```
# spectrum-scale-driver-snap.sh -n ibm-spectrum-scale-csi-driver
Collecting "ibm-spectrum-scale-csi" logs...
The log files will be saved in the folder [./ibm-spectrum-scale-csi-logs_11-18-2019-01:58:22]
oc logs --namespace csi StatefulSet/ibm-spectrum-scale-csi-attacher
oc logs --namespace csi StatefulSet/ibm-spectrum-scale-csi-provisioner
oc logs --namespace csi pod/ibm-spectrum-scale-csi-cr4zt
oc logs --namespace csi pod/ibm-spectrum-scale-csi-lwjnk
oc logs --namespace csi pod/ibm-spectrum-scale-csi-lzscw
oc logs --namespace csi pod/ibm-spectrum-scale-csi-npczx
oc describe CSIScaleOperator --namespace ibm-spectrum-scale-csi-driver
oc logs --namespace ibm-spectrum-scale-csi-driver pod/ibm-spectrum-scale-csi-operator-
7c49568fd4-4hfzv
oc describe all,cm,secret,storageclass,pvc,ds,serviceaccount -l product=ibm-spectrum-scale-csi
--namespace csi
oc describe clusterroles/external-provisioner-runner clusterrolebindings/csi-provisioner-role
clusterroles/external-attacher-runner clusterrolebindings/csi-provisioner-role
clusterroles/csi-nodeplugin clusterrolebindings/csi-nodeplugin --namespace csi
oc get all,cm,secret,storageclass,pvc,ds,serviceaccount --namespace csi -l
product=ibm-spectrum-scale-csi
oc get pod --namespace ibm-spectrum-scale-csi-driver -o wide  -l product=ibm-spectrum-scale-csi
oc get configmap spectrum-scale-config --namespace csi -o yaml
oc get nodes
oc describe nodes
oc describe scc csiaccess
oc cluster-info dump --namespaces kube-system
--output-directory=./ibm-spectrum-scale-csi-logs_11-18-2019-01:58:22
```

```
Finished collecting "ibm-spectrum-scale-csi" logs in the folder ->
./ibm-spectrum-scale-csi-logs_11-18-2019-01:58:22
```

The resultant folder contains the following files with debug information:

- ibm-spectrum-scale-csi-k8snodes
- ibm-spectrum-scale-csi-configmap
- ibm-spectrum-scale-csi-get-all-by-label
- ibm-spectrum-scale-csi-describe-all-by-label
- ibm-spectrum-scale-csi-attacher.log
- ibm-spectrum-scale-csi-scc.log
- ibm-spectrum-scale-csi-provisioner.log
- ibm-spectrum-scale-csi-xxxxx-driver-registrar.log
- ibm-spectrum-scale-csi-xxxxx.log
- ibm-spectrum-scale-csi-operator-xxx-XXXXX-operator-previous.log
- ibm-spectrum-scale-csi-operator-xxx-XXXXX-ansible-previous.log
- ibm-spectrum-scale-csi-operator-xxx-XXXXX-operator.log
- ibm-spectrum-scale-csi-operator-xxx-XXXXX-ansible.log

`ibm-spectrum-scale-csi-xxxxx-driver-registrar.log` and `ibm-spectrum-scale-csi-xxxxx.log` are daemonset logs present for every worker node where the driver is running. For detailed descriptions of these files, see the following section.

7.2 Understanding log files

The first step of troubleshooting is to check the state of the system and services. For CSI driver to function well, it is essential that the Kubernetes or OpenShift cluster is in a good state and properly configured.

7.2.1 Checking the state of Kubernetes cluster

Here are the descriptions of the Kubernetes cluster log files:

ibm-spectrum-scale-csi-k8snodes

This file contains the description of all nodes in the cluster and their status. Things to check here are the node roles, labels, any taints that are applied, system information, and a list of pods that are running on this node. This file also gives information about specific node conditions, such as MemoryPressure, DiskPressure, and PIDPressure, which might cause a node to fail or not be in a ready state.

ibm-spectrum-scale-csi-get-all-by-label

This file contains a list of CSI driver resources, such as pods, serviceaccounts, statefulsets, and their status.

ibm-spectrum-scale-csi-describe-all-by-label

This file contains detailed information about CSI driver resources, such as pods, serviceaccounts, statefulsets, and containers running within the pods. Things to check here are any events listed under the pods and statefulsets. If labels are used for resource creation, then this file contains information about storageclasses and PVCs.

7.2.2 Checking for issues during driver initialization

Here are the descriptions of logs to check during driver initialization:

ibm-spectrum-scale-csi-configmap

This file contains the configmap details for the CSI driver, which includes cluster, file system, and GUI server details.

ibm-spectrum-scale-csi-xxxxx-driver-registrar.log

This file contains logs for registration of the CSI driver with kubelet.

ibm-spectrum-scale-csi-xxxx.log

This file contains detailed logs of the CSI driver initialization process.

7.2.3 Checking for issues during provisioning of volumes

Here are the descriptions of logs to check for issues during provisioning of volumes.

ibm-spectrum-scale-csi-provisioner.log

The volume creation and deletion requests are logged in this file. Use this file to determine the request that failed. Identify the PVC name from the failed request. For example:
`pvc-f531f55d-d90f-4ee0-8ad9-e81c55fe5684`

ibm-spectrum-scale-csi-xxxx.log

Use this file to look for detailed logs of the failed request. The failed request can be looked up by searching for the PVC name identified from `ibm-spectrum-scale-csi-provisioner.log`.

7.2.4 Checking for issues during attaching of volumes

Here are the descriptions of logs to check for issues during attaching of volumes.

ibm-spectrum-scale-csi-attacher.log

The volume mount/unmount requests are logged in this file. Use this file to determine the request that failed. Identify the volume ID from the failed request. For example:

```
volume_id:”17797813605352210071;AC10D811:5DA2D1D1;
path=/gpfs/fs1/csifset1/.volumes/pvc-f531f55d-d90f-4ee0-8ad9-
e81c55fe5684”
```

ibm-spectrum-scale-csi-xxxx.log

Use this file to look for detailed logs of the failed volume attach request. The failed request can be looked up by searching for the volume ID identified from `ibm-spectrum-scale-csi-attacher.log`.

For more details about troubleshooting and error scenarios, see IBM Knowledge Center for IBM Spectrum Scale Container Storage Interface Driver Troubleshooting:

https://www.ibm.com/support/knowledgecenter/STXKQY_5.0.4/com.ibm.spectrum.scale.csi.v5r04.doc/bl1cs_sec_ug_troubleshooting_file.html

8

Migration from SEC to CSI driver

In this chapter we will cover the following migration paths from existing IBM Storage Enabler for Containers (SEC) installations to switch to the IBM Spectrum Scale CSI driver:

- 8.1, “Migration from one community Kubernetes version using SEC to a community Kubernetes version using CSI” on page 54
- 8.2, “Migration scenario from IBM Cloud Private using SEC to OpenShift 4.2 or community Kubernetes using CSI” on page 54
- 8.3, “Migration from OpenShift 3.x using SEC to OpenShift 4.2 using CSI” on page 54

Understanding these migration scenarios helps for IBM Spectrum Scale CSI driver planning and deployment. It is recommended for readers who use SEC today and plan to upgrade to the CSI driver. All the listed migration paths are manual and disruptive.

8.1 Migration from one community Kubernetes version using SEC to a community Kubernetes version using CSI

The IBM Spectrum Scale CSI Driver is supported with Kubernetes version 1.13 and later. SEC is supported until Kubernetes version 1.12. Detailed steps of migration are documented on the following website:

`https://www.ibm.com/support/knowledgecenter/STXKQY_5.0.4/com.ibm.spectrum.scale.csi.v5r04.doc/bl1csi_migrate.html`

The steps do not cover migration of one community Kubernetes to another community Kubernetes.

8.2 Migration scenario from IBM Cloud Private using SEC to OpenShift 4.2 or community Kubernetes using CSI

The IBM Spectrum Scale CSI Driver is not supported with ICP. Customer using IBM Cloud™ Private has to move to OpenShift 4.2 or community Kubernetes to use IBM Spectrum Scale CSI Driver. Steps for migration are documented on the following website:

`https://www.ibm.com/support/knowledgecenter/STXKQY_5.0.4/com.ibm.spectrum.scale.csi.v5r04.doc/bl1csi_migrate.html`

The steps do not cover migration of IBM Cloud Private to OpenShift 4.2 or community Kubernetes.

8.3 Migration from OpenShift 3.x using SEC to OpenShift 4.2 using CSI

The IBM Spectrum Scale CSI Driver is supported with OpenShift 4.2 and later. SEC is supported with OpenShift 3.x. One has to migrate from OpenShift 3.x to OpenShift 4.2 and later to start using IBM Spectrum Scale CSI Driver. Steps for Migrating SEC to CSI for an OpenShift environment are documented on the following website:

`https://www.ibm.com/support/knowledgecenter/STXKQY_5.0.4/com.ibm.spectrum.scale.csi.v5r04.doc/bl1csi_migrate.html`

The steps do not cover migration of OpenShift 3.x to OpenShift 4.2.

9

Support

IBM Spectrum Scale CSI driver can be deployed in varied environments (OS, architecture, and IBM Spectrum Scale version) and can work with multiple Kubernetes distributions (for example, Community Kubernetes or Red Hat OpenShift). This chapter describes the support matrix for IBM Spectrum Scale CSI driver and also gives details of the configurations that have been tested. The chapter also shares findings from our performance tests.

This chapter describes the following topics:

- 9.1, “Support Matrix” on page 56
- 9.2, “Tested Configurations” on page 56
- 9.3, “Performance” on page 56

9.1 Support Matrix

This topic describes the support matrix for IBM Spectrum Scale CSI driver and hardware and software requirements that must be met for using IBM Spectrum Scale CSI driver.

Table 9-1 shows the support matrix for IBM Spectrum Scale CSI driver,

Table 9-1 Support matrix for IBM Spectrum Scale CSI driver

Component	Level	Architecture
IBM Spectrum Scale	5.0.4.1	x86_64
OpenShift	4.2	x86_64
Kubernetes	1.13,1.14,1.15,1.16, 1.17	x86_64
OS - RHEL	7.5, 7.6, 7.7	x86_64

9.2 Tested Configurations

Kubernetes can be deployed using many different tools and can be configured with various container runtime environments. Other variable components in a Kubernetes deployment are the network plug-in and the distribution. Setting up Kubernetes is the customer's responsibility and a prerequisite for deploying the IBM Spectrum Scale CSI driver. Although there are many other Kubernetes deployment configurations possible, Table 9-2 describes the combinations that we have tested.

Table 9-2 Tested Kubernetes configurations for IBM Spectrum Scale CSI driver

Configuration	Comments
Kubernetes with docker runtime	Docker v18.03, 18.06
Kubernetes with CRI-O runtime	CRI-O v1.13, 1.14
OpenShift on VMWare with CoreOS on master nodes and RHEL on worker nodes	RHEL 7.6, 7.7
OpenShift on baremetal with CoreOS on master nodes and RHEL on worker nodes	RHEL 7.6, 7.7
Kubernetes with Calico network plug-in	Calico v3.7, 3.8.2
Kubernetes with Flannel network plug-in	Flannel 0.11.0

9.3 Performance

This section describes the findings from our performance testing of the IBM Spectrum Scale CSI driver. We did a comparative I/O performance study of container based workload versus traditional non-container based workload on IBM Spectrum Scale. We also tested performance of volume provisioning and volume deletion.

9.3.1 I/O from container

We compared I/O performance of a workload running from container application accessing IBM Spectrum Scale file system with the workload running from the host machine and accessing IBM Spectrum Scale file system. We observed that there was negligible performance difference between the two.

9.3.2 Volume provisioning

During our performance tests for volume provisioning we observed that serial PVC creation was faster in getting the PVC in a “Bound” state as compared to parallel PVC creation in batches of 10. This performance difference was more significant in case of independent file set based volume provisioning.

9.3.3 PVC deletion

The following items describe PVC deletion:

- The PVC deletion with files and directories using **`kubectl delete pvc`** ***`<pvc-name>`*** is quick. This is because **`kubectl delete pvc`** ***`<pvc-name>`*** is an asynchronous operation and does not wait for the deletion of its related object (for example, an Independent file set).
- The total time taken for the deletion (unlink + delete) of an Independent file set object corresponding to a PVC depends on the number of files and directories stored within the independent file set. because the independent file set deletion is a background operation during PVC deletion, this does not impact the foreground PVC deletion (using **`kubectl delete pvc`** ***`<pvc-name>`***) time.

A

Implementation details for use cases

This appendix provides the implementation details, including commands and the GUI steps needed to set up the required environment described in the various use cases in Chapter 3, “Solutions and Use Cases” on page 11. You can use these details as a step-by-step guide during implementation and testing of these use cases.

This appendix describes the following topics:

- A.1, “Multiple cluster use cases” on page 60
- A.2, “Compression use case” on page 61
- A.3, “Encryption use case” on page 63

A.1 Multiple cluster use cases

In the example below, the network bandwidth between node `c71f1c7p1ib0` and `c71f1c9p1ib0` over the TCP/IP and RDMA network is assessed. The “nsdperf in server mode” is started in the client and server node:

```
mmdsh -N c71f1c7p1ib0,c71f1c9p1ib0 "/opt/benchmarks/nsdperf-ib -s
</dev/null>/dev/null 2>&1 &"
```

Then, execute **`nsdperf`** (Example A-1) from an administrative node (for example, login node, gateway node, or any cluster node permitting interactive job execution).

Example A-1 An nsdperf example with a single client and a single server

```
# /opt/benchmarks/nsdperf-ib
# Designate the nodes as clients using "client" parameter nsdperf-ib> client c71f1c7p1ib0
Connected to c71f1c7p1ib0
# Designate the nodes as servers using "server" parameter nsdperf-ib> server c71f1c9p1ib0
Connected to c71f1c9p1ib0
# Set the run time to 30 seconds for the tests using "ttime" parameter nsdperf-ib> ttime 30
Test time set to 30 seconds
# Perform the desired nsdperf network tests using "test" parameter.
# TCP/IP network mode - Use "status" command to verify client node connectivity to the server # node
nsdperf-ib> status test time: 30 sec
data buffer size: 4194304
TCP socket send/receive buffer size: 0 tester threads: 4
parallel connections: 1 RDMA enabled: no clients:
c71f1c7p1ib0 (10.168.117.199) -> c71f1c9p1ib0
servers:
c71f1c9p1ib0 (10.168.117.205)
# Perform performance tests from clients to servers.
# The "test" command sends a message to all clients nodes to begin network performance testing
# to the server nodes. By default, write and read tests are performed.

nsdperf-ib> test
1-1 write 3170 MB/sec (756 msg/sec), cli 2% srv 3%, time 30, buff 4194304
1-1 read 3060 MB/sec (728 msg/sec), cli 3% srv 2%, time 30, buff 4194304
```

In Example A-1, the aggregate TCP/IP write bandwidth between 1 nsdperf client and 1 nsdperf server1 (each node with 1 × FDR-IB link) is 3,170 MBps for a 4,194,034 bytes (4 MiB) buffer size. The message rate for a 4 MiB buffer size is 756 per second. The nsdperf test duration is 30 seconds. The CPU busy (non-idle) percentage is 3% on average for all the nsdperf-server nodes, and 2% on average for all the client nodes.

The aggregate TCP/IP read bandwidth between 1 nsdperf client and 1 nsdperf server1 (each node with 1 × FDR-IB link) is 3,060 MBps for a 4,194,034 bytes (4 MiB) buffer size. The message rate for a 4 MiB buffer size is 728 per second. The nsdperf test duration is 30 seconds. The CPU busy (non-idle) percentage is 2% on average for all the server nodes, and 3% on average for all the client nodes.

A.2 Compression use case

Example A-2 depicts the steps to compress the log files stored in a persistent volume by containerized application.

Example A-2

```
A. List PVC - Get PV Name
# kubectl get pvc prod-scale-fset-pvc
NAME                  STATUS   VOLUME                                     CAPACITY   ACCESS
MODES   STORAGECLASS                      AGE
prod-scale-fset-pvc   Bound    pvc-2969aa3d-ad12-4bf8-9793-05745c3b704f   1Gi        RWX
prod-csi-spectrum-scale-fileset   24h
#

B. Describe PVC
# kubectl describe pvc prod-scale-fset-pvc
Name:          prod-scale-fset-pvc
Namespace:     default
StorageClass:  prod-csi-spectrum-scale-fileset
Status:        Bound
Volume:        pvc-2969aa3d-ad12-4bf8-9793-05745c3b704f
Labels:        <none>
Annotations:   pv.kubernetes.io/bind-completed: yes
               pv.kubernetes.io/bound-by-controller: yes
               volume.beta.kubernetes.io/storage-provisioner: csi-spectrum-scale
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:      1Gi
Access Modes:  RWX
VolumeMode:    Filesystem

C. Describe PV - Get the Softlink Path
# kubectl describe pv pvc-2969aa3d-ad12-4bf8-9793-05745c3b704f
Name:            pvc-2969aa3d-ad12-4bf8-9793-05745c3b704f
Labels:          <none>
Annotations:     pv.kubernetes.io/provisioned-by: csi-spectrum-scale
Finalizers:      [kubernetes.io/pv-protection]
StorageClass:    prod-csi-spectrum-scale-fileset
Status:          Bound
Claim:           default/prod-scale-fset-pvc
Reclaim Policy:  Delete
Access Modes:    RWX
VolumeMode:      Filesystem
Capacity:        1Gi
Node Affinity:   <none>
Message:
Source:
    Type:              CSI (a Container Storage Interface (CSI) volume source)
    Driver:            csi-spectrum-scale
    VolumeHandle:
17258972170939727157;09762E6B:5D36FE7F;fileset=2;path=/ibm/gpfs1/production/.volumes/pvc-2969aa3
d-ad12-4bf8-9793-05745c3b704f
    ReadOnly:          false
    VolumeAttributes:      clusterId=17258972170939727157
```

```
storage.kubernetes.io/csiProvisionerIdentity=1570731582191-8081-csi-spectrum-scale
                           volBackendFs=gpfs1
Events:                <none>
#
```

D. List - Get the actual Gpfs Path

```
# ls -al /ibm/gpfs1/production/.volumes/pvc-2969aa3d-ad12-4bf8-9793-05745c3b704f
lrwxrwxrwx 1 root root 51 Oct 10 23:57
/ibm/gpfs1/production/.volumes/pvc-2969aa3d-ad12-4bf8-9793-05745c3b704f ->
/ibm/gpfs1/pvc-2969aa3d-ad12-4bf8-9793-05745c3b704f
#
```

E. Existing file sizes

```
# du -hs /ibm/gpfs1/pvc-f1cc724f-d730-49f2-a6e9-fcc94aaa2369/*
32M        /ibm/gpfs1/pvc-f1cc724f-d730-49f2-a6e9-fcc94aaa2369/health-app.log
2.8M       /ibm/gpfs1/pvc-f1cc724f-d730-49f2-a6e9-fcc94aaa2369/health-app.log-20190916
784K       /ibm/gpfs1/pvc-f1cc724f-d730-49f2-a6e9-fcc94aaa2369/health-app.log-20190922
1008K      /ibm/gpfs1/pvc-f1cc724f-d730-49f2-a6e9-fcc94aaa2369/health-app.log-20190930
744K       /ibm/gpfs1/pvc-f1cc724f-d730-49f2-a6e9-fcc94aaa2369/health-app.log-20191006
```

F. Write Compression Policy

```
# cat compressed_fileset.policy
RULE  'COMPRESS_FILESET'  MIGRATE COMPRESS('true') FOR
FILESET('pvc-f1cc724f-d730-49f2-a6e9-fcc94aaa2369') WHERE NAME LIKE 'health-app.log-%'
#
```

G. Apply Compression Policy

```
# mmapplypolicy gpfs1 -I yes -P compressed_fileset.policy
[I] GPFS Current Data Pool Utilization in KB and %
Pool_Name                      KB_Occupied               KB_Total         Percent_Occupied
system                              4513792               41943040             10.761718750%
[I] 4123 of 2250752 inodes used: 0.183183%.
[I] Loaded policy rules from compressed_fileset.policy.
Evaluating policy rules with CURRENT_TIMESTAMP = 2019-10-11@19:21:27 UTC
Parsed 1 policy rules.
RULE  'COMPRESS_FILESET'  MIGRATE COMPRESS('true') FOR
FILESET('pvc-f1cc724f-d730-49f2-a6e9-fcc94aaa2369') WHERE NAME LIKE 'health-app.log-%'
[I] 2019-10-11@19:21:27.452 Directory entries scanned: 36.
[I] Directories scan: 25 files, 9 directories, 2 other objects, 0 'skipped' files and/or errors.
[I] 2019-10-11@19:21:28.303 Parallel-piped sort and policy evaluation. 36 files scanned.
[I] 2019-10-11@19:21:28.354 Piped sorting and candidate file choosing. 4 records scanned.
[I] Summary of Rule Applicability and File Choices:
 Rule#           Hit_Cnt             KB_Hit             Chosen          KB_Chosen
KB_Ill        Rule
     0                 4              5344                  4              5344
0       RULE 'COMPRESS_FILESET' MIGRATE COMPRESS('true') FOR FILESET(.) WHERE(.)

[I] Filesystem objects with no applicable rules: 20.

[I] GPFS Policy Decisions and File Choice Totals:
 Chose to migrate 5344KB: 4 of 4 candidates;
Predicted Data Pool Utilization in KB and %:
Pool_Name                      KB_Occupied               KB_Total         Percent_Occupied
system                              4513792               41943040             10.761718750%
```

```
[I] 2019-10-11@19:21:28.498 Policy execution. 4 files dispatched.
[I] A total of 4 files have been migrated, deleted or processed by an EXTERNAL EXEC/script;
        0 'skipped' files and/or errors.

H. Files Are compressed
# du -hs /ibm/gpfs1/pvc-f1cc724f-d730-49f2-a6e9-fcc94aaa2369/*
32M        /ibm/gpfs1/pvc-f1cc724f-d730-49f2-a6e9-fcc94aaa2369/health-app.log
256K        /ibm/gpfs1/pvc-f1cc724f-d730-49f2-a6e9-fcc94aaa2369/health-app.log-20190916
72K        /ibm/gpfs1/pvc-f1cc724f-d730-49f2-a6e9-fcc94aaa2369/health-app.log-20190922
80K        /ibm/gpfs1/pvc-f1cc724f-d730-49f2-a6e9-fcc94aaa2369/health-app.log-20190930
64K        /ibm/gpfs1/pvc-f1cc724f-d730-49f2-a6e9-fcc94aaa2369/health-app.log-20191006
```

A.3 Encryption use case

Example A-3 depicts the steps to encrypt the `*.doc` files stored in a persistent volume.

Example A-3

```
1. Create Independent Fileset and link it

# mmcrfileset gpfs1 encrypted_datestore --inode-space new --inode-limit 5000:5000
Fileset encrypted_datestore created with id 8 root inode 655363.

# mmlinkfileset gpfs1 encrypted_datestore -J /ibm/gpfs1/encrypted_datestore
Fileset encrypted_datestore linked at /ibm/gpfs1/encrypted_datestore
#

2. Create encryption Policy to encrypt *.doc files under fileset created in step 1.
# cat encryption_policy.stanza
RULE 'p1' SET POOL 'system' /* one placement rule is required at all times */


RULE 'simpleEncRule' ENCRYPTION 'E1' IS
ALGO 'DEFAULTNISTSP800131A'
KEYS('KEY-a31f678-065b50aa-6c64-4fad-9655-3d4a0a5278b9:node2_devG1')


RULE 'CryptStorageClass' SET ENCRYPTION 'E1'
    [FOR FILESET ('encrypted_datestore')]
    [WHERE NAME LIKE '%.doc']

# mmchpolicy gpfs1 encryption_policy.stanza
Validated policy 'encryption_policy.stanza': Parsed 3 policy rules.
Policy `encryption_policy.stanza' installed and broadcast to all nodes.

# mmlspolicy gpfs1 -L
RULE 'p1' SET POOL 'system' /* one placement rule is required at all times */


RULE 'simpleEncRule' ENCRYPTION 'E1' IS
ALGO 'DEFAULTNISTSP800131A'
KEYS('KEY-a31f678-065b50aa-6c64-4fad-9655-3d4a0a5278b9:node2_devG1')
```

```
RULE 'CryptStorageClass' SET ENCRYPTION 'E1'
    [FOR FILESET ('encrypted_datestore')]
    [WHERE NAME LIKE '%.doc']
#
```

3. Create the Storage Class to create lightweight volumes by specifying details of Fileset

```
# cat storageclass.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
   name: csi-spectrum-scale-gpfs1-lightweight-volumes-selective-encryption
provisioner: csi-spectrum-scale
parameters:
    volBackendFs: "gpfs1"
    volDirBasePath: "encrypted_datestore"
reclaimPolicy: Delete
```

Note : volDirBasePath: "encrypted_datestore" is the relative path of fileset junction from filesystem mountpoint.

```
# kubectl create -f  storageclass.yaml
storageclass.storage.k8s.io/csi-spectrum-scale-gpfs1-lightweight-volumes-selective-encryption
created
```

4. Create pvc by specifying the storage class created in step 3.

```
# cat examples/dynamic/fileset/pvc.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gpfs1-encrypted-store-pvc
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: csi-spectrum-scale-gpfs1-lightweight-volumes-selective-encryption

# kubectl create -f pvc.yaml
persistentvolumeclaim/gpfs1-encrypted-store-pvc created

# kubectl get pvc gpfs1-encrypted-store-pvc
NAME                        STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS                                                        AGE
gpfs1-encrypted-store-pvc  Bound   pvc-c6637db9-5b0a-4d1a-8a31-9ea8336b92df  1Gi      RWX
csi-spectrum-scale-gpfs1-lightweight-volumes-selective-encryption   9s
```

5. Get the volume path details from the pv handle

```
# kubectl get pv pvc-c6637db9-5b0a-4d1a-8a31-9ea8336b92df -o yaml
apiVersion: v1
kind: PersistentVolume
….
….
  csi:
    driver: csi-spectrum-scale
    fsType: ext4
    volumeAttributes:
      storage.kubernetes.io/csiProvisionerIdentity: 1572974492998-8081-csi-spectrum-scale
      volBackendFs: gpfs1
      volDirBasePath: encrypted_datestore
    volumeHandle:
17258972170939727157;09762E6B:5D36FE7F;path=/ibm/gpfs1/production/.volumes/pvc-c6637db9-5b0a-4d1
a-8a31-9ea8336b92df
  persistentVolumeReclaimPolicy: Delete
  storageClassName: csi-spectrum-scale-gpfs1-lightweight-volumes-selective-encryption
  volumeMode: Filesystem
status:
  phase: Bound
#

# kubectl get pv -o jsonpath={.items[0].spec.csi.volumeHandle}
17258972170939727157;09762E69:5D36FE8D;fileset=3;path=/ibm/gpfs1/production/.volumes/pvc-763412d
9-2cad-4d03-bb31-164d1e80cef8
```

6. Create pod with pvc created in step 5.

```
# cat pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: csi-scale-fsetdemo-pod
  labels:
    app: nginx
spec:
  containers:
   - name: web-server
     image: nginx
     volumeMounts:
       - name: mypvc
         mountPath: /usr/share/nginx/html/scale
     ports:
     - containerPort: 80
  volumes:
   - name: mypvc
     persistentVolumeClaim:
       claimName: gpfs1-encrypted-store-pvc
       readOnly: false

# kubectl create -f pod.yaml
pod/csi-scale-fsetdemo-pod created

# kubectl get pod csi-scale-fsetdemo-pod
NAME                     READY   STATUS    RESTARTS   AGE
csi-scale-fsetdemo-pod   1/1     Running   0          48s
```

7. Create sample.txt and sample.doc inside container in pvc mount path

```
# kubectl exec -it csi-scale-fsetdemo-pod touch /usr/share/nginx/html/scale/sample.txt
# kubectl exec -it csi-scale-fsetdemo-pod touch /usr/share/nginx/html/scale/sample.doc
```

8. Check on Spectrum scale if sample.txt is encrypted or not

```
# mmlsattr -L /ibm/gpfs1/production/.volumes/pvc-c6637db9-5b0a-4d1a-8a31-9ea8336b92df/sample.txt
```

```
file name:
/ibm/gpfs1/production/.volumes/pvc-c6637db9-5b0a-4d1a-8a31-9ea8336b92df/sample.txt
....
Misc attributes:      ARCHIVE
Encrypted:            no
#

9. Check on Spectrum Scale if sample.doc is encrypted or not
# mmlsattr -L /ibm/gpfs1/production/.volumes/pvc-c6637db9-5b0a-4d1a-8a31-9ea8336b92df/sample.doc
file name:
/ibm/gpfs1/production/.volumes/pvc-c6637db9-5b0a-4d1a-8a31-9ea8336b92df/sample.doc
....
Misc attributes:      ARCHIVE
Encrypted:            yes
#
```

Related publications

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

IBM Redbooks

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

- *IBM Spectrum Scale (formerly GPFS)*, SG24-8254
- *IBM Spectrum Scale Security*, REDP-5426
- *Enhanced Cyber Security with IBM Spectrum Scale and IBM QRadar*, REDP-5560
- *IBM Storage for Red Hat OpenShift Blueprint Version 1 Release 3*, REDP-5565

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:

- *Performability Analysis of I/O Bound Application on Container-based Server Virtualization Cluster*

  `https://www.researchgate.net/publication/288125229_Performability_analysis_of_IO_bound_application_on_container-based_server_virtualization_cluster`

- *Wharf: Sharing Docker Images in a Distributed File System*

  `http://ccl.cse.nd.edu/research/papers/wharf-socc-2018.pdf`

- *Performance Analysis of Containerized Applications on Local and Remote Storage*

  `https://www.cs.utah.edu/~manua/pubs/msst17.pdf`

IBM Spectrum Scale resources from IBM.com

- IBM Knowledge Center: IBM Spectrum Scale 5.0.4

  `https://www.ibm.com/support/knowledgecenter/STXKQY_5.0.4/ibmspectrumscale504_welcome.html`

- Filesets

  `https://www.ibm.com/support/knowledgecenter/en/STXKQY_5.0.4/com.ibm.spectrum.scale.v5r04.doc/bl1adv_filesets.htm`

- Active File Management

  https://www.ibm.com/support/knowledgecenter/en/STXKQY_5.0.4/com.ibm.spectrum.scale.v5r04.doc/bl1ins_activefilemanagement.htm

- Multi-cluster and Remote Mounts

  https://www.ibm.com/support/knowledgecenter/en/STXKQY_5.0.4/com.ibm.spectrum.scale.v5r04.doc/bl1adv_admmcch.htm

- IBM Spectrum Scale on AWS

  https://www.ibm.com/support/knowledgecenter/en/STXKQY_AWS_SHR/com.ibm.spectrum.scale.aws.v5r03.doc/bl1cld_aws_introtoaws.htm

- IBM Spectrum Scale GUI

  https://www.ibm.com/support/knowledgecenter/en/STXKQY_5.0.4/com.ibm.spectrum.scale.v5r04.doc/bl1ins_introtogui.htm

- IBM Spectrum Scale Developer Edition

  https://www.ibm.com/account/reg/us-en/signup?formid=urx-41728

- IBM Spectrum Scale REST API

  https://www.ibm.com/support/knowledgecenter/en/STXKQY_5.0.4/com.ibm.spectrum.scale.v5r04.doc/bl1adm_restapi_main.htm

- IBM Knowledge Center: IBM Spectrum Scale mmapplypolicy command

  https://www.ibm.com/support/knowledgecenter/en/STXKQY_5.0.4/com.ibm.spectrum.scale.v5r04.doc/bl1adm_mmapplypolicy.htm

- Compression support in IBM Spectrum Scale 5.0.0

  https://developer.ibm.com/storage/2018/01/11/compression-support-spectrum-scale-5-0-0

- IBM Knowledge Center: IBM Spectrum Scale File compression

  https://www.ibm.com/support/knowledgecenter/en/STXKQY_5.0.4/com.ibm.spectrum.scale.v5r04.doc/bl1adm_compression.htm

- IBM Knowledge Center: IBM Spectrum Scale Container Storage Interface Driver Limitations

  https://www.ibm.com/support/knowledgecenter/STXKQY_5.0.4/com.ibm.spectrum.scale.csi.v5r04.doc/bl1csi_limitations.html

- IBM Knowledge Center: IBM Spectrum Scale Container Storage Interface Driver configurations

  https://www.ibm.com/support/knowledgecenter/en/STXKQY_5.0.4/com.ibm.spectrum.scale.csi.v5r04.doc/bl1csi_config_csi.html

- IBM Knowledge Center: IBM Spectrum Scale Container Storage - Kubernetes to IBM Spectrum Scale node mapping

  https://www.ibm.com/support/knowledgecenter/STXKQY_5.0.4/com.ibm.spectrum.scale.csi.v5r04.doc/bl1csi_config_kubernet_SS_mapping_procedure_final.html

- IBM Knowledge Center: Upgrading IBM Spectrum Scale Container Storage Interface driver

  https://www.ibm.com/support/knowledgecenter/STXKQY_5.0.4/com.ibm.spectrum.scale.csi.v5r04.doc/bl1csi_instal_upgrade.html

- IBM Knowledge Center: Ensuring high availability of the GUI service

  `https://www.ibm.com/support/knowledgecenter/en/STXKQY_5.0.4/com.ibm.spectrum.scale.v5r04.doc/bl1adm_guihaconfig.htm`

- IBM Knowledge Center: IBM Spectrum Scale GUI

  `https://www.ibm.com/support/knowledgecenter/STXKQY_5.0.4/com.ibm.spectrum.scale.v5r04.doc/bl1ins_quickrefforgui.htm`

- IBM Knowledge Center: Installation of IBM Spectrum Scale Container Storage Interface driver - Performing pre-installation tasks

  `https://www.ibm.com/support/knowledgecenter/STXKQY_5.0.4/com.ibm.spectrum.scale.v5r04.doc/bl1ins_quickrefforgui.htm`

- IBM Knowledge Center: IBM Spectrum Scale - Preparation for encryption

  `https://www.ibm.com/support/knowledgecenter/STXKQY_5.0.4/com.ibm.spectrum.scale.v5r04.doc/bl1ins_quickrefforgui.htm`

- IBM Knowledge Center: IBM Spectrum Scale - Security mode

  `https://www.ibm.com/support/knowledgecenter/en/STXKQY_5.0.4/com.ibm.spectrum.scale.v5r04.doc/bl1adm_securitymode.htm`

- IBM Knowledge Center: IBM Spectrum Scale - Mounting a remote GPFS file system

  `https://www.ibm.com/support/knowledgecenter/en/STXKQY_5.0.4/com.ibm.spectrum.scale.v5r04.doc/bl1adv_admrmsec.htm`

- IBM Knowledge Center: IBM Spectrum Scale - Accessing a remote GPFS file system

  `https://www.ibm.com/support/knowledgecenter/en/STXKQY_5.0.4/com.ibm.spectrum.scale.v5r04.doc/bl1adv_admmcch.htm`

- IBM Knowledge Center: Installation of IBM Spectrum Scale Container Storage Interface driver - Troubleshooting

  `https://www.ibm.com/support/knowledgecenter/STXKQY_5.0.4/com.ibm.spectrum.scale.csi.v5r04.doc/bl1cs_sec_ug_troubleshooting_file.html`

- IBM Knowledge Center: Migrating from IBM Storage Enabler for Containers to IBM Spectrum Scale Container Storage Interface driver

  `https://www.ibm.com/support/knowledgecenter/STXKQY_5.0.4/com.ibm.spectrum.scale.csi.v5r04.doc/bl1csi_migrate.html`

Online resources

These websites are also relevant as further information sources:

- Red Hat OpenShift: Interactive Learning Portal

  `https://learn.openshift.com`

- Red Hat OpenShift Documentation

  `https://docs.openshift.com`

- Red Hat OpenShift: Machine requirements for a cluster with user-provisioned infrastructure

  `https://docs.openshift.com/container-platform/4.2/installing/installing_bare_metal/installing-bare-metal.html#installation-requirements-user-infra_installing-bare-metal`

- Red Hat OpenShift: Installing a cluster on vSphere

  `https://docs.openshift.com/container-platform/4.2/installing/installing_vsphere/installing-vsphere.html`
- Red Hat OpenShift: Configuring your firewall for OpenShift Container Platform

  `https://docs.openshift.com/container-platform/4.2/installing/install_config/configuring-firewall.html`
- Kubernetes Documentation

  `https://kubernetes.io/docs/home`
- Kubernetes CSI Sidecar Containers

  `https://kubernetes-csi.github.io/docs/sidecar-containers.html`
- Kubernetes: Production-Grade Container Orchestration

  `https://kubernetes.io`
- Kubernetes: Overview of kubeadm

  `https://kubernetes.io/docs/reference/setup-tools/kubeadm/kubeadm`
- Kubernetes: Installing Kubernetes with Kubespray

  `https://kubernetes.io/docs/setup/production-environment/tools/kubespray`
- Kubernetes: Hello Minikube

  `https://kubernetes.io/docs/tutorials/hello-minikube`
- An overview of the architecture for OpenShift Container Platform 4.2

  `https://access.redhat.com/documentation/en-us/openshift_container_platform/4.2/html/architecture/index`
- A management framework for extending Kubernetes with Operators

  `https://github.com/operator-framework/operator-lifecycle-manager`
- IBM Spectrum Scale Container Storage Interface (CSI) project

  `https://github.com/IBM/ibm-spectrum-scale-csi`
- Container Storage Interface (CSI) Specification

  `https://github.com/container-storage-interface/spec`
- Container Storage Interface (CSI)

  `https://github.com/container-storage-interface/spec/blob/master/spec.md`
- operator-lifecycle-manager: Installing OLM

  `https://github.com/operator-framework/operator-lifecycle-manager/blob/master/doc/install/install.md`
- IBM Spectrum Scale Secure - Secure Data in Motion and Rest

  `https://www.slideshare.net/SandeepPatil154/ibm-spectrum-scale-secure-secure-data-in-motion-and-rest`
- Active File Management (AFM): IBM Spectrum Scale User Group 2017, Manchester

  `https://github.com/operator-framework/operator-lifecycle-manager`
- IBM Spectrum Scale AFM/AFM DR: April 19th 2018, UK User Group Event

  `http://files.gpfsug.org/presentations/2018/London/2-B-3_IBM_Spectrum_Scale_AFM_UK_User_Group_April_19_2018.pdf`

- IBM Spectrum Scale CSI driver tool, spectrum-scale-driver-snap.sh, to collect driver debut data

  https://raw.githubusercontent.com/IBM/ibm-spectrum-scale-csi/v1.0.1/tools/spectrum-scale-driver-snap.sh
- How to set up a remote cluster with IBM Spectrum Scale – steps, limitations and troubleshooting

  https://developer.ibm.com/storage/2020/01/27/how-to-set-up-a-remote-cluster-with-ibm-spectrum-scale-steps-limitations-and-troubleshooting
- Container Runtimes Part 1: An Introduction to Container Runtimes

  https://www.ianlewis.org/en/container-runtimes-part-1-introduction-container-r
- IBM Garage Methodology

  https://www.ibm.com/garage/method
- IBM: Complimentary Cloud Adoption Briefing

  https://www.ibm.com/account/reg/us-en/signup?formid=urx-34892
- IBM Cloud Transformation Advisor

  https://www.ibm.com/garage/method/practices/learn/ibm-transformation-advisor
- Docker overview

  https://docs.docker.com/engine/docker-overview
- cri-o: Lightweight Container Runtime for Kubernetes

  https://cri-o.io

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

REDP-5589-00

ISBN 0738458643

Printed in U.S.A.