

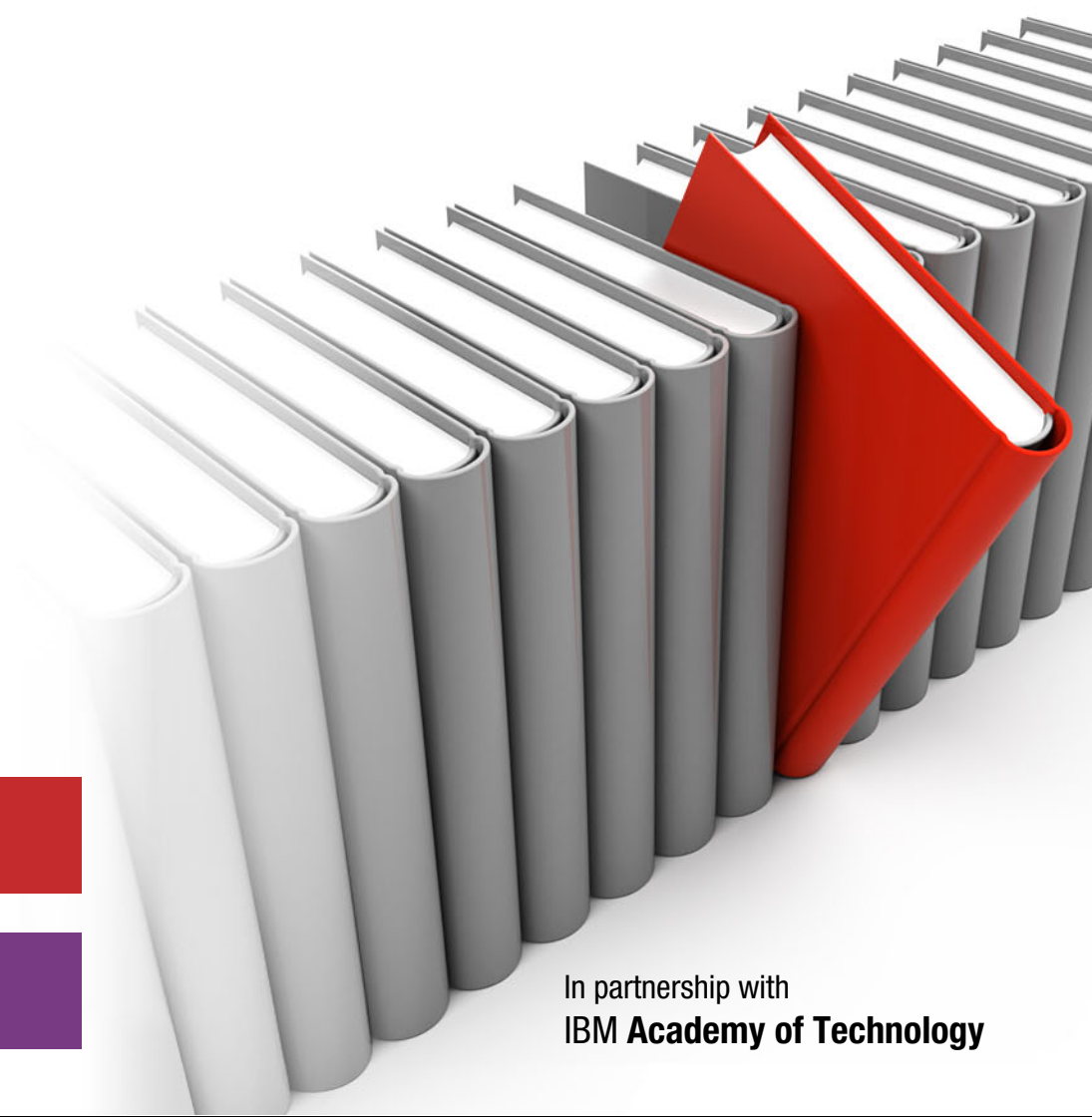
IBM Spectrum Scale in an OpenStack Environment

Bill Owen
Dean Hildebrand
Sandeep Ramesh
Gautam Shah
Gaurang Tapase
Kumar Nachiketa
Kedar Karmarkar
Larry Coyne



Cloud

Storage



In partnership with
IBM Academy of Technology

Find and read thousands of IBM Redbooks publications

- ▶ Search, bookmark, save and organize favorites
- ▶ Get personalized notifications of new content
- ▶ Link to the latest Redbooks blogs and videos

Get the latest version of the Redbooks Mobile App



Download
Now

Android



Promote your business in an IBM Redbooks publication

Place a Sponsorship Promotion in an IBM® Redbooks® publication, featuring your business or solution with a link to your web site.

Qualified IBM Business Partners may place a full page promotion in the most popular Redbooks publications. Imagine the power of being seen by users who download millions of Redbooks publications each year!



ibm.com/Redbooks

About Redbooks → Business Partner Programs

THIS PAGE INTENTIONALLY LEFT BLANK



Abstract

OpenStack is open source software that is widely used as a base with which to build cloud and infrastructure as a service solutions. OpenStack often is deployed on commodity hardware and used to virtualize various parts of the infrastructure (compute, storage, and network) to ease the sharing of the infrastructure across applications, use cases, or workloads.

IBM® Spectrum Scale is software that is used to manage storage and provide massive scale, a global namespace, and high-performance data access with many enterprise features. IBM Spectrum™ Scale is used in clustered environments and provides file protocol (POSIX, NFS, and SMB) and object protocol (Swift and S3) access methods.

Configuring IBM Spectrum Scale™ in systems that use OpenStack software offers benefits that are provided by the many enterprise features in IBM Spectrum Scale and the ability to consolidate storage for various OpenStack components and applications that are running on top of the OpenStack infrastructure under a single storage management plane.

This IBM Redpaper™ publication describes the benefits and best practice recommendations of the use of IBM Spectrum Scale in OpenStack environments. It also describes the steps that are used to configure the storage infrastructure that uses IBM Spectrum Scale with various OpenStack components.

The intended audience for this publication is technical decision makers, cloud architects, IT architects, and those readers who want to learn more about deploying an OpenStack cloud environment with Spectrum Scale storage.

Introduction

Spectrum Scale, Spectrum Scale Protocols, and OpenStack Cloud storage services are described in this section.

Spectrum Scale and Spectrum Scale Protocols

IBM Spectrum Scale is proven, highly scalable, and highly available enterprise software that can be used to build robust high-performance storage and data management solutions in various environments. Spectrum Scale is based on the IBM General Parallel File System (GPFS™). Spectrum Scale can be used to build file or object solutions by using various storage configuration options. GPFS is available on several platforms, including Linux distributions that are running on x86 and Power Systems (big and little endian), IBM AIX® that is running on Power Systems, Windows that is running on x86 and Linux on IBM z™ Systems. GPFS is available since 1998 and is used by thousands of customers.

Spectrum Scale provides a global name space for data access in a cluster. It is typically deployed on a cluster of nodes and supports many different use cases that include file serving for High Performance Computing applications that are running on the cluster, Analytics that is running on the cluster and accessing data that is served from this cluster, or sharing of data between applications that are running on the cluster with other applications.

To use Spectrum Scale, a Internet Protocol network that is configured is needed to allow all nodes in a cluster to communicate with each other. In addition, if you have low latency networks (such as InfiniBand), Spectrum Scale also can be configured to use RDMA on InfiniBand.

The following deployment options are available:

- ▶ All nodes in the cluster have access to the same storage via the storage area network, such as Fibre Channel or InfiniBand.
- ▶ Some nodes have connectivity to storage (often at least two nodes have access to the same storage). These nodes are called Network Shared Disk (NSD) servers. Other GPFS client nodes run applications and access data on storage via these NSD servers.
- ▶ All nodes in a cluster have only local storage that is not directly accessible by other nodes. A GPFS spans all of this local storage. This option is known as File Placement Optimizer (FPO). Unlike the first two options in which data availability is provided by the storage subsystem, GPFS replication ensures data availability in the FPO option.

Note: A Spectrum Scale deployment can include any mix of these deployment scenarios.

Three editions of IBM Spectrum Scale are available: Express, Standard, and Advanced. In each edition, nodes in the cluster must be designated with a license designation of server, client, or FPO. The features that are described in this section are part of the Standard Edition unless otherwise noted.

Spectrum Scale offers many features to enable building a robust data management solution. In terms of scaling, Spectrum Scale supports file systems with sizes of tens of petabytes that contain billions of files and can be accessed by thousands of nodes in a cluster. (Data throughput performance of over 400 GBps was demonstrated.)

Spectrum Scale provides a choice of separating metadata and data on separate storage. This capability has several benefits, including the ability to limit the use of more expensive storage, such as Flash or solid-state drives (SSD) for metadata only.

Replication of data or metadata can be independently configured to give control of your data's durability. Although replication of metadata is recommended, replication of data is a choice that is based on the nature of the data. Native access to Spectrum Scale file systems support standard POSIX interface and provides POSIX semantics.

Spectrum Scale features infrastructure to enable information lifecycle management (ILM) by grouping storage into different storage pools, manage namespaces by using filesets, and providing a policy engine to express the relationship of the namespace to the storage in terms of placement and movement of data.

The policy engine can scan the metadata of a file system quickly. This speed allows policy operations to run efficiently for large data sets. For example, we demonstrated scanning 10 billion files in 43 minutes. The ILM capability also is integrated with Hierarchical Storage Management (HSM) software (such as IBM Spectrum Protect™ HSM) to allow data to flow automatically from disk to tape. Snapshots of file systems and filesets can be taken to provide a space efficient point-in-time copy of the namespace

File clone capability is provided for space efficient copies of individual files. Features, such as local read-only cache and highly available write cache, improve performance for certain workloads by using Flash or SSD storage.

To support different administrative domains, Spectrum Scale offers multi-cluster support that provides controls to allow access on a cluster that is managed by a different administrator. In addition to multi-cluster support, the Active File Management (AFM) feature allows a remote file system (NFS or GPFS) to be locally cached. The namespace is managed by Spectrum Scale, so the user does not have to explicitly manage local or remote copies of a file.

To support analytics applications on shared nothing clusters, Spectrum Scale's File Place Optimizer (FPO) feature allows controlling the placement of replicas and allocation of storage on a device in larger chunks. The use of FPO with OpenStack deployments is described in "Deployment models" on page 11.

IBM Elastic Storage™ Server (ESS) is a storage solution that uses Spectrum Scale Native RAID capability that implements de-clustered raid in software on disks that are attached to dual servers without a storage controller. Another feature that is available in Spectrum Scale is policy-driven compression to enable better storage utilization.

Spectrum Scale has several system operations that can perform I/O and can take some time or use a large portion of system resources, such as file system restripe or policy scans. These operations can affect user I/O. Spectrum Scale quality of service (QoS) function allows controlling the ratio of system I/O activity to the user I/O activity.

Spectrum Scale offers encryption for data at rest and secure deletion. Asynchronous Disaster Recovery (ADR) (which is based on AFM) allows Spectrum Scale deployments to have a copy of data at another site to facilitate dealing with site failures. Encryption and ADR are available only in the advanced edition of Spectrum Scale. Spectrum Scale can be managed by using a command-line interface (CLI) or a recently introduced GUI.

To enable access to data that is managed by Spectrum Scale by clients or nodes that are not part of the Spectrum Scale cluster, Spectrum Scale also accesses the data by using the following protocol access methods:

- ▶ NFS
- ▶ SMB
- ▶ Object protocol (by using OpenStack Swift or Amazon S3 protocol)

This file and object access integration enables customers to consolidate various sources of data efficiently in one global namespace. This integration creates NFS exports, SMB shares, and Swift containers that have data in Spectrum Scale file systems for access by client systems that do not run GPFS. Two or more nodes in the Spectrum Scale cluster must be designated as protocol nodes from which non-Spectrum Scale clients can access these artifacts (exports, shares, or containers). Standard and Advanced editions of the product are required to use these features.

Note: Protocol nodes are also referred to as Cluster Export Services (CES) nodes in Spectrum Scale documentation.

The Spectrum Scale protocol nodes must have GPFS server license designations. The protocol nodes must be configured with external network addresses that are used to access artifacts from clients.

Note: The external network addresses are different from the GPFS cluster address that used to add the protocol nodes to the GPFS cluster.

The integration that is provided allows the artifacts to be accessed from any of the protocol nodes via the configured network addresses. The integration also supports the automatic failover of network addresses from a failed protocol node to the remaining protocol nodes.

Currently, all protocol nodes must use the RHEL7 operating system. The protocol nodes also must all be the same architecture; that is, all Power (in big endian mode) or all Intel x86. Other nodes in the Spectrum Scale cluster can use other platforms and operating systems. The Spectrum Scale protocol functionality is delivered as software, as is the rest of Spectrum Scale.

The NFS server (Ganesha Stack), SMB server (SAMBA stack), and OpenStack Swift Server are all packaged with the Spectrum Scale Software delivery. In an OpenStack environment, you must use the provided OpenStack packages at least on the protocol nodes (which are the nodes that are responsible for providing OpenStack Swift access) to avail the integration of OpenStack Swift that is provided with Spectrum Scale.

For more information about the features and capabilities, see the IBM Spectrum Scale product documentation that is available this website:

https://www.ibm.com/support/knowledgecenter/STXKQY/ibmspectrumscale_welcome.html

Solution and use cases provided by OpenStack components

The OpenStack platform is open source software that is widely used as a base to build cloud and Infrastructure as a Service solutions. The benefits of these solutions are many and varied, including reduced operational cost and increased operational efficiency, the ability to deploy applications more quickly and with standardized APIs, and the ability to deploy these solutions by using the compute, storage, and networking infrastructure that is best suited for the workload.

For more information, see this website:

<https://ibm.biz/Bd4Lnw>

The OpenStack platform consists of a set of loosely coupled services, each delivering a specific set of functions through RESTful APIs. The core of OpenStack is made up of the following essential services:

- ▶ Nova: Management of Compute Resources
- ▶ Keystone: Identity Management
- ▶ Glance: Image Storage
- ▶ Cinder: Virtual Block Storage
- ▶ Swift: Object Storage
- ▶ Neutron: Management of Network Resources

There are many other optional services that can be configured in an OpenStack environment, including mature projects, such as the Manila shared file system service and newer projects, such as Magnum for container management. For more information, see this website:

<http://www.openstack.org/software/project-navigator>

In this Redpaper publication, we focus on the following OpenStack cloud storage projects and the Keystone service for identity management:

- ▶ Cinder

This project provides virtualized block storage to virtual machines (VMs). By using Cinder, a user can create volumes from scratch or from data, including a Glance image, another Cinder volume, or a Cinder volume snapshot. Cinder volumes that are created from a Glance image or from another bootable Cinder volume are designated as bootable and can be used by Nova to boot a VM. Other Cinder volumes can be attached to VMs.

The Cinder architecture supports a pluggable backend for volume drivers. This ability allows storage vendors to write drivers that specialize the volume management operations for their storage platform. The Spectrum Scale volume driver (also known as the GPFS driver) is written to make full use of the Spectrum Scale enterprise features.

- ▶ Glance

This project provides the capability to manage VM images. VMs can be booted from Glance images or Cinder bootable volumes can be created from Glance images. When Glance is configured to use the same Spectrum Scale fileset that stores Cinder volumes, bootable images can be created almost instantly by using the copy-on-write file clone capability.

► Swift

This project provides object storage to any user or application that requires access to data through a RESTful API from anywhere; that is, from VMs that are running in the OpenStack platform or applications that are running anywhere in the world. The Swift object storage configuration is optimized for the Spectrum Scale environment, which provides high availability and simplified management. Swift object storage also supports native Swift API and Amazon S3 APIs for accessing data. Swift object storage also supports access to the same data through object interface or file interface (POSIX, NFS, and SMB) without creating a copy.

► Manila

This project provides shared file system access to any client (virtual or physical). For example, you can use Manila to deploy shared file system storage to a collection of VMs and external clients that require access to the same data. As with Cinder, Manila uses a pluggable backend for volume drivers. The Spectrum Scale share driver (also known as the GPFS driver) is written to make full use of the Spectrum Scale enterprise features.

Note: At the time of this writing, the GPFS Manila share driver uses a version of the Ganesha NFS server that is not compatible with Spectrum Scale V4.2.0. For this reason, we do not describe Manila deployment in this paper. An update of this Redpaper publication, will describe the deployment and use of Manila in a Spectrum Scale environment.

In this Redpaper publication, we describe an end-to-end OpenStack cloud scenario and highlight the advantages of the OpenStack platform with Spectrum Scale storage. After you configure your system, you can upload one or more boot images into the Glance image repository. By using the Cinder volume service, you can create bootable volumes that are persistent so that the state of a VM image can be maintained across boot operations and begin the use of those images in VMs that were created by Nova.

Deployment options

One purpose of this paper is to describe the steps that are used to configure the OpenStack platform with Spectrum Scale. The range of deployments can be single VM up to a full-scale production environment.

To begin, you need is a GPFS cluster and access to an OpenStack distribution. If you do not have GPFS, an easy way to start is by using the Spectrum Scale Trial VM, which is available for download from this website:

<http://www.ibm.com/developerworks/servicemanagement/tc/gpfs/evaluate.html>

Licensing reference to FAQ

In this paper, we describe two primary architectures for deploying OpenStack with Spectrum Scale. There are different licensing requirements for each architecture. We do not provide specific information about licensing options in this paper. For more information, see the following IBM Spectrum Scale FAQ licensing section:

<https://ibm.biz/Bd4LbE>

Advantages of using Spectrum Scale Storage for OpenStack

The success of an OpenStack deployment depends heavily on the chosen storage solution. Many of the benefits of OpenStack can be reduced or eliminated if the storage system cannot perform the job for which OpenStack was chosen. The benefits of the use of Spectrum Scale for OpenStack deployments and why it can deliver on the OpenStack promise are described in this section.

Realizing the promise of OpenStack

OpenStack services are heavy users of any storage system. The number and performance of the virtual hosts determines the amount of work that is accomplished by your cloud infrastructure and leads directly to the speed of a company and its ability to innovate.

Spectrum Scale helps an organization realize the promise of OpenStack by enabling the deployment of massively scalable and high-performance clouds. With Spectrum Scale, extra capacity and bandwidth can be incrementally added seamlessly and on demand without disrupting your applications. By enabling your storage to grow as needed, you also can grow your OpenStack deployment.

When designed correctly, a company's cloud infrastructure increases its ability to innovate and compete. Downtime, unnecessary data movement, poor performance, complexity, and security gaps are roadblocks in running your business and helping it to move faster.

Spectrum Scale provides a secure, mature, and robust storage system that is designed for high availability and efficiency. This efficiency allows your cloud solution to do more with fewer resources, which saves money through reduced management and hardware costs. It also avoids the delays that are caused by inefficient data access and slow performing solutions by providing a single and highly scalable data plane for all of your data. This single data plane enables a tight integration of all the OpenStack storage projects, which provides a unique solution that propels your business forward.

Just as OpenStack promises elasticity of your cloud resources with on-demand provisioning, Spectrum Scale can manage the workload no matter how large it grows. The OpenStack scalable system management tools support this growth, but are not involved with managing the storage system. Time and resources can be wasted performing storage management if the storage solution's management tools cannot keep pace with your cloud. The Spectrum Scale management framework allows management of thousands of servers and petabytes of data, all from a single management pane, which greatly increases operational efficiency and allows your organization to focus on getting work done.

Expect more from your storage with Spectrum Scale and OpenStack

Your OpenStack solution performs many jobs in your organization, and you should expect the same from your storage solution. Do not settle for a siloed storage solution that serves only your VM needs but then requires you to support more storage solutions for all of your other application needs.

Spectrum Scale is also designed to perform many jobs, which creates a single massive capacity pool that can support storage for VMs, multi-platform data access through various file and object access protocols, data analytics through Spark, Hadoop, and so on, and many more use cases.

Spectrum Scale also includes the following enterprise features that are beneficial to OpenStack deployments:

- Scalability and high performance

Experience seamless and practically unlimited growth of your OpenStack cloud solution. The efficiency and performance of Spectrum Scale gives you the best file access storage solution on the market today. It also allows you get more work done by using less hardware, which reduces time to value and cost.

- Highly available

Spectrum Scale is designed with redundancy in mind, which ensures that your OpenStack deployment is continuously available. In addition, Nova VMs can be seamlessly live migrated between servers (or immediately relaunched on another node if there is a failure) because data is always available from every server in your cloud.

- Information lifecycle management

As the size of your storage system grows, ensuring data is in the correct place becomes more of a challenge. Spectrum Scale provides powerful, policy-driven automated tiered storage management. For example, Cinder volumes can be automatically moved from a slow tier to a fast tier as it is used by a VM, while infrequently accessed Glance images continue to be in the slowest tier. The oldest Manila file data and Glance images can be automatically based on policy and moved to lower-cost tiers, such as tape, by using the Spectrum Scale policy engine.

- Compression

Although placing data on the correct tier lowers cost, the use of compression can lower costs even further. Spectrum Scale can compress your virtual images and file data seamlessly in the background, which ensures your applications are not affected.

- Scalable backup and disaster recovery

Enterprise data requires proven and robust data management tools to ensure that data is not lost because of user or software errors or hardware failures and errors. Although simple data replication protects against hardware failures, it is not an end-to-end solution that allows recovering data that was incorrectly modified by a user, software application, or in specific cases, hardware corruption.

Spectrum Scale can create consistent point-in-time copies of Cinder volumes, Cinder consistency groups, the Glance repository, and Manila data to protect against data loss or corruption. These copies can be created and stored instantly within Spectrum Scale by using snapshots that are stored in an external mass capacity storage system by using backups or replicated to another site to protect against a site failure.

- Security

OpenStack was designed to be a secure cloud platform. When Spectrum Scale storage is used, this focus continues through support for data encryption at rest and optional strong security modes that authenticate nodes in the cluster and encrypt all communication.

- Scalable search tools

As the number of Glance and Cinder images grows and more Manila file data is created, searching for the correct information for user and administrative purposes becomes more difficult. Spectrum Scale can scan millions of pieces of information per second by scaling out the search across the entire system, which allows data to be quickly found when it is needed.

► Flexible shared block, file, and object storage

Spectrum Scale offers block storage for your Cinder volume, POSIX, SMB, NFS, Amazon S3, and OpenStack Swift APIs for file and object access, and HDFS for analytics.

Multi-protocol support allows applications to use the correct protocol for the job. VMs can access their Cinder volumes via NFS or POSIX from the hypervisor and then use one or more of the protocols to access data from inside the VM. Manila also can use NFS for multi-tenant access to file data. But most importantly, multiprotocol access allows Spectrum Scale to be used for much more than just providing storage for VMs. Your entire organization can use Spectrum Scale for most if not all of its storage needs, no matter what they might be.

Helping OpenStack Storage work together

Spectrum Scale allows for a unique integration of the OpenStack storage projects.

First, one of the most common tasks in an OpenStack environment is provisioning bootable storage volumes for VMs. Starting operating system volumes are initially stored in Glance. These images are read-only and provide the collection of different operating system (OSs) and software configurations (these images are sometimes referred to as *gold* images). Before booting a VM, its OS image must be copied to a read/write version. Therefore, the speed with which these volumes can be copied controls the speed at which new VMs can be provisioned and started.

Copying these images can be costly in terms of resources and time because they tend to be gigabytes in size. Spectrum Scale avoids copying, and instead can almost instantly provision these OS disk images from Glance by using “cloning”. With a clone, the new image initially points to the gold image and then any changes are written to the new image only. This process creates a more scalable cloud and reduces the load on the storage system. Therefore, it does not affect running applications (as happens when the Glance images are copied).

Second, OpenStack includes a file service that is called Manila and an object service that is called Swift; however, they do not inherently work together. Normally, Swift objects cannot be accessed via NFS, and Manila files cannot be accessed via Swift. But, when Spectrum Scale storage is used, these data access limitations are removed through the use of Spectrum Scale’s unified file and object capability. Without requiring support for a file gateway to the Swift service (which limits the performance of your system), Spectrum Scale can allow clients to access Manila shares via Swift and to provision Manila NFS access to Swift containers.

The features that are provided by Spectrum Scale working together with OpenStack storage components are listed in Table 1.

Table 1 Spectrum Scale and OpenStack features

Spectrum Scale and OpenStack features	Support status
Volume and Image Management (Cinder and Glance)	
Volume Creation and Deletion	Yes
Volume Attach and Detach to a VM instance	Yes
Volume Snapshot management	Yes
Volume Creation from Snapshot	Yes
Efficient cloning of Volumes	Yes

Spectrum Scale and OpenStack features	Support status
Extend Volumes	Yes
Copy Image to Volume and Copy Volume to Image	Yes
Instantaneous Boot Volume Create From Glance Repo	Yes
Live migration of Instances	Yes
Backup of Volumes	Yes
Sharing of Data between instances (with file system support)	Yes
Quality of service that uses multitier storage (with Flash support)	Yes
Encryption of Volumes	Yes
Tiering of Volumes	Yes
Compression of Volumes	Yes
Identity management (Keystone)	
Integrated High Availability across Spectrum Scale Protocol Nodes	Yes
Easy configuration, management, and monitoring	Yes
AD and LDAP Support	Yes
External Keystone Support	Yes
Object Store features (Swift)	
Unified file and Object Support	Yes
In-Place Analytics with Hadoop compatibility	Yes
High Performance and High Availability	Yes
Object Compression	Yes
Object Encryption	Yes
Multi-Region Support	Yes
WAN caching with Active File Management (AFM)	Yes
Policy-based information lifecycle management	Yes
Easy installation, configuration, and management	Yes
Integrated monitoring	Yes
Swift and S3 API Support	Yes
Large Object Support (5 TB)	Yes
Support for OpenStack Swift object store features	Yes
Manila	
Support for NFS	Yes
Support for SMB	No

How Spectrum Scale can provide access to the same Manila and Swift data sets is shown in Figure 1.

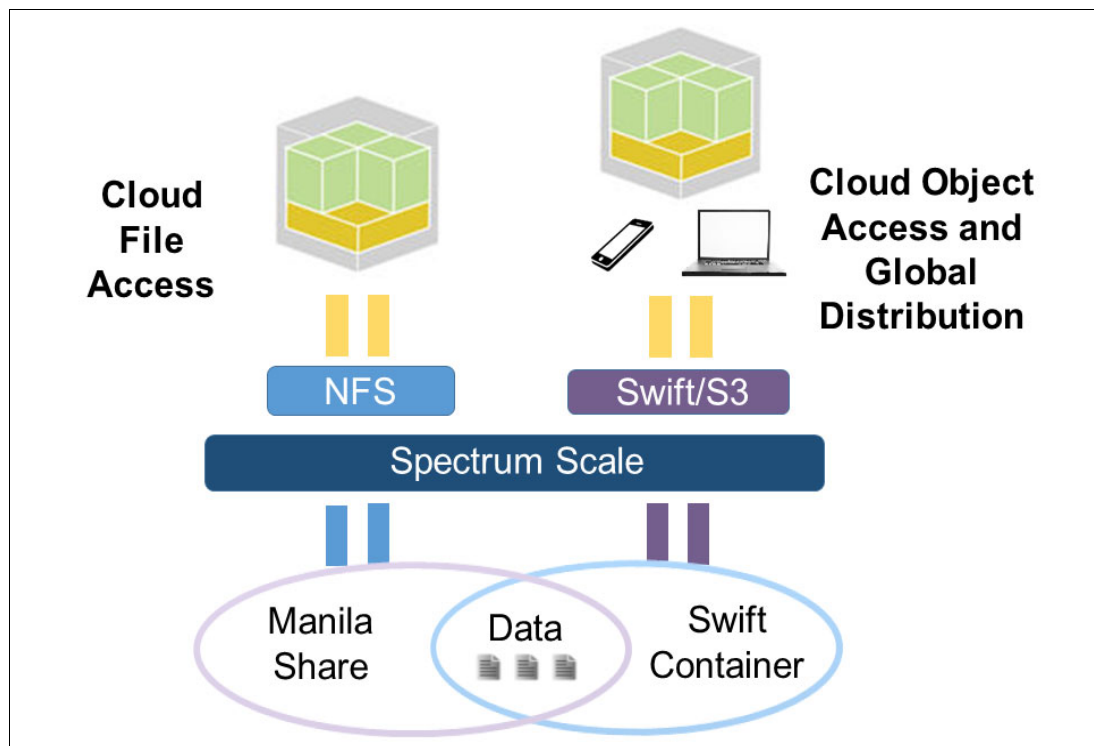


Figure 1 Spectrum Scale can provide access to the same Manila and Swift data sets

The combination of file and object is powerful for many use cases because it allows applications to use the best protocol for their requirements. Global access to data can be provided via Swift, which supports strong authentication via Keystone and HTTP (REST-based) access, while applications that require file semantics (such as media editing applications) can update the same data. For example, file-based applications can generate the data to a Manila share, and then share this data globally via Swift. Also, data can be uploaded from anywhere in the world via Swift, analyzed and processed by using Spark or Hadoop to a Manila share, and then share the results globally.

Deployment models

This section describes the planning and recommended deployment models when OpenStack software is used with Spectrum Scale storage.

Planning for OpenStack components

This section describes the required planning to deploy an OpenStack environment by using Spectrum Scale storage.

Obtaining the OpenStack drivers for Spectrum Scale

Storage that is consolidated and managed by IBM Spectrum Scale can be used to provision (Cinder) block storage, (Manila) file storage, and (Swift) object storage in an OpenStack deployment. In addition, Glance images and Nova instances can be stored in this Spectrum Scale managed storage infrastructure. Object storage is available as a part of Spectrum Scale release.

From the OpenStack Liberty release, there are new updates to the Spectrum Scale Cinder driver, including support for NFS (when Cinder Volume Service is not running on GPFS node). OpenStack distributions that are compatible with these releases can be used for deploying Spectrum Scale Cinder drivers.

Spectrum Scale inherently provides Swift-based object store as a part of its protocol nodes and can be directly used for object-based services that are required in the deployment. Similarly, when configuring object protocol, Spectrum Scale gives the option to configure Keystone on the Spectrum Scale protocol nodes. Alternatively, the Keystone service can be used by the other OpenStack services for the deployment. OpenStack Cinder and Manila drivers for Spectrum Scale are available with all leading OpenStack distributions.

The driver source code can be downloaded from the following websites:

- ▶ <https://github.com/openstack/cinder>
- ▶ <https://github.com/openstack/manila>

The drivers also are available from any other current OpenStack distribution.

OpenStack component deployment planning

At a high level, the role of the nodes in an OpenStack environment can be categorized based on the OpenStack services they provide.

Controller nodes

Controller nodes run API services for Glance image service, Cinder block storage service, Nova compute service, and the OpenStack Horizon Dashboard. Often, these nodes also run the identity service (Keystone). If Object storage Swift from IBM Spectrum Scale is used, Keystone services can run on IBM Spectrum Scale Protocol Nodes.

Compute nodes

Compute nodes run the OpenStack Compute service (Nova-Compute) with the hypervisor, which manages the VM instances. In this Redpaper publication, we describe the use of the KVM hypervisor.

Object nodes and protocol nodes

Object nodes and protocol nodes are Spectrum Scale nodes that host the Object, NFS, and SMB protocols.

Storage nodes

There often are no dedicated storage nodes in a Spectrum Scale environment. Storage is made available to controller and compute nodes by mounting Spectrum Scale file systems on those nodes by using GPFS Client or NFS protocol.

The following approaches can be used for deploying OpenStack software with IBM Spectrum Scale storage:

- ▶ **Shared Storage:** Spectrum Scale that is running with Elastic Storage Server or other Shared Storage subsystem as a storage backend.
- ▶ **Local Storage:** Spectrum Scale that is running with storage-rich servers in an FPO-based cluster configuration.

Organizing OpenStack storage with Spectrum Scale

Spectrum Scale provides many options to organize your data, including file systems, and dependent and independent filesets. This section describes the recommended organization that uses independent filesets.

Independent fileset per Swift policy

Spectrum scale allows users to create multiple Swift policies that are based on their requirements. Every policy creates a separate independent fileset. For example, you can create an object storage policy for encryption so that data that is stored in containers that are associated with that policy is encrypted on disk. This configuration internally creates an independent fileset for that object data.

Independent fileset for Cinder consistency group

Spectrum Scale Cinder Driver allows creating volumes in consistency groups. Each consistency group maps to a separate independent fileset. Therefore, similar volumes can be grouped and stored in a single fileset under a consistency group. If you create a Consistency Group snapshot, the volumes in that group are saved at the same point to provide consistent data.

Independent fileset for hosting Glance, Cinder, and Nova

A common independent fileset for Glance, Cinder, and Nova allows all of the OpenStack storage to be under a same namespace. Having Glance and Cinder storage under the same fileset also uses Spectrum Scale's copy on write feature for creating volumes and images, which saves time and storage space.

Deployment model topologies

This section describes the different topologies that are supported by Spectrum Scale for OpenStack deployment.

Model 1: Shared Storage

Model 1 is an OpenStack deployment on Spectrum Scale cluster that is running with ESS or supported shared storage servers, as shown in Figure 2.

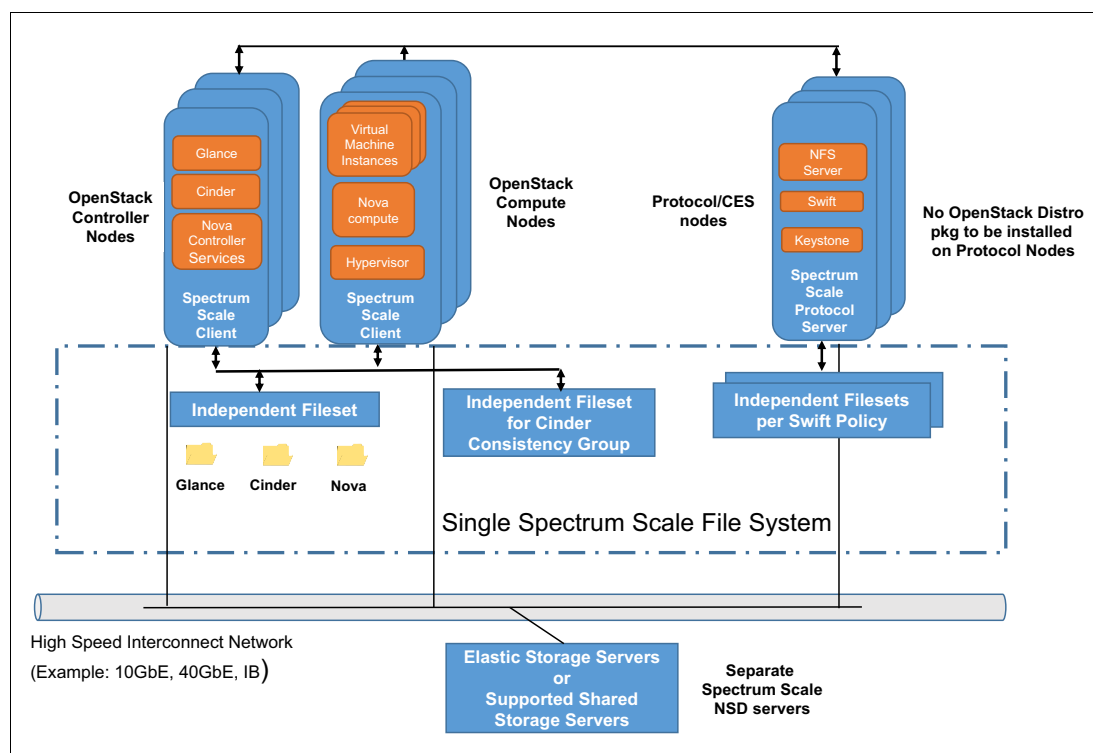


Figure 2 Shared Storage

In the architecture that is shown in Figure 2, there are independent sets of OpenStack controller nodes and dedicated OpenStack compute nodes. We recommend that all of these nodes are part of single Spectrum Scale cluster that is created with Shared Storage Servers and have at least Spectrum Scale client license designation.

It is assumed that the Shared Storage NSD servers have Spectrum Scale server licenses and meet the data availability requirement. Additionally, the protocol nodes serving object and Keystone are separate dedicated nodes with no other OpenStack service running on them.

In this model, the storage is served by IBM Elastic Storage Servers (or other storage controllers that are connected to NSD servers). IBM Elastic Storage Servers offer the benefit of densely packaged storage with declustered RAID capabilities in software. By using this model, you can scale compute separately from storage, which provides granular elasticity.

Model 2: Storage-rich servers with separate compute nodes

Model 2 features an OpenStack deployment with Spectrum Scale that is running on storage-rich servers in an FPO cluster and has OpenStack Compute nodes that are separate from FPO nodes, as shown in Figure 3.

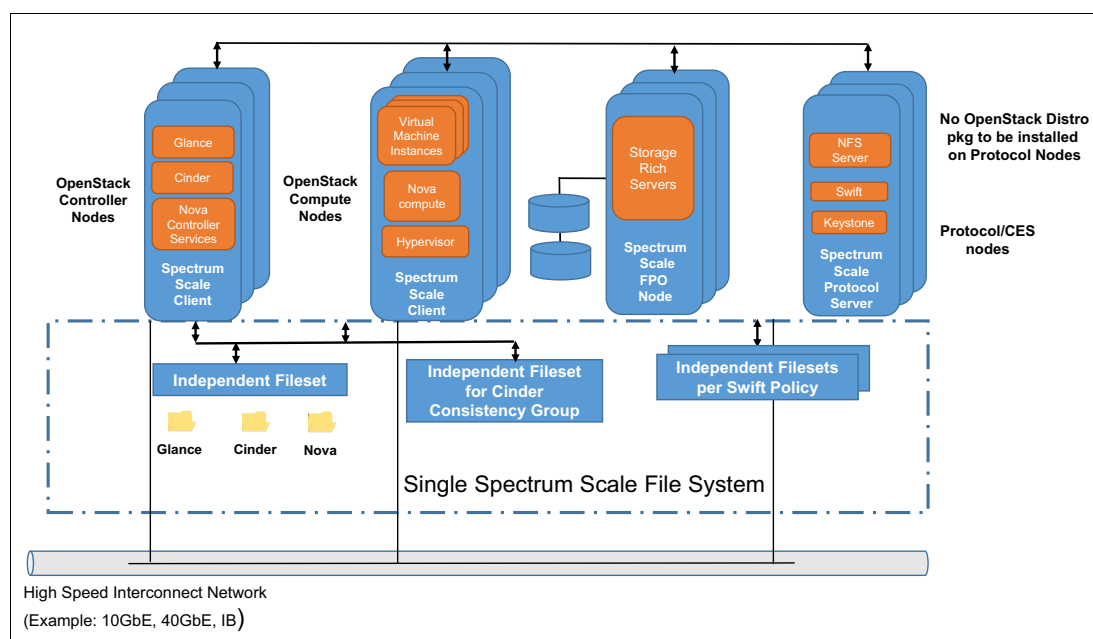


Figure 3 Storage-rich servers with separate compute nodes

In this model, the Spectrum Scale cluster is configured with storage-rich servers by using the Spectrum Scale file placement optimizer (FPO) feature. As in the previous shared storage architecture deployment model, we suggest configuring an independent set of controller nodes and compute nodes.

In this deployment, all of the storage-rich server nodes should have at least a GPFS FPO license. The compute nodes and controller nodes do not contribute to the overall storage infrastructure can be configured with GPFS client license. Additionally, the protocol nodes serving object and Keystone are separate dedicated nodes with no other OpenStack service running on them. It is assumed these nodes do not contribute to the overall storage infrastructure and used to serve the object data only. These nodes should have GPFS protocol server license.

By using this model, you can scale compute independently from storage that is providing the granular elasticity in which storage-rich servers can be used for storage.

Model 3: Storage-rich servers with converged storage and compute nodes

Model 3 is an OpenStack deployment with Spectrum Scale that is running on storage-rich servers in an FPO cluster and has OpenStack Compute nodes overlapping with FPO nodes, as shown in Figure 4.

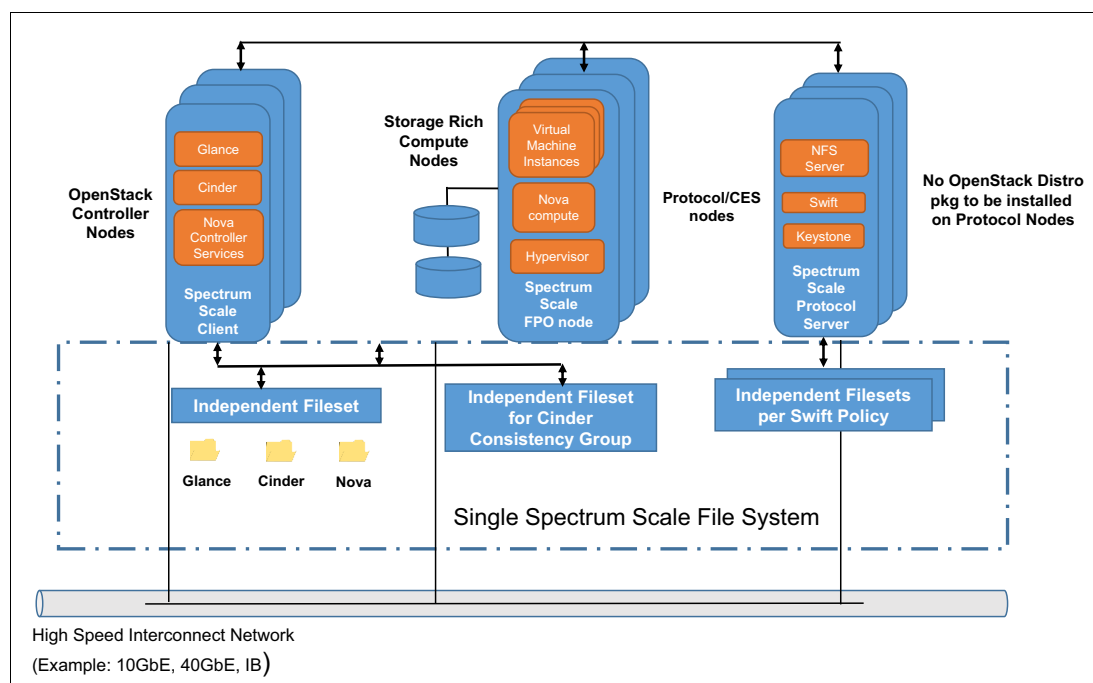


Figure 4 Storage rich servers with converged storage and compute nodes

In this model, the Spectrum Scale cluster is configured with storage rich servers by using the Spectrum Scale file placement optimizer feature. In this architecture, we have compute node and storage node functionality that is running on the same physical nodes (such as converged compute and storage). These nodes should have at least a GPFS FPO license. The controller services should be configured on other nodes that do not run storage or compute functionality and these nodes must have just Spectrum Scale.

Additionally, the protocol nodes serving object and Keystone are separate dedicated nodes with no other OpenStack service running on them. It is assumed these nodes do not contribute to the overall storage infrastructure and are used to serve objects only.

These nodes should have Spectrum Scale protocol server license. Because that compute and storage can run on the same set of nodes, this converged deployment model potentially offers a configuration option with fewer physical servers, although in such deployments the storage access might have a greater effect on the compute workloads that are running on a node (and vice versa) compared to the non-converged deployment models.

You might choose to have a hybrid model in which some of the compute nodes overlap with the FPO nodes that contribute to storage and some run independently as dedicated compute servers.

Model 4: Storage-rich servers in a hyper-converged environment

Model 4 is an OpenStack deployment in a hyper-converged environment, as shown in Figure 5.

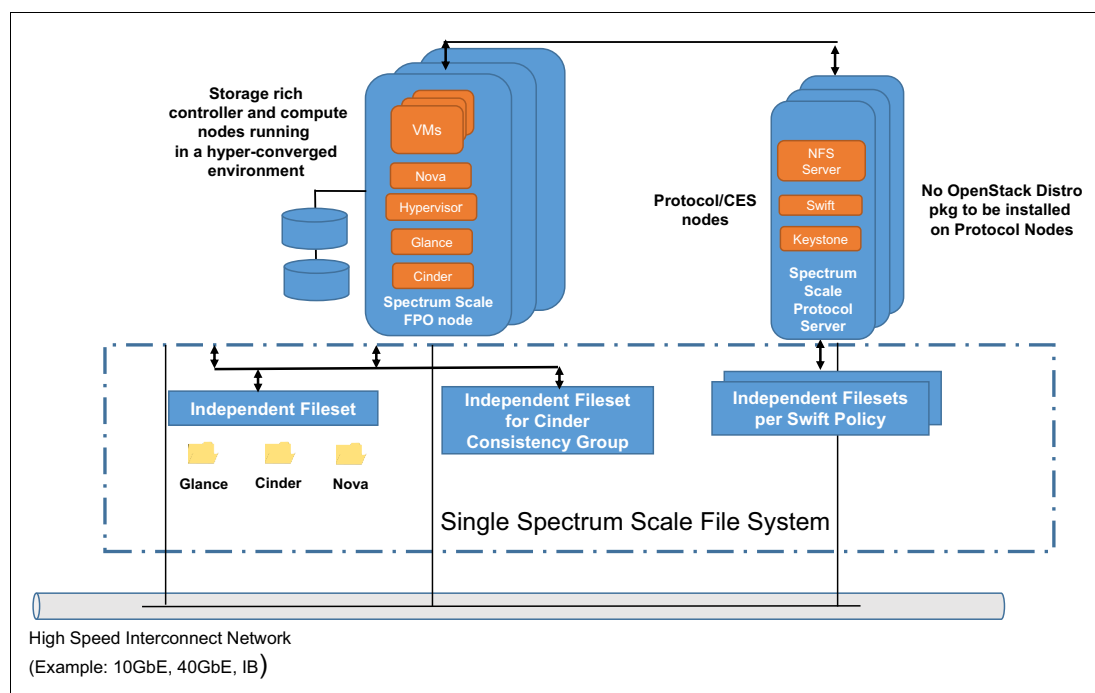


Figure 5 Storage-rich servers in a hyper-converged environment

In this deployment model, all of the OpenStack services are combined in a single set of servers that also contribute to storage. In this model, Spectrum Scale cluster is configured with the FPO feature. All of the servers should at least possess Spectrum Scale FPO license.

The OpenStack controller and Compute services run on the same set of nodes. This configuration might allow you to run with fewer servers. Additionally, the protocol nodes serving object and Keystone are separate dedicated nodes with no other OpenStack service running on them. It is assumed these nodes do not contribute to the overall storage infrastructure and are used to serve objects only. These nodes should have the Spectrum Scale server license.

For more information about planning, configuration, and networking different Spectrum Scale topologies, see *IBM Spectrum Scale (Formerly GPFS)*, SG24-8254, which is available at this website:

<http://www.redbooks.ibm.com/abstracts/sg248254.html>

Recommendations

Models 1 and 2 often are recommended because they scale Spectrum Scale storage nodes and OpenStack compute nodes separately, which is wanted in most scenarios. Models 3 and 4 provide a denser deployment and can be the correct deployment model for workloads that require the smallest server footprint or can benefit from the data locality features that FPO can provide.

All deployment models include the following recommendations:

- Nova compute nodes and Controller nodes should be part of the Spectrum Scale cluster.

Note: If controller and compute nodes cannot be a part of a GPFS cluster, Spectrum Scale storage can be provided via NFS. An NFS export is created on Spectrum Scale protocol servers and is mounted locally onto controller nodes. In this case, the IBM Spectrum Scale Cinder driver must be configured with NFS-based deployment. The support for NFS protocol is available in Spectrum Scale Cinder driver beginning with the OpenStack Liberty release.

- Do not install any of the OpenStack components on Spectrum Scale protocol nodes. Spectrum Scale protocol nodes should be dedicated to serve only protocols: Object (Swift), NFS, and SMB.
- When enabling Object protocol on Spectrum Scale protocol nodes, use the internal Keystone that is included with Spectrum Scale, which has built-in high availability for Keystone. However, if the deployment considerations demand an external Keystone server, that choice also is supported.
- The Glance image store and Cinder volume store should be on a single Spectrum Scale fileset to use the benefits of file cloning when volumes are created from images, and vice versa.
- The Spectrum Scale file system's metadata should be on an SSD if SSD storage is available.

Note: To configure OpenStack services, such as Nova, Cinder, Glance, Keystone (if external), in a highly available configuration, follow the recommendation that is provided by the OpenStack distribution that you use.

Configuring OpenStack with IBM Spectrum Scale

This section describes the installation and configuration steps for an OpenStack cloud environment with minimal required components. All of the backend storage is configured with Spectrum Scale. A high-level use case also is described that demonstrates the use of Spectrum Scale storage in OpenStack cloud environment. Finally, this section describes various OpenStack configuration steps.

System preparation

The goal of this section is to guide the administrator through the installation and configuration of the components that are needed to create an OpenStack cloud computing environment by using Spectrum Scale storage. The following assumptions are made:

- The reader is already familiar with OpenStack components (Nova, Glance, Cinder, Manila, and Swift). For more information, see the resources that are listed in “Related publications” on page 61.
- The reader is already familiar with Spectrum Scale. This Redpaper publication is not intended to serve as a comprehensive Spectrum Scale reference. For more information about Spectrum Scale, see “Related publications” on page 61.

The OpenStack Kilo version is used for the demonstration that is described next.

Before beginning

According to the deployment architecture, identify nodes that are to be used as Spectrum Scale NSD servers, protocol servers, OpenStack controllers, and OpenStack compute nodes. Complete the following steps:

1. Install IBM Spectrum Scale 4.2.0 on all the identified nodes. During installation, choose to enable object service on Spectrum Scale protocol nodes so that Keystone and Swift are configured. For more information, see the following reference materials:
 - *Planning for GPFS*, which is available at this website:
<https://ibm.biz/Bd4Luu>
 - *Installing IBM Spectrum Scale and Deploying Protocols on Linux Nodes*, which is available at this website:
<https://ibm.biz/Bd4Luu>
 - *Understanding and managing object services to help enable object services on CES/Protocol node*, which is available at this website:
<https://ibm.biz/Bd4LuC>
2. Identify Controller nodes and Compute nodes for setting up the OpenStack Cloud environment. For more information, see the following resources:
 - “Deployment models” on page 11
 - *Prepare your environment to configure OpenStack cloud environment*, which is available at this website:
<https://ibm.biz/Bd4LuT>
3. For the storage layout, we recommend the use of a single file system with multiple independent filesets for use by OpenStack infrastructure. We also recommend a block size of 1 MB for this file system regardless of the hypervisor that is used.

Some GPFS features in OpenStack rely on files that are shared by components that use the same fileset. For example, creating Cinder bootable volumes from Glance images by using copy on write. Other features (for example, Cinder consistency groups), create independent filesets for each consistency group. In this case, data must be copied between components rather than the use of GPFS copy on write.

You might also want to create other file systems that can be used by your workloads that are running in the OpenStack cloud. For these file systems, the block size should be chosen carefully based on the chosen storage system and RAID method, required streaming bandwidth, and expected number of small files (such as files of size less than 1 MB). It is important to choose a block size that optimizes for the storage hardware, but also strikes a balance for the expected range of workloads. For more information, see this website:

<https://ibm.biz/Bd4Luk>

Server layout

As an example, we created an OpenStack environment in a virtual setup that uses Spectrum Scale storage, as shown in Figure 6.

Note: In a production environment, we recommend that compute nodes be deployed on physical servers.

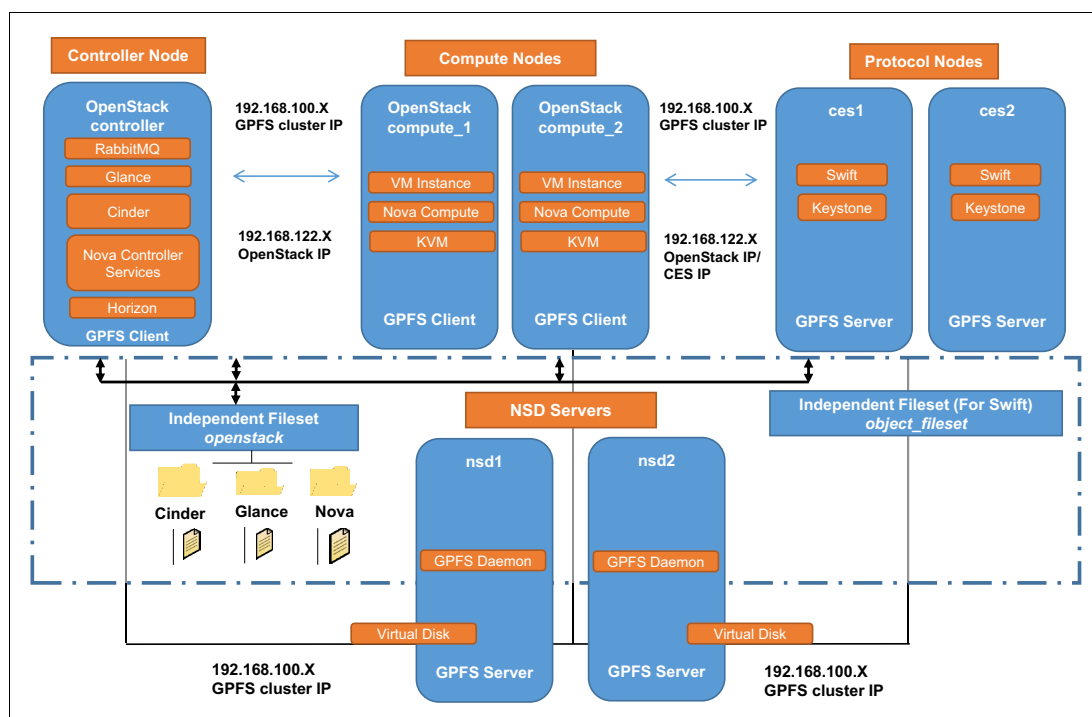


Figure 6 OpenStack environment in a virtual setup with Spectrum Scale as storage backend

The following nodes are featured in the setup:

- ▶ 2 NSD servers (GPFS Servers)
- ▶ 2 Protocol servers (GPFS Servers)
- ▶ 2 OpenStack compute nodes (GPFS Clients)
- ▶ 1 OpenStack controller node (GPFS Client)

NSD servers are used as a storage server in which the disks are directly attached. Here, nsd1 and nsd2 each have a single virtual disk.

The protocol servers, named ces1 and ces2, provide object and Keystone services. In this example, we use the Keystone service (and postgres database instance) that is provided by Spectrum Scale protocol servers.

The OpenStack controller node, named `openstack_controller`, runs RabbitMQ server, Glance, Cinder, and Nova management services.

We can also run OpenStack dashboard (Horizon) on the controller node.

The OpenStack compute nodes, named `openstack_compute_1` and `openstack_compute_2`, run the Nova compute service with the KVM hypervisor where the VM instances are booted.

RHEL 7.1 is the operating system level that we used in this example.

Each node has two network interfaces that are attached with IP addresses assigned in the subnets 192.168.122.XXX and 192.168.100.XXX, as listed in Table 2.

Table 2 Node network interface IP addresses

Node	Network interfaces 1 IP addresses	Network interfaces 2 IP addresses
nsd1	192.168.122.245	192.168.100.171
nsd2	192.168.122.177	192.168.100.218
ces1	192.168.122.24	192.168.100.228
ces2	192.168.122.205	192.168.100.132
openstack_controller	192.168.122.127	192.168.100.246
openstack_compute_1	192.168.122.197	192.168.100.162
openstack_compute_2	192.168.122.135	192.168.100.211

All of the nodes are a part of single GPFS cluster. We used IP 192.168.100.XXX subnet for configuring GPFS cluster.

It is expected the user created a suitable cluster before beginning deployment. The details of the cluster that we are using are shown Example 1.

Example 1 GPFS cluster information

```
# mmlscluster
```

```
GPFS cluster information
=====
```

```
GPFS cluster name:      spectrumscale.ibm.com
GPFS cluster id:        12845755401446978013
GPFS UID domain:        spectrumscale.ibm.com
Remote shell command:   /usr/bin/ssh
Remote file copy command: /usr/bin/scp
Repository type:        CCR
```

Node	Daemon node name	IP address	Admin node name	Designation
1	nsd1_gpfs	192.168.100.171	nsd1_gpfs	quorum-manager-perfmon
2	nsd2_gpfs	192.168.100.218	nsd2_gpfs	quorum-manager-perfmon
3	ces1_gpfs	192.168.100.228	ces1_gpfs	quorum-perfmon
4	ces2_gpfs	192.168.100.132	ces2_gpfs	quorum-perfmon
5	openstack_controller_gpfs	192.168.100.246	openstack_controller_gpfs	perfmon
6	openstack_compute_1_gpfs	192.168.100.162	openstack_compute_1_gpfs	perfmon
7	openstack_compute_2_gpfs	192.168.100.211	openstack_compute_2_gpfs	perfmon

A file system that is named gpfs0 is created and mounted at /ibm/gpfs0.

Spectrum Scale CES and OpenStack services are configured by using IP addresses in the 192.168.122.XXX subnet. An independent filesset named openstack is created to provide storage for glance/cinder/nova.

This fileset is different from the fileset that is used by Spectrum Scale object service. The object service is configured to use a separate independent fileset `object_fileset`, as shown in Example 2.

Example 2 Fileset configured to use a separate independent fileset object_fileset

```
# mmlsfileset gpfs0
  Filesets in file system 'gpfs0':
  Name                Status    Path
  root                Linked   /ibm/gpfs0
  object_fileset      Linked   /ibm/gpfs0/object_fileset
  openstack           Linked   /ibm/gpfs0/openstack
```

The following directory structure is created under the openstack file set link path for each component:

```
# tree
.
|-- cinder
|   |-- volumes
|-- glance
|   |-- images
|-- nova
|   |-- instances
```

These paths are used in respective configuration files for Glance, Cinder, or Nova to configure Spectrum Scale as a storage backend. Ownership and permissions must be correctly assigned to these paths for OpenStack services to work correctly.

OpenStack component configuration

This section describes various OpenStack components and configuration steps that must be followed to set it up in your environment. For more information about how various OpenStack components fit from architectural view, see “Deployment models” on page 11.

Note: In this demonstration example, we are configuring all of the services in a minimal configuration. Also, this example is not deploying Cinder, Glance, and Nova in a highly available configuration.

Keystone

The Keystone service provides identity management services to all other services in an OpenStack environment. The Spectrum Scale installation bundles the Keystone service and installs and configures the internal Keystone service when Object service is enabled on your protocol nodes.

IBM Spectrum Scale automated installer helps you to configure the internal Keystone service to use local database for identify management or to integrate with identity management databases, such as LDAP or Microsoft Active Directory (AD).

However, you can choose to use an external Keystone service. This service runs on a node that is not part of the Spectrum Scale Cluster.

We recommended that you configure and use the internal Keystone service that is bundled with Spectrum Scale installation unless a Keystone is deployed in your environment.

The example installation that is described in this section uses the internal Keystone that is provided by Spectrum Scale.

Note: The internal Keystone deployment creates an instance of the postgres database server. The example installation used this server instance for hosting the component databases of OpenStack service, such as Glance, Cinder, and Nova.

Also, this instance of postgres uses a service name postgresql-obj.service and port 5431. These values must be specified when the postgresql databases are accessed or managed.

For more information about Object and Keystone installation and configuration, see the following sections from the IBM Spectrum Scale Information Center:

► Understanding authentication for Object access:

- <https://ibm.biz/Bd4LLd>
- <https://ibm.biz/Bd4LLD>

► Keystone configuration with local authentication:

<https://ibm.biz/Bd4LLF>

► Keystone configuration with AD:

<https://ibm.biz/Bd4LLE>

► Keystone configuration with LDAP:

<https://ibm.biz/Bd4LLH>

► External Keystone configuration:

<https://ibm.biz/Bd4LL4>

RabbitMQ

OpenStack uses a message queue to coordinate operations and status information among services. The message queue service often runs on the controller node. We used RabbitMQ message queue service in this example.

To install and configure RabbitMQ, run following commands on the Controller node as shown in Example 3.

Example 3 Commands to install and configure RabbitMQ

```
# yum install rabbitmq-server

# systemctl enable rabbitmq-server.service
# systemctl start rabbitmq-server.service

# rabbitmqctl add_user openstack Passw0rd
Creating user "openstack" ...

# rabbitmqctl set_permissions openstack ".*" ".*" ".*"
Setting permissions for user "openstack" in vhost "/" ...
```

Glance

OpenStack Glance provides services for discovering, registering, and retrieving VM images. Glance allows querying of VM image metadata and retrieving the actual image.

Glance images can be stored in many different types of backend storage, including a shared file system, such as GPFS, or in object storage that is provided by Swift. We recommend configuring Glance to use a shared file system so that the copy on write functionality can be used for instantaneous start volume creation.

Complete the following steps to configure Glance:

1. Determine the node that the postgres database server is running. It is running on the node that is assigned the IP address with the `object_database_node` attribute.
2. Identify `object_database_node` by running the `mmces address list` command, as shown in Example 4.

Example 4 mmces address list command

```
# mmces address list
```

Address	Node	Group	Attribute
-----	-----	-----	-----
192.168.122.24	ces1_gpfs	none	none
192.168.122.205	ces2_gpfs	none	object_database_node,object_singleton_node

In the following examples, the postgres instance is running on the `ces2` node, which uses the IP address `192.168.122.205`.

Complete the following steps to create the Glance database:

1. Log in to object database node and switch the user to postgres and connect to the postgres CLI by using port 5431, as shown in the following example:

```
# su postgres

$ psql -p 5431
could not change directory to "/ibm/gpfs0/openstack" psql (9.2.7)
Type "help" for help.
```

```
postgres=#
```

2. Create the Glance database and Glance user, as shown in the following example:

```
postgres=# CREATE DATABASE glance;
CREATE DATABASE
```

```
postgres=# CREATE USER glance;
CREATE ROLE
```

3. Change the password on the Glance user and grant all privileges on the Glance database to the Glance user, as shown in the following example:

```
postgres=# ALTER USER glance WITH PASSWORD 'Passw0rd';
ALTER ROLE

postgres=# GRANT ALL PRIVILEGES ON DATABASE glance TO glance;
GRANT
```

4. (Optional) List the following databases and roles:

```
postgres=# \l
List of databases
```

Name	Owner	Encoding	Collate	Ctype	Access privileges
glance	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	=Tc/postgres + postgres=CTc/postgres+ glance=CTc/postgres
keystone	keystone	UTF8	en_US.UTF-8	en_US.UTF-8	
postgres	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	
template0	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	=c/postgres + postgres=CTc/postgres
template1	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	=c/postgres + postgres=CTc/postgres

(5 rows)

```
postgres=# \du
```

Role name	Attributes	Member of
glance		{}
keystone		{}
postgres	Superuser, Create role, Create DB, Replication	{}

The database files for the postgres instance is at `<gpfs mount point base>/ces/object/keystone/`.

5. Edit `<gpfs mount point base>/ces/object/keystone/pg_hba.conf` to add a host entry for Glance to allow remote nodes to access this database, as shown in the following example:

```
#START Object Protocol customizations
host    keystone          keystone          all          md5
host    glance            glance          all          md5
```

Note: The `<gpfs mount point base>` refers to the mount point of the GPFS that is used for storing CES data.

6. Restart PostgreSQL service by using the `postgresql-obj` service, as shown in the following example:

```
# systemctl restart postgresql-obj.service
```

7. Create the Glance service users and assign roles in Keystone.

This process can be done from any node in which the OpenStack client is installed. We use a protocol node in this example because a current `openrc` file is created during object protocol installation. To begin, source the `openrc` file to use **openstack cli** commands, as shown in the following example:

```
# cat openrc
# Tue Jan 5 15:48:10 EST 2016
export OS_AUTH_URL="http://127.0.0.1:35357/v3"
export OS_IDENTITY_API_VERSION=3
export OS_AUTH_VERSION=3
export OS_USERNAME="admin"
export OS_PASSWORD="Passw0rd"
```

```
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_NAME=admin
export OS_PROJECT_DOMAIN_NAME=Default
```

OS_AUTH_URL is pointed to localhost IP because we are running OpenStack administration commands from protocol nodes where the Keystone service is running, as shown in the following example:

```
# source openrc
```

```
# openstack user create --password Passw0rd glance
```

Field	Value
domain_id	default
enabled	True
id	6407cc204eea4f40b4b24be7fa81ccb7
name	glance

8. Give the admin role to user Glance in admin tenant, as shown in the following example:

```
# openstack role add --project admin --user glance admin
```

9. Give the admin role to user Glance in default domain, as shown in the following example:

```
# openstack role add --domain default --user glance admin
```

10. Create the Glance image service, as shown in the following example:

```
# openstack service create --name glance --description "OpenStack Image service" image
```

Field	Value
description	OpenStack Image service
enabled	True
id	356edf8c09734d9e9117a8dc7d48c359
name	glance
type	image

11. Create the image endpoints. Because Glance service is running on the controller node, we use openstack_controller node to create the image service endpoint, as shown in the following example:

```
# openstack endpoint create image admin http://openstack_controller:9292
```

Field	Value
enabled	True
id	a12111beba6c4032828882145e42191d
interface	admin
region	None
region_id	None
service_id	356edf8c09734d9e9117a8dc7d48c359
service_name	glance
service_type	image
url	http://openstack_controller:9292

```
# openstack endpoint create image internal http://openstack_controller:9292
```

Field	Value
enabled	True
id	61e677f3047149d2935f51aad578a7a8
interface	internal
region	None
region_id	None
service_id	356edf8c09734d9e9117a8dc7d48c359
service_name	glance
service_type	image
url	http://openstack_controller:9292

```
# openstack endpoint create image public http://openstack_controller:9292
```

Field	Value
enabled	True
id	eec696f9279143f9bac2d306138ec8f3
interface	public
region	None
region_id	None
service_id	356edf8c09734d9e9117a8dc7d48c359
service_name	glance
service_type	image
url	http://openstack_controller:9292

12. Install the required Glance packages on every controller node, as shown in the following example:

```
# yum install openstack-glance python-glance python-glanceclient
```

13. Edit `/etc/glance/glance-api.conf`. In the `[database]` section, configure database access, as shown in the following example:

```
[database]
...
connection = postgresql://glance:Passw0rd@192.168.122.205:5431/glance
```

Note: The IP address that is used to connect to postgres database server should refer to the only CES IP address that is designated as `object_database_node`.

14. Add the identity service access. The `auth_uri` and `auth_url` IP address can be any CES IP address, as shown in the following example:

```
[keystone_authtoken]
...
auth_uri = http://192.168.100.132:5000
auth_url = http://192.168.100.132:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = admin
```

```
username = glance
password = Passw0rd
```

15. Configure the file system store and location of image files on GPFS, as shown in the following example:

```
[glance_store]
default_store = file
filesystem_store_datadir = /ibm/gpfs0/openstack/glance/images/

[DEFAULT]
notification_driver = noop
```

16. Edit `/etc/glance/glance-registry.conf` similar to `glance-api.conf`, as shown in the following example:

```
[database]
...
connection = postgresql://glance:Passw0rd@192.168.122.205:5431/glance
```

Note: The database connection IP address should refer to the CES IP address that is designated as `object_database_node`.

```
[keystone_authtoken]
...
auth_uri = http://192.168.100.132:5000
auth_url = http://192.168.100.132:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = admin
username = glance
password = Passw0rd
```

```
[DEFAULT]
notification_driver = noop
```

17. After the Glance configuration file changes are made, distribute the updated configuration files to each controller node.

18. Populate the Glance database, as shown in the following example:

```
# su -s /bin/sh -c "glance-manage db_sync" glance
```

19. Start the `glance-api` and `glance-registry` services, as shown in the following example:

```
# systemctl enable openstack-glance-api.service openstack-glance-registry.service
# systemctl start openstack-glance-api.service openstack-glance-registry.service
```

20. Add the following entry in the `openrc` file that was created by the Spectrum Scale installation on every protocol node, as shown in the following example:

```
export OS_IMAGE_API_VERSION=2
```

21. Confirm that the following entry exists:

```
# cat openrc
# Tue Jan 5 15:48:10 EST 2016
export OS_AUTH_URL="http://127.0.0.1:35357/v3"
export OS_IDENTITY_API_VERSION=3
export OS_AUTH_VERSION=3
export OS_USERNAME="admin"
```



```
export OS_PASSWORD="Passw0rd"
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_NAME=admin
export OS_PROJECT_DOMAIN_NAME=Default
export OS_IMAGE_API_VERSION=2
```

Note: All of the OpenStack administration steps from protocol nodes are shown.

22. Verify the installation by completing the following steps to upload an image to Glance via the protocol node:

a. For this example, download a cirros image, as shown in the following example:

```
# wget -P /tmp/images \
http://download.cirros-cloud.net/0.3.4/cirros-0.3.4-x86_64-disk.img
```

```
# source openrc
```

b. Convert the image type to raw so that Spectrum Scale clone operations can be used when creating volumes from the following image:

```
qemu-img convert -O raw cirros-0.3.4-x86_64-disk.img \
cirros-0.3.4-x86_64-disk.img.raw
```

c. Upload the image to the Glance repository:

```
# glance image-create --name "cirros-0.3.4-x86_64" --file \
/tmp/images/cirros-0.3.4-x86_64-disk.img.raw --disk-format raw \
--container-format bare --progress
[=====>] 100%
```

Property	Value
checksum	ee1eca47dc88f4879d8a229cc70a07c6
container_format	bare
created_at	2016-01-06T00:25:05Z
disk_format	raw
id	b31ddf4e-8c72-4dc8-9c35-62aaf485991d
min_disk	0
min_ram	0
name	cirros-0.3.4-x86_64
owner	ce3813adb35d40ff9c3504f20f6f453d
protected	False
size	13287936
status	active
tags	[]
updated_at	2016-01-06T00:25:05Z
virtual_size	None
visibility	private

23. Verify that the images is stored at the GPFS location that is specified in `glance-api.conf`, as shown in the following example:

```
# ll /ibm/gpfs0/openstack/glance/images/
total 12984
-rw-r-----. 1 glance glance 13287936 Jan  5 19:25
b31ddf4e-8c72-4dc8-9c35-62aaf485991d
```

24. Review the Glance image list, as shown in the following example:

```
# glance image-list
+-----+-----+
| ID                               | Name                               |
+-----+-----+
| b31ddf4e-8c72-4dc8-9c35-62aaf485991d | cirros-0.3.4-x86_64 |
+-----+-----+
```

Cinder

OpenStack Cinder forms a Block Storage service for OpenStack. It is designed to virtualize pools of block storage devices and to provide users with a self-service API to request and use those resources without requiring any knowledge of where the storage is deployed or on what type of device. For more information about Cinder, see this website:

<https://wiki.openstack.org/wiki/Cinder>

The following features are supported by IBM Spectrum Scale GPFS driver that is provided in the OpenStack Kilo release:

- ▶ Volume Create/Delete
- ▶ Volume Attach/Detach
- ▶ Snapshot Create/Delete
- ▶ Create Volume from Snapshot
- ▶ Copy Image to Volume/ Copy Volume to Image
- ▶ Clone Volume
- ▶ Extend Volume
- ▶ Volume Stats (driver_version, vendor_name, volume_backend_name, storage_protocol, total_capacity_gb, free_capacity_gb, and reserved_percentage)

Table 3 lists the available configuration options for IBM Spectrum Scale Driver.

Table 3 Configuration options available with IBM Spectrum Scale driver

Configuration option = Default values	Description
gpfs_images_dir = None	(StrOpt) Specifies the path of the Image service repository in GPFS. Leave undefined if not storing images in GPFS.
gpfs_images_share_mode = None	<p>(StrOpt) Specifies the type of image copy to be used. Set this specification when the Image service repository also uses GPFS so that image files can be transferred efficiently from the Image service to the Block Storage service.</p> <p>There are two valid values: “copy” specifies that a full copy of the image is made; “copy_on_write” specifies that copy-on-write optimization strategy is used and unmodified blocks of the image file are shared efficiently.</p>
gpfs_max_clone_depth = 0	(IntOpt) Specifies an upper limit on the number of indirections that are required to reach a specific block because of snapshots or clones. A lengthy chain of copy-on-write snapshots or clones can have a negative affect on performance, but improves space utilization. A “0” indicates unlimited clone depth.
gpfs_mount_point_base = None	(StrOpt) Specifies the path of the GPFS directory where Block Storage volume and snapshot files are stored.

Configuration option = Default values	Description
gpfs_sparse_volumes = True	(BoolOpt) Specifies that volumes are created as sparse files that initially use no space. If set to False, the volume is created as a fully allocated file, in which case, creation can take longer.
gpfs_storage_pool = system	(StrOpt) Specifies the storage pool to which volumes are assigned. By default, the system storage pool is used.

Note: The `gpfs_images_share_mode` flag is only valid if the Glance Image Service is configured to store its image data on GPFS at the path that is specified by the `gpfs_images_dir` flag. When the value of this flag is `copy_on_write`, the paths that are specified by the `gpfs_mount_point_base` and `gpfs_images_dir` flags must be in the same GPFS and GPFS fileset.

For more information, see this website:

<http://docs.openstack.org/kilo/config-reference/content/GPFS-driver.html>

Cinder services configuration

Because Keystone that is shipped with Spectrum Scale is used, complete the following steps to create the necessary users and tables in postgres for Cinder:

1. From `object_database_node`, switch the user to `postgres` and connect to the `postgres` CLI, as shown in the following example:

```
# su postgres

$ psql -p 5431
psql (9.2.7)
Type "help" for help.
```

```
postgres=#
```

2. Create the Cinder database. Create a Cinder user, change the user's password, and grant all privileges on database Cinder to Cinder user, as shown in the following example:

```
postgres=# CREATE DATABASE cinder;
CREATE DATABASE

postgres=# CREATE USER cinder;
CREATE ROLE

postgres=# ALTER USER cinder WITH PASSWORD 'Passw0rd';
ALTER ROLE

postgres=# GRANT ALL PRIVILEGES ON DATABASE cinder TO cinder;
GRANT
```

- List the roles and databases, as shown in the following example:

```
postgres=# \du
```

List of roles		
Role name	Attributes	Member of
cinder		{}
glance		{}
keystone		{}
postgres	Superuser, Create role, Create DB, Replication	{}

```
postgres=# \l
```

List of databases					
Name	Owner	Encoding	Collate	Ctype	Access privileges
cinder	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	=Tc/postgres + postgres=CTc/postgres+
glance	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	=Tc/postgres + postgres=CTc/postgres+ glance=CTc/postgres
keystone	keystone	UTF8	en_US.UTF-8	en_US.UTF-8	
postgres	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	
template0	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	=c/postgres + postgres=CTc/postgres
template1	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	=c/postgres + postgres=CTc/postgres

(6 rows)

- Edit `/etc/keystone/keystone.conf` to add the Cinder entry, as shown in the following example:

```
#START Object Protocol customizations
host    keystone    keystone    all    md5
host    glance      glance      all    md5
host    cinder      cinder     all    md5
```

```
restart postgresql service
```

```
# systemctl restart postgresql-obj.service
```

- Create the Cinder service user and assign roles in Keystone. Source the `openrc` file to use **openstack cli** commands. This process can be done from any node in which the OpenStack client is installed (for example, any protocol node), as shown in the following example:

```
# source openrc
```

```
# openstack user create --password Passw0rd cinder
```

Field	Value
domain_id	default
enabled	True
id	4635da2952e3451b8bb2350b13f9f5ef
name	cinder

6. Give the admin role to user Cinder in admin tenant, as shown in the following example:

```
# openstack role add --project admin --user cinder admin
```

7. Give the admin role to user Cinder in default domain, as shown in the following example:

```
# openstack role add --domain default --user cinder admin
```

8. Create Cinder services, as shown in the following example:

```
# openstack service create --name cinderv2 --description "OpenStack Block Storage" volumev2
```

Field	Value
description	OpenStack Block Storage
enabled	True
id	547e3167df3b47cd877ec747f48915fa
name	cinderv2
type	volumev2

9. Create the volume endpoints. We use the controller node to create the endpoint because all of the Cinder services are running on the controller node, as shown in the following example:

```
# openstack endpoint create volumev2 admin
http://openstack_controller:8776/v2/%(tenant_id)s
```

Field	Value
enabled	True
id	a7aee81404ac4cfe850b309f3eb70053
interface	admin
region	None
region_id	None
service_id	547e3167df3b47cd877ec747f48915fa
service_name	cinderv2
service_type	volumev2
url	http://openstack_controller:8776/v2/%(tenant_id)s

```
# openstack endpoint create volumev2 internal
http://openstack_controller:8776/v2/%(tenant_id)s
```

Field	Value
enabled	True
id	0fd938dedaf8437a981ecd2b038d5f6f
interface	internal
region	None
region_id	None
service_id	547e3167df3b47cd877ec747f48915fa
service_name	cinderv2
service_type	volumev2
url	http://openstack_controller:8776/v2/%(tenant_id)s

```
# openstack endpoint create volumev2 public
http://openstack_controller:8776/v2/%(tenant_id)s
```

Field	Value
enabled	True
id	b7c4f225aef84c95abf97a61ad552bc2
interface	public
region	None
region_id	None
service_id	547e3167df3b47cd877ec747f48915fa
service_name	cinderv2
service_type	volumev2
url	http://openstack_controller:8776/v2/%(tenant_id)s

10. Install required Cinder packages on the controller node, as shown in the following example:

```
# yum install openstack-cinder python-cinderclient python-oslo-db
python-oslo-log
```

11. Configure Cinder management services (Cinder API and Cinder scheduler). Edit /etc/cinder/cinder.conf, as shown in the following example:

In the [database] section, configure database access:

```
[database]
...
connection = postgresql://cinder:Passw0rd@192.168.122.205:5431/cinder
```

Note: The database connection IP address should refer to the CES IP address that is designated as: object_database_node.

12. In the [DEFAULT] and [oslo_messaging_rabbit] sections, configure RabbitMQ message queue access, as shown in the following example:

```
[DEFAULT]
...
rpc_backend = rabbit

[oslo_messaging_rabbit]
rabbit_host = openstack_controller
rabbit_userid = openstack
rabbit_password = Passw0rd
```

13. In the [DEFAULT] and [keystone_authtoken] sections, configure Identity service access, as shown in the following example:

```
[DEFAULT]
...
auth_strategy = keystone

[keystone_authtoken]
auth_uri = http://192.168.100.132:5000
auth_url = http://192.168.100.132:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
```

```
project_name = admin
username = cinder
password = Passw0rd
```

14. In the [DEFAULT] section, configure the my_ip option to use the node name or management interface IP address of the controller node, as shown in the following example:

```
[DEFAULT]
...
my_ip = openstack_controller
```

15. In the [oslo_concurrency] section, configure the lock path, as shown in the following example:

```
[oslo_concurrency]
lock_path = /var/lock/cinder
```

16. Populate the Block Storage database, as shown in the following example:

```
# su -s /bin/sh -c "cinder-manage db sync" cinder
```

17. Start the cinder-api and cinder-scheduler services, as shown in the following example:

```
# systemctl enable openstack-cinder-api.service openstack-cinder-scheduler.service
# systemctl start openstack-cinder-api.service openstack-cinder-scheduler.service
```

18. Complete the following steps to configure the Cinder volume service by editing /etc/cinder/cinder.conf:

- Add a section named [gpfs].
- In that section, configure the storage backend by using the GPFS driver.
- Enter the appropriate locations for the volume and image directories. Setting the gpfs_images_share_mode to copy_on_write allows the GPFS driver to use clone functionality between Glance images and Cinder volumes, as shown in the following example:

```
[gpfs]
volume_driver = cinder.volume.drivers.ibm.gpfs.GPFSDriver
gpfs_mount_point_base = /ibm/gpfs0/openstack/cinder/volumes/
gpfs_images_dir = /ibm/gpfs0/openstack/glance/images/
gpfs_images_share_mode = copy_on_write
gpfs_max_clone_depth = 3
gpfs_sparse_volumes = True
gpfs_storage_pool = system
volume_backend_name = GPFS
```

19. Enable the GPFS backend for Cinder by adding the following entry in the [DEFAULT] section:

```
[DEFAULT]
...
enabled_backends = gpfs
```

20. In the [DEFAULT] section, configure the following location of the Image service:

```
[DEFAULT]
...
glance_host = openstack_controller
```

21. Create a /var/lock/cinder directory and grant the following permissions:

```
# mkdir /var/lock/cinder
# chmod 777 /var/lock/cinder
```

22.Enable and boot the Cinder-volume service, as shown in the following example:

```
# systemctl enable openstack-cinder-volume.service
# systemctl start openstack-cinder-volume.service
```

23.Add the following environment variables to openrc to enable the Cinder client:

```
export OS_TENANT_NAME=admin
export OS_VOLUME_API_VERSION=2
```

24.Create a sample volume, as shown in the following example:

```
# cinder create --name volume1 1
```

Property	Value
attachments	[]
availability_zone	nova
bootable	false
consistencygroup_id	None
created_at	2016-01-06T18:33:23.164847
description	None
encrypted	False
id	c813aac9-df3a-4321-91fd-3485f7281fe7
metadata	{}
multiattach	False
name	volume1
os-vol-host-attr:host	None
os-vol-mig-status-attr:migstat	None
os-vol-mig-status-attr:name_id	None
os-vol-tenant-attr:tenant_id	ce3813adb35d40ff9c3504f20f6f453d
os-volume-replication:driver_data	None
os-volume-replication:extended_status	None
replication_status	disabled
size	1
snapshot_id	None
source_volid	None
status	creating
user_id	b814abc37e084b1085f22e9227280dc8
volume_type	None

Note: The volume type is listed as None because only one backend is enabled. If multiple backends are enabled, ensure that the appropriate volume types are created and use them when volumes are created.

```
[root@openstack ~]# cinder list
```

ID	Status	Name	Size	Volume Type	Bootable	Attached to
c813aac9-df3a-4321-91fd-3485f7281fe7	available	volume1	1	None	false	

25. Verify that the volume file is created at the location that is specified in the [gpfs] section of cinder.conf, as shown in the following example:

```
# ll /ibm/gpfs0/openstack/cinder/volumes/
total 0
-rw-rw----. 1 root root 1073741824 Jan  6 13:33
volume-c813aac9-df3a-4321-91fd-3485f7281fe7
```

Because we are using Spectrum Scale, the volume that was created from the Controller node is automatically available from every compute node if GPFS client software is installed on all these nodes, as shown in Figure 7 on page 57.

Nova

The Nova service manages compute instances in an OpenStack environment. It is responsible for bootable instances, attaching block storage to those instances, and deleting instances. There are advantages when Nova is used in a Spectrum Scale environment today, such as enabling live migration of instances, and more planned for the future.

In this section, we first describe the Nova configuration steps. Then, we demonstrate the use of Nova with Spectrum Scale storage.

Because Keystone that is shipped with Spectrum Scale is used, complete the following steps to create the necessary users and table in postgres for Nova:

1. Log in to the object database node and Switch the user to postgres and connect to the postgres CLI, as shown in the following example:

```
# su postgres

$ psql -p 5431
psql (9.2.7)
Type "help" for help.
```

```
postgres=#
```

2. Create the Nova database and then create a Nova user, change its password, and grant all privileges on database Nova to Nova user, as shown in the following example:

```
postgres=# CREATE DATABASE nova;
CREATE DATABASE
```

```
postgres=# CREATE USER nova;
CREATE ROLE
```

```
postgres=# ALTER USER nova WITH PASSWORD 'Passw0rd';
ALTER ROLE
```

```
postgres=# GRANT ALL PRIVILEGES ON DATABASE nova TO nova;
GRANT
```

3. (Optional) List all of the roles and databases, as shown in the following example:

```
postgres=# \du
```

List of roles		
Role name	Attributes	Member of
cinder		{}
glance		{}
keystone		{}
nova		{}
postgres	Superuser, Create role, Create DB, Replication	{}

```
postgres=# \l
```

List of databases					
Name	Owner	Encoding	Collate	Ctype	Access privileges
cinder	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	=Tc/postgres + postgres=CTc/postgres+
glance	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	=Tc/postgres + postgres=CTc/postgres+ glance=CTc/postgres
keystone	keystone	UTF8	en_US.UTF-8	en_US.UTF-8	
nova	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	=Tc/postgres + postgres=CTc/postgres+ nova=CTc/postgres
postgres	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	
template0	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	=c/postgres + postgres=CTc/postgres
template1	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	=c/postgres + postgres=CTc/postgres

(7 rows)

4. Edit /<gpfs mount point>/ces/object/keystone/pg_hba.conf to add a Nova entry, as shown in the following example:

```
#START Object Protocol custimizations
host    keystone          keystone          all             md5
host    glance            glance            all             md5
host    cinder            cinder           all             md5
host    nova              nova             all             md5
```

5. Restart postgresql service, as shown in the following example:

```
# systemctl restart postgresql-obj.service
```

6. Source the openrc file to use OpenStack CLI commands, as shown in the following example:

```
# source ~/openrc
```

7. Create the Nova service user, as shown in the following example:

```
# openstack user create --password Passw0rd nova
```

Field	Value
domain_id	default
enabled	True
id	c79437dcbbda4eef95d1356bb8806abe
name	nova

8. Give the admin role to the Nova user in the admin tenant, as shown in the following example:

```
# openstack role add --project admin --user nova admin
```

9. Give the admin role to user Nova in the default domain, as shown in the following example:

```
# openstack role add --domain default --user nova admin
```

10. Create the Compute service, as shown in the following example:

```
# openstack service create --name nova --description "OpenStack Compute" compute
```

Field	Value
description	OpenStack Compute
enabled	True
id	5d31031fdcc44962a422aab9fd7eea43
name	nova
type	compute

11. Create endpoints, as shown in the following example:

Note: Compute endpoints are created with openstack_controller node IP because Nova management services are running on Controller nodes.

```
# openstack endpoint create compute admin http://openstack_controller:8774/v2/%(tenant_id)s
```

Field	Value
enabled	True
id	ceda05ed31354a3ca04a9be0a7a55bc4
interface	admin
region	None
region_id	None
service_id	5d31031fdcc44962a422aab9fd7eea43
service_name	nova
service_type	compute
url	http://openstack_controller:8774/v2/%(tenant_id)s

```
# openstack endpoint create compute internal http://openstack_controller:8774/v2/%(tenant_id)s
```

Field	Value
enabled	True
id	b9202ea9320b46d59e31c096fe3930b8
interface	internal
region	None
region_id	None
service_id	5d31031fdcc44962a422aab9fd7eea43
service_name	nova
service_type	compute
url	http://openstack_controller:8774/v2/%(tenant_id)s

```
# openstack endpoint create compute public http://openstack_controller:8774/v2/%(tenant_id)s
```

Field	Value
enabled	True
id	c7e3b35c31cb496ebac81878b4f7f3cd
interface	public
region	None
region_id	None
service_id	5d31031fdcc44962a422aab9fd7eea43
service_name	nova
service_type	compute
url	http://openstack_controller:8774/v2/%(tenant_id)s

12. Install and configure Nova services on the Controller node by installing the required Nova packages, as shown in the following example:

```
# yum install openstack-nova-api openstack-nova-cert openstack-nova-conductor \  
> openstack-nova-console openstack-nova-novncproxy openstack-nova-scheduler \  
> python-novaclient
```

13. Complete the following steps to edit `/etc/nova/nova.conf`:

- In the `[database]` section, configure database access, as shown in the following example:

```
[database]  
...  
connection = postgresql://nova:PasswOrd@192.168.122.205:5431/nova
```

Note: The database connection IP address should refer to the CES IP address that is designated as `object_database_node`.

- b. In the [DEFAULT] and [oslo_messaging_rabbit] sections, configure RabbitMQ message queue access, as shown in the following example:

```
[DEFAULT]
...
rpc_backend = rabbit

[oslo_messaging_rabbit]
rabbit_host = openstack_controller
rabbit_userid = openstack
rabbit_password = Passw0rd
```

- c. In the [DEFAULT] and [keystone_authtoken] sections, configure Identity service access, as shown in the following example:

```
[DEFAULT]
...
auth_strategy = keystone

[keystone_authtoken]
...
auth_uri = http://192.168.100.132:5000
auth_url = http://192.168.100.132:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = admin
username = nova
password = Passw0rd
```

- d. In the [DEFAULT] section, configure the instances path to be a Spectrum Scale path, as shown in the following example:

```
[DEFAULT]
...
Instances_path = /ibm/gpfs0/openstack/nova/instances/
```

- e. In the [DEFAULT] section, configure the my_ip option to use the host name or management interface IP address of the Controller node, as shown in the following example:

```
[DEFAULT]
my_ip = openstack_controller
```

- f. In the [DEFAULT] section, configure the VNC proxy to use the management interface IP address of the controller node, as shown in the following example:

```
vncserver_listen = openstack_controller
vncserver_proxyclient_address = openstack_controller
```

- g. In the [glance] section, configure the location of the Image service, as shown in the following example:

```
[glance]
...
host = openstack_controller
```

- h. In the [oslo_concurrency] section, configure the lock path, as shown in the following example:

```
[oslo_concurrency]
...
lock_path = /var/lib/nova/tmp
```

14. Populate the Compute database, as shown in the following example:

```
# su -s /bin/sh -c "nova-manage db sync" nova
```

15. Start the Compute services on the Controller node and configure them to start when the system boots, as shown in the following example:

```
# systemctl enable openstack-nova-api.service openstack-nova-cert.service \
openstack-nova-consoleauth.service openstack-nova-scheduler.service \
openstack-nova-conductor.service openstack-nova-novncproxy.service
# systemctl start openstack-nova-api.service openstack-nova-cert.service \
openstack-nova-consoleauth.service openstack-nova-scheduler.service \
openstack-nova-conductor.service openstack-nova-novncproxy.service
```

16. Install and configure Nova on the compute nodes:

```
# yum install openstack-nova-compute sysfsutils
```

17. Complete the following steps to edit the /etc/nova/nova.conf file:

- a. In the [DEFAULT] and [oslo_messaging_rabbit] sections, configure RabbitMQ message queue access, as shown in the following example:

```
[DEFAULT]
...
rpc_backend = rabbit

[oslo_messaging_rabbit]
rabbit_host = openstack_controller
rabbit_userid = openstack
rabbit_password = Passw0rd
```

- b. In the [DEFAULT] and [keystone_authtoken] sections, configure Identity service access, as shown in the following example:

```
[DEFAULT]
...
auth_strategy = keystone

[keystone_authtoken]
...
auth_uri = http://192.168.100.132:5000
auth_url = http://192.168.100.132:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = admin
username = nova
password = Passw0rd
```

- c. In the [DEFAULT] section, configure the instances path to be a Spectrum Scale path, as shown in the following example:

```
[DEFAULT]
...
Instances_path = /ibm/gpfs0/openstack/nova/instances/
```

- d. In the [DEFAULT] section, configure the my_ip option to the node name or management IP of the Compute node, as shown in the following example:

```
[DEFAULT]
my_ip = openstack_compute_1
```

- e. In the [DEFAULT] section, enable and configure remote console access, as shown in the following example:

```
vnc_enabled = True
vncserver_listen = 0.0.0.0
vncserver_proxyclient_address = openstack_compute_1
novncproxy_base_url = http://openstack_controller:6080/vnc_auto.html
```

- f. In the [glance] section, configure the location of the Image service, as shown in the following example:

```
[glance]
...
host = openstack_controller
```

- g. In the [oslo_concurrency] section, configure the lock path, as shown in the following example:

```
[oslo_concurrency]
...
lock_path = /var/lib/nova/tmp
```

18. Start the Compute service (including its dependencies) and configure them to boot automatically when the system starts, as shown in the following example:

```
# systemctl enable libvirtd.service openstack-nova-compute.service
# systemctl start libvirtd.service openstack-nova-compute.service
```

19. If libvirt service fails to start, ensure that the following packages are installed:

```
libvirt-python
libvirt-daemon-config-nwfilter
libvirt-daemon-driver-nwfilter
```

Similar steps are performed to configure Nova service on other Compute nodes (in this example, openstack_compute_2).

Networking

Note: The example that is used in this Redpaper publication features minimal configuration and should not be used as a reference because network requirements and configuration might be different for production OpenStack cloud environments.

OpenStack provides a choice of Neutron or existing Nova network for networking as a service. This section explains how to install and configure the existing Nova network component. The Nova network service enables you to deploy one network type per instance and is suitable for basic network functionality. OpenStack Neutron Networking enables you to deploy multiple network types per instance and includes plug-ins for various products that support virtual networking.

Complete the following steps to configure the Controller node to use Nova network:

1. Edit the /etc/nova/nova.conf file. In the [DEFAULT] section, configure the network and security group APIs, as shown in the following example:

```
[DEFAULT]
...
network_api_class = nova.network.api.API
security_group_api = nova
```

2. Restart the Compute services, as shown in the following example:

```
systemctl restart openstack-nova-api.service openstack-nova-scheduler.service \
openstack-nova-conductor.service
```

3. Complete the following steps to configure the Compute node:

- a. Install older networking components, as shown in the following example:

```
# yum install openstack-nova-network openstack-nova-api
```

- b. Edit the /etc/nova/nova.conf file. In the [DEFAULT] section, configure the network parameters, as shown in the following example:

```
[DEFAULT]
...
network_api_class = nova.network.api.API
security_group_api = nova
firewall_driver = nova.virt.libvirt.firewall.IptablesFirewallDriver
network_manager = nova.network.manager.FlatDHCPManager
network_size = 254
allow_same_net_traffic = False
multi_host = True
send_arp_for_ha = True
share_dhcp_address = True
force_dhcp_release = True
flat_network_bridge = br100
```

4. Start the services and configure them to start when the system boots, as shown in the following example:

```
# systemctl enable openstack-nova-network.service
openstack-nova-api-metadata.service
# systemctl start openstack-nova-network.service
openstack-nova-api-metadata.service
```

5. Complete the following steps to create the initial network:

- a. Source the openrc file.

- b. Create the network for our demonstration, as shown in the following example:

```
# source openrc
# nova network-create demo-net --bridge br100 --multi-host T \
--fixed-range-v4 10.0.100.0/24
```

```
# nova net-list
```

ID	Label	CIDR
e3904f71-4461-472a-a3a5-75dc2290bb13	demo-net	10.0.100.0/24

Starting a VM

The following methods are available to boot a Nova instance:

- ▶ From Glance into ephemeral storage
- ▶ From a Cinder bootable volume

Complete the following steps to boot from Glance:

1. Check the list of Glance images, as shown in Example 5.

Example 5 Check the list of Glance images

```
# glance image-list
```

ID	Name	Disk Format	Container
51f7c8de-b052-4ae3-8f0a-51d26e393eda	cirros-0.3.4-x86_64	qcow2	bare
13287936	active		
14fba53d-da3c-4739-a871-97bcdd0985a9	cirros-0.3.4-x86_64.raw	raw	bare
41126400	active		

2. Select an image from which to start (for example, cirros-03.4-x86_64) and run the **nova boot** command with the wanted value (resource specification), as shown in Example 6.

Example 6 Running the nova boot command

```
# nova boot --image cirros-0.3.4-x86_64 --flavor m1.tiny --nic net-id=e3904f71-4461-472a-a3a5-75dc2290bb13 vm1
```

Property	Value
OS-DCF:diskConfig	MANUAL
OS-EXT-AZ:availability_zone	nova
OS-EXT-SRV-ATTR:host	-
OS-EXT-SRV-ATTR:hypervisor_hostname	-
OS-EXT-SRV-ATTR:instance_name	instance-0000000f
OS-EXT-STS:power_state	0
OS-EXT-STS:task_state	scheduling
OS-EXT-STS:vm_state	building
OS-SRV-USG:launched_at	-
OS-SRV-USG:terminated_at	-
accessIPv4	
accessIPv6	
adminPass	L36kdcDjqDs6
config_drive	
created	2015-12-21T20:34:59Z
flavor	m1.tiny (1)
hostId	
id	9c7164ce-0b08-4171-8c4e-b8e61d72e286
image	cirros-0.3.4-x86_64 (51f7c8de-b052-4ae3-8f0a-51d26e393eda)
key_name	-
metadata	{}
name	vm1
os-extended-volumes:volumes_attached	[]

progress	0
security_groups	default
status	BUILD
tenant_id	24ca9212fcf7464cb0530eecd0ec9d6d
updated	2015-12-21T20:35:00Z
user_id	4741279eb50447019a1eea2a447689a9

This method creates a temporary or ephemeral copy of the boot image and then, starts the VM instance by using that boot image. When the instance is destroyed, all of the data for that instance is destroyed. Now, a full copy of the Glance image is made to create the ephemeral system disk. In the future, a Spectrum Scale file clone (copy on write) image can be used for faster and more efficient booting.

Complete the following steps to boot from Cinder:

1. Create the bootable volume as shown in Example 7.

Example 7 Creating bootable cinder volume followed by the boot

```
# cinder create --image-id cirros-0.3.4-x86_64.raw --name cirros-boot 40
```

Property	Value
attachments	[]
availability_zone	nova
bootable	false
consistencygroup_id	None
created_at	2015-12-21T20:44:27.395178
description	None
encrypted	False
id	596c75b7-dd21-4b9f-a207-d3f0c2a11bec
metadata	{}
multiattach	False
name	cirros-boot
os-vol-host-attr:host	None
os-vol-mig-status-attr:migstat	None
os-vol-mig-status-attr:name_id	None
os-vol-tenant-attr:tenant_id	24ca9212fcf7464cb0530eecd0ec9d6d
os-volume-replication:driver_data	None
os-volume-replication:extended_status	None
replication_status	disabled
size	40
snapshot_id	None
source_volid	None
status	creating
user_id	4741279eb50447019a1eea2a447689a9
volume_type	None

The volume listing shows that the volume is bootable, as shown in the “Bootable” column.

```
# cinder list
```

ID	Status	Name	Size	Volume Type	Bootable	Attached to
321f8120-0f61-4dcf-85b0-cb584b5dc73a	available	swap_volume	16	None	false	
596c75b7-dd21-4b9f-a207-d3f0c2a11bec	available	cirros-boot	40	None	true	
95f799ac-52df-42f0-8343-7d4935c90d03	available	db_volume	900	None	false	

2. Add the boot-volume flag, along with the volume ID on the **nova boot** command to specify that a bootable volume is used, as shown in Example 8.

Example 8 Specifying that a bootable volume is used

```
# nova boot --boot-volume 596c75b7-dd21-4b9f-a207-d3f0c2a11bec --flavor ml.tiny --nic \
net-id=e3904f71-4461-472a-a3a5-75dc2290bb13 vm2
```

Property	Value
OS-DCF:diskConfig	MANUAL
OS-EXT-AZ:availability_zone	nova
OS-EXT-SRV-ATTR:host	-
OS-EXT-SRV-ATTR:hypervisor_hostname	-
OS-EXT-SRV-ATTR:instance_name	instance-00000010
OS-EXT-STS:power_state	0
OS-EXT-STS:task_state	scheduling
OS-EXT-STS:vm_state	building
OS-SRV-USG:launched_at	-
OS-SRV-USG:terminated_at	-
accessIPv4	
accessIPv6	
adminPass	w5hPoY7feR8P
config_drive	
created	2015-12-21T20:54:51Z
flavor	ml.tiny (1)
hostId	
id	3337c736-54fc-48f0-9f15-912955ab436d
image	Attempt to boot from volume - no image supplied
key_name	-
metadata	{}
name	vm2
os-extended-volumes:volumes_attached	[{"id": "596c75b7-dd21-4b9f-a207-d3f0c2a11bec"}]
progress	0
security_groups	default
status	BUILD
tenant_id	24ca9212fcf7464cb0530eecd0ec9d6d
updated	2015-12-21T20:54:51Z
user_id	4741279eb50447019a1eea2a447689a9

In this case, the system image is persistent; therefore, any changes that were made to the image (including modifying the software in the image) are preserved. The instance can be destroyed and then rebooted and the system data is saved.

The other benefit to the use of bootable Cinder volumes is that the boot volume can be created quickly and efficiently when the Glance storage and Cinder volume storage are in the same Spectrum Scale fileset. In this case, a file clone (copy on write) image is created as the Cinder volume.

3. After you boot a Nova instance, you can create a Glance image from the system data by using the **nova image-create** command, as shown in Example 9.

Example 9 Nova image-create command used to create Glance image

```
# nova image-create --poll vml new_cirros_image

Server snapshotting... 100% complete
Finished
# glance image-list
```

ID	Name	Disk Format	Container
51f7c8de-b052-4ae3-8f0a-51d26e393eda	cirros-0.3.4-x86_64	qcow2	bare
13287936	active		
14fba53d-da3c-4739-a871-97bcdd0985a9	cirros-0.3.4-x86_64.raw	raw	bare
41126400	active		
e8fd812f-3c49-4b4c-8467-df30216927ac	new_cirros_image	qcow2	bare
22020096	active		

Swift

Swift fulfils the object storage needs in OpenStack deployments. It is commonly used as a repository of static data, such as VM images, archives, static web pages, photo storage, and backups. Based on OpenStack Swift, Spectrum Scale Object Storage provides the following unique benefits:

- ▶ Uniform file and object access in which objects can be accessed via file interface (by using POSIX, NFS, or SMB protocol) and vice versa
- ▶ In-place analytics of object data via Hadoop features that are available with Spectrum Scale
- ▶ Selective compression of object data
- ▶ Secure data at rest via file system encryption
- ▶ Integrated ILM features
- ▶ Support for AD/LDAP based authentication
- ▶ All of the features that are provided by OpenStack Swift software

Spectrum Scale Object Storage supports Swift and S3 API. When Spectrum Scale is used for OpenStack deployment separate Swift object store nodes are not required. One of the most unique values that Spectrum Scale provides is a single namespace for object, block, and file to be shared in OpenStack deployments and managed under a single pane. This value is unlike other storage island offerings in which all the three types of storage are on different systems and managed differently.

Spectrum Scale Object Storage is a part of Spectrum Scale server license and is hosted on dedicated Spectrum Scale protocol nodes.

Spectrum Scale Object Storage in OpenStack deployments can be part of the following key use cases:

► Image repository

One of the storage types that Glance supports at the back-end is Swift. As shown in Example 10, Glance can be configured to store images on Spectrum Scale Swift; however, we recommend that Glance is configured with a file system backend that uses the GPFS client to store and access images. This configuration provides the benefit of the GPFS file cloning functionality. For more information, see this website:

<https://ibm.biz/Bd498h>

► Back up for volumes

Spectrum Scale Object Storage can be configured so that some or all data it holds is stored in low-end disks, which is possible with the use of Spectrum Scale storage pools. Spectrum Scale allows you to group storage devices based on their performance, cost, locality, and reliability characteristics in storage pools.

The rich Spectrum Scale ILM placement policies also enable the automatic placement of new files or objects in a specific storage pool. Because of this feature, backup data can be easily and economically stored on Spectrum Scale Object Storage.

OpenStack cinder-backup service supports backing up volumes to Swift-based object storage. The configuration information and sample output from backing up or restoring Cinder volumes from Spectrum Scale object are described next.

Complete the following steps:

- a. As shown in Example 10, configure the [DEFAULT] section in `/etc/cinder/cinder.conf` to add the Swift backup driver configuration.

Example 10 Adding Swift backup driver configuration

```
[DEFAULT]
...
backup_driver = cinder.backup.drivers.swift
backup_swift_url = http://192.168.122.205:8080/v1/AUTH_
backup_swift_auth = per_user
backup_swift_auth_version = 2
backup_swift_user = admin
backup_swift_key = admin
backup_swift_tenant = admin
backup_swift_container = volumebackups
backup_swift_object_size = 52428800
backup_swift_retry_attempts = 3
backup_swift_retry_backoff = 2
backup_compression_algorithm = zlib
```

- b. Start the backup service on the Controller node, as shown in Example 11.

Example 11 Starting the backup service on the Controller node

```
# systemctl enable openstack-cinder-backup
# ln -s '/usr/lib/systemd/system/openstack-cinder-backup.service' \
'/etc/systemd/system/multi-user.target.wants/openstack-cinder-backup.service'
# systemctl start openstack-cinder-backup
```

Cinder Backup service is shown in the **cinder service-list** command (optional), as shown in Example 12.

Example 12 Showing the cinder service-list to see the Cinder Backup service status and state

```
# cinder service-list
```

Binary	Host	Zone	Status	State	Updated_at	Disabled Reason
cinder-backup	openstack	nova	enabled	up	2016-01-08T23:24:25.650350	None
cinder-scheduler	openstack	nova	enabled	up	2016-01-08T23:24:24.335346	None
cinder-volume	openstack@gpfs	nova	enabled	up	2016-01-08T23:24:25.002687	None

c. Create a backup of the Cinder volume, as shown in Example 13.

Example 13 Creating a backup cinder volume

```
# cinder list
```

ID	Status	Name	Size	Volume Type	Bootable	Attached to
c813aac9-df3a-4321-91fd-3485f7281fe7	available	volume1	1	None	false	


```
# cinder backup-create c813aac9-df3a-4321-91fd-3485f7281fe7
```

Property	Value
id	4fcc0901-a473-4df4-9c59-6d89f9332a1d
name	None
volume_id	c813aac9-df3a-4321-91fd-3485f7281fe7


```
# cinder backup-list
```

ID	Volume ID	Status	Name	Size
4fcc0901-a473-4df4-9c59-6d89f9332a1d	c813aac9-df3a-4321-91fd-3485f7281fe7	available	None	1

Object Count | Container |

22	volumebackups
----	---------------

d. Delete the volume from cinder, as shown in the following example:

```
# cinder delete c813aac9-df3a-4321-91fd-3485f7281fe7
```

e. Restore a volume by using a backup that was created by using the **backup-restore** command, as shown in the following example:

```
# cinder backup-restore 4fcc0901-a473-4df4-9c59-6d89f9332a1d
```

- f. Check the volume list (see Example 14) to verify that the volume is restored.

Example 14 Verifying that the volume is restored

```
# cinder list
```

Type	Bootable	Attached to	ID	Status	Name	Size	Volume
None	false		b42dc63a-1101-4bf6-9278-8c147cee22cd	available	restore_backup_4fcc0901-a473-4df4-9c59-6d89f9332a1d	1	

- Serving as Object Store with file access for applications that are running on VM instances

Applications that are running on the VM instances that are hosted over OpenStack deployment might need object storage, such as a web server-based workload that must store static web pages, photos, and so on, on an object store. Some applications that are running on the VM instances might need object interface and file interface access to the same data. These applications or VM instances might need to share the data with each other. Spectrum Scale Object Storage with Unified File and Object access feature (UFO) enables the application that is running on the VM instances to cater to such requirements.

To see an example of this use case, see the Amalgamating Manila And Swift for Unified Data Sharing Across Instances presentation at the 2016 spring OpenStack Summit, which is available at this website:

<https://ibm.biz/Bd49Lr>

High-level use of an OpenStack environment

Now that we have the base configuration setup, a basic use case is described in this section. The use of the OpenStack platform often begins with identifying a workload that can be migrated from physical to virtual resources. For example, this workload might be a workload that is running on hardware resources that are expensive to maintain, or a workload that is growing rapidly and is difficult to scale with physical resources.

After identifying the workload, the compute, network, and storage requirements of the workload must be understood and plan accordingly. From a storage perspective, consider the following questions:

- Is data shared between VMs?
- What percentage of the data must be persisted and what percentage is scratch?
- What are the performance requirements of the workload?
- How active is the data?
- Is data written once and rarely read, or is data frequently updated?

For example, consider a case in which you have an existing workload that running on aging servers. Each server in the configuration works independently on its own data set and publishes the results of processing for consumption by other teams in your organization. This workload also requires a large amount of scratch space that can be discarded at the conclusion of the day.

Also, you want to make a record of the state of each machine at the end of each day. There are performance spikes on one server or another and you want to better optimize operations across the pool of resources. How does this process work in an OpenStack environment?

In this example, a prerequisite is that you configured your OpenStack environment with the core components, Nova, Glance, Cinder, Swift, and Keystone. The following process for this workload when Spectrum Scale storage is used is possible:

1. Create a bootable Cinder volume from Glance.
2. Create bootable volumes for each instance by cloning new volumes from first bootable volume.
3. Create a scratch volume for each instance that is preformatted for ext4, and use an appropriate (*scratch*) storage pool.
4. Boot all of the instances by using the bootable Cinder volumes.
5. Mount one scratch volume on each VM instance.
6. Run your workload.
7. When there is a performance spike on one VM, use Nova live migration to move one or more VMs off the busy hypervisor to another hypervisor to free compute resources
8. At the end of each work period, create a Cinder volume snapshot of the bootable image for each VM.
9. Optionally, you can use Cinder backup to store a snapshot of Cinder volume data in Swift object store.

Demonstrating the use case

The following sections show how these operations can be supported in a Spectrum Scale OpenStack cloud. We also describe many more features and benefits of this environment.

For this use case example, we assume that the GPFS storage pool named “scratch” was created. We add two GPFS backends in `cinder.conf` and specify the configuration parameter `gpfs_storage_pool` correctly. The first backend is using the system storage pool and the second is using the scratch storage pool, as shown in Example 15.

Example 15 Enabling and creating volume types for each of the backends

```
[gpfs]
volume_driver = cinder.volume.drivers.ibm.gpfs.GPFSDriver
gpfs_mount_point_base = /ibm/gpfs0/openstack/cinder/volumes/
gpfs_images_dir = /ibm/gpfs0/openstack/glance/images/
gpfs_images_share_mode = copy_on_write
gpfs_max_clone_depth = 3
gpfs_sparse_volumes = True
gpfs_storage_pool = system
volume_backend_name = GPFS

[gpfs_scratch]
volume_driver = cinder.volume.drivers.ibm.gpfs.GPFSDriver
gpfs_mount_point_base = /ibm/gpfs0/openstack/cinder/volumes/
gpfs_images_dir = /ibm/gpfs0/openstack/glance/images/
gpfs_images_share_mode = copy_on_write
gpfs_max_clone_depth = 3
gpfs_sparse_volumes = True
gpfs_storage_pool = scratch
volume_backend_name = GPFS_SCRATCH
```

We enable the backends, as shown in the following example:

```
[DEFAULT]
...
enabled_backends = gpfs, gpfs_scratch
```

We create volume types for each of the backend and set the extra-specs correctly, as shown in the following example:

```
# cinder type-create gpfs
```

ID	Name
8ad6f6db-7797-4bb6-adaa-9ebc5c800ffd	gpfs

```
# cinder type-create gpfs_scratch
```

ID	Name
2f4d277f-76be-4b17-be85-6f2fa3146892	gpfs_scratch

```
# cinder type-key gpfs set volume_backend_name=GPFS
```

```
# cinder type-key gpfs_scratch set volume_backend_name=GPFS_SCRATCH
```

```
# cinder extra-specs-list
```

ID	Name	extra_specs
2f4d277f-76be-4b17-be85-6f2fa3146892	gpfs_scratch	{u'volume_backend_name': u'GPFS_SCRATCH'}
8ad6f6db-7797-4bb6-adaa-9ebc5c800ffd	gpfs	{u'volume_backend_name': u'GPFS'}

Complete the following steps:

1. Create a bootable volume of type gpfs from a Glance image, as shown in Example 16.

Example 16 Creating bootable volume from a Glance image

```
# glance image-list
```

ID	Name
b31ddf4e-8c72-4dc8-9c35-62aaf485991d	cirros-0.3.4-x86_64
cba5779a-627c-47f3-8e91-782974547588	cirros-raw-0.3.4-x86_64

```
# cinder create --image-id cba5779a-627c-47f3-8e91-782974547588 --name base_volume -volume-type gpfs 1
```

```
# cinder list
```

ID	Status	Name	Size	Volume Type	Bootable	Attached to
4ffbab40-880b-40db-97eb-64798f757464	available	base_volume	1	gpfs	true	

Note in the following example that the volume file that is created is a clone child of the image file that is on GPFS:

```
# mmclone show volume-4ffbab40-880b-40db-97eb-64798f757464
Parent Depth  Parent inode  File name
-----
no      1          296972  volume-4ffbab40-880b-40db-97eb-64798f757464
```

2. Create bootable volumes for each instance by cloning new volumes from first bootable volume, as shown in Example 17.

Example 17 Creating bootable volumes for each instance by cloning from first bootable volume

```
# cinder create --source-uuid 4ffbab40-880b-40db-97eb-64798f757464 --name volume1 -volume-type gpfs 1
# cinder create --source-uuid 4ffbab40-880b-40db-97eb-64798f757464 --name volume2 -volume-type gpfs 1
```

```
# cinder list
```

ID	status	Name	Size	Volume Type	Bootable	Attached to
09391b0f-5057-411f-88b5-014166e79103	available	volume2	1	gpfs	true	
4ffbab40-880b-40db-97eb-64798f757464	available	base_volume	1	gpfs	true	
cdd43500-b9e4-4984-904d-7c89b38820e2	available	volume1	1	gpfs	true	

3. Create a scratch volume for each instance that is preformatted for ext4, and use an appropriate (scratch) storage pool (as shown in Example 18) that uses volume type as scratch.

Example 18 Creating a scratch volume for each instance

```
#cinder create --name scratch1 -volume-type gpfs_scratch 1 --metadata fstype=ext4
#cinder create --name scratch2 -volume-type gpfs_scratch 1 --metadata fstype=ext4
```

```
# cinder list
```

ID	status	Name	Size	Volume Type	Bootable	Attached to
09391b0f-5057-411f-88b5-014166e79103	available	volume2	1	gpfs	true	
3fff519c-514a-4d8a-8a51-20bdab08e192	available	scratch1	1	gpfs_scratch	false	
4ffbab40-880b-40db-97eb-64798f757464	available	base_volume	1	gpfs	true	
a2d897a9-6bb4-4c61-a39b-d66c5391c4f3	available	scratch2	1	gpfs_scratch	false	
cdd43500-b9e4-4984-904d-7c89b38820e2	available	volume1	1	gpfs	true	

4. Boot Nova instances, as shown in Example 19.

Example 19 Booting Nova instances

```
# nova boot --boot-volume cdd43500-b9e4-4984-904d-7c89b38820e2 --flavor m1.tiny --nic net-id= e3904f71-4461-472a-a3a5-75dc2290bb13 vm1

# nova boot --boot-volume 09391b0f-5057-411f-88b5-014166e79103 --flavor m1.tiny --nic net-id= e3904f71-4461-472a-a3a5-75dc2290bb13 vm2

# nova list
```

ID	Name	Status	Task State	Power State	Networks
9c7164ce-0b08-4171-8c4e-b8e61d72e286	vm1	ACTIVE	-	Running	demo-net=10.0.100.16
3337c736-54fc-48f0-9f15-912955ab436d	vm2	ACTIVE	-	Running	demo-net=10.0.100.17

5. Attach scratch volumes to the instances, as shown in Example 20.

Example 20 Attaching scratch volumes

```
# nova volume-attach 9c7164ce-0b08-4171-8c4e-b8e61d72e286 3fff519c-514a-4d8a-8a51-20bdab08e192 auto

# nova volume-attach 3337c736-54fc-48f0-9f15-912955ab436d a2d897a9-6bb4-4c61-a39b-d66c5391c4f3 auto
```

The workload is now ready to run on the OpenStack cloud.

VM migration

When Nova is backed by a shared file system, such as Spectrum Scale, instance live migration is possible.

There are some other required configuration steps to enable live migration. For more information, see this website:

<http://docs.openstack.org/admin-guide-cloud/compute-configuring-migrations.html>

Because Spectrum Scale storage is used, you must ensure only that the `instances_path` parameter in `nova.conf` is in a GPFS path and is accessible from all Compute nodes. From one of your Compute nodes, run the commands that are shown in Example 21.

Example 21 Verifying `instances_path` parameter is in a GPFS path and is accessible

```
# grep ^instances_path /etc/nova/nova.conf
instances_path = /ibm/gpfs0/openstack/nova/instances/
```

Note: The following parameters also must be set in `/etc/libvirt/libvirtd.conf` for live-migration to be successful:

```
Listen_tcp = 1
Auth_tcp = "none"
```

Complete the following steps to run VM live migration:

1. View the list of hypervisors and the VM instances that are running on each, as shown in Example 22.

Example 22 View the list of hypervisors and VM instances

```
# nova hypervisor-list
```

ID	Hypervisor	hostname	State	Status
1	openstack_compute_1	up	enabled	
2	openstack_compute_2	up	enabled	

Both instances are now are running on the first hypervisor, openstack_compute_1.

ID	Name	Hypervisor ID	Hypervisor Hostname
6b734361-7aa2-49f3-a7e6-f0a779c1dad8	instance-00000001	1	openstack_compute_1
6e4d5a68-49d9-4212-a64a-9a5d7290265d	instance-00000002	1	openstack_compute_1

2. Migrate one of the instances to another hypervisor by using the nova live-migration command. In the list that is shown in Example 22 on page 56, instance-00000002 corresponds to vm2, which is running on hypervisor openstack_compute_1. We can confirm this correspondence by using the commands that are shown in Example 23.

Example 23 instance-00000002 corresponds to vm2, running on hypervisor openstack_compute_1

```
# nova show vm2 | grep hypervisor
| OS-EXT-SRV-ATTR:hypervisor_hostname| openstack_compute_1
# nova show vm2 | grep instance_name
| OS-EXT-SRV-ATTR:instance_name| instance-00000002
```

To migrate vm2 from hypervisor openstack_compute_1 to openstack_compute_2, run the following command:

```
# nova live-migration 6e4d5a68-49d9-4212-a64a-9a5d7290265d openstack_compute_2
```

In Example 24, you can see that vm2 (instance-00000002) is now running on openstack_compute_2.

Example 24 vm2 (instance-00000002) running on openstack_compute_2

```
# nova hypervisor-servers openstack_compute_2
```

ID	Name	Hypervisor ID	Hypervisor Hostname
6e4d5a68-49d9-4212-a64a-9a5d7290265d	instance-00000002	2	openstack_comput_2

```
# nova show vm2 | grep hypervisor
| OS-EXT-SRV-ATTR:hypervisor_hostname | openstack_compute_2
```

If you monitor activity on vm2 during the migration, there is a slight pause when the actual migration occurred; otherwise, operations continue uninterrupted.

- At the end of each work period, create a Cinder volume snapshot of the bootable image for each VM, as shown in Example 25.

Example 25 *Creating a Cinder volume snapshot of the bootable image for each VM*

```
# cinder snapshot-create --name volume1-snapshot cdd43500-b9e4-4984-904d-7c89b38820e2
```

```
# cinder snapshot-create --name volume2-snapshot 09391b0f-5057-411f-88b5-014166e79103
```

```
# cinder snapshot-list
```

ID	Volume ID	Status	Name	Size
89070f5d-0f37-42a8-8b7f-85d9e6f2cb07	cdd43500-b9e4-4984-904d-7c89b38820e2	available	volume1-snapshot	1
df212a8d-81d9-458f-adfd-dfc44a7b9650	09391b0f-5057-411f-88b5-014166e79103	available	volume2-snapshot	1

- Alternatively, you can use the Cinder backup commands that were shared to back up the volumes to Swift object store or some other backup backend and restore those volumes whenever needed.

Sharing storage across virtualized cluster (VM)

This section describes options for sharing storage across a virtualized cluster (VM).

After the VM instances are instantiated, the VMs must have shared file storage across them in which data in the form of files or objects that are created by one VM is available for other VMs. The following approaches are recommended:

- Shared file storage that uses GPFS client

In this approach (see Figure 7), one or more VMs are part of a Spectrum Scale cluster, each installing the GPFS client and mounted a GPFS. This configuration allows applications that are running in that set of VMs to access the same namespace and share file data with each other by using POSIX semantics. This solution is the highest-performing solution for shared data access and is recommended for data centers in which performance is the primary requirement. In addition, this option might be the only feasible option for applications that require POSIX access (for example, MPI).

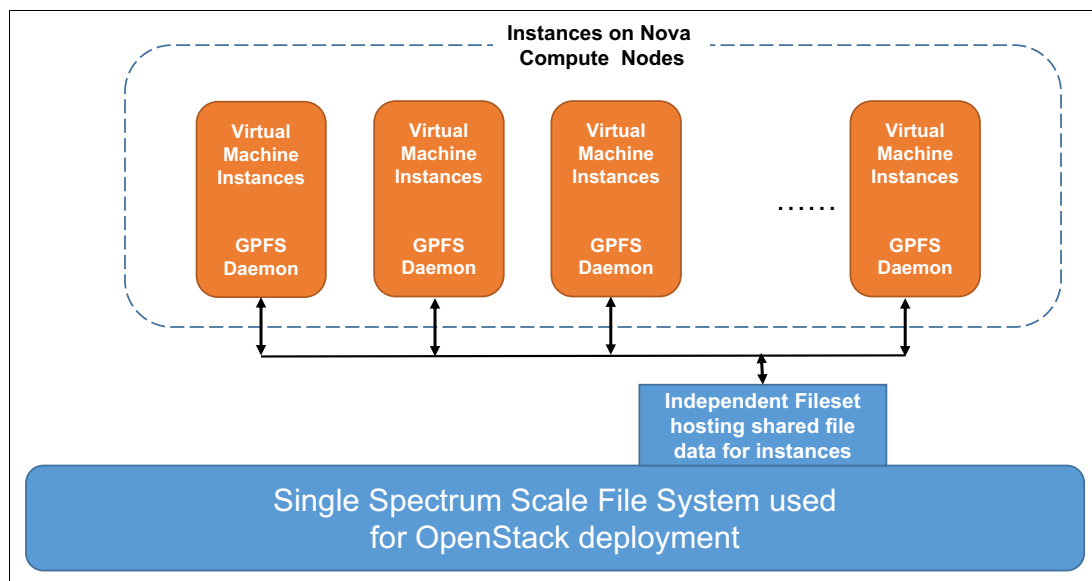


Figure 7 Shared file storage by using GPFS client

Similar to the tradeoffs between the use of the GPFS client directly on bare-metal servers versus a distributed access protocol (such as NFS), care must be taken to ensure smooth operation. For example, GPFSs should be cleanly unmounted and the GPFS system shutdown gracefully, if possible. These precautions prevent GPFS from performing recovery and failover steps that can temporarily suspend access to user data. In the ideal situation, VMs that are part of a Spectrum Scale cluster are long-running VMs that are not quickly provisioned and destroyed.

Assuming several groups of VMs exist and each accesses their own data sets in a single GPFS, it is advisable to have each group of applications access data in their own Spectrum Scale fileset. This configuration allows administrators to manage the data for each group of applications independently, and helps to prevent false resource sharing between the application groups (for example, false sharing of a directory token).

Because many settings in Spectrum Scale apply to an entire cluster, it might also be advantageous to create a Spectrum Scale cluster for each set of VMs (or one cluster for all the VMs) that is separate from the Spectrum Scale cluster that contains the NSD servers and might be hosting the VM disk images. Although this suggestion does entail the use of Spectrum Scale multi-cluster for accessing data, it can help ease administration by allowing different settings for the virtual cluster versus the physical clusters.

In addition, it is important that GPFS manager and quorum daemons are not run in any VM that is low in available memory or is not designed to always be running. Therefore, by using multiple clusters, this configuration helps ensure that these daemons always are running in the physical cluster, which should have sufficient memory and always be running. Another way to control the roles of each node in the GPFS cluster is by using the **mmchnode** command.

A separate, dedicated Spectrum Scale cluster that is used by the VM instances for file sharing also is possible.

- Use of NFS/SMB/Object exported by Spectrum Scale Protocol nodes

In this approach (as shown in Figure 8), data is exported via the Spectrum Scale protocol nodes by using Object (S3 or Swift), NFS, or SMB and accessed by applications that are running in VMs for data sharing. This option does not deliver the raw performance of the shared file storage option (running GPFS clients in VMs), but is much easier to set up and manage because most operating system distributions are bundled with Object, NFS, and SMB clients. In addition, these file-sharing protocols are more lightweight than the GPFS client, which allows VMs to come and go without any interruption to system operation.

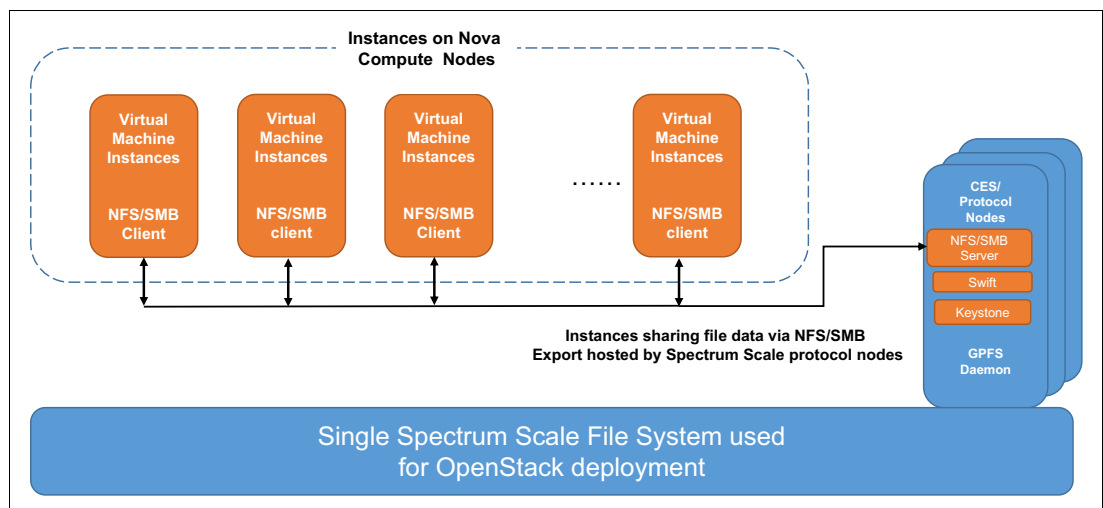


Figure 8 Use of NFS/SMB/Object exported by Spectrum Scale Protocol Nodes

Manila is an OpenStack component that aids in file sharing across the VM instances. As of Spectrum Scale 4.2, Spectrum Scale protocol nodes do not support Manila; however, Manila can be configured to work in a non-CES environment.

Multitier storage architecture for OpenStack deployment

The following key challenges can occur while an effective quality of service (QoS) for multitier storage systems is implemented:

- ▶ Easy and flexible provisioning of storage with different of IOPS performance requirement (hard disk drive and SSD tiered storage pools).
- ▶ Intelligent ILM
- ▶ Seamless HSM processes
- ▶ End-to-end automated and single-window provisioning

These challenges are successfully addressed by IBM Spectrum Scale and the OpenStack Cinder service. This service aids in easy provisioning of volumes from multiple storage tiers to effectively meet the storage requirements of workloads that are running in a set of instances, as shown in Figure 9.

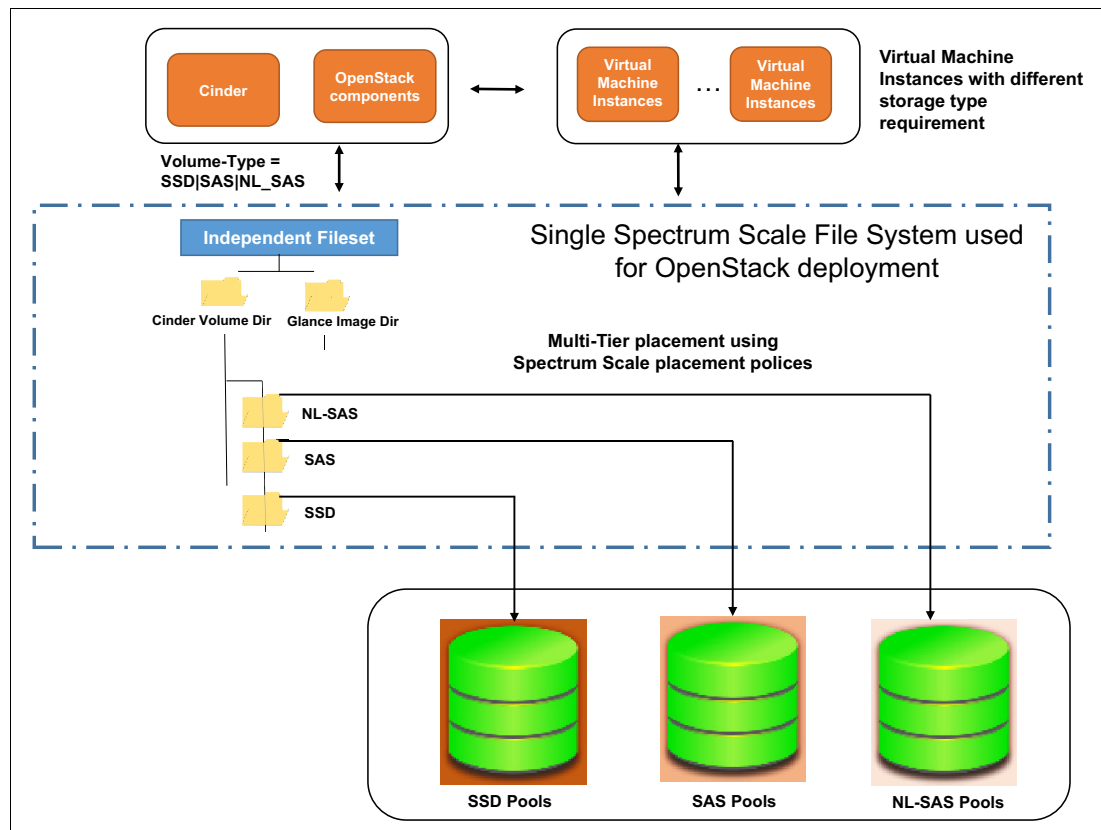


Figure 9 Architecture showing provisioning of volumes from multiple storage tiers

The architecture that is shown in Figure 9 on page 59 includes the following features:

- Creates pools from disk of similar characteristics.
- For clarity, creates directories for hosting Cinder volumes with appropriate nomenclature. This requirement is not a hard requirement, only a recommendation.
- Configures new volume types that correspond to each storage pool and adds storage pool details in the volume backend section in `cinder.conf`.

With this architecture, different classes of storage can be provisioned to different instances that are based on the workload needs. QoS also can be deployed in which user workloads often are categorized as Gold, Silver, or Bronze. With this setup, the storage or cloud admin can deploy storage that is based on the QoS that is associated with the instance owners.

Managing information cycle of volumes

In a multitier storage architecture, there is a common need to tier the VM volumes to appropriate storage tiers for optimal use of the available storage. Spectrum Scale provides migration policies that can migrate infrequently used volumes to lower tiers. This process requires defining Spectrum Scale migration policies and works seamlessly without any OpenStack component configuration or integration.

Figure 10 shows that VM1 (which was provisioned over a volume that is on SSD pool) is migrated to SAS pool because of its less frequent use.

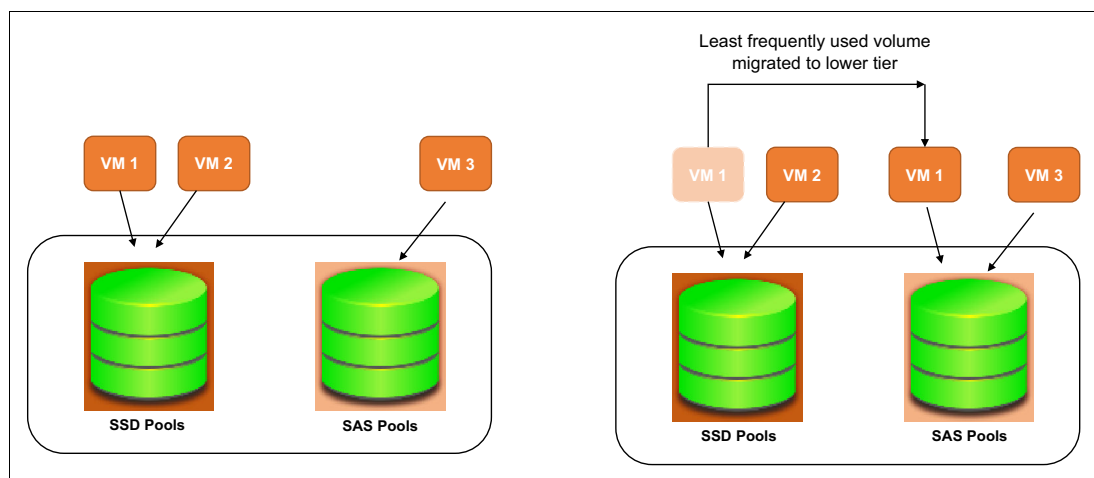


Figure 10 Managing information cycle of volumes

Note: Spectrum Scale also supports HSM, which allows the volumes to be tiered to tapes or cloud.

Conclusion

IBM Spectrum Scale is a proven, enterprise-class file system, and OpenStack is the leading open source Infrastructure-as-a-Service cloud platform. When combined, these technologies provide a unique cloud platform that accelerates time to value by alleviating many of the storage headaches IT administrators, architects, and managers frequently encounter.

By integrating the OpenStack storage services all within a single storage plane, administrators avoid silos of storage and their corresponding data management headaches, which allows enterprises to realize the full value of their data.

In this Redpaper, we provided examples of how you can configure OpenStack to use the high-performance, feature-rich shared storage environment that is provided by Spectrum Scale. Hypothetical use cases and configurations also were reviewed. We also described only a small segment of the potential use cases that can be served by OpenStack and Spectrum Scale working together. We believe that this architecture is the premier architecture for implementing enterprise class OpenStack deployments.

Related publications

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

IBM Redbooks

The following IBM Redbooks publications provide more information about the topic in this document. Some publications that are referenced in this list might be available in softcopy only:

- ▶ *A Deployment Guide for IBM Spectrum Scale Object*, REDP-5113
- ▶ *IBM Private, Public, and Hybrid Cloud Storage Solutions*, REDP-4873
- ▶ *IBM Spectrum Scale (formerly GPFS)*, SG24-8254
- ▶ *Introduction Guide to the IBM Elastic Storage Server*, REDP-5253

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

ibm.com/redbooks

Other publications

The following publications are also relevant as further information sources:

- ▶ *IBM Spectrum Scale V4.2: Concepts, Planning, and Installation Guide*, GA76-0441
- ▶ *IBM Spectrum Scale V4.2: Administration and Programming Reference*, SA23-1452
- ▶ *IBM Spectrum Scale V4.2: Advanced Administration Guide*, SC23-7032
- ▶ *IBM Spectrum Scale V4.2: Data Management API Guide*, GA76-0442
- ▶ *IBM Spectrum Scale V4.2: Problem Determination Guide*, GA76-0443

Online resources and references

For more information, see the following resources:

- ▶ Amalgamating Manila And Swift for Unified Data Sharing Across Instances:
<https://ibm.biz/Bd49Lr>
- ▶ Finding needles in the unified storage haystack:
<https://www.youtube.com/watch?v=Z2SBRsxAevI>
- ▶ IBM Elastic Storage Server:
<http://www.ibm.com/systems/storage/spectrum/ess>

- ▶ IBM Spectrum Scale:
<http://www.ibm.com/systems/storage/spectrum/scale/index.html>
- ▶ IBM Spectrum Scale resources:
<http://www.ibm.com/systems/storage/spectrum/scale/resources.html>
- ▶ IBM Spectrum Scale in the IBM Knowledge Center:
http://www.ibm.com/support/knowledgecenter/SSFKN/gpfs_welcome.html
- ▶ IBM Spectrum Scale Overview and Frequently Asked Questions:
<http://ibm.co/1IK06PN>
- ▶ IBM Spectrum Scale Wiki:
<https://ibm.biz/BdXVxv>
- ▶ Install and Configure Block Storage Service (Cinder):
<https://ibm.biz/Bd49Li>
- ▶ OpenStack Wiki:
<https://wiki.openstack.org/wiki>
- ▶ Write a File, Read as an Object:
<https://www.openstack.org/videos/video/write-a-file-read-as-an-object>

Authors

This Redpaper publication was produced by a team of specialists from around the world working with the International Technical Support Organization, Tucson Center.

Bill Owen is a Senior Engineer with the IBM Spectrum Scale development team. He is responsible for the integration of OpenStack with Spectrum Scale, focusing on the Swift object, Cinder block, and Manila file storage components of OpenStack. He has worked in various development roles within IBM for over 15 years. Before joining IBM, Bill developed and deployed grid management systems for electric utilities. Bill holds B.Sc. and M.S. degrees in Electrical Engineering from New Mexico State University.

Dean Hildebrand is a Master Inventor and the Manager of the Cloud Storage Software Research group at the IBM Almaden Research Center and a recognized expert in the field of distributed and parallel file systems. He has authored numerous scientific publications, created over 24 patents, and chaired and sat on the program committee of numerous conferences. Dr. Hildebrand pioneered pNFS, demonstrating the feasibility of providing standard and scalable access to any parallel file system. He received a B.Sc. degree in Computer Science from the University of British Columbia in 1998 and M.S. and PhD. degrees in Computer Science from the University of Michigan in 2003 and 2007.

Sandeep Ramesh is a Senior Technical Staff Member and working as a Storage Architect with IBM System Labs. He has over 15 years of extensive product architecture and design experience. Sandeep is an IBM Master Inventor, an IBM developerWorks® Master Author, and a member of the IBM Academy of Technology. Sandeep holds a Bachelor of Engineering (Computer Science) degree from the University of Pune, India.

Gautam Shah is a Senior Technical Staff Member with the Spectrum Scale development team. He recently led the delivery of the protocol functionality that is integrated with Spectrum Scale and now leads the team helping with delivery of Spectrum Scale in cloud environments. He has been with IBM for over 20 years with experience in various roles in the high-performance computing, including communication protocols and clustered file system development and deployment of large-scale clusters. He also worked on assignment at the IBM Systems Lab in Pune, India, on the NAS offering based on General Parallel File System. He is a member of the IBM Academy of Technology. He received a B.Sc. in Physics from Loyola College, Chennai and a B.E in Computer Science and Automation from Indian Institute of Science, Bangalore, and a Ph.D. in Information and Computer Science from Georgia Institute of Technology.

Gaurang Tapase is a software developer with IBM Spectrum Scale team. He mainly works on integration of OpenStack storage components (Cinder, Manila, and Swift) with Spectrum Scale. Gaurang joined IBM in 2010 and has worked on development projects that are centered around object stores. Gaurang holds a Bachelor of Technology (Computer Engineering) degree from University of Pune, India.

Kumar Nachiketa is a Lead Consultant with IBM Systems Lab Services, ASEAN, and is based in Jakarta, Indonesia. He previously worked in Oracle as Principal System Administrator and IBM STG Lab services, India as Storage Solution Architect. He holds a Master's degree from BIT Mesra, Ranchi, India. With 10 years of IT experience, Kumar has worked primarily on storage domain and has experience working on multiple vendor storage for solution, implementation, and administration. Recently, he was focusing on Software Defined Storage, Cloud, and OpenStack Integration.

Kedar Karmarkar is a senior engineer and solution architect with IBM Spectrum Scale development team. Kedar is part of Spectrum Scale Client adoption team and was the Storwize V7000 Unified Level 3 support lead in his prior role at IBM. Kedar has over 20 years of infrastructure software, storage development experience in management, and architect roles. He has led development of NAS Storage, Block level virtualization, replication, systems, and storage management products. Kedar has Bachelor of Engineering (Computer Science) degree from University of Pune, India.

Larry Coyne is a Project Leader at the International Technical Support Organization (ITSO), Tucson, Arizona, center. He has 34 years of IBM experience, with 23 years in IBM storage software management. He holds degrees in Software Engineering from the University of Texas at El Paso, and Project Management from George Washington University. His areas of expertise include client relationship management, quality assurance, development management, and support management for IBM Tivoli® Storage Software.

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

Sasikanth Eda
Pallavi Galgali
Simon Lorenz
Ashutosh Mate
Aaron Palazzolo
Harald Seipp
IBM Systems

Now you can become a published author, too!

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

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

ibm.com/redbooks/residencies.html

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>

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®
developerWorks®
GPFS™
IBM®
IBM Elastic Storage™

IBM Spectrum™
IBM Spectrum Protect™
IBM Spectrum Scale™
IBM z™
Redbooks®

Redpaper™
Redpapers™
Redbooks (logo) ®
Tivoli®

The following terms are trademarks of other companies:

Intel, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

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

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

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



REDP-5331-00

ISBN 0738455369

Printed in U.S.A.

Get connected

