OpenStack Deployment and Operations Guide

NetApp, Inc.

February 2017 | Version 5.0

Abstract

This guide is intended to detail design considerations, configuration, deployment and ongoing operational practices for the successful implementation of OpenStack services atop or in concert with NetApp technologies. Please note that this version of the guide applies to the Kilo (Spring 2015) release of OpenStack. The Juno release and prior are documented in version 4.0 available at http://netapp.github.io/openstack-deploy-ops-guide/.

Table of Contents

List of Figures

List of Tables

Chapter 1. Executive Summary

Most options for OpenStack integrated storage solutions aspire to offer scalability, but often lack the features and performance needed for efficient and cost-effective cloud deployment at scale. NetApp® platforms integrated with OpenStack avail a unique combination of advanced storage efficiency, integrated data protection, and non-disruptive operations with the ability to scale while preserving performance.

With NetApp, organizations can lower risk and enable a broad spectrum of cloud SLAs by combining the power and ingenuity of OpenStack cloud management with proven data integrity and fully-developed storage provisioning, data protection, and efficiency.

By leveraging the power of the NetApp clustered Data ONTAP® operating system, enterprise organizations and service providers can build a cloud storage platform with an agile data infrastructure that delivers high-performing, efficient, and scalable open-source cloud services. NetApp provides storage platforms that enable quick deployment, nimble reaction to change with the ability to scale.

With capabilities such as self-healing and integrated data protection for backup and disaster recovery, NetApp solutions are enterprise proven and help reduce risk for OpenStack cloud deployments. A global network of service provider partners already have deployed hundreds of high-SLA cloud services built on NetApp with over a billion users worldwide.

Because NetApp technology is integrated with OpenStack Block Storage Service, OpenStack Object Storage Service, OpenStack Image Service and OpenStack Compute Service, users can build on this proven and highly scalable storage platform to optimize their private and public cloud architectures by reducing risk and increasing efficiency.

This document intends to describe the application of NetApp capabilities to enable production OpenStack deployments. In particular, it's meant to address the needs of system architects, system administrators, and storage administrators who are investigating the use or deployment of OpenStack.

1.1. Authors and Contributors

Bob Callaway, Jon Benedict, Sajid Akhtar, Navneet Singh, Rushi Agrawal, Ben Swartzlander, Stan Skelton, Tim Snider, Robert Esker, Jon Olby, Dustin Schoenbrun

Chapter 2. OpenStack Overview

The OpenStack community is a global collaboration of developers and cloud computing technologists producing the ubiquitous open source cloud computing platform for public and private clouds. The project aims to deliver feature-rich solutions for all types of clouds by being simple to implement yet massively scalable. The technology consists of a series of related projects delivering various components for a cloud infrastructure solution.

OpenStack implements services for establishing infrastructure-as-a-service released under the Apache 2.0 open source license. The project is managed by the OpenStack Foundation, a nonprofit corporate entity established in September 2012 that promotes, protects, and empowers OpenStack software and its community.

This technology consists of a series of modular projects that control large pools of processing, storage, and networking resources throughout a data center, all managed through a single dashboard that gives administrators control while empowering users to provision resources in a self-service fashion.

OpenStack is committed to an open design and development process. The community operates around a six-month, time-based release cycle with frequent development milestones. For additional information, refer to http://www.openstack.org.

OpenStack is available to be deployed on all major Linux® platforms. For additional information, refer to http://wiki.openstack.org/GetOpenStack.

2.1. OpenStack Foundation

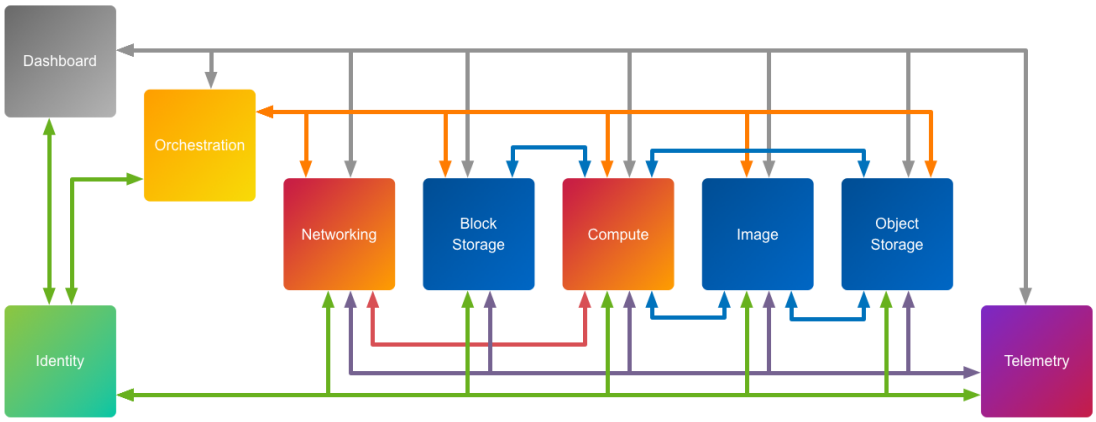
The OpenStack Foundation promotes the development, distribution and adoption of the OpenStack cloud operating system. The goal of the OpenStack Foundation is to serve developers, users, and the entire ecosystem by providing a set of shared resources to grow the footprint of public and private OpenStack clouds, enable technology vendors targeting the platform and assist developers in producing the best cloud software in the industry.

NetApp joined as a charter member of the OpenStack Foundation in August of 2012. The OpenStack Foundation is an independent body providing shared resources to help achieve its mission by protecting, empowering, and promoting OpenStack software and the community around it. As a Gold member of the Foundation, NetApp is pleased to be taking a leadership role in the community.

2.2. OpenStack - A Modular Collection of Cloud Services

The various OpenStack community projects and the services they implement are modular in nature and generally capable of being used independently. They are intended to knit together into a net capability greater than the sum of the individual parts.

Figure 2.1. OpenStack High Level Architecture



OpenStack Compute (project name: Nova)

OpenStack enables enterprises and service providers to offer on-demand computing resources, by provisioning and managing large networks of virtual machines. Compute resources are accessible via APIs for developers building cloud applications and through web interfaces for administrators and users. The compute architecture is designed to scale horizontally on standard hardware. OpenStack Compute is architected to avoid inherent proprietary hardware or software requirements and the ability to integrate with existing systems and third-party technologies. It is designed to manage and automate pools of compute resources and can work with widely available virtualization technologies, as well as bare metal and high-performance computing configurations.

OpenStack Block Storage (project name: Cinder)

OpenStack Block Storage provides a “block storage as a service” capability. It provides persistent block devices mapped to OpenStack compute instances (which are otherwise assumed to be ephemeral). The block storage system manages the creation, attaching and detaching of the block devices to instances. It also optionally supports instance booting and provides mechanisms for creating Snapshot copies and cloning. While fully integrated with OpenStack Compute and Dashboard, it can also be used independent of OpenStack to provide a standardized abstraction for block storage provisioning.

OpenStack Object Storage (project name: Swift)

OpenStack Object Storage provides a fully distributed, scale-out, API-accessible storage platform that can be integrated directly into applications or used for backup, archiving and data retention. Object storage does not present a traditional file system, but rather a distributed storage system for static data such as virtual machine images, photo storage, email storage, backups and archives. The OpenStack Object Storage API (aka Swift API), in a manner somewhat similar to CDMI, proposes an open standard for cloud storage. It can also function as an alternative endpoint for Amazon Web Services S3 and as a CDMI server through the use of add-on components.

OpenStack Dashboard (project name: Horizon)

The OpenStack Dashboard provides administrators and users a graphical interface to access, provision and automate cloud-based resources. The extensible design makes it easy to plug in and expose third-party products and services, such as billing, monitoring, and additional management tools. The dashboard can also be made brand specific for service providers and other Enterprises who require customization. The dashboard is one of several ways to interact with OpenStack resources. Developers can automate access or build tools to manage their resources that use the native OpenStack API or the EC2 compatibility API. The dashboard provides users a self-service portal to provision their own resources within the limits set by administrators.

OpenStack Identity (project name: Keystone)

OpenStack Identity provides a central directory of users mapped to the OpenStack services they can access. It acts as a common authentication system across the cloud operating system and can integrate with existing backend directory services (for example, LDAP). It supports multiple forms of authentication including standard user name and password credentials, token-based systems and AWS-style logins. Additionally, the catalog provides a list of all of the services deployed in an OpenStack cloud in that can be queried in a single registry. Users and third-party tools can programmatically determine which resources they can access. OpenStack Identity enables:

- Configuration of centralized policies across users and systems
- Creation of users and tenants and define permissions for compute, storage and networking resources through the use of role-based access control (RBAC) features
- Integration with existing directories, allowing for a single source of identity authentication
- As a user, get a list of the services that you can access and make API requests or log into the web dashboard to create resources owned by your account

OpenStack Image Service (project name: Glance)

The OpenStack Image Service provides discovery, registration and delivery services for disk and server images. The ability to copy or snapshot a server image and immediately store it away is a powerful capability of the OpenStack cloud operating system. Stored images can be used as a template to get new servers up and running quickly and more consistently if you are provisioning multiple servers than installing a server operating system and individually configuring additional services. It can also be used to store and catalog an unlimited number of backups. The Image Service can store disk and server images in a variety of back-ends, including through NFS and Object Storage. The Image Service API provides a standard REST interface for querying information about disk images and lets clients stream the images to new servers. A multiformat image registry allowing uploads of private and public images in a variety of formats.

OpenStack Network Service (project name: Neutron)

OpenStack Networking is a pluggable, scalable and API-driven system for managing networks and IP addresses. Like other aspects of the cloud operating system, it can be used by administrators and users to increase the value of existing data center assets. OpenStack Networking ensures the network is not the bottleneck or limiting factor in a cloud deployment and provides users self-service over their own network configurations. The pluggable backend architecture lets users take advantage of basic commodity gear or advanced networking services from supported vendors. Administrators can take advantage of software-defined networking (SDN) technology like OpenFlow to allow high levels of multi-tenancy and massive scale. OpenStack Networking has an extension framework allowing additional network services, such as intrusion detection systems (IDS), load balancing, firewalls and virtual private networks (VPN) to be deployed and managed.

OpenStack Telemetry (project name: Ceilometer)

OpenStack Telemetry provides common infrastructure to collect usage and performance measurements within an OpenStack cloud. Its primary initial targets are monitoring and metering, but the framework is expandable to collect data for other needs. Ceilometer was promoted from incubation status to an integrated component of OpenStack in the Grizzly (April 2013) release.

OpenStack Orchestration (project name: Heat)

OpenStack Orchestration implements a service to orchestrate multiple composite cloud applications that use the Amazon Web Services (AWS) CloudFormation template format, through both an OpenStack-native and CloudFormation-compatible API. It is intended, in part, to facilitate movement of workloads from AWS to OpenStack deployments. Heat was promoted from incubation status to an integrated component of OpenStack in the Grizzly (April 2013) release.

OpenStack Database as a Service (project name: Trove)

OpenStack Database as a Service allows users to quickly and easily utilize the features of a relational database without the burden of handling complex administrative tasks. Cloud users and database administrators can provision and manage multiple database instances as needed. Initially, the service focuses on providing resource isolation at high performance while automating complex administrative tasks including deployment, configuration, patching, backups, restores, and monitoring. Trove was promoted from incubation status to an integrated component of OpenStack in the Icehouse (April 2014) release.

OpenStack Hadoop as a Service (project name: Sahara)

The OpenStack Hadoop as a Service project aims to provide users with simple means to provision a Hadoop cluster by specifying several parameters like Hadoop

version, cluster topology, nodes' hardware details, etc. Sahara was promoted from incubation status to an integrated component of OpenStack in the Icehouse (April 2014) release.

OpenStack File Share Service (project name: Manila)

OpenStack File Share Service provides coordinated access to shared or distributed file systems. While the primary consumption of file shares would be across OpenStack Compute instances, the service is also intended to be accessible as an independent capability in line with the modular design established by other OpenStack services. The design and prototype implementation provide extensibility for multiple backends (to support vendor or file system specific nuances / capabilities) but is intended to be sufficiently abstract to accommodate any of a variety of shared or distributed file system types. Manila was officially denoted as an incubated OpenStack program during the Juno release cycle.

Note

The OpenStack Foundation regularly evaluates new project contributions for eventual inclusion as officially integrated. As such, the list declared previously is expected to grow over time.

2.3. OpenStack Releases and Distributions

OpenStack's major release cadence switched from every three months to 6 months starting with the Essex release. The Kilo release date is tentative as of the date of this writing. Successive releases are alphabetically incremented. Hence, the release intended for October of 2015, by convention, will start with the letter "L."

Table 2.1. OpenStack Releases to Date

Release	Date
Austin	October 2010
Bexar	February 2011
Cactus	April 2011
Diablo	September 2011
Essex	April 2012
Folsom	October 2012
Grizzly	April 2013
Havana	October 2013
Icehouse	April 2014
Juno	October 2014
Kilo	April 2015
Liberty	October 2015 (planned)

Additionally, for each major version a stable branch is maintained. These branches represent a centralized effort to maintain bugfixes and security vulnerability patches for released OpenStack project versions in a ready-to-deploy form.

OpenStack is often consumed through one of a variety of prepackaged methods (for example: Red Hat Enterprise Linux - OpenStack Platform, Rackspace Private Cloud Software, Canonical Ubuntu, SUSE OpenStack Cloud, Mirantis OpenStack, and a growing variety of other options). Additional distributions or packaged appliances (for example, Nebula) from leading technology vendors are currently under development, are in preview, or are generally available today.

This document is intended to be broadly applicable to deployment with distributions meeting the OpenStack Foundation's requirements for a compliant software distribution.

2.4. NetApp's Contributions To OpenStack

A suite of NetApp drivers for OpenStack Block Storage (aka Cinder) are built into the Icehouse release of OpenStack available from http://www.openstack.org/software/start/.

NetApp has provided enhancements to the OpenStack Compute Service (Nova) and OpenStack Image Service (Glance) projects to enhance instance creation and storage efficiency as well. NetApp has also published a reference architecture for deploying OpenStack Object Storage (Swift) on top of NetApp's E-Series storage solutions that reduces the overall deployment footprint and replication overhead.

NetApp additionally leads a development effort within the OpenStack community to introduce a new core storage service developed under the project name Manila, which adds a shared file system service to the existing block and object storage services. This addresses a critical gap in OpenStack's storage services coverage and enablesg a new category of on-demand file storage for Infrastructure as a Service (IaaS) deployments. Refer to Chapter 6, *OpenStack Shared File System Service (Manila)* for more information on Manila.

Where possible, NetApp intends to (and has to date) contribute integrations upstream in OpenStack directly. NetApp's direct contributions to OpenStack date back to the Essex release.

Chapter 3. About NetApp Storage Systems and Software

3.1. Description of Data ONTAP

NetApp's Data ONTAP operating system delivers an industry-leading, unified storage platform for unprecedented levels of scalability, and data storage flexibility.

Data ONTAP 8.x provides two operating modes, clustered Data ONTAP and 7-Mode. Clustered Data ONTAP operation enhances NetApp's storage efficiency value by introducing massive scalability and nondisruptive operations. With clustered Data ONTAP 8, two or more controllers (or nodes) operate as one shared resource pool or storage cluster. The storage cluster can be expanded, contracted, and subdivided nondisruptively into one or more secure partitions, or NetApp® Storage Virtual Machine (SVM). A SVM is a logical storage container that includes allocated storage resources from within the cluster as well as security parameters, such as rights and permissions. Logical interfaces allow clients to access data within a SVM from anywhere in the cluster. To the application, a SVM presents a securely partitioned storage pool that can be dynamically deployed and redeployed according to changing business requirements.

Data ONTAP powers NetApp's fabric-attached storage (FAS) hardware line.

Clustered Data ONTAP

Scaling performance while controlling costs is one of the most challenging efforts in the data center. High-performance, technical computing, and digital media content applications place extreme demands on storage systems. Compute clusters running these applications can require multiple gigabytes per second of performance and many terabytes — or even petabytes — of capacity. To maintain peak application performance, users must be able to add storage and move data between systems and tiers of storage without disrupting ongoing operations. At the same time, to control costs, users must be able to effectively manage the storage environment.

Clustered Data ONTAP addresses these challenges and provides high-performance and high-capacity requirements. It enables organizations to address faster time to market by providing massive throughput and the scalability necessary to meet the demanding requirements of high-performance computing and virtualization infrastructures. These high-performance levels address the growing demands of performance, manageability, and reliability for large Linux®, UNIX®, Microsoft®, or VMware® clusters.

Clustered Data ONTAP is an operating system from NetApp that includes:

- Nondisruptive operations based on a clustered file system hosted on interconnected nodes
- Multinode scaling with global namespacing technologies

- NetApp FlexVol® for storage virtualization
- NetApp backup and recovery solutions based on local Snapshot™ copies, replication, and mirroring

NetApp's storage clustering feature within Data ONTAP provides a number of key benefits, including the ability to:

Accelerate performance. Clustered Data ONTAP uses a clustered file system technology to provide maximum input/output (I/O) throughput and remove the bottlenecks that affect production. Information can be striped as volumes across any or all of the storage controllers and disks in the system, which enables balanced levels of throughput for even a single file or volume and allows technical teams to run multiple compute jobs concurrently. When many compute nodes simultaneously require data, you can use load-balancing mirrors within Data ONTAP with a clustering system or add NetApp FlexCache® storage accelerators in front of the system to deliver much higher read throughput.

Simplify storage and data management. Clustered Data ONTAP supports fully integrated storage solutions that are easy to install, manage, and maintain. Enhancing this with its global namespace capability, administrators can simplify client-side management by mapping all data volumes in the cluster into a file system tree structure that automatically maps or remaps servers to their data, even if that data is moved. By offering a single system image across multiple storage nodes, the global namespace eliminates the need for complex automounter maps and symbolic link scripts.

Improve data access. Storage is virtualized at the file system level to enable all compute nodes to mount a single file system, access all stored data, and automatically accommodate physical storage changes that are fully transparent to the compute cluster. Each client can access a huge pool of information residing anywhere in the storage cluster through a single mount point.

Keep resources in balance without disrupting operations. As storage nodes are added to the cluster, physical resources, including CPU, cache memory, network I/O bandwidth, and disk I/O bandwidth, are kept in balance automatically. Clustered Data ONTAP enables you to add storage and move data between storage controllers and tiers of storage without disrupting users and applications. This ushers in a whole new paradigm in which capacity increases, workload balancing, eliminating storage I/O hotspots, and component deprecation become normal parts of the data center without needing to schedule downtime. More importantly, these tasks are accomplished without the need to remount shares, modify client settings, or stop active workloads as is typically the case with traditional or other high-performance computing storage systems.

Simplify installation and maintenance. Using standard Network File System (NFS) and Common Internet File System (CIFS) protocols to access clustered Data ONTAP systems without needing to install special clients, network stack filters, or code on each server in the compute cluster is the value of a unified storage product. The clustered Data ONTAP architecture also reduces or eliminates routine capacity allocation and storage management tasks, resulting in more time to address organizational goals and objectives and less time spent managing storage.

Meet high-availability requirements. Along with stringent performance requirements, high reliability is important for technical applications and cluster

computing. Clustered Data ONTAP leverages core NetApp software such as WAFL® (Write Anywhere File Layout), RAID-DP®, and NetApp Snapshot. RAID-DP, a high-performance implementation of RAID 6, protects against double-disk failures, and transparent node failover automatically bypasses any failed components with no interruption in data availability. In addition to having no single point of failure, clustered Data ONTAP supports the expansion or reconfiguration of the storage infrastructure while online, enabling applications to run uninterrupted as more storage capacity, processing power, and/or throughput is added.

Enable continuous operations. Clustered Data ONTAP is configured for continuous operation with the use of high-performance and modular NetApp storage components. Each system consists of one or more FAS building blocks where each building block is a high-availability pair of controllers (storage nodes). Multiple controller pairs form a single, integrated cluster. Clustered Data ONTAP uses Ethernet technology — Gigabit(Gb) and 10Gb — for server connections and for interconnecting FAS controllers. Servers can also be connected through InfiniBand through the use of a gateway device. Each controller can support any mix of high-performance SAS and cost-effective SATA disk drives. Data can move nondisruptively between nodes or between different tiers of disk as performance requirements change. This capability makes sure that data center and IT administrators can maximize performance where needed while simultaneously improving capacity utilization.

Data ONTAP operating in 7-Mode

While clustered Data ONTAP is the preferred operating mode for nearly all new Data ONTAP installations, and is NetApp's focal point for future delivery of additional enhancement and innovation, there are a significant number of 7-Mode based systems in existence and a variety of valid operational considerations that require ongoing use. While NetApp has provided Cinder driver enablement for 7-Mode systems, NetApp recommends that clustered Data ONTAP be used whenever possible.

3.2. Description of FAS Systems

The FAS family is a unified storage systems family with integrated data protection, robust data management, and built-in storage efficiency for virtualized, shared infrastructure and business applications. FAS storage is excellent for data management efficiency, virtualized infrastructure, shared workloads, and IT as a service delivery. FAS systems are often referred to as "filers." FAS systems can be configured with a variety of media types including hybrid (Flash, SSD, SAS, and SATA) as well as all flash configurations.

Detailed information on the individual options within the FAS family is available at: http://www.netapp.com/us/products/storage-systems/.

Note

NetApp's OpenStack Block Storage drivers interact directly with Data ONTAP and are abstract to the specific FAS platform. Any given hardware platform supported by a version of Data ONTAP that in turn is supported by a particular OpenStack Block Storage driver. Refer to

Section 4.6.9, "Data ONTAP Configuration" for more information on prerequisites for Data ONTAP.

3.3. Description of E-Series and EF-Series Systems

The family of E-series and EF-series systems provides performance-efficient, high-density block storage aimed primarily at application-driven workloads.

NetApp E-Series systems are designed to provide:

- Simple, low-touch administration
- Flexible, reliable SAN storage
- Extreme capacity and density
- High performance GB/s or IOPS
- Performance, power, and space efficiency
- Consistent, low latency
- Consistently high IOPS and throughput
- Enterprise-class capabilities (reliability, availability, manageability)

The NetApp EF-Series is an all-flash storage array that brings together extreme performance and enterprise-grade reliability to create a system optimized for latency-sensitive workloads. Building on a heritage of performance leadership, its core architecture has been proven in the world's most demanding and complex computing environments. Its field-proven design is the culmination of 20-years of industry knowledge focused on designing enterprise-class storage. Leveraging experience from over 650,000 systems shipped, the fully-redundant EF-Series all-flash array is architected to provide the highest levels of reliability, availability and data protection.

The E and EF-Series run on the enterprise-proven SANtricity® software platform. SANtricity is designed to combine sustainable, low-latency performance with enterprise-class availability and protection. Its streamlined firmware is well-suited for the extreme demands of latency-sensitive workloads. And SANtricity helps keep I/O flowing with automated path failover, online administration, advanced data protection, proactive monitoring and repair, non-disruptive upgrades, and extensive diagnostic capabilities.

E-Series' Dynamic Disk Pools (DDP) provide an alternative to standard RAID groups. They simplify protection by removing the complexity of configuring RAID groups and allocating hot spares. Utilization is improved by dynamically spreading data, parity, and spare capacity across all drives in a pool, reducing performance bottlenecks due to hot-spots. Additionally, should a drive failure occur, DDP enables return to optimal state significantly faster than RAID6, while reducing the performance impact during the reconstruction of a failed drive. DDP also offers greater protection from multiple drive failures by prioritizing the reconstruction of the most critical segments.

Chapter 4. OpenStack Block Storage Service (Cinder)

4.1. Overview

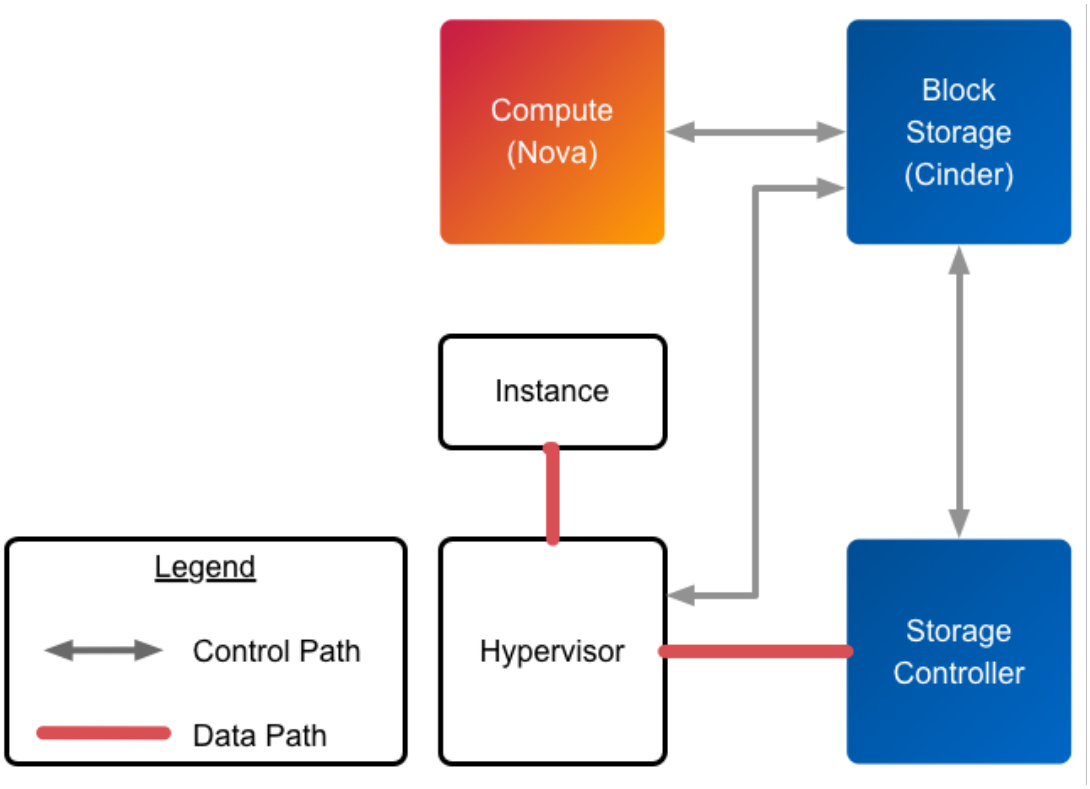
The OpenStack Block Storage service provides management of persistent block storage resources. In addition to acting as secondarily attached persistent storage, you can write images into a Cinder volume for Nova to utilize as a bootable, persistent root volume for an instance. The Block Storage service was originally a component within Nova called **nova-volume**, but emerged as an official, independent project in the Folsom release. Cinder is conceptually similar in function to the well-known Amazon Elastic Block Storage (EBS) offering.

Cinder is typically deployed in conjunction with other OpenStack services (e.g. Compute, Object Storage, Image, etc) as part of a larger, more comprehensive cloud infrastructure. This is not an explicit requirement, as Cinder has been successfully deployed as a standalone solution for block storage provisioning and lifecycle management.

Tip

As a management service, Cinder controls the provisioning and lifecycle management of block storage volumes. It does not reside in the I/O (data) path between the hypervisor and the storage controller, as depicted in Figure 4.1, “Cinder and Nova Logical Architecture”.

Figure 4.1. Cinder and Nova Logical Architecture



4.2. Key Concepts

Volume

A Cinder volume is the fundamental resource unit allocated by the Block Storage service. It represents an allocation of persistent, readable, and writable block storage that could be utilized as the root disk for a compute instance, or as secondary storage that could be attached and/or detached from a compute instance. The underlying connection between the consumer of the volume and the Cinder service providing the volume can be achieved with the iSCSI, NFS, or Fibre Channel storage protocols (dependent on the support of the Cinder driver deployed).

Warning

A Cinder volume is an abstract storage object that may or may not directly map to a "volume" concept from the underlying backend provider of storage. It is critically important to understand this distinction, particularly in context of a Cinder deployment that leverages NetApp storage solutions.

Cinder volumes can be identified uniquely through a UUID assigned by the Cinder service at the time of volume creation. A Cinder volume may also be optionally referred to by a human-readable name, though this string is not guaranteed to be unique within a single tenant or deployment of Cinder.

The actual blocks provisioned in support of a Cinder volume reside on a single Cinder backend. Starting in the Havana release, a Cinder volume can be migrated from one storage backend to another within a deployment of the Cinder service; refer to Section 4.7.3, "Cinder Command Line Interface (CLI)" for an example of volume migration.

The **cinder manage** command allows importing existing storage objects that are not managed by Cinder into new Cinder volumes. The operation will attempt to locate an object within a specified Cinder backend and create the necessary metadata within the Cinder database to allow it to be managed like any other Cinder volume. The operation will also rename the volume to a name appropriate to the particular Cinder driver in use. The imported storage object could be a file, LUN, or a volume depending on the protocol (iSCSI/FC/NFS) and driver (Data ONTAP operating in 7-mode, clustered Data ONTAP, or E-Series) in use. This feature is useful in migration scenarios where virtual machines or other data need to be managed by Cinder; refer to the section called "Cinder manage usage" for an example of the **cinder manage** command.

The **cinder unmanage** command allows Cinder to cease management of a particular Cinder volume. All data stored in the Cinder database related to the volume is removed, but the volume's backing file, LUN, or appropriate storage object is not deleted. This allows the volume to be transferred to another environment for other use cases; refer to the section called "Cinder unmanage usage" for an example of the **cinder unmanage** command.

Snapshot

A Cinder snapshot is a point-in-time, read-only copy of a Cinder volume. Snapshots can be created from an existing Cinder volume that is operational and either attached to an instance or in a detached state. A Cinder snapshot can serve as the content source for a new Cinder volume when the Cinder volume is created with the *create from snapshot* option specified.

Backend

A Cinder backend is the configuration object that represents a single provider of block storage upon which provisioning requests may be fulfilled. A Cinder backend communicates with the storage system through a Cinder driver. Cinder supports multiple backends to be simultaneously configured and managed (even with the same Cinder driver) as of the Grizzly release.

Note

A single Cinder backend may be defined in the `[DEFAULT]` stanza of `cinder.conf`; however, NetApp recommends that the `enabled_backends` configuration option be set to a comma-separated list of backend names, and each backend name have its own configuration stanza with the same name as listed in the `enabled_backends` option. Refer to Section 4.6.1, "Cinder" for an example of the use of this option.

Driver

A Cinder driver is a particular implementation of a Cinder backend that maps the abstract APIs and primitives of Cinder to appropriate constructs within the particular storage solution underpinning the Cinder backend.

Caution

The use of the term "driver" often creates confusion given common understanding of the behavior of “device drivers” in operating systems. The term can connote software that provides a data I/O path. In the case of Cinder driver implementations, the software provides provisioning and other manipulation of storage devices but does not lay in the path of data I/O. For this reason, the term "driver" is often used interchangeably with the alternative (and perhaps more appropriate) term “provider”.

Volume Type

A Cinder volume type is an abstract collection of criteria used to characterize Cinder volumes. They are most commonly used to create a hierarchy of functional capabilities that represent a tiered level of storage services; for example, a cloud administrator might define a `premium` volume type that indicates a greater level of performance than a `basic` volume type, which would represent a best-effort level of performance.

The collection of criteria is specified as a list of key/value pairs, which are inspected by the Cinder scheduler when determining which Cinder backend(s) are able to fulfill a provisioning request. Individual Cinder drivers (and subsequently Cinder backends) may advertise arbitrary key/value pairs (also referred to as capabilities) to the Cinder scheduler, which are then compared against volume type definitions when determining which backend will fulfill a provisioning request.

Extra Spec. An extra spec is a key/value pair, expressed in the style of `key=value`. Extra specs are associated with Cinder volume types, so that when users request volumes of a particular volume type, the volumes are created on storage backends that meet the specified criteria.

Note

The list of default capabilities that may be reported by a Cinder driver and included in a volume type definition include:

- `volume_backend_name`: The name of the backend as defined in `cinder.conf`
- `vendor_name`: The name of the vendor who has implemented the driver (e.g. `NetApp`)
- `driver_version`: The version of the driver (e.g. `1.0`)
- `storage_protocol`: The protocol used by the backend to export block storage to clients (e.g. `iSCSI`, `fc`, or `nfs`)

For a table of NetApp supported extra specs, refer to Table 4.8, "NetApp supported Extra Specs for use with Cinder Volume Types".

QoS Spec. QoS Specs are used to apply generic Quality-of-Service(QoS) support for volumes that can be enforced either at the hypervisor (`front-end`) or at the storage subsystem (`back-end`), or both. QoS specifications are added as standalone objects that can then be associated with Cinder volume types.

Note

The NetApp Cinder drivers do not currently support any back-end QoS specs; however, the NetApp construct of QoS policy groups can be assigned to Cinder volumes managed through a clustered Data ONTAP backend that uses the NFS storage protocol. For more information, see Table 4.8, "NetApp supported Extra Specs for use with Cinder Volume Types".

Storage Pools

With the Juno release of OpenStack, Cinder has introduced the concept of "storage pools". The backend storage may present one or more logical storage resource pools from which Cinder will select as a storage location when provisioning volumes. In releases prior to Juno, NetApp's Cinder drivers contained some logic that determined which FlexVol volume or DDP a Cinder volume would be placed into; with the introduction of pools, all scheduling logic is performed completely within the Cinder scheduler.

For NetApp's Cinder drivers, a Cinder pool is a single container. The container that is mapped to a Cinder pool is dependent on the storage protocol used:

- *iSCSI and Fibre Channel*: a Cinder pool is created for every FlexVol volume within the SVM specified by the configuration option `netapp_vserver`, or for Data ONTAP operating in 7-mode, all FlexVol volumes within the system unless limited by specifying a list of volumes in the configuration option `netapp_volume_list`.
- *NFS*: a Cinder pool is created for each junction path from FlexVol volumes that are listed in the configuration option `nfs_shares_config`.
- *E-Series*: a Cinder pool is created for each DDP listed in the configuration option `netapp_storage_pools`.

For additonal information, refer to Cinder Scheduling and resource pool selection.

Backup & Restore

Cinder offers OpenStack tenants self-service backup and restore operations for their Cinder volumes. These operations are performed on individual volumes. A Cinder backup operation creates a point-in-time, read-only set of data and metadata that can be used to restore the contents of a single Cinder volume either to a new Cinder volume (the default) or to an existing Cinder volume. In contrast

to snapshots, backups are stored in a dedicated repository, independent of the storage pool containing the original volume or the storage backend providing its block storage.

Cinder backup repositories may be implemented either using an object store (such as Swift) or by using an NFS shared filesystem. The Cinder backup service uses a single repository, irrespective of the backends used to provide storage pools for the volumes themselves. For example, a FlexVol volume or Infinite volume exported from a Data ONTAP storage system using NFS can serve as a backup repository for multi-backend, heterogeneous Cinder deployments.

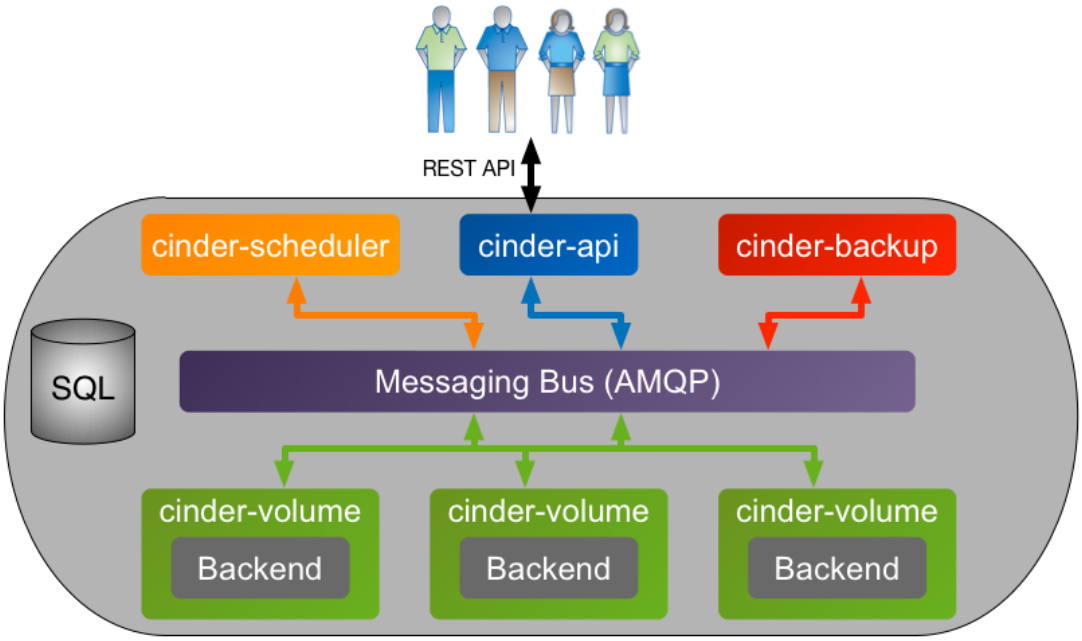
Tenant-controlled, per-volume backup service is complementary to, but not a replacement for, administrative backups of the storage pools themselves that hold Cinder volumes. See http://netapp.github.io/openstack/2015/03/12/cinder-backup-restore/ for a valuable approach to administrative backups when clustered Data ONTAP storage pools are used to host Cinder volumes.

4.3. Process Structure

There are four processes that make up the Cinder service:

- `cinder-api` is an WSGI application that accepts and validates REST (JSON or XML) requests from clients and routes them to other Cinder processes as appropriate over AMQP.
- `cinder-scheduler` determines which backend should serve as the destination for a volume creation or movement request. It maintains non-persistent state for backends (e.g. available capacity, capabilities, and supported extra specs) that can be leveraged when making placement decisions. The algorithm utilized by the scheduler can be changed through Cinder configuration.
- `cinder-volume` accepts requests from other Cinder processes and serves as the operational container for Cinder drivers. This process is multi-threaded and typically has one thread of execution per Cinder backend as defined in the Cinder configuration file.
- `cinder-backup` handles the interaction with potential backup targets (e.g. a file store exported via NFS or an OpenStack Object Storage Service (Swift)) when a client requests a volume backup or restore operation.

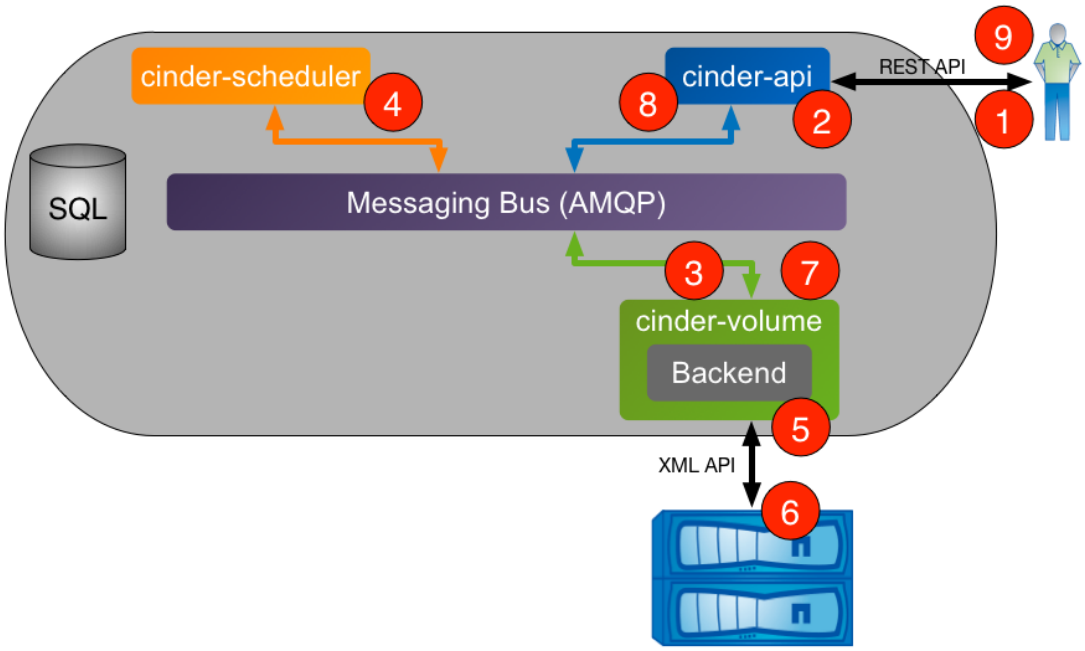
Figure 4.2. Cinder Processes Concept Diagram



Volume Creation Workflow

The following section walks through the steps that occur when a user requests the creation of a new volume from Cinder.

Figure 4.3. Cinder Workflow - Volume Creation



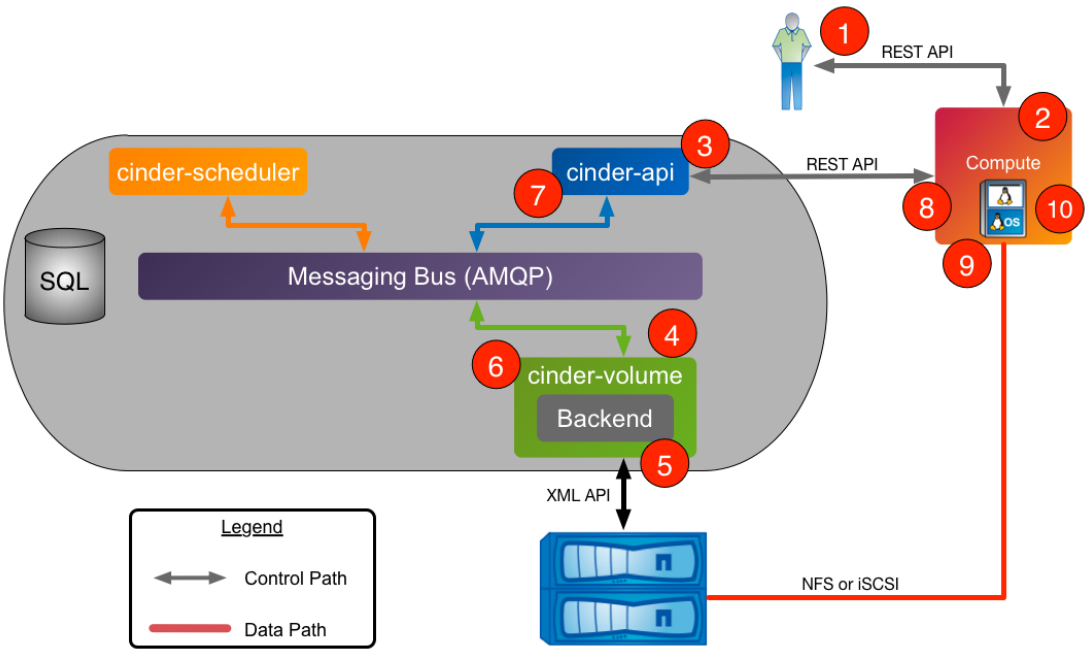
1. Client issues request to create volume through invoking REST API (client may use `python-cinderclient` CLI utility).

2. `cinder-api` process validates request, user credentials; once validated, puts message onto AMQP queue for processing.
3. `cinder-volume` process takes message off of queue, sends message to cinder-scheduler to determine which backend to provision volume into.
4. `cinder-scheduler` process takes message off of queue, generates candidate list based on current state and requested volume criteria (size, availability zone, volume type (including extra specs)).
5. `cinder-volume` process reads response message from `cinder-scheduler` from queue; iterates through candidate list by invoking backend driver methods until successful.
6. NetApp Cinder driver creates requested volume through interactions with storage subsystem (dependent on configuration and protocol).
7. `cinder-volume` process collects volume metadata and connection information and posts response message to AMQP queue.
8. `cinder-api` process reads response message from queue and responds to client.
9. Client receives information including status of creation request, volume UUID, etc.

Volume Attach Workflow

The following section walks through the steps that occur when a user requests that a Cinder volume be attached to a Nova compute instance.

Figure 4.4. Cinder & Nova Workflow - Volume Attach



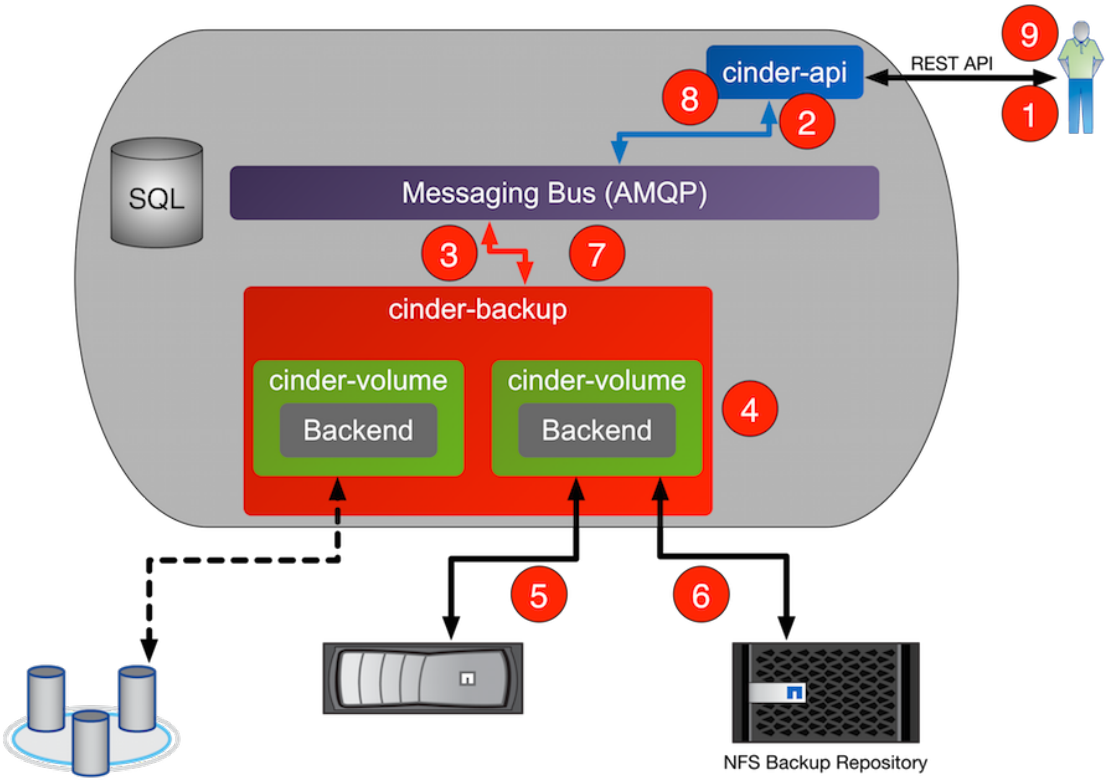
1. Client issues request to attach volume through invoking Nova REST API (client may use `python-novaclient` CLI utility).

2. `nova-api` process validates request, user credentials; once validated, invokes Cinder API to get connection information for specified volume.
3. `cinder-api` process validates request, user credentials; once validated, posts message to volume manager over AMQP.
4. `cinder-volume` reads message from queue, invokes Cinder driver corresponding to volume to be attached.
5. NetApp Cinder driver prepares Cinder volume in preparation for attachment (specific steps dependent on storage protocol used).
6. `cinder-volume` process posts response information to `cinder-api` process via AMQP queue.
7. `cinder-api` process reads response message from `cinder-volume` from queue; passes connection information in RESTful response to Nova caller.
8. Nova creates the connection to the storage with the returned information from Cinder.
9. Nova passes the volume device/file to the hypervisor, who then attaches the volume device/file to the guest VM as an actual or virtualized block device (dependent on storage protocol).

Volume Backup Workflow

The following section walks through the steps that occur when a user requests that a Cinder volume be backed up.

Figure 4.5. Cinder Backup Workflow



1. Client issues request to backup a Cinder volume by invoking REST API (client may use `python-cinderclient` CLI utility).

2. `cinder-api` process validates request, user credentials; once validated, posts message to backup manager over AMQP.

3. `cinder-backup` reads message from queue, creates a database record for the backup and fetches information from the database for the volume to be backed up.

4. `cinder-backup` invokes the `backup_volume` method of the Cinder volume driver corresponding to volume to be backed up, passing the backup record and the connection for the backup service to be used (NFS, Swift, etc.)

5. The appropriate Cinder volume driver attaches to the source Cinder volume.

6. The volume driver invokes the `backup` method for the configured backup service, handing off the volume attachment.

7. The backup service transfers the Cinder volume's data and metadata to the backup repository.

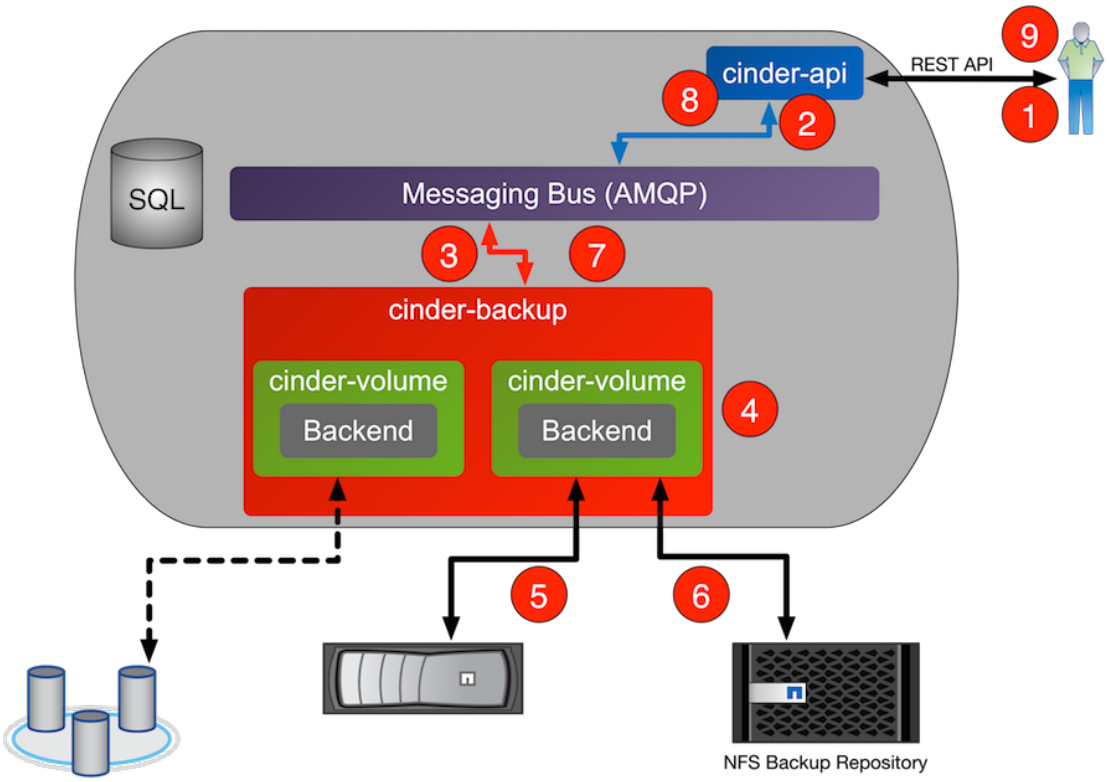
8. The backup service updates the database with the completed record for this backup and posts response information to `cinder-api` process via AMQP queue.

9. `cinder-api` process reads response message from queue and passes results in RESTful response to the client.

Volume Restore Workflow

The following section walks through the steps that occur when a user requests that a Cinder backup be restored.

Figure 4.6. Cinder Restore Workflow



1. Client issues request to restore a Cinder volume by invoking REST API (client may use `python-cinderclient` CLI utility).

2. `cinder-api` process validates request, user credentials; once validated, posts message to backup manager over AMQP.

3. `cinder-backup` reads message from queue, fetches the database record for the backup and a new or preexisting volume database record, depending on whether a pre-existing volume was requested or not.

4. `cinder-backup` invokes the `backup_restore` method of the Cinder volume driver corresponding to volume to be backed up, passing the backup record and the connection for the backup service to be used (NFS, Swift, etc.)

5. The appropriate Cinder volume driver attaches to the destination Cinder volume.

6. The volume driver invokes the `restore` method for the configured backup service, handing off the volume attachment.

7. The backup service locates the backup metadata and data for the Cinder volume in the backup repository and uses these to restore the destination Cinder volume to a state matching the source volume for the original backup operation at the time of that operation.

8. The backup service posts response information to `cinder-api` process via AMQP queue.

9. `cinder-api` process reads response message from `cinder-backup` from queue and passes results in RESTful response to the client.

4.4. API Overview

This section describes some of the most commonly used Cinder API calls and their corresponding CLI commands. It is not meant to be a comprehensive list that is representative of all functionality present in Cinder; for more information, please refer to the OpenStack Configuration Reference. [http://docs.openstack.org/icehouse/config-reference/content/config_overview.html]

Volume API

Table 4.1, "Cinder API Overview - Volume" specifies the valid operations that can be performed on Cinder volumes. Please note that Cinder volumes are identified as CLI command arguments by either their names or UUID.

Table 4.1. Cinder API Overview - Volume

Operation	CLI Command	Description
Create	**cinder create**	Create a Cinder volume of specified size; optional name, availability zone, volume type
Delete	**cinder delete**	Delete an existing Cinder volume; the **cinder force-delete** command may be required if the Cinder volume is in an error state
Edit	**cinder metadata**	Set or unset metadata on a Cinder volume
Extend	**cinder extend**	Increase the capacity of a Cinder volume to the specified size
List	**cinder list**	List all Cinder volumes
Migrate	**cinder migrate**	Move a Cinder volume to a new Cinder backend (specified by name)
Show	**cinder show**	Show details about a Cinder volume
Upload as image	**cinder upload-to-image**	Upload a Cinder volume to the OpenStack Image Service
Manage	**cinder manage**	Bring an existing storage object under Cinder management with specified `source-name`, `source-id`; optional `name`, `availability zone`, `volume type`
Unmanage	**cinder unmanage**	Cease management of an existing Cinder volume without deleting the backing storage object.

Snapshot API

Table 4.2, "Cinder API Overview - Snapshot" specifies the valid operations that can be performed on Cinder snapshots. Please note that Cinder snapshots are identified as CLI command arguments by either their display name or UUID.

Table 4.2. Cinder API Overview - Snapshot

Operation	CLI Command	Description
Create	**cinder snapshot-create**	Create a Cinder snapshot of a specific Cinder volume
Delete	**cinder snapshot-delete**	Delete a Cinder snapshot

Operation	CLI Command	Description
Edit	**cinder snapshot-metadata**	Set or unset metadata on a Cinder snapshot
List	**cinder snapshot-list**	List all Cinder snapshots
Rename	**cinder snapshot-rename**	Change the display-name of a Cinder snapshot
Reset State	**cinder snapshot-reset-state**	Reset the state of a Cinder snapshot
Show	**cinder snapshot-show**	Show details about a Cinder snapshot

Backup API

Table 4.3, "Cinder API Overview - Backup" specifies the valid operations that can be performed on Cinder backups. Please note that Cinder backups are identified as CLI command arguments by either their display name or UUID.

Table 4.3. Cinder API Overview - Backup

Operation	CLI Command	Description
Create	**cinder backup-create**	Create a Cinder backup
Delete	**cinder backup-delete**	Delete a Cinder backup
List	**cinder backup-list**	List all Cinder backups
Restore	**cinder backup-restore**	Restore a Cinder backup into a Cinder volume
Show	**cinder backup-show**	Show details about a Cinder backup

Volume Type API

Table 4.4, "Cinder API Overview - Volume Type" specifies the valid operations that can be performed on Cinder volume types. Please note that Cinder volume types are identified as CLI command arguments by either their display name or UUID.

Table 4.4. Cinder API Overview - Volume Type

Operation	CLI Command	Description
Create	**cinder type-create**	Create a Cinder volume type
Delete	**cinder type-delete**	Delete a Cinder volume type
List	**cinder type-list**	List existing Cinder volume type

Volume Type Extra Specs API

Table 4.5, "Cinder API Overview - Volume Type Extra Specs" specifies the valid operations that can be performed on Cinder volume type extra specs. Please note that Cinder volume type extra specs are properties of Cinder volume types and are identified by their parent object.

Table 4.5. Cinder API Overview - Volume Type Extra Specs

Operation	CLI Command	Description
Set extra specs	**cinder type-key *vtype* set**	Assign extra specs to Cinder volume type
Unset extra specs	**cinder type-key *vtype* unset**	Remove extra specs from Cinder volume type

Volume Type QoS Specs API

Table 4.6, "Cinder API Overview - Volume Type QoS Specs" specifies the valid operations that can be performed on Cinder volume type QoS specs. Please note that Cinder volume type QoS specs are created independently of Cinder volume types and are subsequently associated with a Cinder volume type.

Table 4.6. Cinder API Overview - Volume Type QoS Specs

Operation	CLI Command	Description
Create QoS specs	**cinder qos-create**	Create a Cinder QoS Spec
Delete QoS specs	**cinder qos-delete**	Delete a Cinder QoS Spec
List QoS specs	**cinder qos-list**	List existing Cinder QoS Specs
Show	**cinder qos-show**	Show details about a Cinder QoS Spec
Associate QoS specs	**cinder qos-associate**	Associate a Cinder QoS Spec with a Cinder volume type
Disassociate QoS specs	**cinder qos-disassociate**	Disassociate a Cinder QoS Spec from a Cinder volume type
Edit QoS spec	**cinder qos-key**	Set or unset specifications for a Cinder QoS Spec

4.5. Theory of Operation & Deployment Choices

Construct Mappings between Cinder and Data ONTAP

Cinder Backends and Storage Virtual Machines. Storage Virtual Machines (SVMs, formerly known as Vservers) contain data volumes and one or more LIFs through which they serve data to clients. SVMs can either contain one or more FlexVol volumes, or a single Infinite Volume.

SVMs securely isolate the shared virtualized data storage and network, and each SVM appears as a single dedicated storage virtual machine to clients. Each SVM has a separate administrator authentication domain and can be managed independently by its SVM administrator.

In a cluster, SVMs facilitate data access. A cluster must have at least one SVM to serve data. SVMs use the storage and network resources of the cluster. However, the volumes and LIFs are exclusive to the SVM. Multiple SVMs can coexist in a single cluster without being bound to any node in a cluster. However, they are bound to the physical cluster on which they exist.

Important

When deploying Cinder with clustered Data ONTAP, NetApp recommends that each Cinder backend refer to a single SVM within a cluster through the use of the `netapp_vserver` configuration option. While the driver can operate without the explicit declaration of a

mapping between a backend and an SVM, a variety of advanced functionality (e.g. volume type extra-specs) will be disabled.

Cinder volumes and FlexVol volumes. Data ONTAP FlexVol volumes (commonly referred to as volumes) and OpenStack Block Storage volumes (commonly referred to as Cinder volumes) are not semantically analogous. A FlexVol volume is a container of logical data elements (for example: files, Snapshot™ copies, clones, LUNs, et cetera) that is abstracted from physical elements (for example: individual disks, and RAID groups). A Cinder volume is a block device. Most commonly, these block devices are made available to OpenStack Compute instances. NetApp's various driver options for deployment of FAS as a provider of Cinder storage place Cinder volumes, snapshot copies, and clones within FlexVol volumes.

Important

The FlexVol volume is an overarching container for one or more Cinder volumes.

Note

NetApp's OpenStack Cinder drivers are not supported for use with Infinite Volumes, as Data ONTAP currently only supports FlexClone files and FlexClone LUNs with FlexVol volumes.

Cinder volume representation within a FlexVol volume. A Cinder volume has a different representation in Data ONTAP when stored in a FlexVol volume, dependent on storage protocol utilized with Cinder:

- *iSCSI*: When utilizing the iSCSI storage protocol, a Cinder volume is stored as an iSCSI LUN.
- *NFS*: When utilizing the NFS storage protocol, a Cinder volume is a file on an NFS export.
- *Fibre Channel*: When utilizing the Fibre Channel storage protocol, a Cinder volume is stored as a Fibre Channel LUN.

Cinder Scheduling and resource pool selection. When Cinder volumes are created, the Cinder scheduler selects a resource pool from the available storage pools: see the section called "Storage Pools" for an overview. Table 4.7, "Behavioral Differences in Cinder Volume Placement" details the behavioral changes in NetApp's Cinder drivers when scheduling the provisioning of new Cinder volumes.

Table 4.7. Behavioral Differences in Cinder Volume Placement

Driver	Old Behavior (Icehouse and Prior)	New Behavior (as of Juno)
Clustered Data ONTAP: iSCSI	• Total and available capacity of largest available volume (only) is reported to scheduler. • SSC data is aggregated across all volumes and reported to scheduler.	Each FlexVol volume's capacity and SSC data is reported separately as a pool to the Cinder scheduler. The Cinder filters and weighers decide which pool a new volume goes into, and the driver honors that request.

Driver	Old Behavior (Icehouse and Prior)	New Behavior (as of Juno)
	• During volume creation, driver filters volumes by extra specs and filters/weighs volumes by capacity (largest available space first).	
Data ONTAP operating in 7-mode: iSCSI	• Total and available capacity of all volumes are accumulated and reported to the scheduler as a combined value. • No SSC data is available. • During volume creation, driver filters volumes by capacity but does no weighing.	Each FlexVol volume's capacity is reported separately as a pool to the Cinder scheduler. The Cinder filters and weighers decide which pool a new volume goes into, and the driver honors that request.
Clustered Data ONTAP: NFS	• Total and available capacity of largest available volume (only) is reported to scheduler. • SSC data is aggregated across all volumes and reported to scheduler. • During volume creation, driver filters volumes by extra specs and filters/weighs volumes by capacity (largest available space first).	Each FlexVol volume's capacity and SSC data is reported separately as a pool to the Cinder scheduler. The Cinder filters and weighers decide which pool a new volume goes into, and the driver honors that request.
Data ONTAP operating in 7-mode: NFS	• Total and available capacity of all volumes are accumulated and reported to the scheduler as a combined value. • No SSC data is available. • During volume creation, the base NFS driver filters/weighs volumes by capacity (smallest allocated space first).	Each FlexVol volume's capacity is reported separately as a pool to the Cinder scheduler. The Cinder filters and weighers decide which pool a new volume goes into, and the driver honors that request.
E-Series	• Total and available capacity of all dynamic disk pools are accumulated and reported to the scheduler as a combined value. • E-series volume groups are not supported. • No SSC data is available. • During volume creation, driver filters/weighs volumes by capacity (largest available space first).	Each dynamic disk pool's capacity is reported separately as a pool to the Cinder scheduler. The Cinder filters and weighers decide which pool a new volume goes into, and the driver honors that request.

Cinder snapshots versus NetApp Snapshots. A NetApp Snapshot copy is a point-in-time file system image. Low-overhead NetApp Snapshot copies are made possible by the unique features of the WAFL storage virtualization technology that is part of Data ONTAP. The high performance of the NetApp Snapshot makes it highly scalable. A NetApp Snapshot takes only a few seconds to create — typically less than one second, regardless of the size of the volume or the level of activity on the NetApp storage system. After a Snapshot copy has been created, changes to data objects are reflected in updates to the current version of the objects, as if NetApp Snapshot copies did not exist. Meanwhile, the NetApp Snapshot version of the data remains completely stable. A NetApp Snapshot incurs no performance overhead; users can comfortably store up to 255 NetApp Snapshot copies per FlexVol volume, all of which are accessible as read-only and online versions of the data.

Since NetApp Snapshots are taken at the FlexVol level, they can not be directly leveraged within an OpenStack context, as a user of Cinder requests a snapshot be taken of a particular Cinder volume (not the containing FlexVol volume). As a Cinder volume is represented as either a file on NFS or as a LUN (in the case of iSCSI or Fibre Channel), the way that Cinder snapshots are created is through use

of Data ONTAP's' FlexClone technology. By leveraging the FlexClone technology to facilitate Cinder snapshots, it is possible to create many thousands of Cinder snapshots for a single Cinder volume.

FlexClone files or FlexClone LUNs and their parent files or LUNs that are present in the FlexClone volume continue to share blocks the same way they do in the parent FlexVol volume. In fact, all the FlexClone entities and their parents share the same underlying physical data blocks, minimizing physical disk space usage.

Important

When Cinder is deployed with Data ONTAP, Cinder snapshots are created leveraging the FlexClone feature of Data ONTAP. As such, a license option for FlexClone must be enabled.

Deployment Choice: Direct versus Intermediated

The NetApp Cinder driver can operate in two independent modes: a *direct* mode where the Cinder processes directly interact with NetApp FAS storage systems, and an *intermediated* mode where the Cinder processes interact with an additional software entity that issues provisioning and management requests on behalf of Cinder.

OnCommand® Workflow Automator. OnCommand® Workflow Automator (WFA) is a flexible framework that provides automation for storage-related tasks, customization, scripting capabilities, and integration with higher-order IT systems such as orchestration software through web services.

While WFA can be utilized in conjunction with the NetApp unified Cinder driver, a deployment of Cinder and WFA does introduce additional complexity, management entities, and potential points of failure within a cloud architecture. If you have an existing set of workflows that are written within the WFA framework, and are looking to leverage them in lieu of the default provisioning behavior of the Cinder driver operating directly against a FAS system, then it may be desirable to use the intermediated mode.

SANtricity® Web Services Proxy. The NetApp SANtricity® Web Services Proxy provides access through standard HTTPS mechanisms to configuring management services for E-Series storage arrays. You can install Web Services Proxy on either Linux or Windows. As Web Services Proxy satisfies the client request by collecting data or executing configuration change requests to a target storage array, the Web Services Proxy module issues SYMbol requests to the target storage arrays. Web Services Proxy provides a Representative State Transfer (REST)-style API for managing E-Series controllers. The API enables you to integrate storage array management into other applications or ecosystems.

Recommendation

Unless you have a significant existing investment with OnCommand Workflow Automator that you wish to leverage in an OpenStack deployment, it is recommended that you start with the *direct* mode of operation when deploying Cinder with a NetApp FAS system. When

Cinder is used with a NetApp E-Series system, use of the SANtricity Web Services Proxy in the *intermediated* mode is currently required. The SANtricity Web Services Proxy may be deployed in a highly-available topology using an active/passive strategy.

Deployment Choice: FAS vs E-Series

FAS. If rich data management, deep data protection, and storage efficiency are desired and should be availed directly by the storage, the NetApp FAS product line is a natural fit for use within Cinder deployments. Massive scalability, nondisruptive operations, proven storage efficiencies, and a unified architecture (NAS and SAN) are key features offered by the Data ONTAP storage operating system. These capabilities are frequently leveraged in existing virtualization deployments and thus align naturally to OpenStack use cases.

E-Series. For cloud environments where higher performance is critical, or where higher-value data management features are not needed or are implemented within an application, the NetApp E-Series product line can provide a cost-effective underpinning for a Cinder deployment. NetApp E-Series storage offers a feature called Dynamic Disk Pools, which simplifies data protection by removing the complexity of configuring RAID groups and allocating hot spares. Utilization is improved by dynamically spreading data, parity, and spare capacity across all drives in a pool, reducing performance bottlenecks due to hot-spots. Additionally, should a drive failure occur, DDP enables the pool to return to an optimal state significantly faster than RAID6, while reducing the performance impact during the reconstruction of a failed drive.

Note

As of the Icehouse release, NetApp has integrations with Cinder for both FAS and E-Series, and either storage solution can be included as part of a Cinder deployment to leverage the native benefits that either platform has to offer.

Deployment Choice: Clustered Data ONTAP vs Data ONTAP operating in 7-Mode

Clustered Data ONTAP represents NetApp's platform for delivering future innovation in the FAS product line. Its inherent qualities of virtualization of network interfaces, disk subsystem, and administrative storage controller map well to OpenStack constructs. The Storage Virtual Machine storage server (SVM, historically referred to as Vserver) can span across all nodes of a given clustered Data ONTAP deployment, for example. The elasticity provided to expand or contract a Storage Virtual Machine across horizontally scalable resources are capabilities critical to cloud deployment unique to the clustered Data ONTAP mode of operation.

The Data ONTAP 7-Mode drivers are primarily provided to allow rapid use of prior deployed FAS systems for OpenStack block storage requirements. There is no current intention to enhance the 7-Mode driver's capabilities beyond providing basic bug fixes.

Recommendation

NetApp strongly recommends that all OpenStack deployments built upon the NetApp FAS product set leverage clustered Data ONTAP.

Deployment Choice: NFS versus iSCSI

A frequent question from customers and partners is whether to utilize NFS or iSCSI as the storage protocol with a Cinder deployment on top of the NetApp FAS product line. Both protocol options are TCP/IP-based, deliver similar throughputs and latencies, support Cinder features, snapshot copies and cloning are supported to similar degrees, as well as advertisement of other storage efficienty, data protection, and high availability features.

iSCSI.

- At the time of publishing, the maximum number of iSCSI LUNs per NetApp cluster is either 8,192 or 49,152 - dependent on the FAS model number (refer to Hardware Universe [http://hwu.netapp.com] for detailed information for a particular model). Cinder can be configured to operate with multiple NetApp clusters via multi-backend support to increase this number for an OpenStack deployment.
- LUNs consume more management resources and some management tools also have limitations on the number of LUNs.
- When Cinder is used independently of OpenStack Compute, use of iSCSI is essential to provide direct access to block devices. The Cinder driver used in conjunction with NFS relies on libvirt and the hypervisor to represent files on NFS as virtual block devices. When Cinder is utilized in bare-metal or non-virtualized environments, the NFS storage protocol is not an option.

NFS.

- The maximum number of files in a single FlexVol volume exported through NFS is dependent on the size of the FlexVol volume; a 1TB FlexVol can have 33,554,432 files (assuming 32k inodes). The theoretical maximum of files is roughly two billion.
- NFS drivers require support from the hypervisor to virtualize files and present them as block devices to an instance.
- As of the Icehouse release, the use of parallel NFS (pNFS) is supported with the NetApp unified driver, providing enhanced performance and scalability characteristics.
- There is no difference in the maximum size of a Cinder volume regardless of the storage protocol chosen (a file on NFS or an iSCSI LUN are both 16TB).
- Performance differences between iSCSI and NFS are normally negligible in virtualized environments; for a detailed investigation, please refer to NetApp TR3808: VMware® vSphere and ESX 3.5 Multiprotocol Performance Comparison using FC, iSCSI, and NFS [http://www.netapp.com/us/system/pdf-reader.aspx?m=tr-3808.pdf&cc=us].

Recommendation

Deploying the NetApp Cinder driver with clustered Data ONTAP utilizing the NFS storage protocol yields a more scalable OpenStack deployment than iSCSI with negligible performance differences. If Cinder is being used to provide block storage services independent of other OpenStack services, the iSCSI protocol must be utilized.

Tip

A related use case for the use of iSCSI with OpenStack deployments involves creating a FlexVol volume to serve as the storage for OpenStack compute nodes. As more hypervisor nodes are added, a master boot LUN can simply be cloned for each node, and compute nodes can become completely stateless. Since the configuration of hypervisor nodes are usually nearly identical (except for node-specific data like configuration files, logs, etc), the boot disk lends well to optimizations like deduplication and compression.

Currently this configuration must be done outside of the management scope of Cinder, but it serves as another example of how the differentiated capabilities of NetApp storage can be leveraged to ease the deployment and ongoing operation of an OpenStack cloud deployment.

Fibre Channel Switch Fabric With Cinder

Cinder includes a Fibre Channel zone manager facility for configuring zoning in Fibre Channel fabrics, specifically supporting Cisco and Brocade Fibre Channel switches. The user is required to configure the zoning parameters in the Cinder configuration file (`cinder.conf`). An example configuration using Brocade is given below:

```
zoning_mode=fabric ❶

[fc-zone-manager]
fc_fabric_names=fabricA,fabricB
zoning_policy=initiator-target
brcd_sb_connector=cinder.zonemanager.drivers.brocade.brcd_fc_zone_client_cli.BrcdFCZoneClientCLI
fc_san_lookup_service=cinder.zonemanager.drivers.brocade.brcd_fc_san_lookup_service.BrcdFCSanLookupService
zone_driver=cinder.zonemanager.drivers.brocade.brcd_fc_zone_driver.BrcdFCZoneDriver

[fabricA] ❷
fc_fabric_address=hostname
fc_fabric_user=username
fc_fabric_password=password
principal_switch_wwn=00:00:00:00:00:00:00:00

[fabricB] ❸
fc_fabric_address=hostname
fc_fabric_user=username
fc_fabric_password=password
principal_switch_wwn=00:00:00:00:00:00:00:00
```

❶ This option will need to be set in the `DEFAULT` configuration stanza and its value must be `fabric`.

❷ Be sure that the name of the stanza matches one of the values given in the `fc_fabric_names` option in the `fc-zone-manager` configuration stanza.
❸ Be sure that the name of the stanza matches the other value given in the `fc_fabric_names` option in the `fc-zone-manager` configuration stanza.

Important

While OpenStack has support for several Fibre Channel fabric switch vendors, NetApp has validated their drivers with the use of Brocade switches. For more information on other vendors, refer to the upstream documentation [http://docs.openstack.org/trunk/config-reference/content/section_fc-zoning.html].

Using Cinder Volume Types to Create a Storage Service Catalog

The Storage Service Catalog (SSC) is a concept that describes a set of capabilities that enables efficient, repeated, and consistent use and management of storage resources by the definition of policy-based services and the mapping of those services to the backend storage technology. It is meant to abstract away the actual technical implementations of the features at a storage backend into a set of simplified configuration options.

The storage features are organized or combined into groups based on the customer needs to achieve a particular scenario or use case. Based on the catalog of the storage features, intelligent provisioning decisions are made by infrastructure or software enabling the storage service catalog. In OpenStack, this is achieved together by the Cinder filter scheduler and the NetApp driver by making use of volume type extra-specs support together with the filter scheduler. There are some prominent features which are exposed in the NetApp driver including mirroring, de-duplication, compression, and thin provisioning.

When the NetApp unified driver is used with clustered Data ONTAP and E-Series storage systems, you can leverage extra specs with Cinder volume types to ensure that Cinder volumes are created on storage backends that have certain properties (e.g. QoS, mirroring, compression) configured.

Extra specs are associated with Cinder volume types, so that when users request volumes of a particular volume type, they are created on storage backends that meet the list of requirements (e.g. available space, extra specs, etc). You can use the specs in Table 4.8, "NetApp supported Extra Specs for use with Cinder Volume Types" later in this section when defining Cinder volume types with the **cinder type-key** command.

Table 4.8. NetApp supported Extra Specs for use with Cinder Volume Types

Extra spec	Type	Products Supported	Description
`netapp_raid_type`	String	Clustered Data ONTAP	Limit the candidate volume list based on one of the following raid types: `raid4`, `raid_dp`.
`netapp_disk_type`	String	Clustered Data ONTAP, E-Series	Limit the candidate volume list based on one of the following disk types: `ATA`, `BSAS`, `EATA`, `FCAL`, `FSAS`, `LUN`, `MSATA`, `SAS`, `SATA`, `SCSI`, `XATA`, `XSAS`, or `SSD`.

Extra spec	Type	Products Supported	Description
`netapp:qos_policy_group`[a]	String	Clustered Data ONTAP	Specify the name of a QoS policy group, which defines measurable Service Level Objectives, that should be applied to the Cinder volume at the time of volume creation. Ensure that the QoS policy group object within Data ONTAP should be defined before a Cinder volume is created, and that the QoS policy group is not associated with the destination FlexVol volume.
`netapp_disk_encryption`	Boolean	E-Series	Limit the candidate volume list to only the ones that have Full Disk Encryption (FDE) enabled on the storage controller.
`netapp_mirrored`	Boolean	Clustered Data ONTAP	Limit the candidate volume list to only the ones that are mirrored on the storage controller.
`netapp_unmirrored`[b]	Boolean	Clustered Data ONTAP	Limit the candidate volume list to only the ones that are not mirrored on the storage controller.
`netapp_dedup`	Boolean	Clustered Data ONTAP	Limit the candidate volume list to only the ones that have deduplication enabled on the storage controller.
`netapp_nodedup`[b]	Boolean	Clustered Data ONTAP	Limit the candidate volume list to only the ones that have deduplication disabled on the storage controller.
`netapp_compression`	Boolean	Clustered Data ONTAP	Limit the candidate volume list to only the ones that have compression enabled on the storage controller.
`netapp_nocompression`[b]	Boolean	Clustered Data ONTAP	Limit the candidate volume list to only the ones that have compression disabled on the storage controller.
`netapp_thin_provisioned`	Boolean	Clustered Data ONTAP	Limit the candidate volume list to only the ones that support thin provisioning on the storage controller.
`netapp_thick_provisioned`[b]	Boolean	Clustered Data ONTAP	Limit the candidate volume list to only the ones that support thick provisioning on the storage controller.

[a]Please note that this extra spec has a colon (`:`) in its name because it is used by the driver to assign the QoS policy group to the OpenStack Block Storage volume after it has been provisioned.

[b]In the Juno release, these negative-assertion extra specs are formally deprecated by the NetApp unified driver. Instead of using the deprecated negative-assertion extra specs (for example, `netapp_unmirrored`) with a value of `true`, use the corresponding positive-assertion extra spec (for example, `netapp_mirrored`) with a value of `false`.

4.6. Configuration

4.6.1. Cinder

Cinder is configured by changing the contents of the `cinder.conf` file and restarting all of the Cinder processes. Depending on the OpenStack distribution used, this may require issuing commands such as **service openstack-cinder-api restart** or **service cinder-api restart**.

`cinder.conf`

The `cinder.conf` file contains a set of configuration options (one per line), specified as `option_name=`*`value`*. Configuration options are grouped together into a stanza, denoted by `[`*`stanza_name`*`]`. There must be at least one stanza named `[DEFAULT]` that contains configuration parameters that apply generically to Cinder (and not to any particular backend). Configuration options that are associated with a particular Cinder backend should be placed in a separate stanza.

Note

While it is possible to specify driver-specific configuration options within the `[DEFAULT]` stanza, you are unable to define multiple Cinder backends within the `[DEFAULT]` stanza. NetApp strongly recommends that you specify driver-specific configuration in separate named stanzas, being sure to list the backends that should be enabled as the value for the configuration option `enabled_backends`; for example:

```
enabled_backends=myNfsBackend,myIscsiBackend,myFCBackend,myESeriesBackend
```

The `enabled_backends` option should be specified within the `[DEFAULT]` configuration stanza.

4.6.2. NetApp Data ONTAP Drivers for OpenStack Block Storage (Cinder)

NetApp drivers for clustered Data ONTAP and Data ONTAP operating in 7-Mode are now offered in a single, unified driver (in the Grizzly and prior releases, the drivers were written in two separate variants, namely, iSCSI and NFS drivers). The unified driver provides OpenStack with access to NetApp clustered Data ONTAP and Data ONTAP operating in 7-Mode controllers for provisioning and maintaining OpenStack block storage volumes.

Where to Obtain the Drivers

NetApp's contribution strategy involves adding all new capabilities directly into the upstream OpenStack Block Storage repositories, so all the features are available regardless of which distribution you choose when deploying OpenStack. Bug fixes are delivered into the appropriate branches that represent the different releases of OpenStack (e.g. `trunk`, `stable/icehouse`, `stable/havana`, etc).

On occasion, it may be necessary for NetApp to deliver capability to a previous release of OpenStack that can not be accepted in the upstream OpenStack repositories. In that case, we post the capability at the NetApp Github repository - accessible at https://github.com/NetApp/cinder. Be sure to choose the branch from this repository that matches the release version of OpenStack you are deploying with.

Multiple Deployment Options

A variety of OpenStack block storage deployment options for NetApp Data ONTAP based systems are available in the Kilo OpenStack release and involve making deployment choices between the following:

- Clustered Data ONTAP or Data ONTAP operating in 7-Mode
- iSCSI, Fibre Channel, or NFS storage protocol

While there are multiple supported deployment options, since the Havana release there is a new, single NetApp unified driver that can be configured to

achieve any of the desired deployments. In Grizzly and prior releases, there were multiple drivers segmented by storage family, protocol, and integration with additional NetApp management software. The previous drivers have all been deprecated since the Havana release; see Section A.6, "Deprecated Drivers" for more information on the deprecated capabilities.

The following lists all of the individual options and subsequent sections are intended to offer guidance on which configuration options ought to be employed given varying use cases:

- NetApp clustered Data ONTAP with iSCSI
- NetApp clustered Data ONTAP with NFS
- NetApp clustered Data ONTAP with Fibre Channel
- NetApp Data ONTAP operating in 7-Mode with iSCSI
- NetApp Data ONTAP operating in 7-Mode with NFS
- NetApp Data ONTAP operating in 7-Mode with Fibre Channel

4.6.3. NetApp Unified Driver for Clustered Data ONTAP with iSCSI

The NetApp unified driver for clustered Data ONTAP with iSCSI is a driver interface from OpenStack Cinder to NetApp clustered Data ONTAP storage controllers to accomplish provisioning and management of a storage-area network (SAN) block storage entity; that is, a NetApp LUN that uses the iSCSI protocol.

Configuration Options

To set up the NetApp clustered Data ONTAP iSCSI driver for Cinder, the following stanza should be added to the Cinder configuration file (`cinder.conf`):

```
[myIscsiBackend] ❶
volume_backend_name=myIscsiBackend
volume_driver=cinder.volume.drivers.netapp.common.NetAppDriver
netapp_server_hostname=hostname
netapp_server_port=80
netapp_storage_protocol=iscsi ❷
netapp_storage_family=ontap_cluster
netapp_login=admin_username
netapp_password=admin_password
netapp_vserver=svm_name
```

❶ Be sure that the value of the `enabled_backends` option in the `[DEFAULT]` stanza includes the name of the stanza you chose for the backend.
❷ The value of `netapp_storage_protocol` MUST be set to `iscsi`.

Table 4.9, "Configuration options for clustered Data ONTAP with iSCSI" lists the configuration options available for the unified driver for a clustered Data ONTAP deployment that uses the iSCSI storage protocol.

Table 4.9. Configuration options for clustered Data ONTAP with iSCSI

Option	Type	Default Value	Description
`netapp_server_hostname`	Required		The hostname or IP address for the storage system or proxy server. *The value of this option should be the IP address of either the cluster management LIF or the SVM management LIF.*
`netapp_server_port`	Optional		The TCP port to use for communication with the storage system or proxy server. If not specified, Data ONTAP drivers will use 80 for HTTP and 443 for HTTPS; E-Series will use 8080 for HTTP and 8443 for HTTPS.
`netapp_login`	Required		Administrative user account name used to access the storage system or proxy server.
`netapp_password`	Required		Password for the administrative user account specified in the `netapp_login` option.
`netapp_storage_protocol`	Required		The storage protocol to be used. Valid options are `nfs`, `iscsi`, or `fc`.
`netapp_transport_type`	Required	`http`	Transport protocol for communicating with the storage system or proxy server. Valid options include `http` and `https`.
`netapp_size_multiplier`	Optional	1.2	When creating volumes, the quantity to be multiplied to the requested OpenStack volume size to ensure enough space is available on the SVM (aka Vserver). *This value is currently only used when ISCSI has been selected as the storage protocol to be used.*
`netapp_vserver`	Required		This option specifies the storage virtual machine (previously called a Vserver) name on the storage cluster on which provisioning of block storage volumes should occur.
`netapp_storage_family`	Optional	`ontap_cluster`	The storage family type used on the storage system; valid values are `ontap_7mode` for Data ONTAP operating in 7-Mode, `ontap_cluster` for clustered Data ONTAP, or `eseries` for E-Series.

Caution

If you specify an account in the `netapp_login` option that only has SVM administration privileges (rather than cluster administration privileges), some advanced features of the NetApp unified driver will not work and you may see warnings in the Cinder logs. See the section called "Account Permission Considerations" for more details on the required access level permissions for an SVM admin account.

4.6.4. NetApp Unified Driver for Clustered Data ONTAP with NFS

The NetApp unifed driver for clustered Data ONTAP with NFS is a driver interface from OpenStack block storage to a Data ONTAP cluster system to accomplish provisioning and management of OpenStack volumes on NFS exports provided by the Data ONTAP cluster system. The NetApp unified driver for the Data ONTAP cluster does not require any additional management software to achieve the desired functionality. It uses NetApp APIs to interact with the Data ONTAP cluster.

Configuration Options

To set up the NetApp clustered Data ONTAP NFS driver for Cinder, the following stanza should be added to the Cinder configuration file (`cinder.conf`):


```
[myNfsBackend] ❶
volume_backend_name=myNfsBackend
volume_driver=cinder.volume.drivers.netapp.common.NetAppDriver
netapp_server_hostname=hostname
netapp_server_port=80
netapp_storage_protocol=nfs
netapp_storage_family=ontap_cluster
netapp_login=admin_username
netapp_password=admin_password
netapp_vserver=svm_name
nfs_shares_config=path_to_nfs_exports_file
```

❶ Be sure that the value of the `enabled_backends` option in the `[DEFAULT]` stanza includes the name of the stanza you chose for the backend.

Caution

Please note that exported qtrees are not supported by Cinder.

Note

The file referenced in the `nfs_shares_config` configuration option should contain the NFS exports in the `ip:/share` format, for example:

```
10.63.165.215:/nfs/test
10.63.165.215:/nfs2/test2
```

where `ip` corresponds to the IP address assigned to a Data LIF, and `share` refers to a junction path for a FlexVol volume within an SVM. Make sure that volumes corresponding to exports have read/write permissions set on the Data ONTAP controllers. For more information on other parameters available to affect the behavior of NetApp's NFS driver, please refer to http://docs.openstack.org/trunk/config-reference/content/nfs-driver-options.html.

Table 4.10, "Configuration options for clustered Data ONTAP with NFS" lists the configuration options available for the unified driver for a clustered Data ONTAP deployment that uses the NFS storage protocol.

Table 4.10. Configuration options for clustered Data ONTAP with NFS

Option	Type	Default Value	Description
`netapp_server_hostname`	Required		The hostname or IP address for the storage system or proxy server. *The value of this option should be the IP address of either the cluster management LIF or the SVM management LIF.*
`netapp_server_port`	Optional		The TCP port to use for communication with the storage system or proxy server. If not specified, Data ONTAP drivers will use 80 for HTTP and 443 for HTTPS; E-Series will use 8080 for HTTP and 8443 for HTTPS.

Option	Type	Default Value	Description
`netapp_login`	Required		Administrative user account name used to access the storage system or proxy server.
`netapp_password`	Required		Password for the administrative user account specified in the `netapp_login` option.
`netapp_storage_protocol`	Required		The storage protocol to be used. Valid options are `nfs`, `iscsi`, or `fc`.
`netapp_transport_type`	Required	`http`	Transport protocol for communicating with the storage system or proxy server. Valid options include `http` and `https`.
`netapp_copyoffload_tool_path`	Optional		This option specifies the path of the NetApp copy offload tool binary. Ensure that the binary has execute permissions set which allow the effective user of the **cinder-volume** process to execute the file.
`netapp_vserver`	Required		This option specifies the storage virtual machine (previously called a Vserver) name on the storage cluster on which provisioning of block storage volumes should occur.
`netapp_storage_family`	Optional	`ontap_cluster`	The storage family type used on the storage system; valid values are `ontap_7mode` for Data ONTAP operating in 7-Mode, `ontap_cluster` for clustered Data ONTAP, or `eseries` for E-Series.
`nfs_shares_config`	Required	`/etc/cinder/nfs_shares`	The file referenced by this configuration option should contain a list of NFS shares, each on their own line, to which the driver should attempt to provision new Cinder volumes into.
`thres_avl_size_perc_start`	Optional	20	If the percentage of available space for an NFS share has dropped below the value specified by this option, the NFS image cache will be cleaned.
`thres_avl_size_perc_stop`	Optional	60	When the percentage of available space on an NFS share has reached the percentage specified by this option, the driver will stop clearing files from the NFS image cache that have not been accessed in the last `M` minutes, where `M` is the value of the `expiry_thres_minutes` configuration option.
`expiry_thres_minutes`	Optional	720	This option specifies the threshold for last access time for images in the NFS image cache. When a cache cleaning cycle begins, images in the cache that have not been accessed in the last `M` minutes, where `M` is the value of this parameter, will be deleted from the cache to create free space on the NFS share.

Caution

If you specify an account in the `netapp_login` option that only has SVM administration privileges (rather than cluster administration privileges), some advanced features of the NetApp unified driver will not work and you may see warnings in the Cinder logs. See the section called "Account Permission Considerations" for more details on the required access level permissions for an SVM admin account.

4.6.5. NetApp Unified Driver for Clustered Data ONTAP with Fibre Channel

The NetApp unified driver for clustered Data ONTAP with Fibre Channel is a driver interface from OpenStack Cinder to NetApp clustered Data ONTAP

storage controllers to accomplish provisioning and management of a storage-area network (SAN) block storage entity; that is, a NetApp LUN that uses the Fibre Channel protocol.

Configuration Options

To set up the NetApp clustered Data ONTAP Fibre Channel driver for Cinder, the following stanza should be added to the Cinder configuration file (`cinder.conf`):

```
[myFCBackend] ❶
volume_backend_name=myFCBackend
volume_driver=cinder.volume.drivers.netapp.common.NetAppDriver
netapp_server_hostname=hostname
netapp_server_port=80
netapp_storage_protocol=fc ❷
netapp_storage_family=ontap_cluster
netapp_login=admin_username
netapp_password=admin_password
netapp_vserver=svm_name
```

❶ Be sure that the value of the `enabled_backends` option in the `[DEFAULT]` stanza includes the name of the stanza you chose for the backend.
❷ The value of `netapp_storage_protocol` MUST be set to `fc`.

Table 4.11, "Configuration options for clustered Data ONTAP with Fibre Channel" lists the configuration options available for the unified driver for a clustered Data ONTAP deployment that uses the Fibre Channel storage protocol.

Table 4.11. Configuration options for clustered Data ONTAP with Fibre Channel

Option	Type	Default Value	Description
`netapp_server_hostname`	Required		The hostname or IP address for the storage system or proxy server. *The value of this option should be the IP address of either the cluster management LIF or the SVM management LIF.*
`netapp_server_port`	Optional		The TCP port to use for communication with the storage system or proxy server. If not specified, Data ONTAP drivers will use 80 for HTTP and 443 for HTTPS; E-Series will use 8080 for HTTP and 8443 for HTTPS.
`netapp_login`	Required		Administrative user account name used to access the storage system or proxy server.
`netapp_password`	Required		Password for the administrative user account specified in the `netapp_login` option.
`netapp_storage_protocol`	Required		The storage protocol to be used. Valid options are `nfs`, `iscsi` or `fc`.
`netapp_transport_type`	Required	`http`	Transport protocol for communicating with the storage system or proxy server. Valid options include `http` and `https`.
`netapp_size_multiplier`	Optional	1.2	When creating volumes, the quantity to be multiplied to the requested OpenStack

Option	Type	Default Value	Description
			volume size to ensure enough space is available on the SVM (aka Vserver). *This value is currently only used when iSCSI has been selected as the storage protocol to be used.*
`netapp_vserver`	Required		This option specifies the storage virtual machine (previously called a Vserver) name on the storage cluster on which provisioning of block storage volumes should occur.
`netapp_storage_family`	Optional	`ontap_cluster`	The storage family type used on the storage system; valid values are `ontap_7mode` for Data ONTAP operating in 7-Mode, `ontap_cluster` for clustered Data ONTAP, or `eseries` for E-Series.

Caution

If you specify an account in the `netapp_login` option that only has SVM administration privileges (rather than cluster administration privileges), some advanced features of the NetApp unified driver will not work and you may see warnings in the Cinder logs. See the section called "Account Permission Considerations" for more details on the required access level permissions for an SVM admin account.

Important

In order for Fibre Channel to be set up correctly, you also need to set up Fibre Channel zoning for your backends. See the section called "Fibre Channel Switch Fabric With Cinder" for more details on configuring Fibre Channel zoning.

4.6.6. NetApp Unified Driver for Data ONTAP operating in 7-Mode with iSCSI

The NetApp unified driver for Data ONTAP operating in 7-Mode with iSCSI is a driver interface from OpenStack Cinder to NetApp Data ONTAP operating in 7-Mode storage controllers to accomplish provisioning and management of a storage-area network (SAN) block storage entity; that is, a NetApp LUN that uses the iSCSI protocol.

Configuration Options

To set up the NetApp Data ONTAP operating in 7-Mode iSCSI driver for Cinder, the following stanza should be added to the Cinder configuration file (`cinder.conf`):

```
[myIscsiBackend] ❶
volume_backend_name=myIscsiBackend
volume_driver=cinder.volume.drivers.netapp.common.NetAppDriver
netapp_server_hostname=hostname
```



```
netapp_server_port=80
netapp_storage_protocol=iscsi ❷
netapp_storage_family=ontap_7mode ❸
netapp_login=admin_username
netapp_password=admin_password
```

❶ Be sure that the value of the `enabled_backends` option in the `[DEFAULT]` stanza includes the name of the stanza you chose for the backend.
❷ The value of `netapp_storage_protocol` MUST be set to `iscsi`.
❸ The value of `netapp_storage_family` MUST be set to `ontap_7mode`, as the default value for this option is `ontap_cluster`.

Table 4.12, "Configuration options for Data ONTAP operating in 7-Mode with iSCSI" lists the configuration options available for the unified driver for a clustered Data ONTAP deployment that uses the iSCSI storage protocol.

Table 4.12. Configuration options for Data ONTAP operating in 7-Mode with iSCSI

Option	Type	Default Value	Description
`netapp_server_hostname`	Required		The hostname or IP address for the storage system or proxy server. *The value of this option should be the IP address of either the cluster management LIF or the SVM management LIF.*
`netapp_server_port`	Optional		The TCP port to use for communication with the storage system or proxy server. If not specified, Data ONTAP drivers will use 80 for HTTP and 443 for HTTPS; E-Series will use 8080 for HTTP and 8443 for HTTPS.
`netapp_login`	Required		Administrative user account name used to access the storage system or proxy server.
`netapp_password`	Required		Password for the administrative user account specified in the `netapp_login` option.
`netapp_storage_protocol`	Required		The storage protocol to be used. Valid options are `nfs` or `iscsi`.
`netapp_transport_type`	Required	`http`	Transport protocol for communicating with the storage system or proxy server. Valid options include `http` and `https`.
`netapp_size_multiplier`	Optional	1.2	When creating volumes, the quantity to be multiplied to the requested OpenStack volume size to ensure enough space is available on the SVM (aka Vserver). *This value is currently only used when ISCSI has been selected as the storage protocol to be used.*
`netapp_volume_list`	Optional		This option is only utilized when the storage protocol is configured to use iSCSI. This option is used to restrict provisioning to the specified controller volumes. Specify the value of this option to be a comma separated list of NetApp controller volume names to be used for provisioning.
`netapp_vfiler`	Optional		The vFiler unit on which provisioning of block storage volumes will be done. This option is only used by the driver when connecting to an instance with a storage family of Data ONTAP operating in 7-Mode. Only use this option when utilizing the MultiStore feature on the NetApp storage system.
`netapp_storage_family`	Required	`ontap_cluster`	The storage family type used on the storage system; valid values are `ontap_7mode` for Data ONTAP operating in 7-Mode, `ontap_cluster` for clustered Data ONTAP, or `eseries` for E-Series.

4.6.7. NetApp Unified Driver for Data ONTAP operating in 7-Mode with NFS

The NetApp unifed driver for Data ONTAP operating in 7-Mode with NFS is a driver interface from OpenStack block storage to a Data ONTAP cluster system to accomplish provisioning and management of OpenStack volumes on NFS exports provided by the Data ONTAP cluster system. The NetApp unified driver for the Data ONTAP cluster does not require any additional management software to achieve the desired functionality. It uses NetApp APIs to interact with the Data ONTAP cluster.

Configuration Options

To set up the NetApp Data ONTAP operating in 7-Mode NFS driver for Cinder, the following stanza should be added to the Cinder configuration file (`cinder.conf`):

```
[myNfsBackend] ❶
volume_backend_name=myNfsBackend
volume_driver=cinder.volume.drivers.netapp.common.NetAppDriver
netapp_server_hostname=hostname
netapp_server_port=80
netapp_storage_protocol=nfs
netapp_storage_family=ontap_7mode ❷
netapp_login=admin_username
netapp_password=admin_password
nfs_shares_config=path_to_nfs_exports_file
```

❶ Be sure that the value of the `enabled_backends` option in the `[DEFAULT]` stanza includes the name of the stanza you chose for the backend.
❷ The value of `netapp_storage_family` MUST be set to `ontap_7mode`, as the default value for this option is `ontap_cluster`.

Note

The file referenced in the `nfs_shares_config` configuration option should contain the NFS exports in the `ip:/share` format, for example:

```
10.63.165.215:/nfs/test
10.63.165.215:/nfs2/test2
```

where `ip` corresponds to the IP address assigned to a Data LIF, and `share` refers to a junction path for a FlexVol volume within an SVM. Make sure that volumes corresponding to exports have read/write permissions set on the Data ONTAP controllers. For more information on other parameters available to affect the behavior of NetApp's NFS driver, please refer to http://docs.openstack.org/trunk/config-reference/content/nfs-driver-options.html.

Table 4.13, "Configuration options for Data ONTAP operating in 7-Mode with NFS" lists the configuration options available for the unified driver for a Data ONTAP operating in 7-Mode deployment that uses the NFS storage protocol.

Table 4.13. Configuration options for Data ONTAP operating in 7-Mode with NFS

Option	Type	Default Value	Description
`netapp_server_hostname`	Required		The hostname or IP address for the storage system or proxy server. *The value of this option should be the IP address of either the cluster management LIF or the SVM management LIF.*
`netapp_server_port`	Optional		The TCP port to use for communication with the storage system or proxy server. If not specified, Data ONTAP drivers will use 80 for HTTP and 443 for HTTPS; E-Series will use 8080 for HTTP and 8443 for HTTPS.
`netapp_login`	Required		Administrative user account name used to access the storage system or proxy server.
`netapp_password`	Required		Password for the administrative user account specified in the `netapp_login` option.
`netapp_storage_protocol`	Required		The storage protocol to be used. Valid options are `nfs` or `iscsi`.
`netapp_transport_type`	Required	`http`	Transport protocol for communicating with the storage system or proxy server. Valid options include `http` and `https`.
`netapp_vfiler`	Optional		The vFiler unit on which provisioning of block storage volumes will be done. This option is only used by the driver when connecting to an instance with a storage family of Data ONTAP operating in 7-Mode. Only use this option when utilizing the MultiStore feature on the NetApp storage system.
`netapp_storage_family`	Required	`ontap_cluster`	The storage family type used on the storage system; valid values are `ontap_7mode` for Data ONTAP operating in 7-Mode, `ontap_cluster` for clustered Data ONTAP, or `eseries` for E-Series.
`nfs_shares_config`	Required	`/etc/cinder/nfs_shares`	The file referenced by this configuration option should contain a list of NFS shares, each on their own line, to which the driver should attempt to provision new Cinder volumes into.
`thres_avl_size_perc_start`	Optional	20	If the percentage of available space for an NFS share has dropped below the value specified by this option, the NFS image cache will be cleaned.
`thres_avl_size_perc_stop`	Optional	60	When the percentage of available space on an NFS share has reached the percentage specified by this option, the driver will stop clearing files from the NFS image cache that have not been accessed in the last *`M`* minutes, where *`M`* is the value of the `expiry_thres_minutes` configuration option.
`expiry_thres_minutes`	Optional	720	This option specifies the threshold for last access time for images in the NFS image cache. When a cache cleaning cycle begins, images in the cache that have not been accessed in the last *`M`* minutes, where *`M`* is the value of this parameter, will be deleted from the cache to create free space on the NFS share.

4.6.8. NetApp Unified Driver for Data ONTAP operating in 7-Mode with Fibre Channel

The NetApp unified driver for Data ONTAP operating in 7-Mode with Fibre Channel is a driver interface from OpenStack Cinder to NetApp Data ONTAP operating in 7-Mode storage controllers to accomplish provisioning and management of a storage-area network (SAN) block storage entity; that is, a NetApp LUN that uses the Fibre Channel protocol.

Configuration Options

Note

Both nodes in a 7-Mode HA pair *must* be independently declared as separate Cinder backends with an appropriate cross-reference to one another using the `netapp_partner_backend_name` option.

To set up the NetApp Data ONTAP operating in 7-Mode Fibre Channel driver for Cinder, the following stanza should be added to the Cinder configuration file (`cinder.conf`) for the first node in the HA Pair:

```
[myFCBackend] ❶
volume_backend_name=myFCBackend
netapp_partner_backend_name=myOtherFCBackend ❷
volume_driver=cinder.volume.drivers.netapp.common.NetAppDriver
netapp_server_hostname=hostname
netapp_server_port=80
netapp_storage_protocol=fc ❸
netapp_storage_family=ontap_7mode ❹
netapp_login=admin_username
netapp_password=admin_password
```

❶ Be sure that the value of the `enabled_backends` option in the `[DEFAULT]` stanza includes the name of the stanza you chose for the backend.
❷ Be sure that the value of the `netapp_partner_backend_name` is set to the HA Pair's `volume_backend_name` value and that the HA Pair has this node's `volume_backend_name` value in its configuration stanza under the `netapp_partner_backend_name` option.
❸ The value of `netapp_storage_protocol` MUST be set to `fc`.
❹ The value of `netapp_storage_family` MUST be set to `ontap_7mode`, as the default value for this option is `ontap_cluster`.

To set up the second node in the HA Pair, add the following stanza to the Cinder configuration file (`cinder.conf`):

```
[myOtherFCBackend] ❶
volume_backend_name=myOtherFCBackend
netapp_partner_backend_name=myFCBackend ❷
volume_driver=cinder.volume.drivers.netapp.common.NetAppDriver
netapp_server_hostname=hostname
```



```
netapp_server_port=80
netapp_storage_protocol=fc ❸
netapp_storage_family=ontap_7mode ❹
netapp_login=admin_username
netapp_password=admin_password
```

❶ Be sure that the value of the `enabled_backends` option in the `[DEFAULT]` stanza includes the name of the stanza you chose for the backend.
❷ Be sure that the value of the `netapp_partner_backend_name` is set to the HA Pair's `volume_backend_name` value and that the HA Pair has this node's `volume_backend_name` value in its configuration stanza under the `netapp_partner_backend_name` option.
❸ The value of `netapp_storage_protocol` MUST be set to `fc`.
❹ The value of `netapp_storage_family` MUST be set to `ontap_7mode`, as the default value for this option is `ontap_cluster`.

Table 4.14, "Configuration options for Data ONTAP operating in 7-Mode with Fibre Channel" lists the configuration options available for the unified driver for a clustered Data ONTAP deployment that uses the Fibre Channel storage protocol.

Table 4.14. Configuration options for Data ONTAP operating in 7-Mode with Fibre Channel

Option	Type	Default Value	Description
`netapp_server_hostname`	Required		The hostname or IP address for the storage system or proxy server. *The value of this option should be the IP address of either the cluster management LIF or the SVM management LIF.*
`netapp_server_port`	Optional		The TCP port to use for communication with the storage system or proxy server. If not specified, Data ONTAP drivers will use 80 for HTTP and 443 for HTTPS; E-Series will use 8080 for HTTP and 8443 for HTTPS.
`netapp_login`	Required		Administrative user account name used to access the storage system or proxy server.
`netapp_password`	Required		Password for the administrative user account specified in the `netapp_login` option.
`netapp_storage_protocol`	Required		The storage protocol to be used. Valid options are `nfs`, `iscsi` or `fc`.
`netapp_transport_type`	Required	`http`	Transport protocol for communicating with the storage system or proxy server. Valid options include `http` and `https`.
`netapp_size_multiplier`	Optional	1.2	When creating volumes, the quantity to be multiplied to the requested OpenStack volume size to ensure enough space is available on the SVM (aka Vserver). *This value is currently only used when ISCSI has been selected as the storage protocol to be used.*

Option	Type	Default Value	Description
`netapp_volume_list`	Optional		This option is only utilized when the storage protocol is configured to use iSCSI. This option is used to restrict provisioning to the specified controller volumes. Specify the value of this option to be a comma separated list of NetApp controller volume names to be used for provisioning.
`netapp_vfiler`	Optional		The vFiler unit on which provisioning of block storage volumes will be done. This option is only used by the driver when connecting to an instance with a storage family of Data ONTAP operating in 7-Mode. Only use this option when utilizing the MultiStore feature on the NetApp storage system.
`netapp_storage_family`	Required	`ontap_cluster`	The storage family type used on the storage system; valid values are `ontap_7mode` for Data ONTAP operating in 7-Mode, `ontap_cluster` for clustered Data ONTAP, or `eseries` for E-Series.
`netapp_partner_backend_name`	Required		The name of the `cinder.conf` stanza for a Data ONTAP operating in 7-Mode HA partner. This option is only used by the driver when connecting to an instance with a `netapp_storage_family` value of `ontap_7mode` and is required when `netapp_storage_protocol` is set to `fc`.

Important

In order for Fibre Channel to be set up correctly, you also need to set up Fibre Channel zoning for your backends. See the section called "Fibre Channel Switch Fabric With Cinder" for more details on configuring Fibre Channel zoning.

4.6.9. Data ONTAP Configuration

Data ONTAP Prerequisites

The prerequisites for Data ONTAP (both clustered Data ONTAP and Data ONTAP operating in 7-Mode) are:

- The driver requires a storage controller running Data ONTAP 8.1.1 or later.
- The storage system should have the following licenses applied:
  - Base
  - NFS (if the NFS storage protocol is to be used)
  - iSCSI (if the iSCSI storage protocol is to be used)
  - FCP (if the Fibre Channel protocol is to be used)

- FlexClone
- MultiStore (if vFiler units are used with Data ONTAP operating in 7-Mode)

Storage Virtual Machine Considerations

1. Ensure the appropriate licenses (as described previously) are enabled on the storage system for the desired use case.
2. The SVM must be created (and associated with aggregates) before it can be utilized as a provisioning target for Cinder.
3. FlexVol volumes must be created before the integration with Cinder is configured, as there is a many-to-one relationship between Cinder volumes and FlexVol volumes (see Section 4.5, "Theory of Operation & Deployment Choices" for more information).
4. Regardless of the storage protocol used, data LIFs must be created and assigned to SVMs before configuring Cinder.
5. If NFS is used as the storage protocol:
   a. Be sure to enable the NFS service on the SVM.
   b. Be sure to enable the desired version of the NFS protocol (e.g. `v4.0, v4.1-pnfs`) on the SVM.
   c. Be sure to define junction paths from the FlexVol volumes and refer to them in the file referenced by the `nfs_shares_config` configuration option in `cinder.conf`.
6. If iSCSI is used as the storage protocol:
   a. Be sure to enable the iSCSI service on the SVM.
   b. Be sure to set iSCSI as the data protocol on the data LIF.
   c. Note that iSCSI LUNs will be created by Cinder; therefore, it is not necessary to create LUNs or igroups before configuring Cinder.
7. If Fibre Channel is used as the storage protocol:
   a. Be sure to enable the FCP service on the SVM.
   b. Be sure to set FCP as the data protocol on the data LIF.
   c. Note that Fibre Channel LUNs will be created by Cinder; therefore, it is not necessary to create LUNs or igroups before configuring Cinder.
8. Once FlexVol volumes have been created, be sure to configure the desired features (e.g. deduplication, compression, SnapMirror® relationships, etc) before configuring Cinder. While Cinder will periodically poll Data ONTAP to discover changes in configuration and/or features, there is a delay in time between when changes are performed and when they are reflected within Cinder.
9. NetApp recommends that the autogrow capability for FlexVol volumes within Data ONTAP not be used with a Cinder deployment.

Account Permission Considerations

When configuring the NetApp unified driver to interact with a clustered Data ONTAP instance, you must specify the administrative account to use when operations are invoked by the Cinder driver. While an account with cluster-level administrative permissions is normally utilized, it is possible to use a Cluster-wide scoped account with the appropriate privileges. In order to use an Cluster-scoped account with the Cinder driver and clustered Data ONTAP and have access to the full set of features (including Cinder Volume Type Extra Specs support) availed by the Cinder driver, be sure to add the access levels for the commands shown in Table 4.15, "Common Access Level Permissions Required with Cluster Account", Table 4.16, "Access Level Permissions Required For Extra Specs Support with SVM Administrative Account", Table 4.17, "Access Level Permissions Required For iSCSI Support with SVM Administrative Account", and Table 4.19, "Access Level Permissions Required For NFS Support with SVM Administrative Account".

Table 4.15. Common Access Level Permissions Required with Cluster Account

Command	Access Level
vserver	readonly
event	all
security	readonly

Table 4.16. Access Level Permissions Required For Extra Specs Support with SVM Administrative Account

Command	Access Level
snapmirror	readonly
storage aggregate	readonly
storage disk	readonly
volume	readonly
volume efficiency	readonly

Table 4.17. Access Level Permissions Required For iSCSI Support with SVM Administrative Account

Command	Access Level
lun create	all
lun delete	all
lun resize	all
lun	readonly
lun map	all
lun unmap	all
lun mapped	readonly
lun igroup modify	all
lun igroup add	all
lun igroup create	all
lun igroup	readonly
network interface	readonly

Command	Access Level
vserver iscsi	readonly
vserver iscsi interface	readonly
version	all
volume	readonly
volume file clone create	all

Table 4.18. Access Level Permissions Required For Fibre Channel Support with SVM Administrative Account

Command	Access Level
fcp initiator show	readonly
fcp portname show	readonly
lun create	all
lun delete	all
lun resize	all
lun	readonly
lun map	all
lun unmap	all
lun mapped	readonly
lun igroup modify	all
lun igroup add	all
lun igroup create	all
lun igroup	readonly
vserver fcp	readonly
vserver fcp interface	readonly
version	all
volume	readonly
volume file clone create	all

Table 4.19. Access Level Permissions Required For NFS Support with SVM Administrative Account

Command	Access Level
network interface	readonly
version	all
volume	readonly
volume file clone create	all
volume file modify	all
volume file show-disk-usage	readonly

Creating Role for Cluster-Scoped Account. To create a role with the necessary privilege's required, with access via ONTAP API only, use the following command syntax to create the role and the cDOT ONTAP user:

1. Create role with appropriate command directory permissions (note you will need to execute this command for each of the required access levels as described in the earlier tables).


```
security login role create –role openstack –cmddirname [required
 command from earlier tables] -access [Required Access Level]
```

2. Command to create user with appropriate role

```
security login create –username openstack –application ontapi –
authmethod password –role openstack
```

Creating Role for SVM-Scoped Account. To create a role with the necessary privileges required, with access via ONTAP API only, use the following command syntax to create the role and the cDOT ONTAP user:

1. Create role with appropriate command directory permissions (note you will need to execute this command for each of the required access levels as described in the earlier tables).

```
security login role create –role openstack -vserver [vserver_name] –
cmddirname [required command from earlier tables] -access [Required
 Access Level]
```

2. Command to create user with appropriate role

```
security login create –username openstack –application ontapi –
authmethod password –role openstack -vserver [vserver_name]
```

Tip

For more information on how to grant these access level permissions to a role, and then assign the role to an SVM administrative account, please refer to the System Administration Guide for Cluster Administrators [http://support.netapp.com] document in the Clustered DATA ONTAP documentation.

Storage Networking Considerations

1. Ensure there is segmented network connectivity between the hypervisor nodes and the Data LIF interfaces from Data ONTAP.

2. When NFS is used as the storage protocol with Cinder, the node running the **cinder-volume** process will attempt to mount the NFS shares listed in the file referred to within the `nfs_shares_config` configuration option in `cinder.conf`. Ensure that there is appropriate network connectivity between the **cinder-volume** node and the Data LIF interfaces, as well as the cluster/SVM management interfaces.

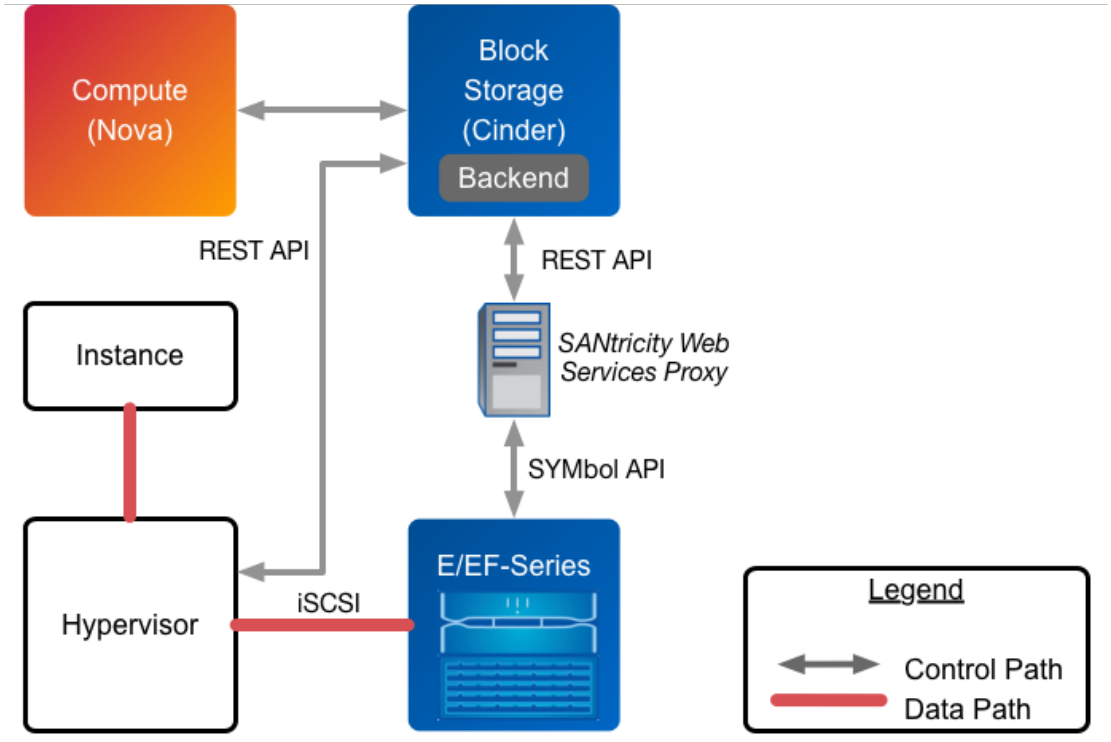
4.6.10. NetApp E-Series Driver for OpenStack Block Storage (Cinder)

The NetApp E-Series driver for Cinder was introduced in the Icehouse release as a new storage family supported within the NetApp unified driver framework. The E-Series driver provides OpenStack with access to NetApp E-Series controllers for provisioning and maintaining OpenStack block storage volumes that use the iSCSI protocol.

Deployment Topology

As described in Section 4.5, "Theory of Operation & Deployment Choices", Cinder with NetApp E-Series requires the use of the NetApp SANtricity Web Services Proxy server deployed as an intermediary between Cinder and the E-Series storage system. A common deployment topology with Cinder, Nova, and an E-Series controller with the SANtricity Web Services Proxy can be seen in Figure 4.7, "Cinder & E-Series Deployment Topology".

Figure 4.7. Cinder & E-Series Deployment Topology



Tip

Documentation (including installation instructions) for the NetApp SANtricity Web Services Proxy server are available for download at https://library.netapp.com/ecm/ecm_download_file/ECMP1506075.

Dynamic disk pools (as described in Section 3.3, "Description of E-Series and EF-Series Systems") are the only supported disk collection strategy when utilizing the

Cinder E-Series driver. For more information on the capabilities of the E-Series storage systems, visit http://support.netapp.com.

Tip

While formally introduced in the Icehouse release of OpenStack, NetApp has backported the E-Series driver to the Grizzly and Havana releases of OpenStack, accessible at https://github.com/NetApp/cinder. Be sure to choose the branch from this repository that matches the release version of OpenStack you are deploying with.

Important

The use of multipath and DM-MP are required when using the OpenStack Block Storage driver for E-Series. Ensure that all unconfigured iSCSI host ports on the E-Series array are disabled for both IPv4 and IPv6 in order for multipath to function properly.

Configuration Options

To set up the NetApp E-Series driver for Cinder, the following stanza should be added to the Cinder configuration file (`cinder.conf`):

```
[myESeriesBackend] ❶
volume_backend_name=myESeriesBackend
volume_driver=cinder.volume.drivers.netapp.common.NetAppDriver
netapp_server_hostname=hostname
netapp_server_port=80
netapp_storage_protocol=iscsi ❷
netapp_storage_family=eseries ❸
netapp_controller_ips=1.2.3.4,1.2.3.5
netapp_sa_password=storage_array_password
netapp_storage_pools=myPool1,myPool2
netapp_login=admin_username
netapp_password=admin_password
use_multipath_for_image_xfer=True
```

❶ Be sure that the value of the `enabled_backends` option in the `[DEFAULT]` stanza includes the name of the stanza you chose for the backend.
❷ As the E-Series solution only provides block storage services, the value of `netapp_storage_protocol` MUST be set to `iscsi`.
❸ The value of `netapp_storage_family` MUST be set to `eseries`, as the default value for this option is `ontap_cluster`.

Important

In order for OpenStack Block Storage and OpenStack Compute to take advantage of multiple paths, the following configuration options must be correctly configured:

- The `use_multipath_for_image_xfer` should be set to `True` in `cinder.conf` within the driver stanza.
- The `iscsi_use_multipath` should be set to `True` in `nova.conf` within the `[libvirt]` stanza.

Table 4.20, "Configuration options for E-Series with iSCSI" lists the configuration options available for the unified driver for a E-Series deployment that use the iSCSI storage protocol.

Table 4.20. Configuration options for E-Series with iSCSI

Option	Type	Default Value	Description
`netapp_server_hostname`	Required		The hostname or IP address for the proxy server.
`netapp_server_port`	Optional		The TCP port to use for communication with the storage system or proxy server. If not specified, Data ONTAP drivers will use 80 for HTTP and 443 for HTTPS; E-Series will use 8080 for HTTP and 8443 for HTTPS.
`netapp_login`	Required		Administrative user account name used to access the proxy server.
`netapp_password`	Required		Password for the administrative user account specified in the `netapp_login` option.
`netapp_storage_protocol`	Required		The storage protocol to be used. Valid options are `nfs` or `iscsi`.
`netapp_transport_type`	Required	`http`	Transport protocol for communicating with the proxy server. Valid options include `http` and `https`.
`netapp_webservice_path`	Optional	`/devmgr/v2`	This option is used to specify the path to the E-Series proxy application on a proxy server. The value is combined with the value of the `netapp_transport_type`, `netapp_server_hostname`, and `netapp_server_port` options to create the URL used by the driver to connect to the proxy application.
`netapp_controller_ips`	Required		This option is only utilized when the storage family is configured to `eseries`. This option is used to restrict provisioning to the specified controllers. Specify the value of this option to be a comma separated list of controller hostnames or IP addresses to be used for provisioning.
`netapp_sa_password`	Optional		Password for the NetApp E-Series storage array.
`netapp_storage_pools`	Required		This option is used to restrict provisioning to the specified storage pools. Only dynamic disk pools are currently supported. Specify the value of this option to be a comma separated list of disk pool names to be used for provisioning.
`netapp_storage_family`	Required	`ontap_cluster`	The storage family type used on the storage system; valid values are `ontap_7mode` for Data ONTAP operating in 7-Mode, `ontap_cluster` for clustered Data ONTAP, or `eseries` for E-Series.
`netapp_eseries_host_type`	Optional	`linux_dm_mp`	This option is used to define how the controllers in the E-Series storage array will work with the particular operating system on the hosts that are connected to it. Valid values include `aix`, `avt`, `factoryDefault`, `hpux`, `linux_atto`, `linux_dm_mp`, `linux_mpp_rdac`, `linux_pathmanager`, `macos`, `ontap`, `svc`, `solaris_v11`, `solaris_v10`, `vmware`, `windows`, `windows_atto`, or `windows_clustered`.

4.6.11. E-Series Configuration

E-Series Prerequisites

The prerequisites for NetApp E-Series are:

- The driver requires the use of the NetApp SANtricity Web Services Proxy version 1.0 or greater.
- The storage controller should have a firmware version installed that is supported by the NetApp SANtricity Web Services Proxy. Refer to the proxy documentation for the most recent list of firmware versions that are supported.

Storage Networking Considerations

1. Ensure there is segmented network connectivity between the hypervisor nodes and the network interfaces present on the E-Series controller.
2. Ensure there is network connectivity between the **cinder-volume** nodes and the interfaces present on the node running the NetApp SANtricity Web Services Proxy software.

4.6.12. NFS Backup Configuration

By default, the Swift object store is used for the backup repository. To use an NFS export as the backup repository, the Cinder backup service must be restarted after adding at least the following two configuration lines to the `[DEFAULT]` section of the Cinder configuration file `cinder.conf`:

```
backup_driver=cinder.backup.drivers.nfs
backup_share=HOST:EXPORT_PATH
```

Other configuration options may be specified in the `[DEFAULT]` section as appropriate. Table 4.21, "Configuration options for NFS backup service" lists the configuration options available for the Cinder Backup service when an NFS export is leveraged as the backup repository.

Table 4.21. Configuration options for NFS backup service

Option	Type	Default Value	Description
`backup_driver`	Required	`cinder.backup.drivers.swift`	Determines the driver that the backup service will load.[a]
`backup_share`	Required		Set to *`HOST:EXPORT_PATH`*.[b]
`backup_mount_options`	Optional		Comma separated list of options to be specified when mounting the NFS export specified in `backup_share`[c].
`backup_compression_algorithm`	Optional	`zlib`	The compression algorithm to be used when sending backup data to the repository. Valid values are `zlib`, `bz2`, and `None`.[d]
`backup_file_size`	Optional	`1999994880`	Data from Cinder volumes larger than this will be stored as multiple files in the backup repository. This option must be a multiple of `backup_sha_block_size_bytes`.

Option	Type	Default Value	Description
backup_sha_block_size_bytes	Optional	32768	Size of Cinder volume blocks for digital signature calculation.

[a]Set the value of the `backup_driver` option to `cinder.backup.drivers.nfs` to enable the NFS backup driver.

[b] Set ***HOST*** to the IP address via which the NFS share is exported (for example, the address on the appropriate NFS data LIF on a Data ONTAP storage system). Set ***PATH*** to the export path used to mount the share.

[c]Mount options such as `nfsvers` should be specified appropriately according to the configuration of the NFS server. For example, if a SteelStore appliance is acting as the backup repository, the appliance will recommend these mount options: `rw,nolock,hard,intr,nfsvers=3,tcp,rsize=131072,wsize=131072,bg`

[d]NetApp recommends that the `backup_compression_algorithm` option be set to `None` in order to avoid consuming excess CPU on the Cinder backup node, and that purpose-built deduplication and compression features be enabled on the Data ONTAP storage systen providing the backup repository to achieve storage efficiency.

4.7. Examples

4.7.1. cinder.conf

This section provides an example Cinder configuration file (`cinder.conf`) that contains three backends - one for clustered Data ONTAP with the NFS storage protocol, one for clustered Data ONTAP with the iSCSI storage protocol, and one for an E-Series deployment (leveraging iSCSI).

```
[DEFAULT]
rabbit_password=netapp123
rabbit_hosts=192.168.33.40
rpc_backend=cinder.openstack.common.rpc.impl_kombu
notification_driver=cinder.openstack.common.notifier.rpc_notifier
periodic_interval=60
lock_path=/opt/stack/data/cinder
state_path=/opt/stack/data/cinder
osapi_volume_extension=cinder.api.contrib.standard_extensions
rootwrap_config=/etc/cinder/rootwrap.conf
api_paste_config=/etc/cinder/api-paste.ini
sql_connection=mysql://root:netapp123@127.0.0.1/cinder?charset=utf8
iscsi_helper=tgtadm
my_ip=192.168.33.40
volume_name_template=volume-%s
verbose=True
debug=True
auth_strategy=keystone
#ceilometer settings
cinder_volume_usage_audit=True
cinder_volume_usage_audit_period=hour
control_exchange=cinder

enabled_backends=cdot-iscsi,cdot-nfs,eseries-iscsi

[cdot-iscsi]
volume_backend_name=cdot-iscsi
volume_driver=cinder.volume.drivers.netapp.common.NetAppDriver
netapp_server_hostname=10.63.40.150
netapp_server_port=80
netapp_storage_protocol=iscsi
```



```
netapp_storage_family=ontap_cluster
netapp_login=admin
netapp_password=netapp123
netapp_vserver=demo-iscsi-svm

[cdot-nfs]
volume_backend_name=cdot-nfs
volume_driver=cinder.volume.drivers.netapp.common.NetAppDriver
netapp_server_hostname=10.63.40.150
netapp_server_port=80
netapp_storage_protocol=nfs
netapp_storage_family=ontap_cluster
netapp_login=admin
netapp_password=netapp123
netapp_vserver=demo-nfs-svm
nfs_shares_config=/etc/cinder/nfs.shares❶

[eseries-iscsi]
volume_backend_name=eseries-iscsi
volume_driver=cinder.volume.drivers.netapp.common.NetAppDriver
netapp_server_hostname=10.63.165.26
netapp_server_port=8081
netapp_storage_protocol=iscsi
netapp_storage_family=eseries
netapp_login=admin
netapp_password=netapp123
netapp_sa_password=password
netapp_controller_ips=10.63.215.225,10.63.215.226
netapp_storage_pools=DDP
```

❶ The content of `/etc/cinder/nfs.shares` is:

```
10.63.40.153:/vol2_dedup
10.63.40.153:/vol3_compressed
10.63.40.153:/vol4_mirrored
10.63.40.153:/vol5_plain
```

4.7.2. Clustered Data ONTAP

This section provides an example configuration script to be executed within Data ONTAP that enables two SVMs, appropriately configured for the Cinder configuration referenced in Section 4.7.1, "`cinder.conf`". Note that you may have to edit IP addresses and feature lists based on the environment and licenses present.

```
# create aggrs
storage aggregate create -aggregate aggr1 -diskcount 24 -nodes \
democluster-1-01

storage aggregate create -aggregate aggr2 -diskcount 24 -nodes \
democluster-1-02
```



```
# create SVMs
vserver create -vserver demo-iscsi-svm -rootvolume vol1 \
-aggregate aggr1 -ns-switch file -rootvolume-security-style unix

vserver create -vserver demo-nfs-svm -rootvolume vol1 \
-aggregate aggr2 -ns-switch file -rootvolume-security-style unix

# iSCSI setup
iscsi create -vserver demo-iscsi-svm

network interface create -vserver demo-iscsi-svm -lif \
demo-iscsi-data -role data -data-protocol iscsi -home-node \
democluster-1-01 -home-port e0d -address 10.63.40.149 \
 -netmask 255.255.192.0

volume create -vserver demo-iscsi-svm -volume vol2 \
-aggregate aggr1 -size 10g

vserver export-policy rule create -vserver demo-iscsi-svm \
-policyname default -clientmatch 0.0.0.0/0 -rorule any -rwrule \
any -superuser any -anon 0

volume create -vserver rcallawa-iscsi-vserver -volume vol1_plain \
-aggregate aggr1 -size 10g

# NFS setup
nfs create -vserver demo-nfs-svm -access true
network interface create -vserver demo-nfs-svm -lif demo-nfs-data \
-role data -home-node democluster-1-02 -home-port e0d -address \
10.63.40.153 -netmask 255.255.192.0

vserver export-policy rule create -vserver demo-nfs-svm \
-policyname default -clientmatch 0.0.0.0/0 -rorule any -rwrule \
any -superuser any -anon 0

volume create -vserver demo-nfs-svm -volume vol2_dedup -aggregate \
aggr2 -size 6g -junction-path /vol2_dedup

volume create -vserver demo-nfs-svm -volume vol3_compressed \
-aggregate aggr2 -size 6g -junction-path /vol3_compressed

volume create -vserver demo-nfs-svm -volume vol4_mirrored \
-aggregate aggr2 -size 5g -junction-path /vol4_mirrored

volume create -vserver demo-nfs-svm -volume vol4_mirror_dest \
-aggregate aggr2 -size 5g -type DP

volume create -vserver demo-nfs-svm -volume vol5_plain \
-aggregate aggr2 -size 6g -junction-path /vol5_plain

# SSC features
volume efficiency on -vserver demo-nfs-svm -volume vol2_dedup

volume efficiency on -vserver demo-nfs-svm -volume vol3_compressed
```



```
volume efficiency modify -vserver demo-nfs-svm -volume \
vol3_compressed -compression true -inline-compression true

snapmirror create -source-path demo-nfs-svm:vol4_mirrored \
-destination-path demo-nfs-svm:vol4_mirror_dest -type DP \
-vserver demo-nfs-svm

snapmirror initialize -source-path demo-nfs-svm:vol4_mirrored \
-destination-path demo-nfs-svm:vol4_mirror_dest -type DP

# enable v4.0, v4.1, pNFS
nfs modify -vserver demo-nfs-svm -v4.0 enabled -v4.1 enabled \
-v4.1-pnfs enabled
```

4.7.3. Cinder Command Line Interface (CLI)

Cinder Service Verification

In this section, we use the Cinder CLI to verify that the configuration presented in Section 4.7.1, "`cinder.conf`" has been properly initialized by Cinder.

```
vagrant@precise64:~/devstack$ cinder service-list
+------------------+--------------------------+------+---------+-------+----------------------------+-----------------+
|      Binary      |           Host           | Zone |  Status | State |         Updated_at         | Disabled Reason |
+------------------+--------------------------+------+---------+-------+----------------------------+-----------------+
| cinder-scheduler |         precise64        | nova | enabled |   up  | 2014-05-20T17:14:12.00 |       None      |
|  cinder-volume   |   precise64@cdot-iscsi❶  | nova | enabled |   up  | 2014-05-20T17:14:10.00 |       None      |
|  cinder-volume   |    precise64@cdot-nfs ❷  | nova | enabled |   up  | 2014-05-20T17:14:11.00 |       None      |
|  cinder-volume   | precise64@eseries-iscsi❸ | nova | enabled |   up  | 2014-05-20T17:14:06.00 |       None      |
+------------------+--------------------------+------+---------+-------+----------------------------+-----------------+
```

❶ This is the backend defined by the configuration stanza `[cdot-iscsi]`.
❷ This is the backend defined by the configuration stanza `[cdot-nfs]`.
❸ This is the backend defined by the configuration stanza `[eseries-iscsi]`.

Creating and Defining Cinder Volume Types

In this section, we create a variety of Cinder Volume Types that leverage both the default capabilities of each driver, as well as the NetApp specific extra specs described in Table 4.8, "NetApp supported Extra Specs for use with Cinder Volume Types".

- The `iscsi` type provisions Cinder volumes onto any backend that uses the iSCSI storage protocol (in this example, that would be `[cdot-iscsi]` and `[eseries-iscsi]`).
- The `nfs` type provisions Cinder volumes onto any backend that uses the NFS storage protocol (in this example, that would be `[cdot-nfs]`).
- The `gold` type provisions Cinder volumes onto any backend that has a SnapMirror relationship (in this example, that would be `[cdot-nfs]`, although only one of the four NFS exports defined in `/etc/cinder/nfs_shares` has this support).

- The `silver` type provisions Cinder volumes onto any backend that has deduplication enabled (in this example, that would be `[cdot-nfs]`, although only one of the four NFS exports defined in `/etc/cinder/nfs_shares` has this support).
- The `bronze` type provisions Cinder volumes onto any backend that has compression enabled (in this example, that would be `[cdot-nfs]`, although only one of the four NFS exports defined in `/etc/cinder/nfs_shares` has this support).
- The `analytics` type provisions Cinder volumes onto the `[eseries-iscsi]` backend.

```
vagrant@precise64:~/devstack$ cinder type-create iscsi
+--------------------------------------+-------+
|                  ID                  |  Name |
+--------------------------------------+-------+
| 46cecec0-a040-476c-9475-036ca5577e6a | iscsi |
+--------------------------------------+-------+

vagrant@precise64:~/devstack$ cinder type-create nfs
+--------------------------------------+------+
|                  ID                  | Name |
+--------------------------------------+------+
| 7564ec5c-a81b-4c62-8376-fdcab62037a2 | nfs  |
+--------------------------------------+------+

vagrant@precise64:~/devstack$ cinder type-create gold
+--------------------------------------+------+
|                  ID                  | Name |
+--------------------------------------+------+
| 0ac5c001-d5fa-4fce-a9e3-e2cce7460027 | gold |
+--------------------------------------+------+

vagrant@precise64:~/devstack$ cinder type-create silver
+--------------------------------------+--------+
|                  ID                  |  Name  |
+--------------------------------------+--------+
| f820211a-ee1c-47ff-8f70-2be45112826d | silver |
+--------------------------------------+--------+

vagrant@precise64:~/devstack$ cinder type-create bronze
+--------------------------------------+--------+
|                  ID                  |  Name  |
+--------------------------------------+--------+
| ae110bfc-0f5a-4e93-abe1-1a31856c0ec7 | bronze |
+--------------------------------------+--------+

vagrant@precise64:~/devstack$ cinder type-create analytics
+--------------------------------------+-----------+
|                  ID                  |    Name   |
+--------------------------------------+-----------+
| 66459c78-5cb5-4a15-a476-f1138a4022bc | analytics |
+--------------------------------------+-----------+


vagrant@precise64:~/devstack$ cinder type-key iscsi set storage_protocol=iSCSI
vagrant@precise64:~/devstack$ cinder type-key nfs set storage_protocol=nfs
vagrant@precise64:~/devstack$ cinder type-key gold set netapp_mirrored=true
```



```
vagrant@precise64:~/devstack$ cinder type-key silver set netapp_dedup=true
vagrant@precise64:~/devstack$ cinder type-key bronze set netapp_compression=true
vagrant@precise64:~/devstack$ cinder type-key analytics set volume_backend_name=eseries-iscsi
vagrant@precise64:~/devstack$ cinder extra-specs-list
+--------------------------------------+-----------+-----------------------------------------+
|                  ID                  |    Name   |               extra_specs               |
+--------------------------------------+-----------+-----------------------------------------+
| 0ac5c001-d5fa-4fce-a9e3-e2cce7460027 |    gold   |       {u'netapp_mirrored': u'true'}     |
| 46cecec0-a040-476c-9475-036ca5577e6a |   iscsi   |     {u'storage_protocol': u'iSCSI'}     |
| 66459c78-5cb5-4a15-a476-f1138a4022bc | analytics | {u'volume_backend_name': u'eseries-iscsi'} |
| 7564ec5c-a81b-4c62-8376-fdcab62037a2 |    nfs    |       {u'storage_protocol': u'nfs'}     |
| ae110bfc-0f5a-4e93-abe1-1a31856c0ec7 |   bronze  |     {u'netapp_compression': u'true'}    |
| f820211a-ee1c-47ff-8f70-2be45112826d |   silver  |        {u'netapp_dedup': u'true'}       |
+--------------------------------------+-----------+-----------------------------------------+
```

Creating Cinder Volumes with Volume Types

In this section, we create volumes with no type, as well as each of the previously defined volume types.

```
vagrant@precise64:~/devstack$ cinder create --display-name myGold --volume-type gold 1
+---------------------------------+--------------------------------------+
|            Property             |                Value                 |
+---------------------------------+--------------------------------------+
|          attachments            |                  []                  |
|       availability_zone         |                 nova                 |
|            bootable             |                false                 |
|           created_at            |      2014-05-20T17:23:57.000000      |
|          description            |                 None                 |
|           encrypted             |                False                 |
|               id                | 3678281e-3924-4512-952a-5b89713fac4d |
|            metadata             |                  {}                  |
|              name               |               myGold                 |
|     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   |   f42d5597fb084480a9626c2ca844db3c   |
|             size                |                  1                   |
|          snapshot_id            |                 None                 |
|          source_volid           |                 None                 |
|             status              |              creating                |
|            user_id              |   a9ef3a9f935f4761861afb003986bdab   |
|          volume_type            |                 gold                 |
+---------------------------------+--------------------------------------+

vagrant@precise64:~/devstack$ cinder create --display-name mySilver --volume-type silver 1
+---------------------------------+--------------------------------------+
|            Property             |                Value                 |
+---------------------------------+--------------------------------------+
|          attachments            |                  []                  |
|       availability_zone         |                 nova                 |
|            bootable             |                false                 |
|           created_at            |      2014-05-20T17:24:12.000000      |
|          description            |                 None                 |
|           encrypted             |                False                 |
|               id                | 6dd3e64d-ca02-4156-8532-24294db89329 |
|            metadata             |                  {}                  |
|              name               |              mySilver                |
|     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   |   f42d5597fb084480a9626c2ca844db3c   |
|             size                |                  1                   |
|          snapshot_id            |                 None                 |
|          source_volid           |                 None                 |
|             status              |              creating                |
|            user_id              |   a9ef3a9f935f4761861afb003986bdab   |
|          volume_type            |                 silver               |
+---------------------------------+--------------------------------------+
```



```
vagrant@precise64:~/devstack$ cinder create --display-name myBronze --volume-type bronze 1
+---------------------------------+--------------------------------------+
|            Property             |                Value                 |
+---------------------------------+--------------------------------------+
|          attachments            |                  []                  |
|       availability_zone         |                 nova                 |
|            bootable             |                false                 |
|           created_at            |      2014-05-20T17:24:28.000000      |
|          description            |                 None                 |
|           encrypted             |                False                 |
|               id                | 459b388f-ae1d-49bf-9c1d-3fe3b18afad3 |
|            metadata             |                  {}                  |
|              name               |               myBronze               |
|     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   |   f42d5597fb084480a9626c2ca844db3c   |
|              size               |                  1                   |
|          snapshot_id            |                 None                 |
|          source_volid           |                 None                 |
|             status              |               creating               |
|            user_id              |   a9ef3a9f935f4761861afb003986bdab   |
|          volume_type            |                bronze                |
+---------------------------------+--------------------------------------+

vagrant@precise64:~/devstack$ cinder create --display-name myISCSI --volume-type iscsi 1
+---------------------------------+--------------------------------------+
|            Property             |                Value                 |
+---------------------------------+--------------------------------------+
|          attachments            |                  []                  |
|       availability_zone         |                 nova                 |
|            bootable             |                false                 |
|           created_at            |      2014-05-20T17:25:42.000000      |
|          description            |                 None                 |
|           encrypted             |                False                 |
|               id                | 93ef9627-ac75-46ae-820b-f722765d7828 |
|            metadata             |                  {}                  |
|              name               |               myISCSI                |
|     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   |   f42d5597fb084480a9626c2ca844db3c   |
|              size               |                  1                   |
|          snapshot_id            |                 None                 |
|          source_volid           |                 None                 |
|             status              |               creating               |
|            user_id              |   a9ef3a9f935f4761861afb003986bdab   |
|          volume_type            |                iscsi                 |
+---------------------------------+--------------------------------------+

vagrant@precise64:~/devstack$ cinder create --display-name myNFS --volume-type nfs 1
+---------------------------------+--------------------------------------+
|            Property             |                Value                 |
+---------------------------------+--------------------------------------+
|          attachments            |                  []                  |
|       availability_zone         |                 nova                 |
|            bootable             |                false                 |
|           created_at            |      2014-05-20T17:26:03.000000      |
|          description            |                 None                 |
|           encrypted             |                False                 |
|               id                | 4ccf1a4c-cfe0-4b35-8435-400547cabcdd |
|            metadata             |                  {}                  |
|              name               |                myNFS                 |
|     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   |   f42d5597fb084480a9626c2ca844db3c   |
|              size               |                  1                   |
|          snapshot_id            |                 None                 |
|          source_volid           |                 None                 |
|             status              |               creating               |
```



```
|            user_id             |   a9ef3a9f935f4761861afb003986bdab   |
|          volume_type           |                 nfs                  |
+--------------------------------+--------------------------------------+

vagrant@precise64:~/devstack$ cinder create --display-name myAnalytics --volume-type
 analytics 1
+--------------------------------+--------------------------------------+
|            Property            |                Value                 |
+--------------------------------+--------------------------------------+
|          attachments           |                  []                  |
|       availability_zone        |                 nova                 |
|            bootable            |                false                 |
|           created_at           |      2014-05-20T17:26:17.000000      |
|          description           |                 None                 |
|           encrypted            |                False                 |
|               id               | 55d0bbfa-6792-406d-8dc8-2bf1fb94b0dc |
|            metadata            |                  {}                  |
|              name              |             myAnalytics              |
|     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  |   f42d5597fb084480a9626c2ca844db3c   |
|              size              |                  1                   |
|          snapshot_id           |                 None                 |
|          source_volid          |                 None                 |
|             status             |              creating                |
|            user_id             |   a9ef3a9f935f4761861afb003986bdab   |
|          volume_type           |              analytics               |
+--------------------------------+--------------------------------------+

vagrant@precise64:/opt/stack/data/cinder$ cinder create --display-name myGenericVol 1
+--------------------------------+--------------------------------------+
|            Property            |                Value                 |
+--------------------------------+--------------------------------------+
|          attachments           |                  []                  |
|       availability_zone        |                 nova                 |
|            bootable            |                false                 |
|           created_at           |      2014-05-20T18:01:02.000000      |
|          description           |                 None                 |
|           encrypted            |                False                 |
|               id               | 12938589-3ca9-49a7-bcd7-003bbcd62895 |
|            metadata            |                  {}                  |
|              name              |             myGenericVol             |
|     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  |   f42d5597fb084480a9626c2ca844db3c   |
|              size              |                  1                   |
|          snapshot_id           |                 None                 |
|          source_volid          |                 None                 |
|             status             |              creating                |
|            user_id             |   a9ef3a9f935f4761861afb003986bdab   |
|          volume_type           |                 None                 |
+--------------------------------+--------------------------------------+

vagrant@precise64:/opt/stack/data/cinder$ cinder list
+--------------------------------------+-----------+--------------+------+-------------+----------+-------------+
|                  ID                  |   Status  |     Name     | Size | Volume Type | Bootable | Attached to |
+--------------------------------------+-----------+--------------+------+-------------+----------+-------------+
| 12938589-3ca9-49a7-bcd7-003bbcd62895 | available | myGenericVol |  1   |     None    |  false   |             |
| 1f71ccef-781b-4628-b0f7-44030acd8181 | available |   myISCSI    |  1   |    iscsi    |  false   |             |
| 3678281e-3924-4512-952a-5b89713fac4d | available |    myGold    |  1   |     gold    |  false   |             |
| 459b388f-ae1d-49bf-9c1d-3fe3b18afad3 | available |   myBronze   |  1   |    bronze   |  false   |             |
| 4ccf1a4c-cfe0-4b35-8435-400547cabcdd | available |    myNFS     |  1   |     nfs     |  false   |             |
| 55d0bbfa-6792-406d-8dc8-2bf1fb94b0dc | available | myAnalytics  |  1   |  analytics  |  false   |             |
| 6dd3e64d-ca02-4156-8532-24294db89329 | available |   mySilver   |  1   |    silver   |  false   |             |
| 93ef9627-ac75-46ae-820b-f722765d7828 | available |   myISCSI    |  1   |    iscsi    |  false   |             |
+--------------------------------------+-----------+--------------+------+-------------+----------+-------------+
```

We'll now look at the local NFS mounts that are present on the node that is running **cinder-volume** and look for the volumes that were created on NFS backends. By mapping the mountpoints to the directories where the volume files exist, we are able to associate that the volumes were created in the appropriate FlexVol volume

that had the NetApp specific features enabled that matched the Cinder volume type definitions.

```
vagrant@precise64:~/devstack$ mount |grep cinder
10.63.40.153:/vol2_dedup on /opt/stack/data/cinder/mnt/6fbcc46d69a86a6be25f3df3e6ae55ba type nfs
 (rw,vers=4,addr=10.63.40.153,clientaddr=192.168.114.157)
10.63.40.153:/vol3_compressed on /opt/stack/data/cinder/mnt/aac4e6312b50b1fd6ddaf25d8dec8aaa type
 nfs (rw,vers=4,addr=10.63.40.153,clientaddr=192.168.114.157)
10.63.40.153:/vol4_mirrored on /opt/stack/data/cinder/mnt/89af08204a543dd0985fa11b16f3d22f type nfs
 (rw,vers=4,addr=10.63.40.153,clientaddr=192.168.114.157)
10.63.40.153:/vol5_plain on /opt/stack/data/cinder/mnt/e15a92dcf98a7b3fdb3963e39ed0796f type nfs
 (rw,vers=4,addr=10.63.40.153,clientaddr=192.168.114.157)
vagrant@precise64:~/devstack$ cd /opt/stack/data/cinder/
vagrant@precise64:/opt/stack/data/cinder$ find . -name volume-\*
./mnt/89af08204a543dd0985fa11b16f3d22f/volume-3678281e-3924-4512-952a-5b89713fac4d ❶
./mnt/aac4e6312b50b1fd6ddaf25d8dec8aaa/volume-459b388f-ae1d-49bf-9c1d-3fe3b18afad3 ❷
./mnt/6fbcc46d69a86a6be25f3df3e6ae55ba/volume-6dd3e64d-ca02-4156-8532-24294db89329 ❸
./mnt/6fbcc46d69a86a6be25f3df3e6ae55ba/volume-4ccf1a4c-cfe0-4b35-8435-400547cabcdd ❹
```

❶ This is the volume of type `gold` which was placed on `10.63.40.153:/vol4_mirrored`.

❷ This is the volume of type `bronze` which was placed on `10.63.40.153:/vol3_compressed`.

❸ This is the volume of type `silver` which was placed on `10.63.40.153:/vol2_dedup`.

❹ This is the volume of type `nfs` which was placed on `10.63.40.153:/vol2_dedup`. It could have been placed on `10.63.40.153:/vol3_compressed`, `10.63.40.153:/vol4_mirrored`, or `10.63.40.153:/vol5_plain` as any of those destinations would have fulfilled the volume type criteria of `storage_protocol=nfs`.

Note

Note that the volumes of type `analytics` and `iscsi`, as well as the volume created without a type did not appear under the NFS mount points because they were created as iSCSI LUNs within the E-Series and CDOT systems, respectively.

Cinder manage usage

In this section we import a Data ONTAP iSCSI LUN by specifying its path in `source-name` or LUN UUID in `source-id`.

```
$ cinder service-list
+------------------+--------------------+------+---------+-------+----------------------------+-----------------+
|      Binary      |        Host        | Zone |  Status | State |         Updated_at         | Disabled Reason |
+------------------+--------------------+------+---------+-------+----------------------------+-----------------+
| cinder-scheduler |     openstack9     | nova | enabled |   up  | 2014-08-25T15:10:22.000000 |       None      |
|  cinder-volume   |     openstack9     | nova | enabled |  down | 2014-08-21T17:38:14.000000 |       None      |
|  cinder-volume   | openstack9@eseries | nova | enabled |   up  | 2014-08-25T15:10:27.000000 |       None      |
|  cinder-volume   | openstack9@iscsi   | nova | enabled |   up  | 2014-08-25T15:10:28.000000 |       None      |
+------------------+--------------------+------+---------+-------+----------------------------+-----------------+
```

```
$ cinder manage --id-type source-name openstack9@iscsi#pool /vol/vol1/lun1
+--------------------------------+----------------------------------------+
|            Property            |                 Value                  |
+--------------------------------+----------------------------------------+
|          attachments           |                   []                   |
|       availability_zone        |                  nova                  |
|            bootable            |                 false                  |
|           created_at           |      2014-08-25T15:11:18.000000        |
```



```
|          description           |                None                |
|           encrypted            |               False                |
|               id               | 9a62ce5f-b125-48e8-8c94-79356b27f2a9 |
|            metadata            |                 {}                 |
|              name              |                None                |
|     os-vol-host-attr:host      |       openstack9@iscsi#pool        |
| os-vol-mig-status-attr:migstat |                None                |
| os-vol-mig-status-attr:name_id |                None                |
|  os-vol-tenant-attr:tenant_id  |   8b4ef3cd82f145738ad8195e6bd3942c   |
|              size              |                 0                  |
|          snapshot_id           |                None                |
|          source_volid          |                None                |
|             status             |              creating              |
|            user_id             |   1b1c9e72e33f4a35b73a8e2d43354d1c   |
|          volume_type           |                None                |
+--------------------------------+-------------------------------------+

$ cinder manage --id-type source-id openstack9@iscsi#pool
 013a7fe0-039b-459e-8cc2-7b59c693139d
+--------------------------------+-------------------------------------+
|            Property            |                Value                |
+--------------------------------+-------------------------------------+
|          attachments           |                 []                  |
|       availability_zone        |                nova                 |
|            bootable            |               false                 |
|           created_at           |      2014-08-25T15:13:18.000000     |
|          description           |                None                 |
|           encrypted            |               False                 |
|               id               | f2c94f4d-adb3-4c3c-a6aa-cb4c52bd2e39 |
|            metadata            |                 {}                  |
|              name              |                None                 |
|     os-vol-host-attr:host      |       openstack9@iscsi#pool         |
| os-vol-mig-status-attr:migstat |                None                 |
| os-vol-mig-status-attr:name_id |                None                 |
|  os-vol-tenant-attr:tenant_id  |   8b4ef3cd82f145738ad8195e6bd3942c   |
|              size              |                 0                   |
|          snapshot_id           |                None                 |
|          source_volid          |                None                 |
|             status             |              creating               |
|            user_id             |   1b1c9e72e33f4a35b73a8e2d43354d1c   |
|          volume_type           |                None                 |
+--------------------------------+-------------------------------------+

$ cinder list
+--------------------------------------+----------------+------+------+-------------+----------+-------------+
|                  ID                  |     Status     | Name | Size | Volume Type | Bootable | Attached to |
+--------------------------------------+----------------+------+------+-------------+----------+-------------+
| 9a62ce5f-b125-48e8-8c94-79356b27f2a9 |   available    | None |  1   |     None    |  false   |             |
+--------------------------------------+----------------+------+------+-------------+----------+-------------+
| f2c94f4d-adb3-4c3c-a6aa-cb4c52bd2e39 |   available    | None |  1   |     None    |  false   |             |
+--------------------------------------+----------------+------+------+-------------+----------+-------------+
```

In this section we import an E-Series volume by specifying its label in `source-name` or world-wide identifier in `source-id`.

```
$ cinder service-list
+------------------+--------------------+------+---------+-------+----------------------------+-----------------+
|      Binary      |        Host        | Zone |  Status | State |         Updated_at         | Disabled Reason |
+------------------+--------------------+------+---------+-------+----------------------------+-----------------+
| cinder-scheduler |     openstack9     | nova | enabled |   up  | 2014-08-25T15:10:22.000000 |       None      |
|  cinder-volume   |     openstack9     | nova | enabled |  down | 2014-08-21T17:38:14.000000 |       None      |
|  cinder-volume   | openstack9@eseries | nova | enabled |   up  | 2014-08-25T15:10:27.000000 |       None      |
|  cinder-volume   | openstack9@iscsi   | nova | enabled |   up  | 2014-08-25T15:10:28.000000 |       None      |
+------------------+--------------------+------+---------+-------+----------------------------+-----------------+

$ cinder manage --id-type source-name openstack9@eseries#pool WCAABGUIYJBAHKOYTNWKH5Y2NU
+--------------------------------+-------------------------------------+
|            Property            |                Value                |
+--------------------------------+-------------------------------------+
|          attachments           |                 []                  |
|       availability_zone        |                nova                 |
|            bootable            |               false                 |
|           created_at           |      2014-08-25T15:21:18.000000     |
```



```
|          description           |                None                |
|           encrypted            |                False               |
|               id               | 206a6731-f23b-419d-8131-8bccbbd83647 |
|            metadata            |                 {}                 |
|              name              |                None                |
|     os-vol-host-attr:host      |       openstack9@eseries#pool      |
| os-vol-mig-status-attr:migstat |                None                |
| os-vol-mig-status-attr:name_id |                None                |
|  os-vol-tenant-attr:tenant_id  |   8b4ef3cd82f145738ad8195e6bd3942c   |
|              size              |                 0                  |
|          snapshot_id           |                None                |
|          source_volid          |                None                |
|             status             |              creating              |
|            user_id             |   1b1c9e72e33f4a35b73a8e2d43354d1c   |
|          volume_type           |                None                |
+--------------------------------+------------------------------------+

$ cinder manage --id-type source-id openstack9@eseries#pool
 60:08:0e:50:00:23:c7:34:00:00:47:33:54:03:7f:b9
+--------------------------------+------------------------------------+
|            Property            |               Value                |
+--------------------------------+------------------------------------+
|          attachments           |                 []                 |
|       availability_zone        |                nova                |
|            bootable            |               false                |
|           created_at           |     2014-08-25T15:25:18.000000     |
|          description           |                None                |
|           encrypted            |                False               |
|               id               | ad0262e0-bbe6-4b4d-8c36-ea6a361d777a |
|            metadata            |                 {}                 |
|              name              |                None                |
|     os-vol-host-attr:host      |       openstack9@eseries#pool      |
| os-vol-mig-status-attr:migstat |                None                |
| os-vol-mig-status-attr:name_id |                None                |
|  os-vol-tenant-attr:tenant_id  |   8b4ef3cd82f145738ad8195e6bd3942c   |
|              size              |                 0                  |
|          snapshot_id           |                None                |
|          source_volid          |                None                |
|             status             |              creating              |
|            user_id             |   1b1c9e72e33f4a35b73a8e2d43354d1c   |
|          volume_type           |                None                |
+--------------------------------+------------------------------------+

$ cinder list
+--------------------------------------+----------------+------+------+-------------+----------+-------------+
|                  ID                  |     Status     | Name | Size | Volume Type | Bootable | Attached to |
+--------------------------------------+----------------+------+------+-------------+----------+-------------+
| 206a6731-f23b-419d-8131-8bccbbd83647 |   available    | None |  1   |     None    |  false   |             |
+--------------------------------------+----------------+------+------+-------------+----------+-------------+
| ad0262e0-bbe6-4b4d-8c36-ea6a361d777a |   available    | None |  1   |     None    |  false   |             |
+--------------------------------------+----------------+------+------+-------------+----------+-------------+
```

Cinder unmanage usage

In this section we unmanage a Cinder volume by specifying its ID.

```
$ cinder list
+--------------------------------------+----------------+------+------+-------------+----------+-------------+
|                  ID                  |     Status     | Name | Size | Volume Type | Bootable | Attached to |
+--------------------------------------+----------------+------+------+-------------+----------+-------------+
| 206a6731-f23b-419d-8131-8bccbbd83647 |   available    | None |  1   |     None    |  false   |             |
+--------------------------------------+----------------+------+------+-------------+----------+-------------+
| ad0262e0-bbe6-4b4d-8c36-ea6a361d777a |   available    | None |  1   |     None    |  false   |             |
+--------------------------------------+----------------+------+------+-------------+----------+-------------+

$ cinder unmanage 206a6731-f23b-419d-8131-8bccbbd83647

$ cinder list
+--------------------------------------+----------------+------+------+-------------+----------+-------------+
|                  ID                  |     Status     | Name | Size | Volume Type | Bootable | Attached to |
+--------------------------------------+----------------+------+------+-------------+----------+-------------+
| ad0262e0-bbe6-4b4d-8c36-ea6a361d777a |   available    | None |  1   |     None    |  false   |             |
+--------------------------------------+----------------+------+------+-------------+----------+-------------+
```


Chapter 5. OpenStack Image Service (Glance)

5.1. Overview

The OpenStack Image Service (Glance) provides discovery, registration and delivery services for disk and server images. The ability to copy or snapshot a server image and immediately store it away is a powerful capability of the OpenStack cloud operating system. Stored images can be used as a template to get new servers up and running quickly and more consistently if you are provisioning multiple servers than installing a server operating system and individually configuring additional services. It can also be used to store and catalog an unlimited number of backups.

Glance can store disk and server images in a variety of back-ends (called stores), including through NFS and Object Storage. The Glance API provides a standard REST interface for querying information about disk images and lets clients stream the images to new servers. A multiformat image registry allowing uploads of private and public images in a variety of formats.

5.2. Key Concepts

Image

A virtual machine image is a single file which contains a virtual disk that has a bootable operating system installed on it. Virtual machine images come in different formats, such as `raw` and `qcow2`.

Store

An image store is where the virtual machine images managed by Glance reside on a persistent medium. While Glance currently has support for many different stores, the most commonly deployed stores are `file` and `swift`.

`file` This store refers to a directory on a local file system where the **glance-registry** service is running. The directory could refer to:

- locally attached storage
- a remote, shared filesystem (e.g. NFS) - see Section 5.4.2, “Configuration of Glance with NetApp FAS”, or
- a remote block device that has been partitioned and a filesystem created within a partition - see Section 5.4.3, “Configuration of Glance with NetApp E-Series and EF-Series”

`swift` This store refers to an instance of the OpenStack Object Storage service (Swift).

5.3. Theory of Operation & Deployment Choices

Glance and Clustered Data ONTAP

Image Formats - `raw` vs. `QCOW2`. As previously mentioned, Glance supports a variety of image formats; however `raw` and `QCOW2` are the most common. While `QCOW2` does provide some benefits (supports copy-on-write, snapshots, dynamic expansion) over the `raw` format, it should be noted that when images are copied into Cinder volumes, they are converted into the `raw` format once stored on a NetApp backend.

Note

Use of the `QCOW2` image format is recommended for ephemeral disks due to its inherent benefits when taking instance snapshots. Use of the `raw` image format can be advantageous when Cinder volumes are used as persistent boot disks as a conversion from an alternate format to `raw` that would be performed by Cinder can be avoided. Both `raw` and `QCOW2` formats respond well to NetApp deduplication technology which is often utilized with Glance deployments.

NFS with Deduplication. Since there is a high probability of duplicate blocks in a repository of virtual machine images, NetApp highly recommends to enable deduplication on the FlexVol volume(s) where the images are stored. You can check the status of deduplication for a particular FlexVol volume by issuing the **volume efficiency show** command as seen below.

```
::> volume efficiency show -vserver demo-vserver -volume vol2

Vserver Name: demo-vserver
Volume Name: vol2
Volume Path: /vol/vol2
State: Disabled
Status: Idle
Progress: Idle for 00:19:53
Type: Regular
Schedule: sun-sat@0
Efficiency Policy Name: -
Blocks Skipped Sharing: 0
Last Operation State: Success
Last Success Operation Begin:
Thu Nov 21 14:19:23 UTC 2013
Last Success Operation End:
Thu Nov 21 14:20:40 UTC 2013
Last Operation Begin:
Thu Nov 21 14:19:23 UTC 2013
Last Operation End:
Thu Nov 21 14:20:40 UTC 2013
Last Operation Size: 0B
```



```
Last Operation Error: -
Changelog Usage: 0%
Logical Data Size: 224KB
Logical Data Limit: 640TB
Logical Data Percent: 0%
Queued Job: -
Stale Fingerprint Percentage: 0
Compression: false
Inline Compression: false
Incompressible Data Detection: false
Constituent Volume: false
Compression Quick Check File Size: 524288000

::> volume efficiency on -vserver demo-vserver -volume vol2
Efficiency for volume "vol2" of Vserver "demo-vserver" is enabled.
Already existing data could be processed by running
"volume efficiency start -vserver demo-vserver -volume vol2
-scan-old-data true".
```

Enhanced Instance Creation

NetApp contributed a capability to enhance instance creation which focuses on booting tenant-requested VM instances by OpenStack Compute Service (Nova) using persistent disk images in the shortest possible time and in the most storage capacity efficient manner possible. This Enhanced Persistent Instance Creation feature (sometimes referred to as Rapid Cloning) is achieved by leveraging NetApp FlexClone technology, as well as the NetApp Copy Offload tool. The Enhanced Instance Creation feature can significantly decrease the time elapsed when the Nova service is fulfilling image provisioning and boot requests.

The NetApp Copy Offload tool was added in the Icehouse release to enable Glance images to be efficiently copied to a destination Cinder volume. When the Cinder and Glance are configured to use the NetApp NFS Copy Offload tool, a controller-side copy is attempted before reverting to downloading the image from Glance. This improves image provisioning times while reducing the consumption of bandwidth and CPU cycles on the host(s) running Glance and Cinder. This is due to the copy operation being performed completely within the storage cluster.

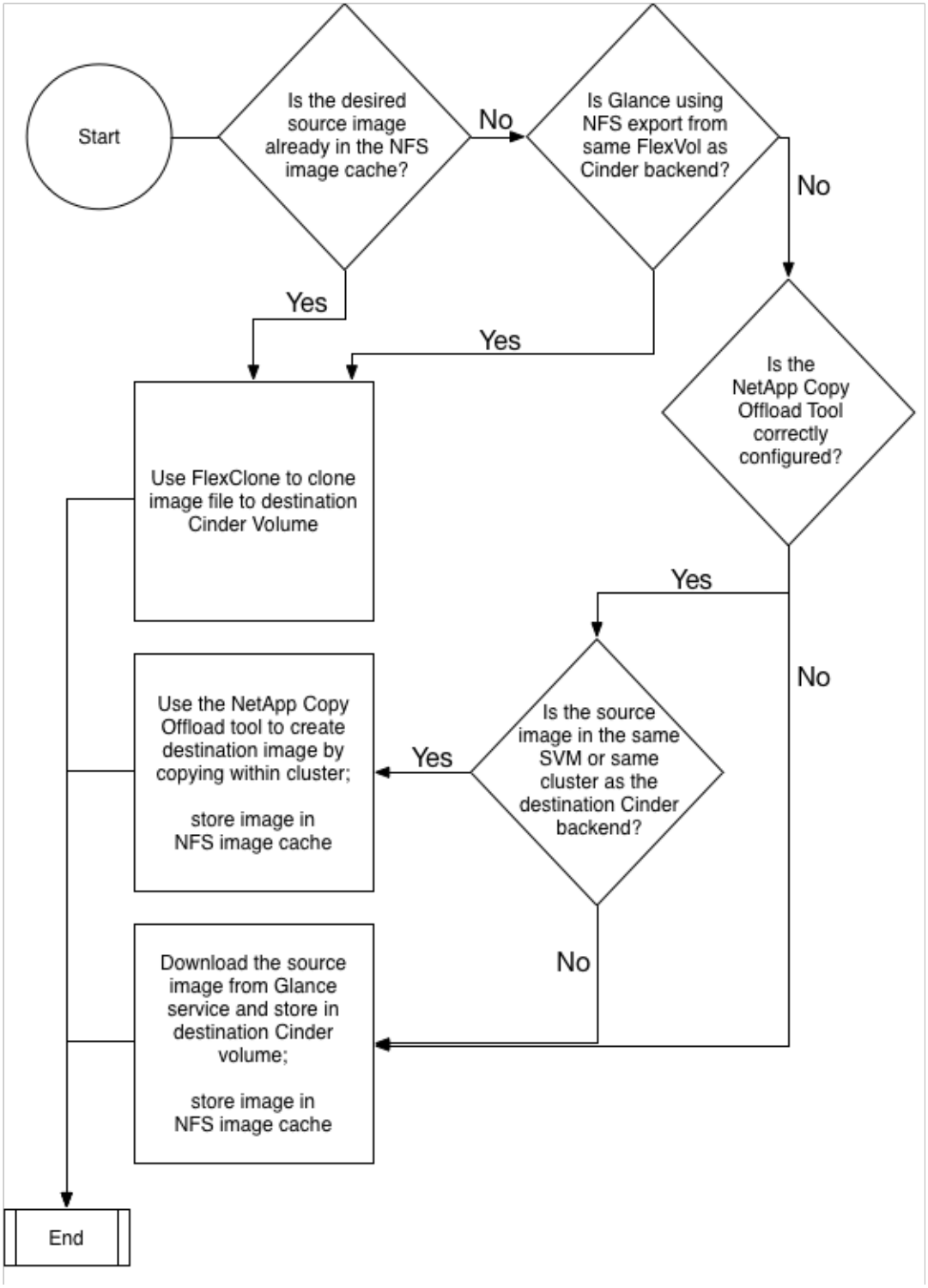
Requirements

The NetApp Copy Offload tool requires that:

- The storage system must have Data ONTAP v8.2 or greater installed.
- To configure the copy offload workflow, enable NFS and export it from the SVM.
- The vStorage feature must be enabled on each storage virtual machine (SVM, also known as a Vserver) that is permitted to interact with the copy offload client. To set this feature, you can use the `nfs modify -vserver` *`openstack_vs1`* `-vstorage enabled –v4.0 enabled` CLI command.

Figure 5.1, "Enhanced Instance Creation Flowchart" describes the workflow associated with the Enhanced Instance Cloning capability of the NetApp driver.

Figure 5.1. Enhanced Instance Creation Flowchart



Note

In the second decision point in the flowchart described in Figure 5.1, "Enhanced Instance Creation Flowchart", Cinder determines if the source image from Glance and the destination volume would reside in the same FlexVol volume. This can be achieved by creating a directory structure within the NFS export to segment the Glance images from Cinder volumes, and appropriately setting the `filesystem_datastore_dir` and `nfs_shares_config`. This configuration could also be used in conjunction with a single NetApp InfiniVol volume.

In order to take advantage of the Enhanced Instance Creation feature, there are several configuration options that must be appropriately set in both Cinder and Glance.

Glance. Modify the glance configuration file `/etc/glance/glance-api.conf` as follows:

- Set the `default_store` configuration option to `file`.
- Set the `filesystem_store_datadir` configuration option in the `[glance_store]` stanza to the path to the NFS export from the desired FlexVol volume.
- Set the `filesystem_store_file_perm` configuration option in the `[glance_store]` stanza to be the file permissions to be assigned to new Glance images; alternatively, make sure that the effective user of the `cinder-volume` process has access to read Glance images from the NFS export (e.g. add the `cinder` user into the `glance` group).
- Set the `show_image_direct_url` configuration option to `True` in the `[default]` stanza.
- Set the `show_multiple_locations` configuration option to `True` in the `[default]` stanza.
- Set the `filesystem_store_metadata_file` configuration option in the `[glance_store]` stanza to point to a metadata file. The metadata file should contain a JSON object that contains the correct information about the NFS export used by Glance, similar to:

```
{
    "id": "NetApp1",
    "share_location": "nfs://192.168.0.1/myGlanceExport",
    "mountpoint": "/var/lib/glance/images",
    "type": "nfs"
}
```

Cinder. Modify the cinder configuration file `/etc/cinder/cinder.conf` as follows:

- Set the `netapp_copyoffload_tool_path` configuration option in Cinder (under the appropriate backend stanza) to the path to the NetApp Copy Offload binary as installed on the system running **cinder-volume**.
- Set the `glance_api_version` configuration option to `2`.

Tip

Leveraging the "boot from image (creates a new volume)" option in Nova, you can leverage the enhanced instance creation capabilities described previously. Normally volumes created as a result of this option are persistent beyond the life of the instance. However, you can select the "delete on terminate" option in combination with the "boot from image (creates a new volume)" option to create an ephemeral volume while still leveraging the enhanced instance creation capabilities described previously. This can provide a significantly faster provisioning and boot sequence than the normal way that ephemeral disks are provisioned, where a copy of the disk image is made from Glance to local storage on the hypervisor node where the instance resides.

There are three tunable parameters within the Cinder driver configuration that can affect the behavior of how often space utilized by the NFS image cache managed by the NetApp unified driver is reclaimed for other uses: namely, `thres_avl_size_perc_start`, `thres_avl_size_perc_stop`, and `expiry_thres_minutes`. For more information on these parameters, refer to Table 4.10, "Configuration options for clustered Data ONTAP with NFS".

5.4. Configuration

5.4.1. Glance

When the file storage backend is used, the `filesystem_store_datadir` configuration option in `glance-api.conf` declares the directory Glance uses to store images (relative to the node running the **glance-api** service).

```
$ #for RHEL/CentOS/Fedora derived distributions
$ sudo openstack-config --get /etc/glance/glance-api.conf \
DEFAULT filesystem_store_datadir
/var/lib/glance/images/

$ #for Ubuntu derived distributions
$ sudo cat /etc/glance/glance-api.conf|grep \
filesystem_store_datadir|egrep -v "^#.*"
filesystem_store_datadir=/var/lib/glance/images/
```

5.4.2. Configuration of Glance with NetApp FAS

By specifying the value of `filesystem_store_datadir` to be a directory that is the mount point for an NFS share that is served from a NetApp FlexVol volume, you can have a single filesystem that can be mounted from one or more **glance-registry** servers.

Warning

The NFS mount for the `filesystem_store_datadir` is not managed by Glance; therefore, you must use the standard Linux mechanisms (e.g. **/**

`etc/fstab` or NFS automounter) to ensure that the NFS mount exists before Glance starts.

Tip

Be sure to refer to the Clustered Data ONTAP NFS Best Practices and Implementation Guide [http://www.netapp.com/us/system/pdf-reader.aspx?pdfuri=tcm:10-61288-16&m=tr-4067.pdf] for information on how to optimally set up the NFS export for use with Glance, and NetApp Data Compression and Deduplication Deployment and Implementation Guide [http://www.netapp.com/us/system/pdf-reader.aspx?pdfuri=tcm:10-60107-16&m=tr-3958.pdf].

5.4.3. Configuration of Glance with NetApp E-Series and EF-Series

E-Series and EF-Series storage systems can alternatively be used as the backing store for Glance images. An E-Series volume should be created (with SANtricity specifying the desired RAID level and capacity) and then mapped to the Glance node. After the volume is visible to the host it is formatted with a file system, mounted, and a directory structure created on it. This directory path can be specified as the `filesystem_store_datadir` in the Glance configuration file `glance-api.conf`.

Steps:

1. Create the LUN from a disk pool or volume group using SANtricity and map it to the host. Assuming that the volume has been mapped to `/dev/sdc` on the host, create a partition on the volume and then create a filesystem on the partition (e.g. ext4):

```
fdisk /dev/sdc
mkfs.ext4 /dev/sdc1
mount /dev/sdc1 /mnt/sdc1
mkdir /mnt/sdc1/glanceImageStore
```

2. Edit the Glance configuration file `glance-api.conf` so that it contains the `filesystem_store_datadir` option, and ensure the value refers to the Glance image store directory created in the previous step:

```
filesystem_store_datadir=/mnt/sdc1/glanceImageStore
```


Chapter 6. OpenStack Shared File System Service (Manila)

6.1. Overview

The OpenStack Shared File System (Manila) service provides management of persistent shared file system resources. The Shared File System service was originally conceived as an extension to the Block Storage service (Cinder), but emerged as an official, independent project in the Grizzly release.

Manila is typically deployed in conjunction with other OpenStack services (e.g. Compute, Object Storage, Image, etc) as part of a larger, more comprehensive cloud infrastructure. This is not an explicit requirement, as Manila has been successfully deployed as a standalone solution for shared file system provisioning and lifecycle management.

Tip

As a management service, Manila controls the provisioning and lifecycle management of shared filesystems. It does not reside in the I/O (data) path between clients and the storage controller hosting the filesystem.

6.2. Key Concepts

Share

A Manila share is the fundamental resource unit allocated by the Shared File System service. It represents an allocation of a persistent, readable, and writable filesystem that can be accessed by OpenStack compute instances, or clients outside of OpenStack (depending on deployment configuration). The underlying connection between the consumer of the share and the Manila service providing the share can be achieved with a variety of protocols, including NFS and CIFS (protocol support is dependent on the Manila driver deployed and the selection of the end user).

Warning

A Manila share is an abstract storage object that may or may not directly map to a "share" concept from the underlying backend provider of storage.

Manila shares can be identified uniquely through a UUID assigned by the Manila service at the time of share creation. A Manila share may also be optionally referred to by a human-readable name, though this string is not guaranteed to be unique within a single tenant or deployment of Manila.

The actual capacity provisioned in support of a Manila share resides on a single Manila backend within a single resource pool.

Snapshot

A Manila snapshot is a point-in-time, read-only copy of a Manila share. Snapshots can be created from an existing Manila share that is operational regardless of whether a client has mounted the file system. A Manila snapshot can serve as the content source for a new Manila share when the Manila share is created with the *create from snapshot* option specified.

Backend

A Manila backend is the configuration object that represents a single provider of resource pools upon which provisioning requests for shared file systems may be fulfilled. A Manila backend communicates with the storage system through a Manila driver. Manila supports multiple backends to be configured and managed simultaneously (even with the same Manila driver).

Note

A single Manila backend may be defined in the `[DEFAULT]` stanza of `manila.conf`; however, NetApp recommends that the `enabled_share_backends` configuration option be set to a comma-separated list of backend names, and each backend name have its own configuration stanza with the same name as listed in the `enabled_share_backends` option. Refer to Section 6.6.1, “Manila” for an example of the use of this option.

Storage Pools

With the Kilo release of OpenStack, Manila has introduced the concept of "storage pools". The backend storage may present one or more logical storage resource pools from which Manila will select as a storage location when provisioning shares. In releases prior to Kilo, NetApp's Manila drivers contained some logic that determined which aggregate a Manila share would be placed into; with the introduction of pools, all scheduling logic is performed completely within the Manila scheduler.

Important

For NetApp's Manila drivers, a Manila storage pool is an aggregate defined within Data ONTAP.

Driver

A Manila driver is a particular implementation of a Manila backend that maps the abstract APIs and primitives of Manila to appropriate constructs within the particular storage solution underpinning the Manila backend.

Caution

The use of the term "driver" often creates confusion given common understanding of the behavior of “device drivers” in operating systems.

The term can connote software that provides a data I/O path. In the case of Manila driver implementations, the software provides provisioning and other manipulation of storage devices but does not lay in the path of data I/O. For this reason, the term "driver" is often used interchangeably with the alternative (and perhaps more appropriate) term “provider”.

Share Type

A Manila share type is an abstract collection of criteria used to characterize Manila shares. They are most commonly used to create a hierarchy of functional capabilities that represent a tiered level of storage services; for example, a cloud administrator might define a `premium` share type that indicates a greater level of performance than a `basic` share type, which would represent a best-effort level of performance.

The collection of criteria is specified as a list of key/value pairs, which are inspected by the Manila scheduler when determining which resource pools are able to fulfill a provisioning request. Individual Manila drivers (and subsequently Manila backends) may advertise arbitrary key/value pairs (also referred to as capabilities) to the Manila scheduler for each pool, which are then compared against share type definitions when determining which pool will fulfill a provisioning request.

Extra Spec. An extra spec is a key/value pair, expressed in the style of `key=value`. Extra specs are associated with Manila share types, so that when users request shares of a particular share type, the shares are created on pools within storage backends that meet the specified criteria.

Note

The list of default capabilities that may be reported by a Manila driver and included in a share type definition include:

- `share_backend_name`: The name of the backend as defined in `manila.conf`
- `vendor_name`: The name of the vendor who has implemented the driver (e.g. `NetApp`)
- `driver_version`: The version of the driver (e.g. `1.0`)
- `storage_protocol`: The protocol used by the backend to export block storage to clients (e.g. `NFS_CIFS`)

For a table of NetApp supported extra specs, refer to Table 6.6, “NetApp supported Extra Specs for use with Manila Share Types”.

Share Access Rules

Share access rules define which clients can access a particular Manila share. Access rules can be declared for NFS shares by listing the valid IP networks (using CIDR notation) which should have access to the share. In the case of CIFS shares, the Windows security identifier (SID) can be specified.

Important

For NetApp's Manila drivers, share access is enforced through the use of export policies configured within the NetApp storage controller.

Security Services

Security services are the concept in Manila that allow Finer-grained client access rules to be declared for authentication or authorization to access share content. External services including LDAP, Active Directory, Kerberos can be declared as resources that should be consulted when making an access decision to a particular share. Shares can be associated to multiple security services.

Share Networks

A share network is an object that defines a relationship between a tenant's network/subnet (as defined in an OpenStack network service (Neutron or Nova-network)) and the Manila shares created by the same tenant; that is, a tenant may find it desirable to provision shares such that only instances connected to a particular OpenStack-defined network have access to the share.

Note

As of Kilo, share networks are no longer required arguments when creating shares.

Share Servers

A share server is a logical entity that manages the shares that are created on a specific share network. Depending on the implementation of a specific Manila driver, a share server may be a configuration object within the storage controller, or it may represent logical resources provisioned within an OpenStack deployment that are used to support the data path used to access Manila shares.

Share servers interact with network services to determine the appropriate IP addresses on which to export shares according to the related share network. Manila has a pluggable network model that allows share servers to work with OpenStack environments that have either Nova-Network or Neutron deployed. In addition, Manila contains an implementation of a standalone network plugin which manages a pool of IP addresses for shares that are defined in the `manila.conf` file. For more details on how share servers interact with the various network services, please refer to Figure 6.2, “Manila Workflow - Share Creation with Share Servers” and Figure 6.3, “Manila Workflow - Share Creation without Share Servers”.

Important

Within the NetApp Manila driver, a share server is defined to be a storage virtual machine (also known as a Vserver) within the clustered Data ONTAP system that is associated with a particular backend. The NetApp Manila driver has two operating "modes":

1. One that supports the dynamic creation of share servers (SVMs) for each share network - this is referred to as the NetApp Manila driver with share server management.

2. One that supports the reuse of a single share server (SVM) for all shares hosted from a backend - this is referred to as the NetApp Manila driver without share server management.

6.3. Process Structure

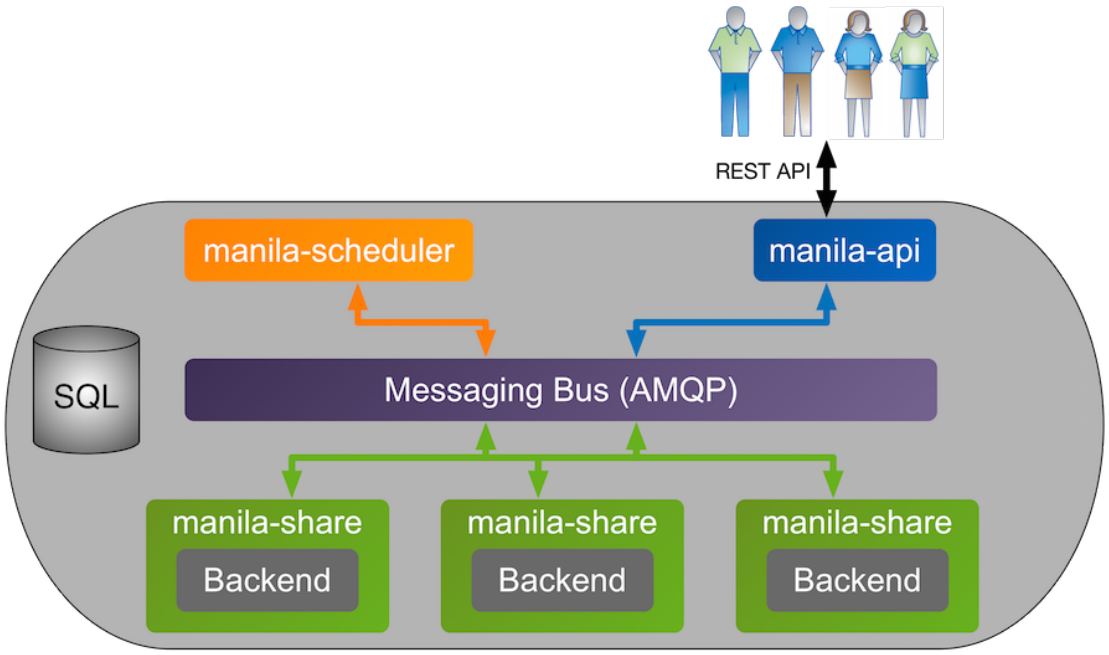
There are three processes that make up the Manila service:

- `manila-api` is a WSGI application that accepts and validates REST (JSON) requests from clients and routes them to other Manila processes as appropriate over AMQP.

- `manila-scheduler` determines which backend should serve as the destination for a share creation request. It maintains non-persistent state for pools and backends (e.g. available capacity, capabilities, and supported extra specs) that can be leveraged when making placement decisions. The algorithm utilized by the scheduler can be changed through Manila configuration.

- `manila-share` accepts requests from other Manila processes and serves as the operational container for Manila drivers. This process is multi-threaded and typically has one thread of execution per Manila backend as defined in the Manila configuration file.

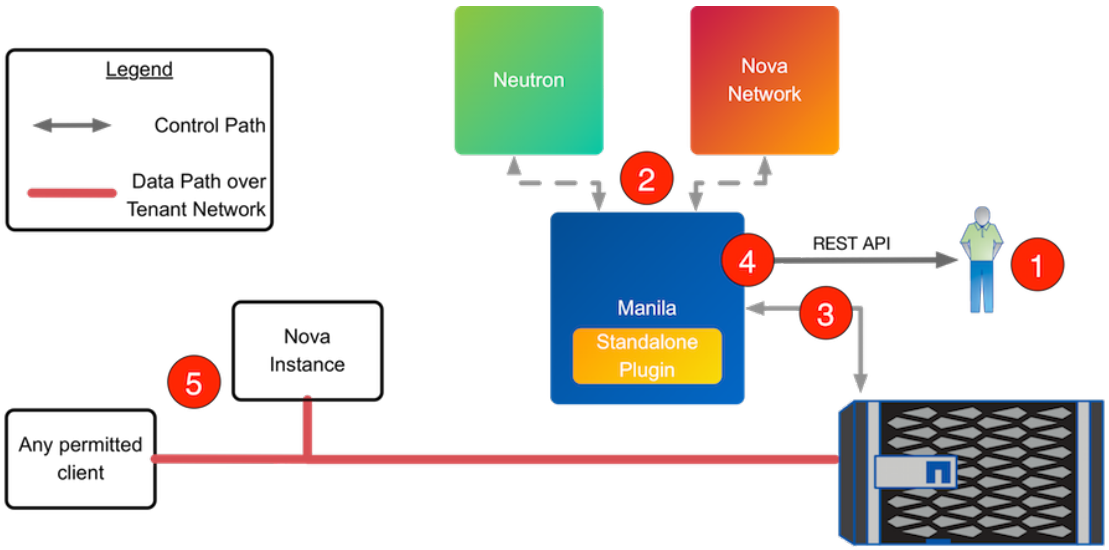
Figure 6.1. Manila Processes Concept Diagram



Share Creation Workflow - With Share Servers

The following section walks through the steps that occur when a user requests the creation of a new share from Manila, and a backend is selected that uses share servers.

Figure 6.2. Manila Workflow - Share Creation with Share Servers



1. Client issues request to create share through invoking REST API (client may use `python-manilaclient` CLI utility).

   The `manila-api` and `manila-scheduler` processes perform the following tasks:

   a. `manila-api` process validates request, user credentials; once validated, puts message onto AMQP queue for processing.

   b. `manila-share` process takes message off of queue, sends message to manila-scheduler to determine which pool and backend to provision share into.

   c. `manila-scheduler` process takes message off of queue, generates candidate list of resource pools based on current state and requested share criteria (size, availability zone, share type (including extra specs)).

   d. `manila-share` process reads response message from `manila-scheduler` from queue; iterates through candidate list by invoking backend driver methods for corresponding pools until successful.

2. The share manager associated with the backend selected by the `manila-scheduler` calls out to the network service associated with the backend (either the standalone plugin, Neutron, or Nova Network; defined through the appropriate stanza in `manila.conf`) to get the required information (including share IP address, network segmentation ID, etc.).

3. If selected by the scheduler, NetApp's Manila driver creates requested share (and, if necessary, a share server) through interactions with storage subsystem (dependent on configuration and protocol).

4. `manila-share` process creates share metadata and posts response message to AMQP queue.

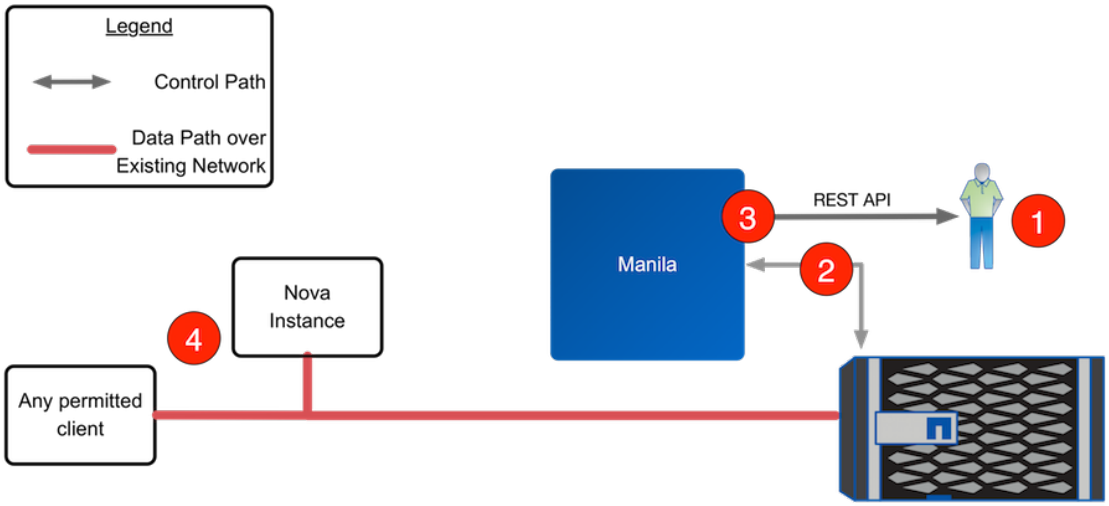
   `manila-api` process reads response message from queue and responds to client with share ID information.

5. After a share is created and exported by the backend, client uses ID information to request updated share details and uses export information from response to mount share (using protocol-specific commands).

Share Creation Workflow - Without Share Servers

The following section walks through the steps that occur when a user requests the creation of a new share from Manila and a backend is selected that does not use share servers.

Figure 6.3. Manila Workflow - Share Creation without Share Servers



1. Client issues request to create share through invoking REST API (client may use `python-manilaclient` CLI utility).

   The `manila-api` and `manila-scheduler` processes perform the following tasks:

   a. `manila-api` process validates request, user credentials; once validated, puts message onto AMQP queue for processing.

   b. `manila-share` process takes message off of queue, sends message to manila-scheduler to determine which pool and backend to provision share into.

   c. `manila-scheduler` process takes message off of queue, generates candidate list of resource pools based on current state and requested share criteria (size, availability zone, share type (including extra specs)).

   d. `manila-share` process reads response message from `manila-scheduler` from queue; iterates through candidate list by invoking backend driver methods for corresponding pools until successful.

2. If selected by the scheduler, NetApp's Manila driver creates requested share through interactions with storage subsystem (dependent on configuration and protocol).

   Without the existence of a share server, NetApp's Manila driver will export shares through the data LIFs that exist within the SVM that is scoped to the Manila backend.

3. `manila-share` process creates share metadata and posts response message to AMQP queue.

   `manila-api` process reads response message from queue and responds to client with share ID information.

4. After a share is created and exported by the backend, client uses ID information to request updated share details and uses export information from response to mount share (using protocol-specific commands).

Share Access Workflow

The following section walks through the steps that are required in order for any client to access a shared filesystem.

1. Client issues request to enable access to a share through invoking Manila REST API (client may use `python-manilaclient` CLI utility), specifying the type of access (either `IP` or `user`) and the value (either IP address or IP network address in CIDR notation or user name in standard Windows notation).

2. `manila-api` process validates request, user credentials; once validated, posts message to share manager over AMQP.

3. `manila-share` reads message from queue, invokes Manila driver corresponding to share to be attached.

4. NetApp Manila driver creates appropriate export policies for the share and access type provided.

5. `manila-share` process posts response information to `manila-api` process via AMQP queue.

6. `manila-api` process reads response message from `manila-share` from queue; passes connection information in RESTful response to caller.

6.4. API Overview

This section describes some of the most commonly used Manila API calls and their corresponding CLI commands. It is not meant to be a comprehensive list that is representative of all functionality present in Manila; for more information, please refer to the help text from running **manila help**.

Share API

Table 6.1, “Manila API Overview - Share” specifies the valid operations that can be performed on Manila shares. Please note that Manila shares are identified as CLI command arguments by either their names or UUID.

Table 6.1. Manila API Overview - Share

Operation	CLI Command	Description
Create	**manila create**	Create a Manila share of specified size; optional name, availability zone, share type, share network, source snapshot
Delete	**manila delete**	Delete an existing Manila share; the **manila force-delete** command may be required if the Manila share is in an error state
Edit	**manila metadata**	Set or unset metadata on a Manila share
List	**manila list**	List all Manila shares
Show	**manila show**	Show details about a Manila share

Share Access API

Table 6.2, "Manila API Overview - Share Access" specifies the valid access operations that can be performed on Manila shares. Please note that Manila shares are identified as CLI command arguments by either their names or UUID.

Table 6.2. Manila API Overview - Share Access

Operation	CLI Command	Description
Allow	**manila access-allow**	Allow access to the specified share for the specified access type and value (IP address or IP network address in CIDR notation or Windows user name).
Deny	**manila access-deny**	Deny access to the specified share for the specified access type and value (IP address or IP network address in CIDR notation or Windows user name).
List	**manila access-list**	List all Manila share access rules

Snapshot API

Table 6.3, "Manila API Overview - Snapshot" specifies the valid operations that can be performed on Manila snapshots. Please note that Manila snapshots are identified as CLI command arguments by either their display name or UUID.

Table 6.3. Manila API Overview - Snapshot

Operation	CLI Command	Description
Create	**manila snapshot-create**	Create a Manila snapshot of a specific Manila share
Delete	**manila snapshot-delete**	Delete a Manila snapshot
List	**manila snapshot-list**	List all Manila snapshots
Rename	**manila snapshot-rename**	Change the display-name of a Manila snapshot
Reset State	**manila snapshot-reset-state**	Reset the state of a Manila snapshot
Show	**manila snapshot-show**	Show details about a Manila snapshot

Share Type API

Table 6.4, "Manila API Overview - Share Type" specifies the valid operations that can be performed on Manila share types. Please note that Manila share types

are identified as CLI command arguments by either their display name or UUID. Creation or deletion of share types normally requires administrative privileges.

Table 6.4. Manila API Overview - Share Type

Operation	CLI Command	Description
Create	**manila type-create**	Create a Manila share type
Delete	**manila type-delete**	Delete a Manila share type
List	**manila type-list**	List existing Manila share type

Share Type Extra Specs API

Table 6.5, "Manila API Overview - Share Type Extra Specs" specifies the valid operations that can be performed on Manila share type extra specs. Please note that Manila share type extra specs are properties of Manila share types and are identified by their parent object. Modifying extra specs or viewing the contents of a share type normally requires administrative privileges.

Table 6.5. Manila API Overview - Share Type Extra Specs

Operation	CLI Command	Description
List extra specs	**manila extra-specs-list**	Print the values of extra specs assigned to Manila share types
Set extra specs	**manila type-key *stype* set**	Assign extra specs to Manila share type
Unset extra specs	**manila type-key *stype* unset**	Remove extra specs from Manila share type

6.5. Theory of Operation & Deployment Choices

Construct Mappings between Manila and Clustered Data ONTAP

Manila backends and Clustered Data ONTAP. Storage Virtual Machines (SVMs, formerly known as Vservers) contain one or more FlexVol volumes and one or more LIFs through which they serve data to clients.

SVMs securely isolate the shared virtualized data storage and network, and each SVM appears as a single dedicated storage virtual machine to clients. Each SVM has a separate administrator authentication domain and can be managed independently by its SVM administrator.

In a cluster, SVMs facilitate data access. A cluster must have at least one SVM to serve data. SVMs use the storage and network resources of the cluster. However, the shares and LIFs are exclusive to the SVM. Multiple SVMs can coexist in a single cluster without being bound to any node in a cluster. However, they are bound to the physical cluster on which they exist.

Manila shares and FlexVol volumes. Data ONTAP FlexVol volumes (commonly referred to as volumes) and OpenStack File Share Storage shares (commonly referred to as Manila shares) are semantically analogous. A FlexVol volume is a

container of logical data elements (for example: files, Snapshot™ copies, clones, LUNs, et cetera) that is abstracted from physical elements (for example: individual disks, and RAID groups).

Manila snapshots versus NetApp Snapshots. A NetApp Snapshot copy is a point-in-time file system image. Low-overhead NetApp Snapshot copies are made possible by the unique features of the WAFL storage technology that is part of Data ONTAP. The high performance of the NetApp Snapshot makes it highly scalable. A NetApp Snapshot takes only a few seconds to create — typically less than one second, regardless of the size of the share or the level of activity on the NetApp storage system. After a Snapshot copy has been created, changes to data objects are reflected in updates to the current version of the objects, as if NetApp Snapshot copies did not exist. Meanwhile, the NetApp Snapshot version of the data remains completely stable. A NetApp Snapshot incurs no performance overhead; users can comfortably store up to 255 NetApp Snapshot copies per FlexVol volume, all of which are accessible as read-only and online versions of the data.

Important

Since NetApp Snapshots are taken at the FlexVol level, they can and are directly leveraged within an Manila context, as a user of Manila requests a snapshot be taken of a particular Manila share.

Deployment Choice: Direct versus Intermediated

The NetApp Manila driver can operate in two independent modes: a *direct* mode where the Manila processes directly interact with NetApp FAS storage systems, and an *intermediated* mode where the Manila processes interact with an additional software entity that issues provisioning and management requests on behalf of Manila.

OnCommand® Workflow Automator. OnCommand® Workflow Automator (WFA) is a flexible framework that provides automation for storage-related tasks, customization, scripting capabilities, and integration with higher-order IT systems such as orchestration software through web services.

While WFA can be utilized in conjunction with the NetApp unified Manila driver, a deployment of Manila and WFA does introduce additional complexity, management entities, and potential points of failure within a cloud architecture. If you have an existing set of workflows that are written within the WFA framework, and are looking to leverage them in lieu of the default provisioning behavior of the Manila driver operating directly against a FAS system, then it may be desirable to use the intermediated mode.

Recommendation

Unless you have a significant existing investment with OnCommand Workflow Automator that you wish to leverage in an OpenStack deployment, it is recommended that you start with the *direct* mode of operation when deploying Manila with a NetApp clustered Data ONTAP system.

Deployment Choice: Utilizing Share Servers

Manila offers the capability for shares to be accessible through tenant-defined networks (defined within Neutron). This is achieved by defining a share network object, which provides the relationship to the Neutron network and subnet from which an IP address should be allocated, as well as configured on the backend storage (along with the appropriate segmentation approach (e.g. VLAN, VXLAN, GRE, etc).

Offering this capability to end users places certain requirements on storage platforms that are integrated with Manila to be able to dynamically configure themselves. Share servers are an object defined by Manila that manages the relationship between share networks and shares. In the case of the reference driver implementation, a share server corresponds to an actual Nova instance that provides the file system service, with raw capacity provided through attached Cinder block storage volumes. In the case of the Manila driver for NetApp clustered Data ONTAP, a share server corresponds to a storage virtual machine (SVM), also referred to as a Vserver.

Note

One share server is created by Manila for each share network that has shares associated with it.

Important

When deploying Manila with NetApp clustered Data ONTAP without share server management, NetApp requires that each Manila backend refer to a single SVM within a cluster through the use of the `netapp_vserver` configuration option.

With Share Servers. Within the clustered Data ONTAP driver with share server support, a storage virtual machine will be created for each share server. While this can provide some advantages with regards to secure multitenancy and integration with a variety of network services within OpenStack, care must be taken to ensure that the scale limits are enforced through Manila quotas. It is a documented best practice to not exceed 200 SVMs running on a single cluster at any given time to ensure consistent performance and responsive management operations.

Without Share Servers. With the clustered Data ONTAP driver without share server support, data LIFs are reused and the provisioning of new Manila shares (i.e. FlexVol volumes) is limited to the scope of a single SVM.

Using Manila Share Types to Create a Storage Service Catalog

The Storage Service Catalog (SSC) is a concept that describes a set of capabilities that enables efficient, repeated, and consistent use and management of storage resources by the definition of policy-based services and the mapping of those services to the backend storage technology. It is meant to abstract away the

actual technical implementations of the features at a storage backend into a set of simplified configuration options.

The storage features are organized or combined into groups based on the customer needs to achieve a particular scenario or use case. Based on the catalog of the storage features, intelligent provisioning decisions are made by infrastructure or software enabling the storage service catalog. In OpenStack, this is achieved together by the Manila filter scheduler and the NetApp driver by making use of share type extra-specs support together with the filter scheduler.

When the NetApp unified driver is used with a clustered Data ONTAP storage system, you can leverage extra specs with Manila share types to ensure that Manila shares are created on storage backends that have certain properties (e.g. thin provisioning, disk type, RAID type) configured.

Extra specs are associated with Manila share types, so that when users request shares of a particular share type, they are created on storage backends that meet the list of requirements (e.g. available space, extra specs, etc). You can use the specs in Table 6.6, "NetApp supported Extra Specs for use with Manila Share Types" later in this section when defining Manila share types with the **manila type-key** command.

Table 6.6. NetApp supported Extra Specs for use with Manila Share Types

Extra spec	Type	Description
`netapp_raid_type`	String	Limit the candidate aggregate (pool) list based on one of the following raid types: `raid4`, `raid_dp`.
`netapp_disk_type`	String	Limit the candidate aggregate (pool) list based on one of the following disk types: `ATA`, `BSAS`, `EATA`, `FCAL`, `FSAS`, `LUN`, `MSATA`, `SAS`, `SATA`, `SCSI`, `XATA`, `XSAS`, or `SSD`.
`netapp:thin_provisioned`	Boolean	Enable thin provisioning (a space guarantee of `None`) on the share.
`netapp:snapshot_policy`	String	Apply the specified snapshot policy to the created FlexVol volume. *Note that the snapshots created by applying a policy will not have corresponding Manila snapshot records.*
`netapp:language`	String	Apply the specified language to the FlexVol volume that corresponds to the Manila share. The language of the FlexVol volume determines the character set Data ONTAP uses to display file names and data for that volume. The default value for the language of the volume is the language of the SVM.
`netapp:max_files`	String	Change the maximum number of files for the FlexVol volume that corresponds to the Manila share. By default, the maximum number of files is proportional to the size of the share. This spec can be used to increase the number of files for very large shares (greater than 1TB), or to place a smaller limit on the number of files on a given share.

Caution

When using the Manila driver without share server management, you can specify a value for the `netapp_login` option that only has SVM administration privileges (rather than cluster administration privileges); you should note some advanced features of the driver may not work and you may see warnings in the Manila logs, unless

appropriate permissions are set. See the section called "Account Permission Considerations" for more details on the required access level permissions for an SVM admin account.

6.6. Configuration

6.6.1. Manila

Manila is configured by changing the contents of the `manila.conf` file and restarting all of the Manila processes. Depending on the OpenStack distribution used, this may require issuing commands such as **service openstack-manila-api restart** or **service manila-api restart**.

`manila.conf`

The `manila.conf` file contains a set of configuration options (one per line), specified as `option_name=value`. Configuration options are grouped together into a stanza, denoted by `[stanza_name]`. There must be at least one stanza named `[DEFAULT]` that contains configuration parameters that apply generically to Manila (and not to any particular backend). Configuration options that are associated with a particular Manila backend should be placed in a separate stanza.

Note

While it is possible to specify driver-specific configuration options within the `[DEFAULT]` stanza, you are unable to define multiple Manila backends within the `[DEFAULT]` stanza. NetApp strongly recommends that you specify driver-specific configuration in separate named stanzas, being sure to list the backends that should be enabled as the value for the configuration option `enabled_share_backends`; for example:

```
enabled_share_backends=clusterOne,clusterTwo
```

The `enabled_share_backends` option should be specified within the `[DEFAULT]` configuration stanza.

6.6.2. Manila Network Plugins

As described in Figure 6.2, "Manila Workflow - Share Creation with Share Servers", there are a set of network plugins that provide for a variety of integration approaches with the network services that are available with OpenStack. These plugins should only be utilized with the NetApp clustered Data ONTAP driver with share server management.

These are the valid network plugins as of the Kilo release:

- *Standalone Network Plugin*: IP settings (address range, subnet mask, gateway, version) are all defined through configuration options within the driver-specific stanza.

- *Nova Network Plugin: Simple*: Use a single Nova network ID for all share servers; the ID of the Nova Network to be leveraged is specified through a configuration option to be defined within the driver-specific configuration stanza.
- *Nova Network Plugin: Configurable*: Allow end-users of Manila to create share networks that map to different Nova networks when defining share networks. Values for segmentation protocol, IP address, netmask, protocol, and gateway are obtained from Nova-network when creating a new share server. Default values for network ID and subnet ID can be specified through configuration options within the driver-specific stanza; note that values specified by end users when defining share networks will take precedence over values declared in configuration.
- *Neutron Network Plugin*: Use Neutron networks and subnets when defining share networks. Values for segmentation protocol, IP address, netmask, protocol, and gateway are obtained from Neutron when creating a new share server. Default values for network ID and subnet ID can be specified through configuration options within the driver-specific stanza; note that values specified by end users when defining share networks will take precedence over values declared in configuration.

The network plugin is chosen by setting the value of the `network_api_class` configuration option within the driver-specific stanza of the `manila.conf` configuration file.

Standalone Network Plugin

To set up the standalone network plugin, the following options should be added to the driver-specific stanza within the Manila configuration file (`manila.conf`):

```
network_api_class=manila.network.StandaloneNetworkPlugin
standalone_network_plugin_allowed_ip_ranges=10.0.0.10-10.0.0.254
standalone_network_plugin_ip_version=4
standalone_network_plugin_segmentation_id=314
standalone_network_plugin_mask=255.255.255.0
standalone_network_plugin_gateway=10.0.0.1
```

Table 6.7, "Configuration options for Standalone Network Plugin" lists the configuration options available for the standalone network plugin:

Table 6.7. Configuration options for Standalone Network Plugin

Option	Type	Default Value	Description
`standalone_network_plugin_gateway`	Required		Specify the gateway IP address that should be configured on the data LIF through which the share is exported.
`standalone_network_plugin_mask`	Required		Specify the subnet mask that should be configured on the data LIF through which the share is exported. You can specify the CIDR suffix (without the slash, e.g. `24`) or the full netmask (e.g. `255.255.255.0`).
`standalone_network_plugin_segmentation_id`	Optional		Specify the segmentation ID that should be assigned to data LIFs through which shares can be exported.

Option	Type	Default Value	Description
`standalone_network_plugin_allowed_ip_ranges`	Optional		Specify the range of IP addresses that can be used on data LIFs through which shares can be exported. An example of a valid range would be `10.0.0.10-10.0.0.254`. If this value is not specified, the entire range of IP addresses within the network computed by applying the value of `standalone_network_plugin_mask` to the value of `standalone_network_plugin_gateway`. In this case, the broadcast, network, and gateway addresses are automatically excluded.
`standalone_network_plugin_ip_version`	Optional	4	Specify the IP version for the network that should be configured on the data LIF through which the share is exported. Valid values are 4 or 6.

Nova Network - Single Network Plugin

To set up the single Nova Network plugin, the following options should be added to the driver-specific stanza within the Manila configuration file (`manila.conf`):

```
network_api_class=manila.network.NovaSingleNetworkPlugin
nova_single_network_plugin_net_id=97fb9f7e-4ffe-4900-8dba-c6d4251e588e
```

Table 6.8, "Configuration options for Nova Network Plugin" lists the configuration options available for the Nova network plugin:

Table 6.8. Configuration options for Nova Network Plugin

Option	Type	Default Value	Description
`nova_single_network_plugin_net_id`	Optional		Specify the ID of the default Nova network that will be used with share servers.

Nova Network - Configurable Network Plugin

To set up the configurable Nova Network plugin, a single option should be added to the driver-specific stanza within the Manila configuration file (`manila.conf`):

```
network_api_class=manila.network.NovaNetworkPlugin
```

Neutron Network Plugin

To set up the Neutron network plugin, the following options should be added to the driver-specific stanza within the Manila configuration file (`manila.conf`):

```
network_api_class=manila.network.NeutronNetworkPlugin
neutron_net_id=37fb9f7e-4ffe-4900-8dba-c6d4251e588e
```



```
neutron_subnet_id=447732be-4cf2-42b0-83dc-4b6f4ed5368c
```

Table 6.9, "Configuration options for Neutron Network Plugin" lists the configuration options available for the Neutron network plugin:

Table 6.9. Configuration options for Neutron Network Plugin

Option	Type	Default Value	Description
`neutron_net_id`	Optional		Specify the ID of a Neutron network from which a port should be created if one is not specified when a share network is created.
`neutron_subnet_id`	Optional		Specify the ID of a Neutron subnet from which a port should be created if one is not specified when a share network is created.

6.6.3. NetApp Data ONTAP Drivers for OpenStack File Share Storage (Manila)

NetApp's Manila drivers for clustered Data ONTAP (with or without the management of share servers) are offered in a single, unified driver.

Where to Obtain the Drivers

NetApp's contribution strategy involves adding all new capabilities directly into the upstream OpenStack Shared File System service repositories, so all the features are available regardless of which distribution you choose when deploying OpenStack. Bug fixes are delivered into the appropriate branches that represent the different releases of OpenStack (e.g. `trunk`, `stable/juno`, `stable/icehouse`, etc).

On occasion, it may be necessary for NetApp to deliver capability to a previous release of OpenStack that can not be accepted in the upstream OpenStack repositories. In that case, we post the capability at the NetApp Github repository - accessible at https://github.com/NetApp/manila. Be sure to choose the branch from this repository that matches the release version of OpenStack you are deploying with. There will be a `README` file in the root of the repository that describes the specific changes that are merged into that repository beyond what is available in the upstream repository.

Multiple Deployment Options

A variety of OpenStack file share storage deployment options for NetApp clustered Data ONTAP based systems are available in the Kilo OpenStack release and involve making deployment choices between the presence or absence of management of share servers (SVM or Vservers) by the driver.

The following lists all of the individual options and subsequent sections are intended to offer guidance on which configuration options ought to be employed given varying use cases:

- NetApp clustered Data ONTAP without share server management

- NetApp clustered Data ONTAP with share server management

6.6.4. NetApp Unified Driver for Clustered Data ONTAP without Share Server management

The NetApp unified driver for clustered Data ONTAP without share server management is a driver interface from OpenStack Manila to NetApp clustered Data ONTAP storage controllers to accomplish provisioning and management of shared file systems within the scope of a single SVM (Vserver).

Configuration Options

To set up the NetApp clustered Data ONTAP driver without Share Server management, the following stanza should be added to the Manila configuration file (`manila.conf`):

```
[cdotSingleSVM] ❶
share_backend_name=cdotSingleSVM
share_driver = manila.share.drivers.netapp.common.NetAppDriver
driver_handles_share_servers=False ❷
netapp_storage_family=ontap_cluster
netapp_server_hostname=hostname
netapp_server_port=80
netapp_login=admin_username
netapp_password=admin_password
netapp_vserver=svm_name
netapp_transport_type=https
netapp_aggregate_name_search_pattern=^((?!aggr0).)*$
```

❶ Be sure that the value of the `enabled_share_backends` option in the `[DEFAULT]` stanza includes the name of the stanza you chose for the backend.

❷ The value of `driver_handles_share_servers` **MUST** be set to `False` if you want the driver to operate without managing share servers.

Table 6.10, "Configuration options for clustered Data ONTAP without Share Server management" lists the configuration options available for the unified driver for a clustered Data ONTAP deployment that does not manage share servers.

Table 6.10. Configuration options for clustered Data ONTAP without Share Server management

Option	Type	Default Value	Description
`driver_handles_share_servers`	Required		Denotes whether the driver should handle the responsibility of managing share servers. This must be set to `false` if the driver is to operate without managing share servers.
`netapp_server_hostname`	Required		The hostname or IP address for the storage system or proxy server. *The value of this option should be the IP address of either the cluster management LIF or the SVM management LIF.*

Option	Type	Default Value	Description
netapp_server_port	Optional		The TCP port to use for communication with the storage system or proxy server. If not specified, Data ONTAP drivers will use 80 for HTTP and 443 for HTTPS.
netapp_login	Required		Administrative user account name used to access the storage system.
netapp_password	Required		Password for the administrative user account specified in the **netapp_login** option.
netapp_transport_type	Required	http	Transport protocol for communicating with the storage system or proxy server. Valid options include **http** and **https**.
netapp_vserver	Required		This option specifies the storage virtual machine (previously called a Vserver) name on the storage cluster on which provisioning of shared file systems should occur. This parameter is required if the driver is to operate without managing share servers (that is, be limited to the scope of a single SVM).
netapp_storage_family	Required	ontap_cluster	The storage family type used on the storage system; valid values are **ontap_cluster** for clustered Data ONTAP.
netapp_volume_name_template	Optional	share_ %(share_id)s	This option specifies a string replacement template that is applied when naming FlexVol volumes that are created as a result of provisioning requests.
netapp_aggregate_name_search_pattern	Optional	(.*)	This option specifies a regular expression that is applied against all available aggregates related to the SVM specified in the **netapp_vserver** option. This filtered list will be reported to the Manila scheduler as valid pools for provisioning new shares.
netapp_trace_flags	Optional		This option is a comma-separated list of options (valid values include **method** and **api**) that controls which trace info is written to the Manila logs when the debug level is set to **True**.

Caution

If you specify an account in the **netapp_login** option that only has SVM administration privileges (rather than cluster administration privileges), some advanced features of the NetApp unified driver will not work and you may see warnings in the Manila logs. See the section called "Account Permission Considerations" for more details on the required access level permissions for an SVM admin account.

6.6.5. NetApp Unified Driver for Clustered Data ONTAP with Share Server management

The NetApp unified driver for clustered Data ONTAP with share server management is a driver interface from OpenStack Manila to NetApp clustered Data ONTAP storage controllers to accomplish provisioning and management of shared file systems across the scope of the entire cluster. This driver will create

a new storage virtual machine (SVM) for each share server that is requested by the Manila service. This driver also creates new data logical interfaces (LIFs) that provide access for clients on a specific share network to access shared file systems exported from the share server.

Configuration Options

To set up the NetApp clustered Data ONTAP driver with Share Server management, the following stanza should be added to the Manila configuration file (`manila.conf`):

```
[cdotMultipleSVM] ❶
share_backend_name=cdotMultipleSVM
share_driver=manila.share.drivers.netapp.common.NetAppDriver
driver_handles_share_servers=True ❷
netapp_storage_family=ontap_cluster
netapp_server_hostname=hostname
netapp_server_port=80
netapp_login=admin_username
netapp_password=admin_password
netapp_transport_type=https
netapp_root_volume_aggregate=aggr1
netapp_aggregate_name_search_pattern=^((?!aggr0).)*$
```

❶ Be sure that the value of the `enabled_share_backends` option in the `[DEFAULT]` stanza includes the name of the stanza you chose for the backend.
❷ The value of `driver_handles_share_servers` **MUST** be set to `True` if you want the driver to manage share servers.

Table 6.11, "Configuration options for clustered Data ONTAP with Share Server management" lists the configuration options available for the unified driver for a clustered Data ONTAP deployment that manages share servers.

Table 6.11. Configuration options for clustered Data ONTAP with Share Server management

Option	Type	Default Value	Description
`driver_handles_share_servers`	Required		Denotes whether the driver should handle the responsibility of managing share servers. This must be set to `true` if the driver is to manage share servers.
`netapp_server_hostname`	Required		The hostname or IP address for the storage system or proxy server. *The value of this option should be the IP address of the cluster management LIF.*
`netapp_server_port`	Optional		The TCP port to use for communication with the storage system or proxy server. If not specified, Data ONTAP drivers will use 80 for HTTP and 443 for HTTPS.
`netapp_login`	Required		Administrative user account name used to access the storage system.
`netapp_password`	Required		Password for the administrative user account specified in the `netapp_login` option.

Option	Type	Default Value	Description
netapp_transport_type	Required	http	Transport protocol for communicating with the storage system or proxy server. Valid options include **http** and **https**.
netapp_storage_family	Required	ontap_cluster	The storage family type used on the storage system; valid values are **ontap_cluster** for clustered Data ONTAP.
netapp_root_volume_aggregate	Required		This option specifies name of the aggregate upon which the root volume should be placed when a new SVM is created to correspond to a Manila share server.
netapp_root_volume_name	Optional	root	This option specifies name of the root volume that will be created when a new SVM is created to correspond to a Manila share server.
netapp_vserver_name_template	Optional	os_%s	This option specifies a string replacement template that is applied when naming SVMs that are created to correspond to a Manila share server.
netapp_lif_name_template	Optional	os_ %(net_allocation_id)s	This option specifies a string replacement template that is applied when naming data LIFs that are created as a result of provisioning requests.
netapp_volume_name_template	Optional	share_%(share_id)s	This option specifies a string replacement template that is applied when naming FlexVol volumes that are created as a result of provisioning requests.
netapp_aggregate_name_search_pattern	Optional	(.*)	This option specifies a regular expression that is applied against all available aggregates. This filtered list will be reported to the Manila scheduler as valid pools for provisioning new shares.
netapp_port_name_search_pattern	Optional	(.*)	This option allows you to specify a regular expression for overriding the selection of network ports on which to create Vserver LIFs.
netapp_trace_flags	Optional		This option is a comma-separated list of options (valid values include **method** and **api**) that controls which trace info is written to the Manila logs when the debug level is set to **True**.

Caution

If you specify an account in the **netapp_login** option that only has SVM administration privileges (rather than cluster administration privileges), some advanced features of the NetApp unified driver will not work and you may see warnings in the Manila logs. See the section called "Account Permission Considerations" for more details on the required access level permissions for an SVM admin account.

6.6.6. Data ONTAP Configuration

Data ONTAP Prerequisites

The prerequisites for Data ONTAP are:

- The driver requires a storage controller running Clustered Data ONTAP 8.2 or later.
- The storage system should have the following licenses applied:
  - Base
  - NFS (if the NFS storage protocol is to be used)
  - CIFS (if the CIFS storage protocol is to be used)
  - FlexClone

Storage Virtual Machine Considerations

When using the NetApp Manila driver in the mode where it does not manage share servers, it is important to pay attention to the following considerations:

1. Ensure the appropriate licenses (as described previously) are enabled on the storage system for the desired use case.
2. The SVM referenced in the `netapp_vserver` option must be created (and associated with aggregates) before it can be utilized as a provisioning target for Manila.
3. Data LIFs must be created and assigned to SVMs before configuring Manila.
4. If NFS is used as the storage protocol:
   a. Be sure to enable the NFS service on the SVM.
   b. Be sure to enable the desired version of the NFS protocol (e.g. `v4.0, v4.1-pnfs`) on the SVM.
5. If CIFS is used as the storage protocol:
   a. Be sure to enable the CIFS service on the SVM.
   b. Be sure to set CIFS as the data protocol on the data LIF.

Account Permission Considerations

When configuring NetApp's Manila drivers to interact with a clustered Data ONTAP instance, it is important to choose the correct administrative credentials to use. While an account with cluster-level administrative permissions is normally utilized, it is possible to use an account with reduced scope that has the appropriate privileges granted to it. In order to use an Cluster-scoped account with the Manila driver and clustered Data ONTAP and have access to the full set of features (including Manila Share Type Extra Specs support) availed by the Manila driver, be sure to add the access levels for the commands shown in Table 6.12, "Common Access Level Permissions Required with Any Manila Driver", Table 6.13, "Access Level Permissions Required For Manila Driver for clustered Data ONTAP with share server management - with Cluster-wide Administrative Account", and Table 6.14, "Access Level Permissions Required For Manila Driver for clustered Data ONTAP without share server management - with Cluster-wide Administrative Account".

Table 6.12. Common Access Level Permissions Required with Any Manila Driver

Command	Access Level
cifs share	all
event	all
network interface	readonly
vserver export-policy	all
volume snapshot	all
version	readonly
system node	readonly
version	readonly
volume	all
vserver	readonly
security	readonly

Table 6.13. Access Level Permissions Required For Manila Driver for clustered Data ONTAP with share server management - with Cluster-wide Administrative Account

Command	Access Level
cifs create	all
cifs delete	all
kerberos-config	all
kerberos-realm	all
ldap client	all
ldap create	all
license	readonly
dns create	all
network interface	all
network port	readonly
network port vlan	all
vserver	all

Table 6.14. Access Level Permissions Required For Manila Driver for clustered Data ONTAP without share server management - with Cluster-wide Administrative Account

Command	Access Level
license	readonly
storage aggregate	readonly
storage disk	readonly

Creating Role for Cluster-Scoped Account. To create a role with the necessary privileges required, with access via ONTAP API only, use the following command syntax to create the role and the cDOT ONTAP user:

1. Create role with appropriate command directory permissions (note you will need to execute this command for each of the required access levels as described in the earlier tables).


```
security login role create –role openstack –cmddirname [required
 command from earlier tables] -access [Required Access Level]
```

2. Command to create user with appropriate role

```
security login create –username openstack –application ontapi –
authmethod password –role openstack
```

Creating Role for SVM-Scoped Account. To create a role with the necessary privileges required, with access via ONTAP API only, use the following command syntax to create the role and the cDOT ONTAP user:

1. Create role with appropriate command directory permissions (note you will need to execute this command for each of the required access levels as described in the earlier tables).

```
security login role create –role openstack -vserver [vserver_name] –
cmddirname [required command from earlier tables] -access [Required
 Access Level]
```

2. Command to create user with appropriate role

```
security login create –username openstack –application ontapi –
authmethod password –role openstack -vserver [vserver_name]
```

Tip

For more information on how to grant access level permissions to a role, and then assign the role to an administrative account, please refer to the System Administration Guide for Cluster Administrators [http://support.netapp.com] document in the Clustered DATA ONTAP documentation.

Storage Networking Considerations

1. Ensure there is segmented network connectivity between the hypervisor nodes and the Data LIF interfaces from Data ONTAP.

2. LIF assignment

6.7. Examples

6.7.1. `manila.conf`

This section provides an example Manila configuration file (`manila.conf`) that contains one backend - for clustered Data ONTAP without share server management.

```
[keystone_authtoken]
signing_dir = /var/cache/manila
admin_password = nomoresecrete
admin_user = manila
admin_tenant_name = service
auth_protocol = http
auth_port = 35357
auth_host = 10.236.168.134

[DEFAULT]
rabbit_userid = stackrabbit
rabbit_password = stackqueue
rabbit_hosts = 10.236.168.134
rpc_backend = rabbit
enabled_share_backends = cdotSingleSVM
enabled_share_protocols = NFS,CIFS
neutron_admin_password = nomoresecrete
cinder_admin_password = nomoresecrete
nova_admin_password = nomoresecrete
default_share_type = default
state_path = /opt/stack/data/manila
osapi_share_extension = manila.api.contrib.standard_extensions
rootwrap_config = /etc/manila/rootwrap.conf
api_paste_config = /etc/manila/api-paste.ini
share_name_template = share-%s
scheduler_driver = manila.scheduler.filter_scheduler.FilterScheduler
verbose = True
debug = True
auth_strategy = keystone

[DATABASE]
connection = mysql://root:stackdb@127.0.0.1/manila?charset=utf8

[oslo_concurrency]
lock_path = /opt/stack/manila/manila_locks

[cdotSingleSVM]
share_backend_name=cdotSingleSVM
share_driver = manila.share.drivers.netapp.common.NetAppDriver
driver_handles_share_servers=False
netapp_storage_family=ontap_cluster
netapp_server_hostname=10.63.40.150
netapp_server_port=80
netapp_login=admin
netapp_password=netapp1!
netapp_vserver=manila-vserver
netapp_transport_type=http
netapp_aggregate_name_search_pattern=^((?!aggr0).)*$
```


6.7.2. Clustered Data ONTAP

This section provides an example configuration script to be executed within Data ONTAP that enables one SVM appropriately configured for the Manila configuration referenced in Section 6.7.1, "`manila.conf`". Note that you may have to edit IP addresses and feature lists based on the environment and licenses present.

```
# create aggrs
storage aggregate create -aggregate aggr1 -diskcount 24 \
-nodes cluster-1-01

storage aggregate create -aggregate aggr2 -diskcount 24 \
-nodes cluster-1-02

# create SVMs
vserver create -vserver manila-vserver -rootvolume vol1 \
-aggregate aggr1 -ns-switch file -rootvolume-security-style unix

# NFS setup
nfs create -vserver manila-vserver -access true
network interface create -vserver manila-vserver \
-lif manila-nfs-data -role data -home-node cluster-1-02 \
-home-port e0d -address 10.63.40.153 -netmask 255.255.192.0

vserver export-policy rule create -vserver manila-vserver \
-policyname default -clientmatch 0.0.0.0/0 -rorule any -rwrule \
any -superuser any -anon 0

# enable v4.0, v4.1, pNFS
nfs modify -vserver manila-vserver -v4.0 enabled -v4.1 enabled \
-v4.1-pnfs enabled

# assign aggregates to vserver
vserver modify -vserver manila-vserver -aggr-list aggr1,aggr2
```

6.7.3. Manila Command Line Interface (CLI)

Manila Service Verfification

In this section, we use the Manila CLI to verify that the configuration presented in Section 6.7.1, "`manila.conf`" has been properly initialized by Manila.

```
[stack@scspr0030615001 devstack]$ manila service-list
+----+------------------+------------------------------+------+---------+-------+----------------------------+
| Id | Binary           | Host                         | Zone | Status  | State | Updated_at                 |
+----+------------------+------------------------------+------+---------+-------+----------------------------+
| 1  | manila-scheduler | scspr0030615001              | nova | enabled | up    | 2015-03-25T12:25:12.000000 |
| 2  | manila-share     | scspr0030615001@cdotSingleSVM | nova | enabled | up    | 2015-03-25T12:25:15.000000 |
+----+------------------+------------------------------+------+---------+-------+----------------------------+
```


Creating and Defining Manila Share Types

In this section, we create a variety of Manila Share Types that leverage both the default capabilities of each driver, as well as the NetApp specific extra specs described in Table 6.6, "NetApp supported Extra Specs for use with Manila Share Types".

- The `general` type provisions Manila shares onto the cDOT backend using all the default settings.
- The `flash` type provisions Manila shares onto any pool that contains SSD disks within the aggregate.
- The `archive` type provisions Manila shares onto any pool that contains SAS drives within the aggregate, and also creates thin provisioned FlexVol volumes.
- The `default` type provisions Manila shares onto any pool of any driver without share server management.

```
[stack@scspr0030615001 devstack]$ manila type-create general False
+--------------------------------------+---------+------------+------------+-------------------------------------+
| ID                                   | Name    | Visibility | is_default | required_extra_specs                |
+--------------------------------------+---------+------------+------------+-------------------------------------+
| deadeebf-19a2-47b1-9d7f-1c3806cfcb72 | general | public     | -          | driver_handles_share_servers : False |
+--------------------------------------+---------+------------+------------+-------------------------------------+
[stack@scspr0030615001 devstack]$ manila type-create flash False
+--------------------------------------+---------+------------+------------+-------------------------------------+
| ID                                   | Name    | Visibility | is_default | required_extra_specs                |
+--------------------------------------+---------+------------+------------+-------------------------------------+
| 37fb9f7e-4ffe-4900-8dba-c6d4251e588f | flash   | public     | -          | driver_handles_share_servers : False |
+--------------------------------------+---------+------------+------------+-------------------------------------+
[stack@scspr0030615001 devstack]$ manila type-create archive False
+--------------------------------------+---------+------------+------------+-------------------------------------+
| ID                                   | Name    | Visibility | is_default | required_extra_specs                |
+--------------------------------------+---------+------------+------------+-------------------------------------+
| 20e4b58d-aab6-42ae-8e9b-8f9d44f17276 | archive | public     | -          | driver_handles_share_servers : False |
+--------------------------------------+---------+------------+------------+-------------------------------------+
```

```
[stack@scspr0030615001 devstack]$ manila type-key general set \
    share_backend_name=cdotSingleSVM driver_handles_share_servers=False
[stack@scspr0030615001 devstack]$ manila type-key flash set \
    netapp_disk_type=SSD driver_handles_share_servers=False
[stack@scspr0030615001 devstack]$ manila type-key archive set \
    netapp:thin_provisioned=True netapp_disk_type=SAS driver_handles_share_servers=False
[stack@scspr0030615001 devstack]$ manila extra-specs-list
+--------------------------------------+---------+-------------------------------------+
| ID                                   | Name    | all_extra_specs                     |
+--------------------------------------+---------+-------------------------------------+
| 20e4b58d-aab6-42ae-8e9b-8f9d44f17276 | archive | driver_handles_share_servers : False |
|                                      |         | netapp_disk_type : SAS              |
|                                      |         | netapp:thin_provisioned : True      |
| 37fb9f7e-4ffe-4900-8dba-c6d4251e588f | flash   | netapp_disk_type : SSD              |
|                                      |         | driver_handles_share_servers : False |
| 447732be-4cf2-42b0-83dc-4b6f4ed5368d | default | driver_handles_share_servers : False |
| deadeebf-19a2-47b1-9d7f-1c3806cfcb72 | general | share_backend_name : cdotSingleSVM  |
|                                      |         | driver_handles_share_servers : False |
+--------------------------------------+---------+-------------------------------------+
```

Creating Manila Shares with Shares Types

In this section, we create shares with the default type, as well as each of the previously defined share types.

```
[stack@scspr0030615001 devstack]$ manila create --name myDefault NFS 1
+-------------------+--------------------------------------+
| Property          | Value                                |
+-------------------+--------------------------------------+
| status            | creating                             |
| description       | None                                 |
| availability_zone | nova                                 |
```



```
| share_network_id  | None                                 |
| export_locations  | []                                   |
| share_server_id   | None                                 |
| host              | None                                 |
| snapshot_id       | None                                 |
| is_public         | False                                |
| id                | 63bd5bef-298d-4e49-bea0-49a4cfb143f9 |
| size              | 1                                    |
| name              | myDefault                            |
| share_type        | 447732be-4cf2-42b0-83dc-4b6f4ed5368d |
| created_at        | 2015-03-25T12:44:11.794842           |
| export_location   | None                                 |
| share_proto       | NFS                                  |
| project_id        | 5bf3e15106dd4333b1f55742fa08f90e     |
| metadata          | {}                                   |
+-------------------+--------------------------------------+

[stack@scspr0030615001 devstack]$ manila create --name myGeneral --share-type general NFS 1
+-------------------+--------------------------------------+
| Property          | Value                                |
+-------------------+--------------------------------------+
| status            | creating                             |
| description       | None                                 |
| availability_zone | nova                                 |
| share_network_id  | None                                 |
| export_locations  | []                                   |
| share_server_id   | None                                 |
| host              | None                                 |
| snapshot_id       | None                                 |
| is_public         | False                                |
| id                | 95f49ca6-723f-42d0-92f3-4be79c9ad7e6 |
| size              | 1                                    |
| name              | myGeneral                            |
| share_type        | deadeebf-19a2-47b1-9d7f-1c3806cfcb72 |
| created_at        | 2015-03-25T12:44:47.223548           |
| export_location   | None                                 |
| share_proto       | NFS                                  |
| project_id        | 5bf3e15106dd4333b1f55742fa08f90e     |
| metadata          | {}                                   |
+-------------------+--------------------------------------+

[stack@scspr0030615001 devstack]$ manila create --name myFlash --share-type flash NFS 1
+-------------------+--------------------------------------+
| Property          | Value                                |
+-------------------+--------------------------------------+
| status            | creating                             |
| description       | None                                 |
| availability_zone | nova                                 |
| share_network_id  | None                                 |
| export_locations  | []                                   |
| share_server_id   | None                                 |
| host              | None                                 |
| snapshot_id       | None                                 |
| is_public         | False                                |
| id                | ec5d2ddb-4573-4ee1-a1e8-2c8532c68e3d |
| size              | 1                                    |
| name              | myFlash                              |
| share_type        | 37fb9f7e-4ffe-4900-8dba-c6d4251e588f |
| created_at        | 2015-03-25T12:44:59.374780           |
| export_location   | None                                 |
| share_proto       | NFS                                  |
| project_id        | 5bf3e15106dd4333b1f55742fa08f90e     |
| metadata          | {}                                   |
+-------------------+--------------------------------------+

[stack@scspr0030615001 devstack]$ manila create --name myArchive --share-type archive NFS 1
+-------------------+--------------------------------------+
| Property          | Value                                |
+-------------------+--------------------------------------+
| status            | creating                             |
| description       | None                                 |
| availability_zone | nova                                 |
```



```
| share_network_id  | None                                 |
| export_locations  | []                                   |
| share_server_id   | None                                 |
| host              | None                                 |
| snapshot_id       | None                                 |
| is_public         | False                                |
| id                | e4774a70-3e70-4a7c-ab76-886f010efe0a |
| size              | 1                                    |
| name              | myArchive                            |
| share_type        | 20e4b58d-aab6-42ae-8e9b-8f9d44f17276 |
| created_at        | 2015-03-25T12:45:11.124722           |
| export_location   | None                                 |
| share_proto       | NFS                                  |
| project_id        | 5bf3e15106dd4333b1f55742fa08f90e     |
| metadata          | {}                                   |
+-------------------+--------------------------------------+
```

```
[stack@scspr0030615001 devstack]$ manila list
+--------------------------------------+-----------+------+-------------+-----------+-----------+------------
+----------------------------------------------------------+-------------------------------------+
| ID                                   | Name      | Size | Share Proto | Status    | Is Public | Share Type |
 Export location                                            | Host                                |
+--------------------------------------+-----------+------+-------------+-----------+-----------+------------
+----------------------------------------------------------+-------------------------------------+
| 63bd5bef-298d-4e49-bea0-49a4cfb143f9 | myDefault | 1    | NFS         | available | False     | default    |
 10.63.40.153:/share_63bd5bef_298d_4e49_bea0_49a4cfb143f9 | scspr0030615001@cdotSingleSVM#aggr1 |
| 95f49ca6-723f-42d0-92f3-4be79c9ad7e6 | myGeneral | 1    | NFS         | available | False     | general    |
 10.63.40.153:/share_95f49ca6_723f_42d0_92f3_4be79c9ad7e6 | scspr0030615001@cdotSingleSVM#aggr1 |
| e4774a70-3e70-4a7c-ab76-886f010efe0a | myArchive | 1    | NFS         | available | False     | archive    |
 10.63.40.153:/share_e4774a70_3e70_4a7c_ab76_886f010efe0a | scspr0030615001@cdotSingleSVM#aggr1 |
| ec5d2ddb-4573-4ee1-a1e8-2c8532c68e3d | myFlash   | 1    | NFS         | error     | False     | flash      | None
                                                            | None                                |
+--------------------------------------+-----------+------+-------------+-----------+-----------+------------
+----------------------------------------------------------+-------------------------------------+
```

We'll now look at the CLI output from the Data ONTAP cluster to see what FlexVol volumes were created for the Manila share objects, as well as the provisioning strategy (thin or thick) for each share. Note how the name of the FlexVol volume corresponds to the share UUID as defined by Manila. You can see that share `e4774a70_3e70_4a7c_ab76_886f010efe0a` was thin provisioned, as declared by the `archive` share type. The rest of the shares were thick provisioned. Also note that the share of type `myFlash` failed to create, as this SVM does not have any aggregates with SSD drives, as seen in the command output below.

```
cluster-1::> volume show -vserver manila-vserver
Vserver   Volume       Aggregate    State      Type       Size  Available Used%
--------- ------------ ------------ ---------- ---- ---------- ---------- -----
manila-vserver
          share_63bd5bef_298d_4e49_bea0_49a4cfb143f9
                       aggr1        online     RW          1GB    972.7MB    5%
manila-vserver
          share_95f49ca6_723f_42d0_92f3_4be79c9ad7e6
                       aggr1        online     RW          1GB    972.7MB    5%
manila-vserver
          share_e4774a70_3e70_4a7c_ab76_886f010efe0a
                       aggr1        online     RW          1GB    972.7MB    5%
manila-vserver
          vol1         aggr1        online     RW         20MB    18.88MB    5%
4 entries were displayed.

cluster-1::>

cluster-1::> volume show -vserver manila-vserver -space-guarantee none
Vserver   Volume       Aggregate    State      Type       Size  Available Used%
--------- ------------ ------------ ---------- ---- ---------- ---------- -----
manila-vserver
          share_e4774a70_3e70_4a7c_ab76_886f010efe0a
                       aggr1        online     RW          1GB    972.7MB    5%

cluster-1::> volume show -vserver manila-vserver -space-guarantee volume
Vserver   Volume       Aggregate    State      Type       Size  Available Used%
--------- ------------ ------------ ---------- ---- ---------- ---------- -----
manila-vserver
          share_63bd5bef_298d_4e49_bea0_49a4cfb143f9
```



```
                      aggr1        online     RW          1GB    972.7MB    5%
manila-vserver
          share_95f49ca6_723f_42d0_92f3_4be79c9ad7e6
                      aggr1        online     RW          1GB    972.7MB    5%
manila-vserver
          vol1        aggr1        online     RW         20MB    18.88MB    5%
3 entries were displayed.

cluster-1::>

cluster-1::> disk show -type SSD
There are no entries matching your query.
```

We'll now add access rules for any IP-connected client to mount these NFS shares with full read/write privileges.

```
[stack@scspr0030615001 devstack]$ manila access-allow
 63bd5bef-298d-4e49-bea0-49a4cfb143f9 ip 0.0.0.0/0
+--------------+--------------------------------------+
| Property     | Value                                |
+--------------+--------------------------------------+
| share_id     | 63bd5bef-298d-4e49-bea0-49a4cfb143f9 |
| deleted      | False                                |
| created_at   | 2015-03-25T13:25:24.577736           |
| updated_at   | None                                 |
| access_type  | ip                                   |
| access_to    | 0.0.0.0/0                            |
| access_level | rw                                   |
| state        | new                                  |
| deleted_at   | None                                 |
| id           | c400bdd7-7e4f-49a4-b73d-5aa417af95c3 |
+--------------+--------------------------------------+
[stack@scspr0030615001 devstack]$ manila access-allow
 95f49ca6-723f-42d0-92f3-4be79c9ad7e6 ip 0.0.0.0/0
+--------------+--------------------------------------+
| Property     | Value                                |
+--------------+--------------------------------------+
| share_id     | 95f49ca6-723f-42d0-92f3-4be79c9ad7e6 |
| deleted      | False                                |
| created_at   | 2015-03-25T13:25:47.417271           |
| updated_at   | None                                 |
| access_type  | ip                                   |
| access_to    | 0.0.0.0/0                            |
| access_level | rw                                   |
| state        | new                                  |
| deleted_at   | None                                 |
| id           | 09f8f699-1cec-4519-8aaa-a30d346ad54c |
+--------------+--------------------------------------+
[stack@scspr0030615001 devstack]$ manila access-allow
 e4774a70-3e70-4a7c-ab76-886f010efe0a ip 0.0.0.0/0
+--------------+--------------------------------------+
| Property     | Value                                |
+--------------+--------------------------------------+
| share_id     | e4774a70-3e70-4a7c-ab76-886f010efe0a |
| deleted      | False                                |
| created_at   | 2015-03-25T13:26:03.344004           |
```



```
| updated_at   | None                                 |
| access_type  | ip                                   |
| access_to    | 0.0.0.0/0                            |
| access_level | rw                                   |
| state        | new                                  |
| deleted_at   | None                                 |
| id           | d0565115-8369-455e-ad8f-3dd7c56037ea |
+--------------+--------------------------------------+
```


Chapter 7. OpenStack Compute Service (Nova)

7.1. Overview

The OpenStack Compute Service (Nova) is a cloud computing fabric controller, which is the main part of an IaaS system. You can use OpenStack Compute to host and manage cloud computing systems.

Nova is typically deployed in conjunction with other OpenStack services (e.g. Block Storage, Object Storage, Image, etc) as part of a larger, more comprehensive cloud infrastructure.

7.2. Key Concepts

Instance

An instance is the fundamental resource unit allocated by the OpenStack Compute service. It represents an allocation of compute capability (most commonly but not exclusively a virtual machine), along with optional ephemeral storage utilized in support of the provisioned compute capacity.

Caution

Unless a root disk is sourced from Cinder (see the section called “Root Disk Choices When Booting Nova Instances”, the disks associated with VMs are "ephemeral," meaning that (from the user's point of view) they effectively disappear when a virtual machine is terminated.

Instances can be identified uniquely through a UUID assigned by the Nova service at the time of instance creation. An instance may also be optionally referred to by a human-readable name, though this string is not guaranteed to be unique within a single tenant or deployment of Nova.

Flavor

Virtual hardware templates are called "flavors" in OpenStack, defining sizes for RAM, disk, number of cores, and so on. The default install provides five flavors, and are configurable by admin users.

Flavors define a number of parameters, resulting in the user having a choice of what type of virtual machine to run—just like they would have if they were purchasing a physical server. The table lists the elements that can be set. Flavors may also contain extra_specs, which can be used to define free-form characteristics, giving a lot of flexibility beyond just the size of RAM, CPU, and Disk in determining where an instance is provisioned.

Root (and ephemeral) disks

Each instance needs at least one root disk (that contains the bootloader and core operating system files), and may have optional ephemeral disk (per the definition of the flavor selected at instance creation time). The content for the root disk either comes from an image stored within the Glance repository (and copied to storage attached to the destination hypervisor) or from a persistent block storage volume (via Cinder). For more information on the root disk strategies available during instance creation, refer to the section called “Root Disk Choices When Booting Nova Instances”.

7.3. Theory of Operation & Deployment Choices

Root Disk Choices When Booting Nova Instances

There are several choices for how the root disk should be created which are presented to cloud users when booting Nova instances.

Boot from image	This option allows a user to specify an image from the Glance repository to copy into an ephemeral disk.
Boot from snapshot	This option allows a user to specify an instance snapshot to use as the root disk; the snapshot is copied into an ephemeral disk.
Boot from volume	This option allows a user to specify a Cinder volume (by name or UUID) that should be directly attached to the instance as the root disk; no copy is made into an ephemeral disk and any content stored in the volume is persistent.
Boot from image (create new volume)	This option allows a user to specify an image from the Glance repository to be copied into a persistent Cinder volume, which is subsequently attached as the root disk for the instance.
Boot from volume snapshot (create new volume)	This option allows a user to specify a Cinder volume snapshot (by name or UUID) that should be used as the root disk; the snapshot is copied into a new, persistent Cinder volume which is subsequently attached as the root disk for the instance.

Tip

Leveraging the "boot from volume", “boot from image (creates a new volume)”, or "boot from volume snapshot (create new volume)" options in Nova normally results in volumes that are persistent beyond the life of a particular instance. However, you can select the “delete on terminate” option in combination with any of the aforementioned options to create

an ephemeral volume while still leveraging the enhanced instance creation capabilities described in the section called "Enhanced Instance Creation". This can provide a significantly faster provisioning and boot sequence than the normal way that ephemeral disks are provisioned, where a copy of the disk image is made from Glance to local storage on the hypervisor node where the instance resides.

Instance Snapshots vs. Cinder Snapshots

Instance snapshots allow you to take a point in time snapshot of the content of an instance's disk. Instance snapshots can subsequently be used to create an image that can be stored in Glance which can be referenced upon subsequent boot requests.

While Cinder snapshots also allow you to take a point-in-time snapshot of the content of a disk, they are more flexible than instance snapshots. For example, you can use a Cinder snapshot as the content source for a new root disk for a new instance, or as a new auxiliary persistent volume that can be attached to an existing or new instance. For more information on Cinder snapshots, refer to Section 4.2, "Key Concepts".

Instance Storage Options at the Hypervisor

The Nova configuration option `instances_path` specifies where instances are stored on the hypervisor's disk. While this may normally point to locally attached storage (which could be desirable from a performance perspective), it prohibits the ability to support live migration of instances between hypervisors. By specifying a directory that is a mounted NFS export (from a NetApp FlexVol volume), it is possible to support live migration of instances because their root disks are on shared storage which can be accessed from multiple hypervisor nodes concurrently.

Note

Assuming shared storage (NFS) is used to store Nova instances, there are several other requirements that must be met in order to fully support live migration scenarios. More information can be found at http://docs.openstack.org/trunk/openstack-ops/content/compute_nodes.html

Warning

The NFS mount for the `instances_path` is not managed by Nova; therefore, you must use the standard Linux mechanisms (e.g. `/etc/fstab` or NFS automounter) to ensure that the NFS mount exists before Nova starts.

Chapter 8. OpenStack Object Storage Service (Swift)

OpenStack Object Storage provides a fully distributed, scale-out, API-accessible storage platform that can be integrated directly into applications or used for backup, archiving and data retention. Object storage does not present a traditional file system, but rather a distributed storage system for static data such as: virtual machine images, photo storage, email storage, backups and archives.

The Swift API proposes an open standard for cloud storage. It can also function as an alternative endpoint for Amazon Web Services S3 and as a CDMI server through the use of add on components.

Swift requires node accessible media for storing object data. This media can be drives internal to the node or external storage devices such as the NetApp E-Series storage array. This section provides information that enables an NetApp E-Series storage array to be used as the backing store for Swift object storage.

8.1. Overview

There are several major advantages of using E-Series storage for Swift object storage nodes. These include:

- A dramatic reduction in the storage capacity and physical hardware required to facilitate data protection through Swift's consistent hashing ring. The unique characteristics of E-series' Dynamic Disk Pools (DDP) enable the use of a parity protection scheme for data protection as an alternative to the default approach involving creating 3 or more copies of data. Within a single site, the capacity required for object storage along with the parity overhead is an approximate 1.3 multiple of the object(s) stored. The default Swift behavior involves storing a multiple of 3.

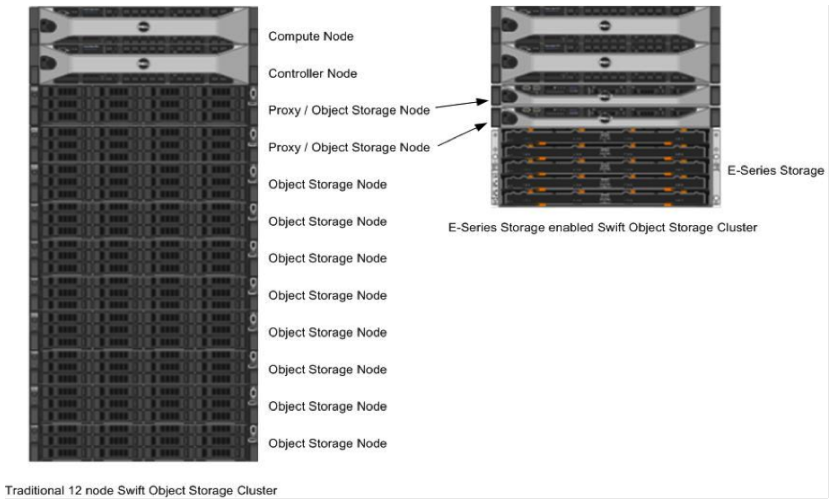
- The reduction of replicas made possible by the use of DDP has the effect of significantly reducing a typically major inhibitor to the scale a given Swift cluster can achieve. It has been observed that the weight of replication traffic can become a limitation to scale in certain use cases.

- The associated storage capacity efficiency associated with employing DDP

- Reduced Swift node hardware requirements: Internal drive requirements for storage nodes are reduced, only operating system storage is required. Disk space for Swift object data, and optionally the operating system itself, is supplied by the E-Series storage array.

- Reduced rack space, power, cooling and footprint requirements: Since a single storage subsystem provides storage space for multiple Swift nodes, smaller and possibly lower power 1U nodes can be used in the cluster.

Figure 8.1. Traditional and E-Series Swift Stack Comparison



On the left of Figure 8.1, "Traditional and E-Series Swift Stack Comparison" is a traditional Swift cluster, which has a total storage capacity of 240TB. This requires 10 Swift object storage nodes with 12 2TB drives per system, which results in approximately 80 TB of effective storage capacity assuming that Swift uses the default replica count of 3.

Compare this traditional system to the E-Series based cluster, shown on the right in Figure 8.1, "Traditional and E-Series Swift Stack Comparison". The E-Series cluster has identical controller and compute nodes as the traditional system. In the E-Series cluster the effective 80TB storage capacity of the traditional system can be obtained by using a single 4U storage subsystem. The dynamic disk pools (DDP) data reconstruction feature on E-Series replaces the data replication implementation of Swift. As mentioned above, this enables a 1U server (with similar memory and CPU resources as the traditional cluster object nodes) to be used in the E-Series stack. This results in 42% less rack footprint and approximately 55% in drive savings (120 drives vs. ~54 drives for an E-Series based cluster). Additionally the number of attached Swift object storage nodes attached to the E-Series can be increased if additional object storage processing capacity is required.

Tip

Swift may also be deployed in conjunction with the NetApp FAS product line, as an iSCSI LUN could be used as a block device to provide storage for object, container, or account data. This deployment may be used in situations where the scale of an object storage deployment is small, or if it is desirable to reuse existing FAS systems.

DDP Reconstruction

E-Series storage can effectively serve as the storage medium for OpenStack Object Storage. The data reconstruction capabilities associated with DDP eliminate the need for data replication within zones. DDP reconstruction provides RAID-6 data protection against multiple simultaneous drive failures within the storage subsystem. Data that resides on multiple failed drives is given top priority during reconstruction. This data has the highest potential for being lost if a 3rd

drive failure occurs is reconstructed first on the remaining optimal drives in the storage subsystem. After this critical data is reconstructed all other data on the failed drives is reconstructed. This prioritized data reconstruction dramatically reduces the possibility of data loss due to drive failure.

8.2. Swift Zones and NetApp E-Series Storage

Swift uses zoning to isolate the cluster into separate partitions to isolate the cluster from failures. Swift data is replicated across the cluster in zones that are as unique as possible. A zone is an arbitrary grouping of nodes; typically zones are established that use physical attributes of the cluster, such as geographical locations, separate networks, equipment racks, storage subsystems, or even single drives. Zoning allows the cluster to tolerate equipment failures within the cluster without data loss or loss of connectivity to the remaining cluster.

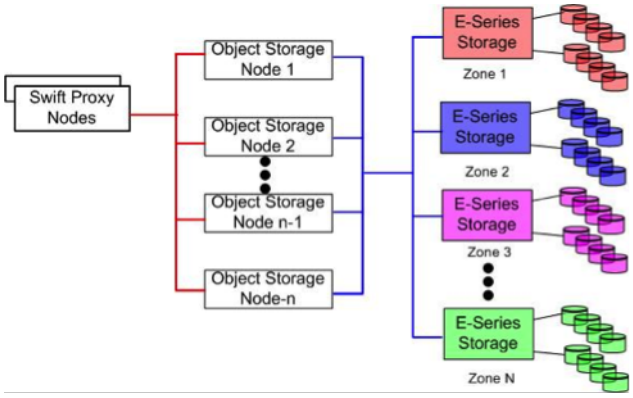
By default, Swift replicates data 3 times across the cluster. Swift replicates data across zones in a unique pattenr that attempts to ensure high availability and high durability for data. Swift chooses to place a replica of data in a server in an unused zone before placing it into an unused server in a zone that already has a replica of the data.

The data reconstruction feature of E-Series ensures that clients always have access to their data - regardless of drive or other component failures within the storage subsystem. When E-Series storage is used Swift data replication counts specified when rings are built can be reduced from 3 to 1.

E-Series storage offers flexible configuration options that satisfy practically all Swift zoning requirements. DDP reconstruction also eliminates the requirement of Swift data replication within a single storage array. Zoning based on E-Series storage can be done on a storage subsystem, individual controller, or drive tray basis.

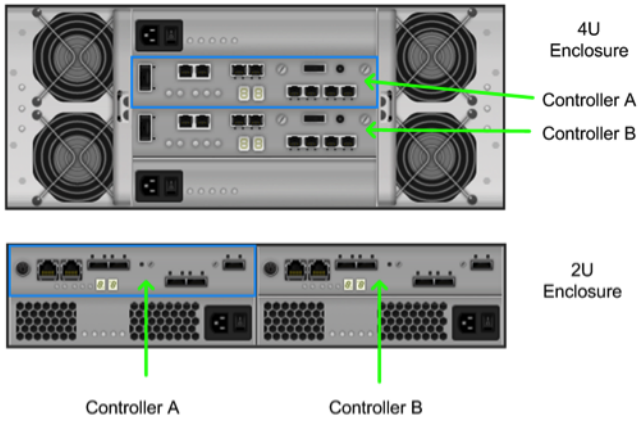
Controller-subsystem based zoning. In a cluster that contains several E-Series storage subsystems zoning may be done through the use of 1 or more E-Series subsystems as a zone. An example of this is shown in Figure 8.2, “Controller-subsystem based zoning”.

Figure 8.2. Controller-subsystem based zoning



Controller based zoning. E-Series storage subsystems contain two independently configurable controller modules (shown in Figure 8.3, "E-Series Controller Layout") which in turn communicate with drives contained within the storage subsystem and optionally other externally attached E-Series drive enclosures.

Figure 8.3. E-Series Controller Layout



For controller based zoning, each E-Series storage controller supports 2 Swift object storage nodes. Each node is connected to a single controller in the storage array. LUNs for each Swift node are configured separately on drives located within the enclosure. Additional drive enclosures may also be attached to the controller module for added storage capacity.

Drive-tray or enclosure-based zoning. If Swift cluster requirements require unique zones for each object node E-Series storage arrays can effectively provide storage capacity for multiple Swift object nodes. Disk pools are created according to desired capacity. Individual drives which comprise a disk pool are preselected by the system. Automated drive selection helps ensure that:

- Subsystem I/O performance is maximized.
- The impact of hardware failures within a drive enclosure or tray is minimized.
- I/O load on the subsystem is distributed as evenly as possible across all the drive channels in the subsystem.

If unique zoning is not required, node connectivity is only limited by the host connectivity capacity of the E-Series storage subsystem being used.

8.3. Disk Pools, Volumes, and Mapping

Disk pools should be created and sized based on the number of object storage nodes connecting to the storage subsystem. A minimum of 11 drives per drive pool is required, but the recommended number of drives in a drive pool is equal to `N`, where `N` is the total number of drives in the storage subsystem divided by the total number of attached object storage nodes.

Assuming no SSD drives are present, create 3 volumes of equal capacity on each drive pool. Be sure to select the "Map Later" option to ensure the mapping to a

host takes place after all volumes are created. If SSDs are present, be sure to create separate disk pools that only contain SSDs. Swift documentation recommends that SSDs be leveraged for account and container type objects.

The default mapping for volumes to hosts (via LUN mapping) is to expose all volumes to all hosts. To ensure multiple hosts are not accessing the same LUN concurrently, it is recommended that each volume be explicitly mapped to the WWN for the appropriate host it should connect to. If the default mapping is utilized, extreme care must be exercised to ensure the correct mapping is retained to prevent data corruption.

8.4. Partitioning and File System Considerations

After volumes are created and mapped to Swift nodes, they need to be partitioned and have a file system created on them. For each LUN that was created on the E-Series storage array create a single, new primary partition that utilizes the entire capacity available on the LUN.

NetApp recommends the use of **dm-multipath** to provide support for redundant paths between an object storage node and the E-Series storage controller. For details on how to configure **dm-multipath**, refer to the NetApp E-Series Storage Systems Failover Drivers Guide, located at https://library.netapp.com/ecm/ecm_get_file/ECMP1394845.

Partitioning with Multipath

Assuming that three volumes were created from the disk pool, and if multipath is enabled, you should see a total of 6 mapped devices, as in the following example:

```
rroot@stlrx300s7-102:~# ls -l /dev/mapper
total 0
lrwxrwxrwx 1 root root       7 May  5 15:20
 360080e50003220a80000017353450e3f -> ../dm-0
lrwxrwxrwx 1 root root       7 May  5 15:20
 360080e5000322230000019153450e18 -> ../dm-1
lrwxrwxrwx 1 root root       7 May  5 15:20
 360080e5000322230000019053450e18 -> ../dm-2
crw------- 1 root root 10, 236 May  5 15:20 control
```

Now we use the **parted** command to partition the mapped devices:

```
root@stlrx300s7-102:/dev/mapper# luns=`ls|grep -v control`
root@stlrx300s7-102:/dev/mapper# for i in $luns
> do
> parted -a optimal -s --  /dev/mapper/$i mklabel gpt mkpart primary xfs 0% 100%
> done
root@stlrx300s7-102:/dev/mapper# ls -l /dev/dm-
dm-0  dm-1  dm-2  dm-3  dm-4  dm-5  dm-6  dm-7  dm-8
root@stlrx300s7-102:/dev/mapper# ls -l /dev/mapper
total 0
lrwxrwxrwx 1 root root       7 May  5 15:29 360080e50003220a80000017353450e3f -> ../dm-0
lrwxrwxrwx 1 root root       7 May  5 15:29 360080e50003220a80000017353450e3f1 -> ../dm-3
```



```
lrwxrwxrwx 1 root root       7 May  5 15:29 360080e50003220a80000017353450e3f-part1 -> ../dm-4
lrwxrwxrwx 1 root root       7 May  5 15:29 360080e5000322230000019053450e18 -> ../dm-2
lrwxrwxrwx 1 root root       7 May  5 15:29 360080e5000322230000019053450e18p1 -> ../dm-5
lrwxrwxrwx 1 root root       7 May  5 15:29 360080e5000322230000019053450e18-part1 -> ../dm-6
lrwxrwxrwx 1 root root       7 May  5 15:29 360080e5000322230000019153450e18 -> ../dm-1
lrwxrwxrwx 1 root root       7 May  5 15:29 360080e5000322230000019153450e18p1 -> ../dm-7
lrwxrwxrwx 1 root root       7 May  5 15:29 360080e5000322230000019153450e18-part1 -> ../dm-8
crw------- 1 root root 10, 236 May  5 15:20 control
```

Swift currently requires that the underlying filesystem have support for extended attributes of the file system. While this requirement may be removed in a future release of Swift, as of the Havana release the recommended filesystem type is XFS.

Internal volumes created in a DDP layout resemble a traditional RAID 6 volume with the following parameters:

- Configuration: 8+2 RAID 6
- Segment Size: 128K
- Stripe Width: 1MB (total number of data drives * segment size = 128K * 8)

These parameters can be leveraged to configure the file system for optimal performance with the LUN. When a file system is created on a logical volume device, **mkfs.xfs** automatically queries the logical volume to determine appropriate stripe unit and stripe width values, unless values are passed at the time of filesystem creation; for example:

```
# ls -l /dev/mapper/|grep part|awk '{print $9}'
360080e50003220a80000017353450e3f-part1
360080e5000322230000019053450e18-part1
360080e5000322230000019153450e18-part1
# parts=`ls -l /dev/mapper/|grep part|awk '{print $9}'`
# for i in $parts
> do
> mkfs.xfs -d su=131072,sw=8 -i size=1024 $i
> done
```

Tip

You can verify that the partition was successfully created and is properly aligned by using the **parted** command:

```
# parted /dev/mapper/360a98000433468525634447 17a513571 align-
check optimal 1
1 aligned
```

You can verify that the underlying filesystem has the correct values for stripe unit and stripe width by using the **xfs_info** command:

```
mount -t xfs -o nobarrier,noatime,nodiratime,inode64 /dev/
mapper/360080e50003220a80000017353450e3f-part1 /disk1
```



```
# xfs_info /disk1
meta-data=/dev/mapper/360080e50003220a80000017353450e3f-part1 isize=1024   agcount=4,
 agsize=83623808 blks
         =                       sectsz=512   attr=2
data     =                       bsize=4096   blocks=334495232, imaxpct=5
         =                       sunit=32     swidth=256 blks
naming   =version 2              bsize=4096   ascii-ci=0
log      =internal               bsize=4096   blocks=163328, version=2
         =                       sectsz=512   sunit=32 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
```

`sunit` and `swidth` are shown in `bsize` (block size) units in the **xfs_info** command output.

```
stripe unit= 32 sunits * 4096 bsize (block size)= 131072 bytes = 128K
stripe width= 256 blocks * 4096 bsize = 1M = 128K * 8 drives
```

The sysctl `fs.xfs.rotorstep` can be used to change how many files are put into an XFS allocation group. Increasing the default number from 1 to 255 reduces seeks to multiple allocation groups. NetApp has observed improved performance in some cases by increasing this number. You can put the following line in `/etc/sysctl.conf` to ensure this change is affected on each boot of the system:

```
fs.xfs.rotorstep = 255
```

When mounting the XFS filesystem that resides on the LUNs offered from the E-Series storage, be sure to use the following mount options:

```
mount –t xfs –o “nobarrier,noatime,nodiratime,inode64” \
/dev/mapper/nodeX /srv/node/sdb1
```

Warning

The mount points for the account, container, and object storage are not managed by Swift; therefore, you must use the standard Linux mechanisms (e.g. `/etc/fstab`) to ensure that the mount points exist and are mounted before Swift starts.

8.5. Swift Ring Considerations with DDP

A Swift ring represents a mapping between the names of entities stored on disk and their physical location. There are separate rings for accounts, containers, and objects. When other components need to perform any operation on an object, container, or account, they need to interact with the appropriate ring to determine its location in the cluster.

The ring maintains this mapping using zones, devices, partitions, and replicas. Each partition in the ring is replicated 3 times, by default, across the cluster. The locations for a partition are stored in the mapping maintained by the ring. The ring

is also responsible for determining which devices are used for handoff in failure scenarios.

Tip

When leveraging the DDP capabilities of the E-Series storage system, it is only necessary to have Swift maintain 1 replica of the data as the storage subsystem automatically is providing sufficient replication of the data within the storage cluster to withstand multiple failures.

The number of replicas for partitions within Swift rings is set at the time the ring is created. To create the Swift rings for the Swift object types of account, container, and object, use the **swift-ring-builder** CLI command with the `replicas` parameter set to ***1***:

```
#swift-ring-builder ring-name create part-power replicas min-hours
swift-ring-builder account.builder create 10 1 1
swift-ring-builder container.builder create 10 1 1
swift-ring-builder object.builder create 10 1 1
```

Tip

When creating the Swift rings, the number of partitions per ring must be calculated. However, the default computation for computing the number of partitions (expressed by the exponent of 2) - otherwise known as the partition power (`part-power`) - is based on the number of disks in a Swift cluster. While some Swift deployments (without NetApp E-Series storage solutions) utilize direct attached storage without RAID, with E-Series virtual disks are attached and leveraged that are created from a dynamic disk pool. As such, be sure to calculate the partition power as a function of the virtual disk count, rather than the number of physical (spinning or SSD) drives present.

Chapter 9. Operational Concerns

9.1. Operational Concerns with Data ONTAP

Considerations for the use of thin provisioning with FlexVol volumes

Using thin provisioning, you can configure your volumes so that they appear to provide more storage than they have available, provided that the storage that is actually being used does not exceed the available storage.

To use thin provisioning with FlexVol volumes, you create the volume with a guarantee of none. With a guarantee of none, the volume size is not limited by the aggregate size. In fact, each volume could, if required, be larger than the containing aggregate. The storage provided by the aggregate is used up only as data is written to the LUN or file.

If the volumes associated with an aggregate show more storage as available than the physical resources available to that aggregate, the aggregate is overcommitted. When an aggregate is overcommitted, it is possible for writes to LUNs or files in volumes contained by that aggregate to fail if there is not sufficient free space available to accommodate the write.

If you have overcommitted your aggregate, you must monitor your available space and add storage to the aggregate as needed to avoid write errors due to insufficient space.

Aggregates can provide storage to FlexVol volumes associated with more than one Storage Virtual Machine (SVM). When sharing aggregates for thin-provisioned volumes in a multi-tenancy environment, be aware that one tenant's aggregate space availability can be adversely affected by the growth of another tenant's volumes.

For more information about thin provisioning, see the following technical reports:

- TR 3965: NetApp Thin Provisioning Deployment and Implementation Guide [http://media.netapp.com/DOCUMENTS/TR-3965.PDF]
- TR 3483: Thin Provisioning in a NetApp SAN or IP SAN Enterprise Environment [http://media.netapp.com/DOCUMENTS/TR3483.PDF]

NFS Best Practices

Be sure to refer to the Clustered Data ONTAP NFS Best Practices and Implementation Guide [http://www.netapp.com/us/system/pdf-reader.aspx?pdfuri=tcm:10-61288-16&m=tr-4067.pdf] for information on how to optimally set up NFS exports for use with the various OpenStack services described in this guide.

Volume Migration

Volume migration is a new feature added to the OpenStack Block Storage service in the Havana release. Volume migration allows the movement of data associated with a Cinder volume from one storage backend to another, completely transparent to any instances that may have the volume attached. Volume migration is agnostic to the storage protocol used to access the volume, and can be utilized to switch from leveraging iSCSI to NFS, or vice versa, even on the same physical NetApp system.

Please be aware of the following caveats:

1. Volume migrations can only be performed by a cloud administrator, not a tenant or user.

2. Migration of volumes that have existing snapshots is not currently allowed. If it is important to retain the data stored in a snapshot for a volume you wish to migrate, one option is to create another Cinder volume from the snapshot (which creates a copy of the snapshot), and then delete the snapshot that is associated with the volume you wish to migrate.

3. Transparent migration of volumes that are attached to Nova instances are only supported when the Nova instances are running on a hypervisor that supports live migration. NetApp has qualified live migrations with the KVM hypervisor on Ubuntu 13.04 with the Havana release; note that earlier versions of Ubuntu (including 12.04 LTS) are known to not support migration of attached volumes.

Migrating from Data ONTAP operating in 7-Mode to clustered Data ONTAP.
The volume migration feature of the OpenStack Block Storage service can be used to aid in the transition from Data ONTAP operating in 7-Mode to clustered Data ONTAP seamlessly. If you have volumes managed by Cinder on a Data ONTAP operating in 7-Mode storage system, you can configure the clustered Data ONTAP instance as a new backend in the Cinder configuration and leverage the migration feature to move existing volumes to the new backend and then retire the Data ONTAP operating in 7-Mode system, if desired.

Once you have configured both the storage systems to operate with Cinder, you can verify both backends have been enabled successfully and are ready to support the migration process.

```
$ cinder service list
+------------------+--------------------+------+---------+-------+----------------------------+
|      Binary      |        Host        | Zone |  Status | State |         Updated_at         |
+------------------+--------------------+------+---------+-------+----------------------------+
| cinder-scheduler |     openstack1     | nova | enabled |   up  | 2013-1-1T19:01:26.000000 |
|  cinder-volume   |  openstack1@7mode | nova | enabled |   up  | 2013-1-1T19:01:18.000000 |
|  cinder-volume   |  openstack1@cDOT  | nova | enabled |   up  | 2013-1-1T19:01:27.000000 |
+------------------+--------------------+------+---------+-------+----------------------------+
```

The host `openstack1@7mode` represents the backend representing the Data ONTAP operating in 7-Mode system, and `openstack1@cDOT` represents the backend representing the clustered Data ONTAP system. Volumes can be migrated individually to the new backend, through the use of the **cinder migrate** CLI command. For example, suppose we have a volume with ID `781501e1-af79-4d3e-be90-f332a5841f5e` on the `openstack1@7mode` storage backend. We can move it to the `openstack1@cDOT` backend with the following CLI command:


```
cinder migrate 781501e1-af79-4d3e-be90-f332a5841f5e openstack1@cDOT
```

The command is asynchronous and completes in the background. You can check the status of the migration with the show command:

```
cinder show 781501e1-af79-4d3e-be90-f332a5841f5e
```

While a volume migration is in progress, Cinder commands from tenants that involve operations on the volume (such as attach/detach, snapshot, clone, etc) will fail.

During this process, the data is copied inefficiently through the cinder-volume node. This compares unfavorably to data-efficient migration techniques, but has the significant advantage that it can be completely non-disruptive (if live migration is supported) and it can be done in small increments of one Cinder volume at a time, so the operations can be distributed over periods when load is minimal.

If you are utilizing a hypervisor that does not support live migration of volumes and the volume is currently attached, it is necessary to detach the volume from the Nova instance before performing the migration. If the volume is the boot volume or otherwise critical to the operation of the instance, you need to shutdown the Nova instance before performing the migration.

Appendix A. Packaging

A.1. Packaging and Downloading Requirements

Refer to the following link for packaging and downloading requirements: http://wiki.openstack.org/Getopenstack

A.2. Installation and Uninstallation

Refer to the following link for install/uninstall-related information: http://wiki.openstack.org/Getopenstack

Refer to the following link for documentation on the E-Series SANtricity software: http://support.netapp.com/documentation/productlibrary/index.html?productID=61197

Refer to the following link for documentation on configuring **dm-multipath** on Linux: https://library.netapp.com/ecmdocs/ECMP1217221/html/GUID-34FA2578-0A83-4ED3-B4B3-8401703D65A6.html

A.3. Upgrading and Reverting

Refer to the following link for upgrading/reverting-related information: http://wiki.openstack.org/Getopenstack

A.4. Licensing

OpenStack is released under the Apache 2.0 license.

A.5. Versioning

Presently, there is no separate versioning system for NetApp's Cinder drivers, but are instead distinguished by the enclosing OpenStack release's versioning and release system.

A.6. Deprecated Drivers

In the OpenStack Havana release, NetApp deprecated a variety of management-integrated drivers that had been available in previous OpenStack releases. The driver names include:

```
cinder.volume.drivers.netapp.iscsi.NetAppCmodeISCSIDriver
cinder.volume.drivers.netapp.nfs.NetAppCmodeNfsDriver
cinder.volume.drivers.netapp.iscsi.NetAppISCSIDriver
cinder.volume.drivers.netapp.nfs.NetAppNfsDriver
```


The direct (and now unified) driver available in the Havana release provides equivalent functionality to these drivers for the various OpenStack services, but without the complex configuration and requirements to operate additional management software and infrastructure.

In situations where it is necessary to leverage the management-integrated drivers in a Havana-based OpenStack deployment, NetApp has ported the drivers to the Havana code base and made them available for download from a separate git repository located at https://github.com/NetApp/cinder/tree/stable/havana.

Appendix B. Troubleshooting

B.1. Common Problems

Common problems listed below are followed by the **cinder** or **nova** CLI command and possible reasons for the occurrence of the problem.

1. Create volume operation fails with an error status.

```
cinder create size_gb
```

- No space left on the NetApp volume or NetApp volume does not have sufficient free space to host the specified OpenStack volume. Here NetApp volume refers to a FlexVol volume inside the configured Storage Virtual Machine (SVM aka Vserver) for Clustered Data ONTAP driver. It refers to NetApp volume names configured in Cinder as parameter `netapp_volume_list` or NetApp volumes in the configured vFiler unit as parameter `netapp_vfiler` or system wide NetApp volumes for 7-mode storage system.
- Cinder API service is down.
- Cinder scheduler service is down.
- Cinder scheduler reports sufficient available space on NetApp backend but Cinder volume fails to create backend:
  - The Grizzly-based iSCSI driver for Clustered Data ONTAP and 7-mode driver report available space as infinite and hence failure may occur when no NetApp volume can host the OpenStack volume.
  - The Grizzly-based NFS driver Clustered Data ONTAP and 7-mode report available space as the sum of available space of all configured NFS exports and hence failure may occur when no single NetApp volume can host the OpenStack volume of the given size.
  - The Havana-based iSCSI and NFS driver for Clustered Data ONTAP report the available capacity for largest NetApp volume in the configured Storage Virtual Machine (SVM aka Vserver). Capacity mismatch might fail volume creation.
  - The Havana-based iSCSI and NFS driver for 7-mode storage system report the available capacity as sum of available space of all configured NetApp volumes and hence failure may occur when no single NetApp volume can host the OpenStack volume of the given size.
- The Havana based NFS driver for Clustered Data ONTAP has the configuration option `netapp_vserver` to specify the Storage Virtual Machine (SVM aka Vserver) to use for provisioning. It may so happen that the NFS exports specified in the configuration and the NetApp volumes in the SVM do not coincide.
- NFS service is not running on the NetApp storage server in case of NFS drivers.
- NFS mount for exports failed on the Cinder node due to incorrect export policy or insufficient privileges in case of NFS drivers.

- NetApp volumes getting full because snapshots occupying storage space.
- NetApp volumes are shared between OpenStack Cinder and other client side applications.

2. Create volume with volume-type operation fails with error status.

```
cinder create --volume-type volume_type size_gb
```

- All the reasons mentioned under Item 1 in this appendix.
- The NetApp backend(s) with available space do not support at least one of the extra-specs bound to the volume-type requested. Hence, it does not return the extra spec in volume stats call to the Cinder scheduler.
- In Clustered Data ONTAP drivers operation fails due to:
  - No NetApp volume supports all extra specs bound to the volume-type.
  - The configured storage admin user does not have sufficient privileges to query specific storage service catalog features required to support the volume-type configuration.
  - The configured IP address/host name is on a SVM network interface but the volume-type support requires cluster wide API access.

3. Create volume from image-id operation fails with an error status.

```
cinder create --image-id image-id size_gb
```

- All the reasons mentioned under Item 1 in this appendix.
- The Grizzly-based NFS driver does not have the mentioned operation supported. It may be required to use the latest code from the NetApp git repository, from the `stable/grizzly` branch in order to get a supported version.
- Glance related services are down.
- The image could not be downloaded from glance because of download error.
- Havana-based NFS drivers may experience a shortage in storage capacity due to space occupied by image cache files. Image cache files are files with prefix img-cache, and are periodically cleaned by the driver.

4. Create volume from image-id with volume-type operation fails with an error status.

```
cinder create --image-id image-id --volume-type volume_type size_gb
```

- All the reasons mentioned under Items 1, 2, and 3 in this appendix.

5. Create snapshot operation fails with an error status.


```
cinder snapshot-create volume-id
```

- The FlexClone license is not installed.
- The NetApp volume hosting the source OpenStack volume does not have sufficient available space.
- Any maintenance operation by a storage admin directly at the storage backend causing LUN or file unavailability.

6. Create volume from snapshot operation fails with an error status.

```
cinder create --snapshot-id snapshot-id size_gb
```

- All reason mentioned under Items 1 & 5 in this appendix.

7. Create cloned volume operation fails with an error status.

```
cinder create --source-volid volume-id size_gb
```

- All reason mentioned under Items 1 & 5 in this appendix.

8. Volume attach operation in nova fails.

```
nova volume-attach instance-id volume-id path size_gb
```

- iSCSI drivers:
  - The iSCSI license may not be installed.
  - The iSCSI service on the **nova-compute** host may not be running.
  - The iSCSI portal can not be found. No network interface of type iSCSI has been created.
  - The network is not reachable due to firewall, configuration, or transient issues.

9. Volume extend operation fails for Havana based drivers.

```
cinder extend volume-id new_size_gb size_gb
```

- The NetApp volume hosting the OpenStack volume has insufficient space.
- iSCSI drivers
  - Reason mentioned under Item 5 in this appendix.
- NFS drivers

- The disk image format of the Cinder volume is not `raw` or `qcow2`.

10. Volume upload to image operation fails.

```
cinder upload-to-image volume-id image size_gb
```

- The Glance service is down.
- All reasons mentioned under Item 8 in this appendix.

11. Volume backup and restore operation fails.

```
cinder backup-create volume-id size_gb
cinder backup-restore volume-id size_gb
```

- The Cinder backup service is not running.
- All reasons mentioned under Item 8 in this appendix.

12. Volume migration operation fails.

```
cinder migrate volume-id host
```

- All reasons mentioned under Item 8 in this appendix.

B.2. Triage and Data Collection

Please use the NetApp OpenStack Communities site to track or report issues related to Cinder. In case of issues, the data can be collected from logs printed by each of the below mentioned process. Logs need to be collected for Cinder related processes. For Glance and Nova verifying the service up status is sufficient.

Cinder processes

- **cinder-api**
- **cinder-backup**
- **cinder-scheduler**
- **cinder-volume**

Manila processes

- **manila-api**
- **manila-scheduler**
- **manila-share**

Nova processes

- nova-api
- nova-scheduler
- nova-cpu

Glance processes

- glance-api
- glance-registry

Swift processes

- swift-object-server
- swift-object-replicator
- swift-object-updator
- swift-object-auditor
- swift-container-server
- swift-container-replicator
- swift-container-updator
- swift-container-auditor
- swift-account-server
- swift-account-replicator
- swift-account-auditor

B.3. References

The following references were used in this paper:

- NIST Cloud Definition http://www.nist.gov
- OpenStack Foundation http://www.openstack.org
- Cloud Data Management Interface (CDMI) http://www.snia.org/cdmi

For additional information, visit:

- For more information on the operation, deployment of, or support for NetApp's OpenStack integrations: http://communities.netapp.com/groups/openstack
- For source code for OpenStack, including NetApp contributions, available through Github: http://www.github.com/openstack

- For more information about NetApp's participation in OpenStack, visit the NetApp Community site: http://www.netapp.com/openstack
- For more information about OpenStack history: http://www.openstack.org or http://en.wikipedia.org/wiki/OpenStack

B.4. Support

Community support is available through the NetApp Communities site: http://communities.netapp.com/groups/openstack.

NetApp's Interoperability Matrix (IMT) details components and versions of qualified configurations. Since the majority of OpenStack provides a control plane it's not presently explicitly called out, but host operating system, hypervisor, and other components involved in the data path should be noted.

http://support.netapp.com/matrix/

The NetApp OpenStack team presently intends to provide maintenance of the two most recently released versions of OpenStack. For example, during Juno development, all code that is part of the Havana and Icehouse official branches are supported. Upon Juno release, direct maintenance for Havana would be dropped and maintenance for Icehouse is added.

NetApp can provide customized support options for production requirements. For more information, please contact your sales team.

About NetApp

NetApp creates innovative storage and data management solutions that help our customers accelerate business breakthroughs and achieve outstanding cost efficiency. Our dedication to principles of simplicity, innovation, and customer success has made us one of the fastest-growing storage and data management providers today.

Version History

Version	Date	Document Version History
Version 1.0	April 2013	Initial OpenStack Deployment and Operations Guide.
Version 1.1	April 2013	Minor enhancements.
Version 2.0	December 2013	Updated with OpenStack Havana changes - including the new unified driver support, and enhanced instance creation for boot from volume.
Version 2.1	December 2013	Addition of Swift on E-Series deployment information.
Version 2.2	January 2014	Minor enhancement to clarify NFS configuration.
Version 2.3	January 2014	Addition of Glance on E-series deployment information, add note of additional required Glance configuration in support of enhanced instance creation feature.
Version 3.0	June 2014	Major revision for Icehouse release; theory of operation and concept discussion for Cinder, Glance, Nova, Swift; expanded configuration examples and deployment advice.
Version 4.0	November 2014	Update for Juno release
Version 5.0	April 2015	Update for Kilo release; added chapter on Manila, Cinder Fibre Channel drivers.