

**ORACLE®**

ZFS STORAGE  
APPLIANCE

# Using the Oracle ZFS Storage Appliance as Storage Back End for OpenStack Cinder

An Example Architecture Using Oracle ZFS Storage Appliance  
and Oracle Solaris

ORACLE WHITE PAPER | DECEMBER 2015



**ORACLE®**

## Table of Contents

Introduction	3
Introducing the OpenStack Project	5
Providing a Reference Model for Cloud Services	5
Defining Cloud Deployment Models	5
Private Cloud	5
Community Cloud	6
Public Cloud	6
Hybrid Cloud	6
OpenStack Main Services Overview	6
Compute Service	7
Block Storage Service	7
Networking Service	7
Image Service	7
Object Storage	7
Identity Service	8
Dashboard Service	8
OpenStack on the Oracle Solaris Operating System	8
Architecture of the Oracle OpenStack Implementation on Oracle Solaris	9
Meeting Storage Requirements in an OpenStack Environment	10
Gaining Flexibility with the Oracle ZFS Storage Appliance Architecture	10
Storage Pool	11
Project	12
Shares	12
Data Services	12
Snapshots	12
Clones	12
Remote Replication	12
Shadow Migration	12
Data Security	13
Virus Protection	13
Encryption	13
Analytics	13
Using the Oracle ZFS Storage Appliance in an Oracle OpenStack Environment	13
Designing an OpenStack Architecture with Oracle Solaris and Oracle ZFS Storage Appliance	14
Establishing a Network Architecture	16
Installing and Configuring Oracle OpenStack in Oracle Solaris	20
Implementing the Network Configuration	21
Setting Up Storage Network Interfaces	21
Setting Up the OpenStack Network Connections	22
Setting Up OpenStack Cinder and the Oracle ZFS Storage Appliance	24

Using iSCSI as the Communication Link	24
Understanding the Role of the Oracle OpenStack ZFSSA Cinder Driver	25
Installing and Configuring the Oracle OpenStack ZFSSA Cinder Driver	28
Setting Up Network Interfaces on the Oracle ZFS Storage Appliance	28
Setting Up Storage Connections on Oracle Solaris OpenStack Nodes	29
Setting Up the Cinder Configuration File	29
Setting Up Cinder Service with Oracle ZFS Storage Appliance	
Cluster Configurations	31
Using Multiple Cinder Storage Back-end Definitions	33
Administering Oracle ZFS Storage Appliance Volumes in the OpenStack Environment	35
Best Practices for Deploying Oracle OpenStack on Oracle Solaris with Oracle ZFS Storage Appliance	38
Planning for Growth with a Multiple Network Architecture	38
Incorporating Uniformity into the Network Design	39
Using Logical Hostnames	40
Using NTP to Synchronize OpenStack Nodes Time	40
Employing a Uniform Symmetrical Hardware Design	40
Using Configuration to Manage Storage Capacity for Various Cinder Guests	41
Use Multiple Backend Volume Definitions	41
Increase Block Storage API Service Throughput	42
Managing Volumes	43
Use Volume Labels	43
Troubleshooting Configurations	44
Troubleshooting iSCSI Connectivity	45
Troubleshooting Using Analytics	47
Appendix A: Cinder Configuration File	48
Appendix B: Oracle OpenStack on Oracle Solaris Open Issues	50
References	51



## Introduction

As the use of cloud computing continues to grow, so too do industry expectations for increasing corporate spending on IT cloud implementations. The cost benefits of being able to standardize on server hardware and software and share those resources in a more economical way within a cloud service model are key factors for the uptake of cloud computing. Furthermore, cloud computing offers flexibility to adapt to organizational changes, and control over critical issues like security and capacity management.

The OpenStack flexible Cloud platform has helped with the wide adoption and deployment of cloud computing. OpenStack offers seamless scaling for both private and public clouds.

Designing and implementing a cloud using the OpenStack software is a challenging task and requires a thorough understanding of the requirements and needs of cloud customers/users. The OpenStack platform is very flexible in accommodating and implementing such requirements in a proper cloud implementation.

Cloud deployments often focus around sharing compute, network infrastructure resources and application services. The development of the technology of providing storage capacity resources based upon organization level Service Agreements around performance, availability, reliability, capacity and costs has not kept up with the speed of development of modeling SLAs for compute and network cloud services.

This paper will describe how the Oracle ZFS Storage Appliance and its data services can be utilized to implement multi-level storage service level agreement (SLA) requirements in an enterprise OpenStack-based cloud architecture using an Oracle Solaris SPARC compute platform.

The described architecture provides a secure and highly available storage subsystem architecture for OpenStack cloud implementations. Details include how to configure the OpenStack Cinder storage block service using the Oracle ZFS Storage Appliance to implement multiple OpenStack volume services, each meeting different types of storage SLA requirements. These SLA requirements include use of data encryption.

A dual data path outside the OpenStack Neutron layer, between the OpenStack Compute nodes and the Oracle ZFS Storage Appliance, is used to create a highly available, secure connection between virtual machine (VM) instances and the storage subsystem, avoiding any possible bottlenecks going through the OpenStack Neutron network layers.



This paper also offers best practices for fully utilizing the rich Oracle ZFS Storage Appliance features in an OpenStack architecture.

NOTE: References to Sun ZFS Storage Appliance, Sun ZFS Storage 7000, and ZFS Storage Appliance all refer to the same family of Oracle ZFS Storage Appliances.



## Introducing the OpenStack Project

The OpenStack project is an ongoing development effort to provide an open source cloud software-computing platform that can provide to organizations cloud-computing services running on commodity hardware. OpenStack controls and manages large pools of compute, storage and networking resources throughout the data center. The OpenStack Foundation promotes and manages the development, distribution, and adoption of the OpenStack cloud operating system. Many organizations, including Oracle, support and contribute to the OpenStack project.

### Providing a Reference Model for Cloud Services

Cloud computing has gained market attention. To create a common understanding and a reference model for comparing various cloud services, the United States government's National Institute of Standards and Technology (NIST) has described a reference model, related characteristics, service models and deployment models for cloud computing.

NIST provides the following definition of cloud computing:

"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."

Within cloud computing, NIST distinguishes three different cloud service models:

- **Software as a Service (SaaS)** – The capability for consumers to use the provider's applications running on a cloud infrastructure.
- **Platform as a Service (PaaS)** – The capability for consumers to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services and tools supported by the provider.
- **Infrastructure as a Service (IaaS)** – The capability for the consumer to provision processing, storage, networks and other fundamental computing resources, where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications.

The OpenStack cloud implementation follows the NIST IaaS services model through a set of interrelated software services.

### Defining Cloud Deployment Models

OpenStack offers several cloud deployment models, commonly known as Public, Private and Hybrid models. NIST, however, distinguishes the following four different cloud deployment models, which all are supported by the OpenStack cloud offering.

#### Private Cloud

As defined by NIST, the private cloud is provisioned for exclusive use by a single organization comprising multiple consumers (that is, business units). It may be owned, managed, and operated by the organization,

a third party, or some combination of them, and it may exist on or off premises. Separation of data and services between consumers is important and no access to external resources is available.

### Community Cloud

NIST defines this type of cloud as an infrastructure provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns (such as mission, security requirements, policy, and compliance considerations). It may be owned, managed, and operated by one or more of the organizations in the community, a third party, or some combination of them, and it may exist on or off premises.

### Public Cloud

The public cloud model is at the opposite spectrum of the private cloud model and is defined by NIST as the cloud infrastructure provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud provider. From a network and data separation perspective, this is the most challenging type of cloud deployment models.

### Hybrid Cloud

The hybrid cloud model is defined by NIST as the cloud infrastructure being a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability (such as cloud bursting for load balancing between clouds).

Understanding which deployment model is going to be used in your organization helps you to understand, define and scope security, availability and performance requirements of the envisioned cloud solution. It will also help you to determine physical network requirements and architecture, as this paper will explore later.

## OpenStack Main Services Overview

OpenStack provides an IaaS deployment model type solution by using a set of modular services. Each service offers an Application Programming Interface (API) that enables the services to communicate with each other and offer public API functions for third party applications. There are a number of mandatory core services and some optional services that can be installed, depending on the customer's cloud implementation requirements. The following diagram shows the most common OpenStack services.

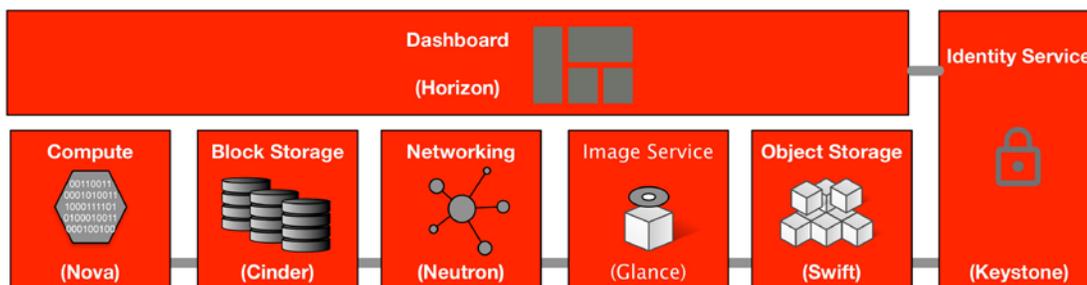


Figure 1. OpenStack components



Authentication and access control are provided by the OpenStack Identity Service (Keystone) for all other OpenStack services. Configuring, managing and monitoring the OpenStack services is provided by the Dashboard (Horizon) service. This is available as a Browser User Interface (BUI), enabling the administrator's remote access functionality.

### **Compute Service**

The OpenStack Compute Service (Nova) manages the lifecycle of compute instances in an OpenStack environment. It manages functions including spawning, scheduling and decommissioning of virtual machines on demand. The compute service facilitates this management through an abstraction layer that interfaces with supported hypervisors.

### **Block Storage Service**

The OpenStack Block Storage Service (Cinder) provides persistent block storage to running instances for both the Instance boot image and any specific application block storage volumes. Cinder is responsible for managing the lifecycle of the volumes, including creation, attachment to guest instances, snapshot, cloning and deleting of volumes. The pluggable driver architecture facilitates the creation and management of those volumes. The Oracle-provided Cinder driver plug-in for the Oracle ZFS Storage Appliance is the focus of this paper.

### **Networking Service**

The OpenStack Networking Service (Neutron) provides various networking services to cloud users (tenants) such as IP address management, Domain Name Service (DNS), Dynamic Host Configuration Protocol (DHCP), load balancing, and security (network access rules, like firewall policies). Neutron provides a framework for software-defined networking (SDN) that allows pluggable integration like the Oracle Elastic Virtual Switch (EVS) plug-in for Oracle Solaris.

OpenStack Networking enables cloud tenants to manage their guest network configurations. When you are setting up a (virtual) network architecture, pay careful attention to network traffic isolation, availability, integrity, and confidentiality.

### **Image Service**

The OpenStack Image Service (Glance) provides disk image management services. The Image Service provides image discovery, registration, and delivery services to the Compute Service as needed.

Trusted processes for managing the lifecycle of disk images, as well as other data security aspects, are required.

### **Object Storage**

The OpenStack Object Storage Service (Swift) provides support for storing and retrieving arbitrary unstructured data in the cloud. The Object Storage Service provides a RESTful, HTTP-based API. The service provides a high degree of resiliency through data replication and can handle petabytes of data. Object storage is typically used for storing large, static data objects, including media files, virtual machine images and backup images.

## Identity Service

The OpenStack Identity Service (Keystone) provides an authentication and authorization service for all other OpenStack services. The Identity Service has pluggable support for multiple forms of authentication.

## Dashboard Service

The OpenStack Dashboard (Horizon) provides a web-browser-based user interface (BUI) for both cloud administrators and cloud tenants. This BUI enables administrators and tenants to provision, manage, and monitor cloud resources.

## OpenStack on the Oracle Solaris Operating System

The Oracle OpenStack software for Oracle Solaris is available as Oracle Solaris software packages. They are fully integrated into the Oracle Solaris Image Package Repository, and thus provide integral release update functionality. (See the References section at the end of this document for locations.) The packages contain all the OpenStack services, including the following modules:

- **Compute (Nova):** Takes advantage of Oracle Solaris Zones, supporting both native non-global zones and a new feature of Oracle Solaris 11.2 called Oracle Solaris Kernel Zones. Kernel Zones enable greater isolation and independence with a separate kernel instance.
- **Networking (Neutron):** Uses Oracle Solaris virtual networking and the new Elastic Virtual Switch (EVS) feature in Oracle Solaris. EVS extends network virtualization to virtual switches and spans those virtual switches across multiple physical servers or compute nodes as if they were a single switch.
- **Block Storage (Cinder):** The Oracle OpenStack for Oracle Solaris distribution contains two Oracle block storage Cinder drivers. One supports the use of the ZFS file system on an OpenStack Oracle Solaris node as back end for volumes managed by Cinder. The other driver supports Oracle ZFS Storage Appliance as volume repository for Cinder. Both drivers take advantage of ZFS's numerous capabilities, including instant snapshot and cloning, encryption, redundancy and data integrity. The Cinder driver for the Oracle ZFS Storage Appliance option is the focus of this white paper.
- **Image Management (Glance):** The Image service provides disk management services, like image discovery, registration and delivery services to the Compute service as needed. A new form of image archive introduced in Oracle Solaris 11.2 is called Unified Archives. This is the primary integration point for Glance. It allows for fast creation and cloning of system images, including virtualization, into the cloud.
- **Configuration Management (RAD):** The Remote Administration Daemon (RAD) is a new feature in Oracle Solaris 11.2. This framework provides tools and a protocol for remote administration on the operating system and is the seamless glue that is used to create new compute nodes, configure networking, or provision storage.

- **Service Management (SMF):** All the OpenStack services have been integrated with the Service Management Facility, providing fast startup and recovery should a service fail for whatever reason.

## Architecture of the Oracle OpenStack Implementation on Oracle Solaris

The OpenStack implementation on Oracle Solaris utilizes the kernel zone partitioning functionality to create the OpenStack Guest instances. The Oracle Solaris Zones partitioning is an Operating System (OS) virtualization method providing an isolated and secure environment for running applications. A zone provides isolation for application execution, meaning that no interaction can occur between processes running on different zones. Likewise, these processes have no access to physical resources of the platform on which the zones are configured.

With the Oracle OpenStack on Oracle Solaris implementation, Oracle Solaris zone instances are used as OpenStack guest instances and are created and administered by the related OpenStack services.

Because the Oracle Solaris Zone architecture is not using a hypervisor, zones offer a near native performance to applications.

The following diagram illustrates the interaction between the OpenStack Nova service and the Oracle Solaris zones. Other OpenStack services like Neutron and Cinder provide the Oracle Solaris Zone administrative interface with the appropriate device specification for the zone configuration file of the related zone instance.

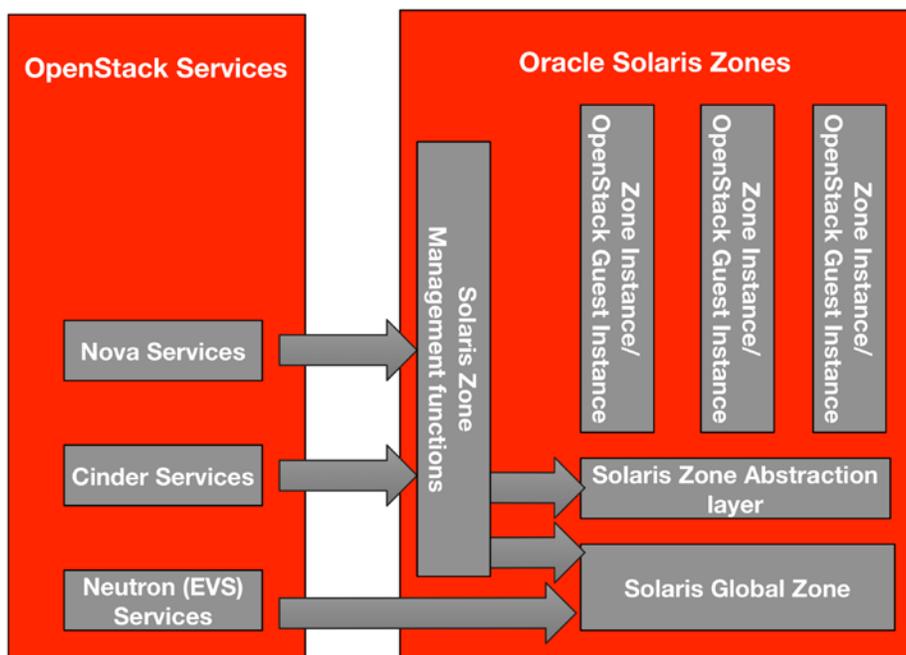


Figure 2. Interaction of Oracle Solaris Zoning and OpenStack

Note that after an OpenStack guest instance is started, no OpenStack layers are involved on a compute node, either in the block I/O path or in the compute resource layers. Some network traffic might flow



between the OpenStack node where the guest instance is running and the OpenStack node where the Neutron network service is active.

## Meeting Storage Requirements in an OpenStack Environment

Cloud platforms provide both Compute and Network as a Service functions with SLA-type properties to customize the services to consumer requirements. This is not always the case for storage services in a cloud environment. The implementation of storage SLA-type services has not kept up with the cloud compute and network services' developments. This is especially true for block-based storage services. It is still the cloud administrator's responsibility to match the consumer's storage SLA requirements –like availability, security, performance and cost – with the various low-level I/O properties – such as RAID levels, cache settings, and type of I/O interface used – of the storage subsystems.

With the OpenStack Cinder block storage service, multiple storage back-end definitions can be created. The Oracle Solaris ZFSSA Cinder OpenStack driver enables the administrator to set up different back-end definitions, each definition having a specific set of I/O properties that match a certain set of storage SLAs. For example, one definition might be customized for a storage back-end optimized for high performance, low latency or high availability dedicated to mission-critical transactional services. Another back-end definition can be optimized for archiving large media type files.

This Cinder driver functionality enables an Oracle ZFS Storage Appliance to be used for storage capacity consolidation in an OpenStack cloud environment while still providing storage capacity provisioning tailored to a variety of SLA types.

## Gaining Flexibility with the Oracle ZFS Storage Appliance Architecture

The Oracle ZFS Storage Appliance combines multiple protocol connectivity, data services for business continuity, and ease of management into a single storage appliance. The Oracle ZFS Storage Appliance supports Network File System (NFS), Common Internet File System (CIFS), Internet Small Computer System Interface (iSCSI), InfiniBand (IB), and Fibre Channel (FC) protocols for data access. The Oracle ZFS Storage Appliance also supports Network Data Management Protocol (NDMP) for backing up and restoring the data. The Oracle ZFS Storage Appliance is available either as single head or a clustered head for high availability. Its browser-based user interface offers intuitive, easy navigation, with layered, detailed information displays that ease management activities.

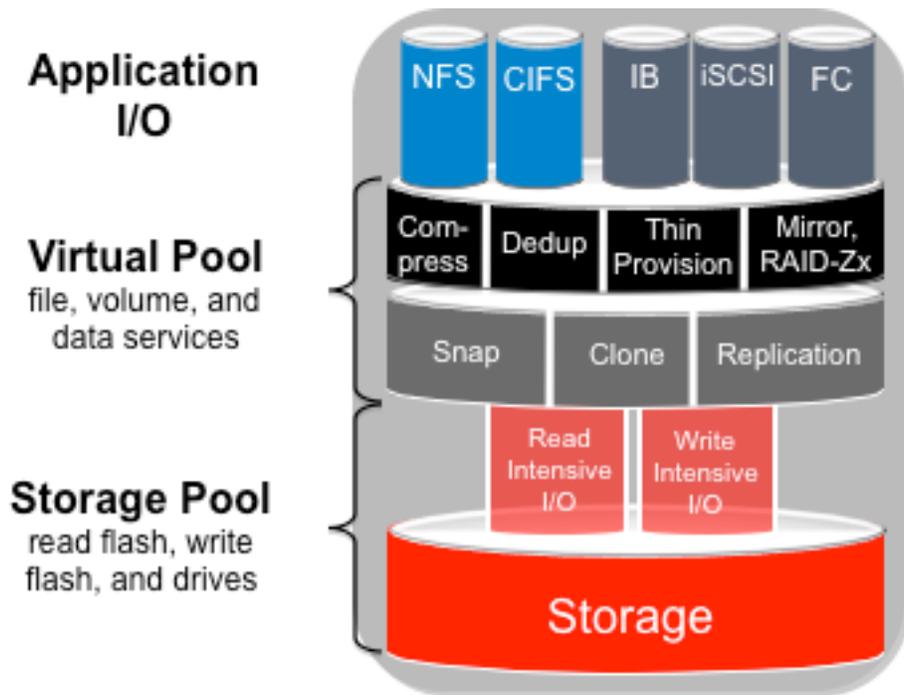


Figure 3. Oracle ZFS Storage Appliance Stack

The Oracle ZFS Storage Appliance architecture utilizes the Hybrid Storage Pool (HSP) model where the integrated direct random access memory (DRAM), flash and physical disks are seamlessly integrated for efficient data placement. Based on the application I/O request and pattern, Oracle ZFS Storage Appliance automatically handles the data movement among these tiers. This model ensures scalable and predictable performance when consolidating multiple applications with different workloads onto a single Oracle ZFS Storage Appliance.

The storage also includes a powerful performance monitoring tool called Analytics which provides details about the performance of components such as the network, storage, file systems, and client access. The Oracle ZFS Storage Appliance also offers a variety of RAID protections to balance capacity, protection, and performance requirement of the applications.

Test results of industry-standard benchmarks like SPECsfs, SPC-1 and SPC-2 illustrate the superior benefits of this architectural design, making the Oracle ZFS Storage Appliance highly cost effective.

The following features and constructs of the Oracle ZFS Storage Appliance provide the building blocks for its high functionality and are key to its suitability to the OpenStack cloud environment. They are provided here as a quick reference.

### Storage Pool

The storage pool, similar to a volume group, is created over a set of physical disks. File systems and LUNs are then created over the storage pool. One or more storage pools are created over the available physical disks and flash drives for use as secondary cache are assigned. The storage pool is configured with a RAID layout such as Mirrored, RAID-Z (single parity), RAID-Z2 (dual parity), and so on.

## Project

Capacity can easily be managed by grouping related file systems and/or LUNs into so-called projects. Projects share capacity from a pool of disks. If needed, a quota can be set up for each file system so that IT staff can easily balance capacity over various file systems without having to move them around.

A project can be considered a “consistency group.” A project defines a common administrative control point for managing shares. All shares within a project can share common settings, and quotas can be enforced at the project level in addition to the share level.

## Shares

Shares are file systems and LUNs. They are exported over the Oracle ZFS Storage Appliance data protocols to clients of the Appliance. File systems export a file-based hierarchy and can be accessed over CIFS, NFS, HTTP/WebDav, and FTP. Both CIFS and NFS have file-sharing capabilities, using locking mechanisms to prevent concurrent updates from different clients. LUNs export block-based shares and can be accessed over iSCSI or FC. The application server needs to create a file system on a LUN. File system shares share free capacity from the pool to which they belong. Quota can be used to limit the amount of capacity used by a file system share or by a project. Projects and shares have compression, encryption and de-duplication options. LUNs can be thin provisioned.

## Data Services

The Oracle ZFS Storage Appliance offers a rich set of data services. Remote replication, snapshots and cloning are key features for a complete disaster recovery (DR) and business continuity solution. For further details, please refer to the documents listed in the Reference section of this paper.

## Snapshots

The Oracle ZFS Storage Appliance has unlimited snapshot capability. Snapshots are the read-only point-in-time copies of a file system. They are instantaneously created, initially with no space allocated. Blocks are allocated as changes are made to the base file system (copy-on-write). Snapshots are either initiated manually or can be automatically scheduled at specific intervals. These snapshot data sets can be directly accessed for any backup purposes. Taking project snapshots is the equivalent of performing snapshots on all shares within the project.

## Clones

The Oracle ZFS Storage Appliance supports an unlimited number of clones. A clone is an instantaneously created read and writable copy of a snapshot. One or more clones can be created from a single snapshot. These clones are presented to the users as a normal file system(s). All the regular operations are allowed on the clones, including taking a snapshot from the clone. The clones are typically used in test, development, QA, and backup environments.

## Remote Replication

Data is replicated from the primary to the secondary location, with data blocks asynchronously streamed to the secondary location. Data replication can be set up between two nodes, a primary and secondary site, or within the same node between two different pools. Source data is modified at the granularity of a ZFS transaction; therefore the data is always consistent. Modified data is replicated to the secondary site, which ensures the data at the secondary site is also consistent.

## Shadow Migration



The Shadow Migration Data Service offers the ability to migrate data from any NFS volume to an NFS volume on the Oracle ZFS Storage Appliance. Once set up, the original NFS volume is under the control of the Oracle ZFS Storage Appliance and access to the volume is handled by the Appliance. The Oracle ZFS Storage Appliance moves the file structure, including the ACLs, from the source location to the new NFS volume on the Appliance without any interruption of services to the clients using the NFS volume.

### **Data Security**

Managing and monitoring data access tasks are provided by LDAP and Active Directory (AD) services together with functions to set up local user accounts. User access can be monitored through the extensive Analytics functionality, with drill-down capabilities to observe user access to projects and shares.

### **Virus Protection**

The Oracle ZFS Storage Appliance has an antivirus protection service to provide to file shares protection against virus infection attacks.

### **Encryption**

The Oracle ZFS Storage Appliance offers transparent data encryption for projects or individual shares inside of projects. The Appliance offers both local key management and central key management administration using the Oracle Key Manager (OKM) system.

### **Analytics**

Analytics is an advanced function in Oracle ZFS Storage Appliance offering administrators information on various storage performance statistics. It is the only storage platform offering such unique capabilities through the DTrace function. It visualizes run-time performance statistics of the storage subsystem, enabling quick and easy diagnosis of application storage workloads and providing valuable information for making capacity planning decisions.

## **Using the Oracle ZFS Storage Appliance in an Oracle OpenStack Environment**

The use of the Oracle ZFS Storage Appliance to consolidate the storage capacity required for an OpenStack implementation has many advantages, including centralized management, with volume management well integrated with the OpenStack Cinder functionality. For instance, both Cinder snapshot and 'create volume from snapshot' functions take full advantage of the Oracle ZFS Storage Appliance's snapshot and clone data services. As previously noted, combining the option to use multiple back-end definitions for OpenStack Cinder and using the Oracle ZFS Storage Appliance properties that define specific volume I/O characteristics, different OpenStack volume types can be defined, each meeting specific customer and/or application type SLAs including properties like security, availability, performance and costs.

For an environment requiring the use of encryption, the Oracle ZFS Storage Appliance encryption service can be used to offload the encryption burden from the OpenStack environment.



## Designing an OpenStack Architecture with Oracle Solaris and Oracle ZFS Storage Appliance

The OpenStack cloud environment provides a framework to design a virtual compute and network environment on top of a number of physical servers and a physical network infrastructure. Before setting up the OpenStack software framework, a detailed understanding is needed on the number of virtual servers, as well as storage, compute and network resources that are required for your particular application environment. Because a good understanding of the OpenStack components and their interaction is a must, it is recommended that you first set up an OpenStack sandbox environment to gain experience on its functionality before deploying it in a production environment.

One of the most complex elements is the design and configuration of the OpenStack virtual network using the OpenStack Neutron services.

Most OpenStack architecture examples and demos use a configuration in which only a single network interface is used on each node. All network communication to and from the virtual compute instances as well as internal OpenStack, SSH management, and OpenStack API use the same interface. This can quickly lead to network bandwidth congestion and long service response times. Using only one network interface is highly unsuitable for a production environment.

As with the traditional enterprise server network architecture, it is good practice to identify the different types of networking traffic streams and keep them separated from each other. Company security requirements, regulatory requirements, performance requirements and availability requirements will lead to classification of different types of network traffic and subsequently to isolation and performance requirements. Consider separating subnets for: strict administrative access, secure exclusive access to data repositories by certain applications, VMs' external internet access, and OpenStack internal data traffic.

Be particularly careful to avoid creating bottlenecks or single points of failure on physical network ports by mapping multiple virtual data streams/subnets over a single physical network.

Start by identifying the different types of network traffic in your OpenStack virtual IT infrastructure and how each would have to be mapped on a reliable physical network to satisfy the identified network security and performance requirements.

OpenStack distinguishes four different types of networks:

- Network traffic between VMs and networks outside the OpenStack virtual environment, like the corporate intranet or the external worldwide internet. In OpenStack documentation, this is often referred to as the External network.
- Network traffic between VMs themselves; in the OpenStack documentation, it is referred to as the Guest or Tenant network.
- Network traffic used by the OpenStack framework components to manage and monitor the virtual components within the OpenStack framework, referred to as data using the API network.
- Administrative access to the physical nodes of the OpenStack system. OpenStack documentation refers to this type of network as the Management network.

The following figure illustrates the OpenStack network model.

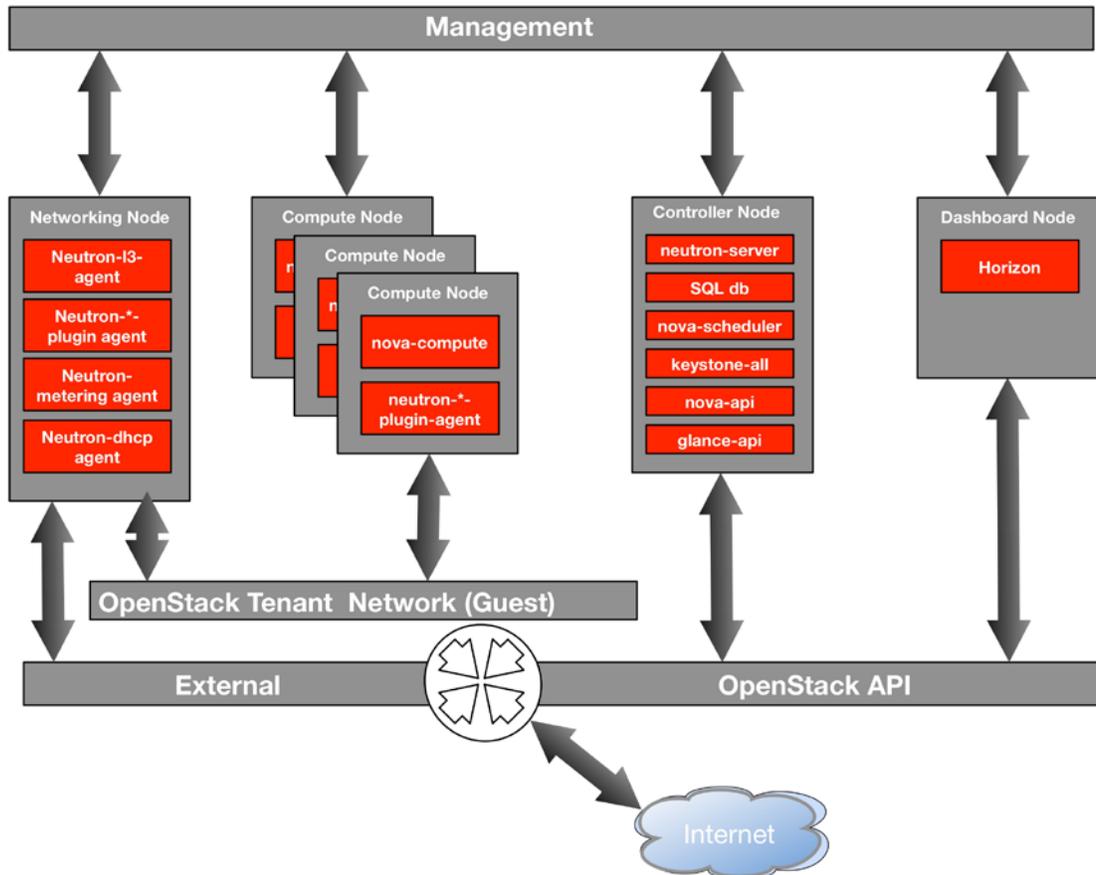


Figure 4. OpenStack network model

What is missing from the OpenStack network definitions is a separate network to be used by the compute nodes to access the storage subsystem(s) using the Cinder block storage service. In the previous diagram, either using the management interface or the external interface would accomplish this. Neither of the two options provides secure access to the data with data streams separated from the other data traffic functions. In both cases, all storage I/O traffic would have to go through the Neutron layers on the OpenStack networking node.

In the network architecture used in this paper, an extra storage network subnet – for network traffic between VMs and the VMs' application datasets on external storage – is used as shown in the next diagram, to separate all storage data traffic from the rest of the data networks.

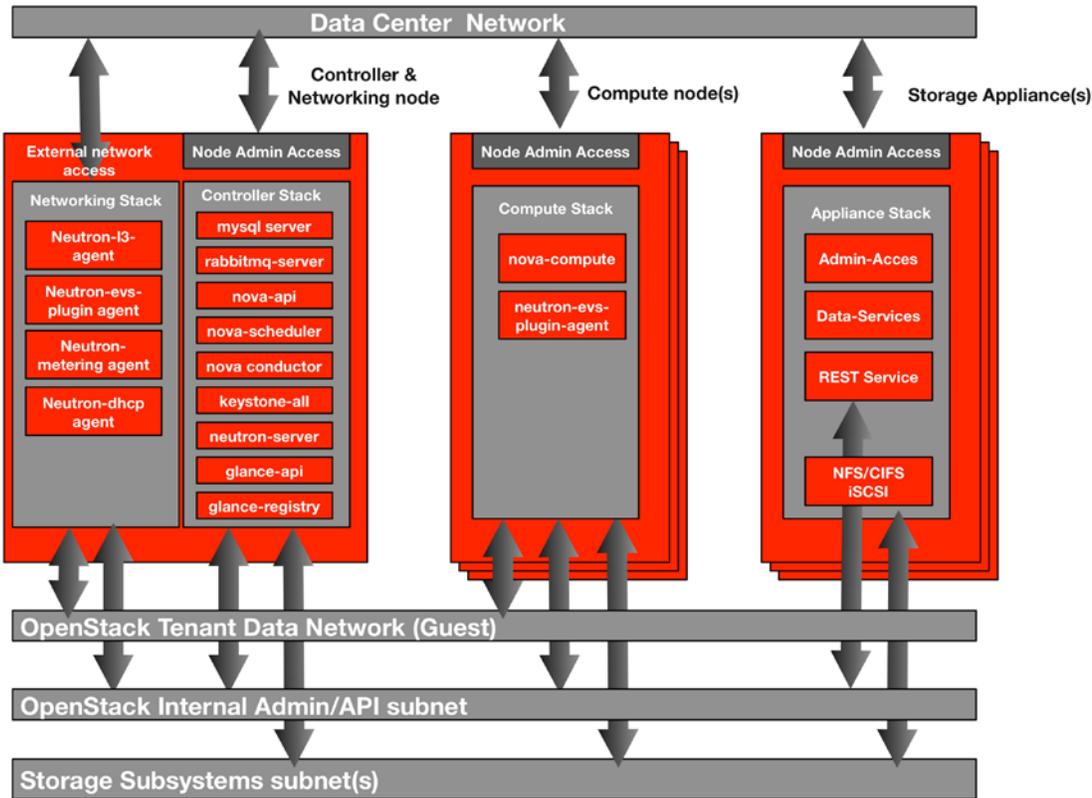


Figure 5. OpenStack network architecture used in this paper

In the network architecture described in this paper, the OpenStack external network function is configured using the Data Center Network, so the OpenStack external access and OpenStack admin access are both configured using the Data Center Network.

Based on performance, reliability and security requirements described previously, a physical infrastructure can be designed.

Note that from the previously listed categories of data traffic, only the data traffic between VMs on the Tenant Network remains within the virtual environment of the OpenStack architecture. In OpenStack, networks that host traffic between VMs are referred to as Tenant or Guest networks. The network(s) carrying the other categories of data traffic are referred to as Provider networks. Provider networks map directly to a physical network in the data center. Thus, the physical network must already exist before an OpenStack framework can be installed and configured. In many examples used in current OpenStack documentation, these Provider networks are mapped on a single physical network and a single subnet using a single network port on each OpenStack node.

### Establishing a Network Architecture

The OpenStack architecture described in this paper deploys the Oracle OpenStack for Oracle Solaris on Oracle SPARC servers with the following configured segregated networks:

- A subnet used for administrative access to each node. In this subnet, IP addresses are used to access a node for administrative purposes, like installing or updating software packages and performing administrative tasks on the global Oracle Solaris instance.

This subnet is also used to configure all OpenStack API-based data services. It is used to expose both the API to users of the OpenStack cloud and the listener endpoints of the various OpenStack services like Neutron, Keystone and Glance. If the listener endpoints cannot be exposed for security reasons, then configure these on an isolated separate subnet.

The ILOM management connections of each node are also connected to this subnet, each with its own IP address.

- Two physical separated subnets are used to form redundant access to the Oracle ZFS Storage Appliance's storage subsystems. Data repositories are made available to the VM instances through the Cinder iSCSI interface.
- A physical network to map the OpenStack tenants' subnets and route them to the data center network so VMs can access the data center network.

The following diagram shows the physical network connections.

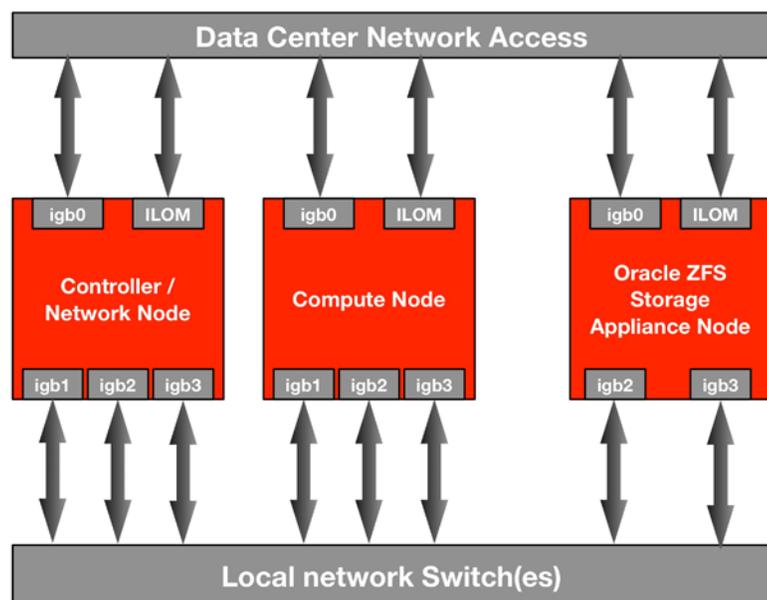


Figure 6. Physical network structure

Consider the following:

- When using a single physical network and a single switch for the local network, VLAN tagging can be used to keep the two storage subnets separated. In this case, the switch used in the local network must be able to support the VLAN tagging as used by the network set-up for the Oracle ZFS Storage Appliance. The architecture example in this paper uses this type of configuration.

- All network components in each OpenStack node are of the same type and are on the same hardware I/O expansion slots.
- All network connections on each node are wired in the same way and the same network port on each OpenStack node is configured to use the same IP subnet.

The following diagram shows where all the OpenStack software modules are installed and how the logical IP subnets are mapped to the hardware network ports.

The OpenStack subnets used by the guest instances are mapped to the network device net1 on each OpenStack node and the OpenStack admin subnet is mapped to the network device net0. These subnets are configured on the OpenStack Neutron and Oracle Solaris EVS layer. The network connections from the OpenStack guest instance to these subnets are handled by the Oracle Solaris zoning layer.

The storage subnets are configured on each OpenStack node in the Oracle Solaris global zone.

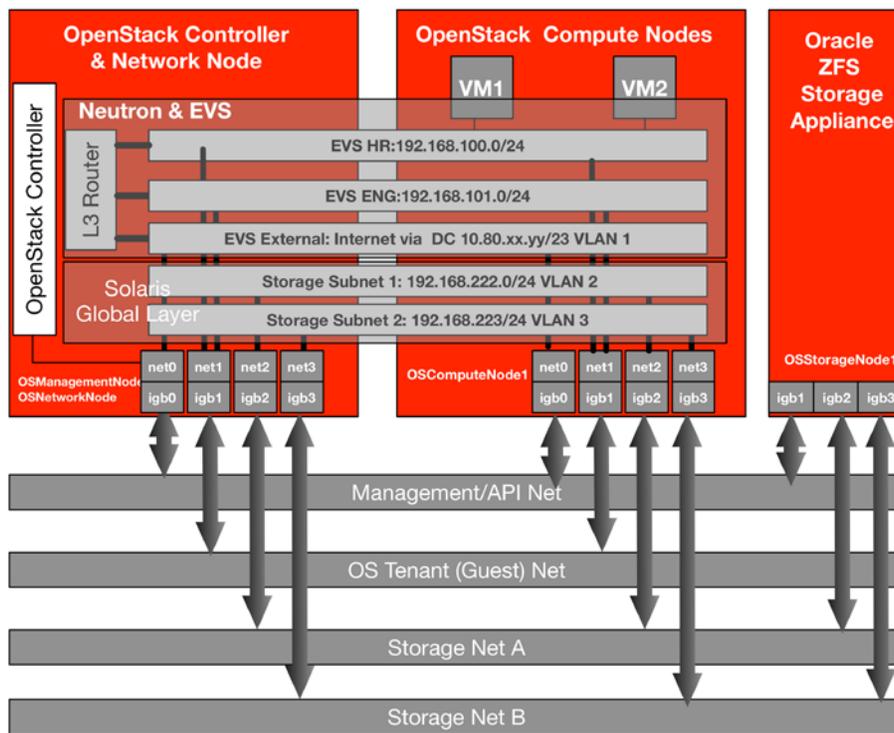


Figure 7. Network setup for example architecture

Oracle Solaris uses the EVS to implement an OpenStack Neutron L3 router, and follows the deployment model of a single virtual provider router with private networks. In this model, each tenant can use one or more private networks and all the tenant private networks share the same router. The EVS router is created, owned, and managed by the data center administrator and owned by the OpenStack service tenant. Thus, the router should not be visible in the tenant's network topology view. Furthermore, tenant networks cannot use overlapping IPs. Thus, administrators need to create the private networks on behalf of tenants.

By default, the EVS router prevents routing between private networks that are part of the same tenant. That is, VMs within one private network cannot communicate with the VMs in another private network,



even though they are all part of the same tenant. You can change this behavior by changing the setting `allow_forwarding_between_networks` to 'true' in the `/etc/neutron/l3_agent.ini` configuration file and restarting the `neutron-l3-agent` SMF service.

The EVS router provides the tenants' VMs the connectivity to the outside world. It does this by performing Bidirectional Network Address Translation (NAT) or Source NAT on the interface that connects the router to the external network. Tenants that need to be accessed from the outside world can create as many floating IPs (public IPs) as they need or as are allowed by the floating IP quota. Then associate these floating IPs with the VMs that need access from the outside world.

More info about the use of an EVS router in an OpenStack Neutron environment can be found in the *Installing and Configuring OpenStack in Oracle Solaris* document listed in the References section.

## Installing and Configuring Oracle OpenStack in Oracle Solaris

Before installing and configuring Oracle OpenStack on Oracle Solaris, be familiar with the specific requirements and installation steps as documented in *Installing and Configuring OpenStack in Oracle Solaris*.

The installation process can be broken down into the following steps:

1. Set up physical machines.

Verify that all servers meet the minimum hardware requirements to support the required Oracle Solaris version, as well as Oracle Solaris kernel Zone functionality. The servers must also meet the minimum disk, RAM memory, and CPU resources requirements to host the OpenStack services.

Install planned network Host Bus Adapters (HBAs), then connect network HBA and ILOM ports to network switches.

2. Installing the correct Oracle Solaris version on each physical machine, set up network for Oracle Solaris admin and ILOM access, and set up access to the Oracle Solaris pkg repository.
3. Configure and test network DNS functionality on each machine.
4. Set up the storage subsystem subnets and enable Oracle Solaris multipathing for iSCSI.
5. Install Oracle OpenStack modules as required for the OpenStack Controller/Network node and Compute nodes. See the document *Installing and Configuring OpenStack in Oracle Solaris* for further details.
6. Configure each OpenStack service by modifying the required parameters in the various related configuration files for each OpenStack service. Wait with the configuration and startup of the OpenStack Cinder service until the Oracle ZFS Storage Appliance has been set up as mentioned below.
7. Set up the required OpenStack virtual Tenant (Guest) network(s) and network ports to be used by the Oracle Solaris EVS (Elastic Virtual Switch ) for subnet routing. Pay special attention to the section 'Configuring the Neutron L3 Agent' in the *Installing and Configuring OpenStack in Oracle Solaris* document
8. Configure the Oracle ZFS Storage Appliance with the Appliance properties and user credentials as described in the section of this paper titled "Setting Up OpenStack Cinder and the Oracle ZFS Storage Appliance." Configure Cinder for use of the Oracle ZFS Storage Appliance.
9. Create virtual guest instances boot images.
10. Create virtual guest instances.

Note that this paper only details configuration steps for setting up the communication between the OpenStack nodes and the Oracle ZFS Storage Appliance and how to define back-end definitions in the OpenStack Cinder service for the Oracle ZFS Storage Appliance. All steps required to install and configure the various OpenStack services are described in the previously mentioned document *Installing and Configuring OpenStack in Oracle Solaris*. When required, some extra information is given in this paper



specific to the example architecture in this paper. This paper focuses on the process to configure the Cinder service for use with the Oracle ZFS Storage Appliance.

## Implementing the Network Configuration

The OpenStack architecture presented in this paper uses multiple network interfaces on each OpenStack node in order to separate network I/O for performance and security reasons. All nodes used in the architecture described in this paper use the same number of networks ports and all are configured in the same way.

The net0 interface is used to make the connection to the external world, so Oracle Solaris patches and updates can easily be retrieved through the specified patch repository in the Oracle Solaris pkg utility. This network interface must be set up during the installation and configuration of Oracle Solaris on the node.

The net1 interface is used for the OpenStack Guest virtual LAN(s). This network interface does not need to be given an IP address.

The net2 and net3 interfaces are used to set up dual redundant network connections to the Oracle ZFS Storage Appliance. Two independent subnets are used, each with its own respective network. The Oracle Solaris stms multipathing feature is used to handle the dual paths connections to the iSCSI LUNs on the Appliance.

## Setting Up Storage Network Interfaces

To match the VLAN tagging that is used in the configuration of the Oracle ZFS Storage Appliance, the following script is used to configure the net2 and net3 interfaces on each OpenStack node.

```
NODE_IP=$1
dladm set-linkprop -t -p mtu=9000 net2
dladm set-linkprop -t -p mtu=9000 net3
dladm create-vnic -l net2 -v 222 net222002
dladm create-vnic -l net3 -v 223 net223003
ipadm create-ip net222002
ipadm create-ip net223003
ipadm create-addr -T static -a 192.168.222.${NODE_IP} net222002/v4static
ipadm create-addr -T static -a 192.168.223.${NODE_IP} net223003/v4static
```

Next step is to enable Oracle Solaris multipath support for iSCSI on all OpenStack Compute nodes using the following command:

```
stmsboot -e -D iscsi
```

Note that a change in the multipath setup requires a reboot of Oracle Solaris for it to rebuild the devices in the /dev directory.

## Setting Up the OpenStack Network Connections

The OpenStack guest network access is set up on the OpenStack controller-node as part of the Neutron installation process. Follow the process steps in the section "Installing and Configuring Neutron" from the document *Installing and Configuring OpenStack in Oracle Solaris*.

Special consideration is required to assign the physical network ports to the OpenStack Guest network(s) and the interface used for guests instances to reach the Internet. This is done by using `evsadm` to map a vlan or vlan range to a network interface.

Use the following steps to set up the network connections:

1. Notify EVS which VLANs to use:

```
evsadm set-controlprop -p vlan-range=1,200-300
```

2. Specify which VLANs are mapped on which network interface by using the `evsadm up-linkport` option.

```
root@controller-node:~#evsadm set-controlprop -p uplink-port=net1,vlan-range=200-300  
root@controller-node:~#evsadm set-controlprop -p uplink-port=net0,vlan-  
range='',flat=yes
```

To view the settings, use the `evsadm -o all` option.

```
root@controller-node:~#evsadm show-controlprop -o all
```

PROPERTY	PERM	VALUE	DEFAULT	FLAT	VLAN_RANGE	VXLAN_RANGE	HOST
l2-type	rw	vlan	vlan	--	--	--	--
uplink-port	rw	net1	--	no	200-300	--	--
uplink-port	rw	net0	--	yes		--	--
uri-template	rw	ssh://	ssh://	--	--	--	--
uuid	r-	eb0c27be-edc1-11e4-b338-7d5f11dc4417	--	--	--	--	--
vlan-range	rw	200-300	--	--	--	--	--
vlan-range-avail	r-	201-300	--	--	--	--	--
vxlan-addr	rw	0.0.0.0	0.0.0.0	--	--	--	--
vxlan-ipvers	rw	v4	v4	--	--	--	--
vxlan-mgroup	rw	0.0.0.0	0.0.0.0	--	--	--	--
vxlan-range	rw	--	--	--	--	--	--
vxlan-range-avail	r-	--	--	--	--	--	--

3. Set up one or more OpenStack internal Tenant subnets as described in the section "How to Create an Internal Network" in *Installing and Configuring OpenStack in Oracle Solaris*.
4. Configure the Neutron L3 agent.

In this step, a virtual provider router is created and a route to the external network is established. The Tenant subnets created in the previous step and the subnet related to the external network as created

in this step will be added to the router. The process of setting up the Neutron L3 agent is documented in the section "How to Configure the External Network in OpenStack" in the *Installing and Configuring OpenStack in Oracle Solaris* document.

Note that when there is no need to reach a guest instance from the external internet, no floating IPs need to be set up. Setting the `enable_snat` option in the EVS router that was created in the previous step is enough to give guest instances access to the external network.

The `evsadm`, `d1adm` and `ipadm` utilities can be used to verify the network configuration on the OpenStack controller node.

```
root@controller-node:~# d1adm
```

LINK	CLASS	MTU	STATE	OVER
net1	phys	1500	up	--
net3	phys	1500	up	--
net0	phys	1500	up	--
net2	phys	1500	up	--
net222002	vlan	1500	up	net2
net223003	vlan	1500	up	net3
13e4d3a277e_1_0	vnic	1500	up	net0
131b93b59e2_d_0	vnic	1500	up	net1
dh64945c23_9d_0	vnic	1500	up	net1

```
root@controller-node:~# ipadm
```

NAME	CLASS/TYPE	STATE	UNDER	ADDR
dh64945c23_9d_0	ip	ok	--	--
dh64945c23_9d_0/v4	static	ok	--	192.168.99.2/24
13e4d3a277e_1_0	ip	ok	--	--
13e4d3a277e_1_0/v4	static	ok	--	10.xx.yy.168/23
lo0	loopback	ok	--	--
lo0/v4	static	ok	--	127.0.0.1/8
lo0/v6	static	ok	--	::1/128
net0	ip	ok	--	--
net0/v4	static	ok	--	10.xx.yy.26/23
net222002	ip	ok	--	--
net222002/v4static	static	ok	--	192.168.222.26/24
net223003	ip	ok	--	--
net223003/v4static	static	ok	--	192.168.223.26/24

## Setting Up OpenStack Cinder and the Oracle ZFS Storage Appliance

The Oracle OpenStack ZFSSA Cinder driver plug-in provides storage provisioning for the OpenStack block volume create and manage functions by creating and managing LUNs on an Oracle ZFS Storage Appliance.

The Oracle ZFS Storage Appliance is used to provide a central managed volume repository for the OpenStack guest instances. Volumes are created as LUNs on the Oracle ZFS Storage Appliance and made available to the OpenStack nodes using the iSCSI interface over the network.

The Oracle OpenStack ZFSSA Cinder driver exposes a number of Oracle ZFS Storage Appliance properties. These properties allow for tuning LUNs specific to certain I/O workloads and security requirements. By using the Cinder function to specify multiple back ends, each containing specific information for volumes created with their associated back-end definition, a customizable storage provisioning capability can be achieved. Volume are created according to specific application or workgroup storage requirements, like transactional type workloads requiring low latency response times, or high security storage requiring encryption and physically separate storage locations for different workgroups. More detail on the use of multiple Cinder back-end definitions are provided later in this paper.

It is good practice to separate the storage data traffic paths between the OpenStack nodes and the storage subsystem from the other OpenStack network traffic. As previously described, two independent separate IP subnets are provisioned over two physical separate networks to fulfill the data separation, security and reliability requirements.

The creation of LUNs on the Oracle ZFS Storage Appliance and the setup of the iSCSI communication are handled by the Oracle OpenStack ZFSSA Cinder driver that is active on the OpenStack controller node. Specific Appliance configuration information and volume characteristics have to be specified in the Cinder configuration `cinder.conf` file.

### Using iSCSI as the Communication Link

iSCSI enables two nodes to exchange SCSI commands over an IP network. The iSCSI targets and initiators are identified using unique IQNs (iSCSI Qualified Names). Block volumes on the target side are made available as SCSI LUNs, each with a unique LUN number.

iSCSI uses TCP as a transport layer. TCP port number 3260 is reserved as a TCP listener port for the iSCSI service at the target side.

Note that iSCSI nodes can be configured using multiple networks, targets can be made visible to those networks by the storage subsystem, and the iSCSI initiator node can detect iSCSI targets from multiple network interfaces. When a single LUN is made visible over multiple networks, iSCSI multipathing software on the initiator must be used to utilize multiple paths between an initiator and target LUN.

You can add extra security by specifying CHAPS authentication, to be used when establishing an initiator-target communication session.

So to set up communication between two nodes, the following information is needed to identify the SCSI object:

- The SCSI node (initiator and target) DNS name or IP address
- The TCP iSCSI listener port (default 3260) at the target side
- The iSCSI IQN of the initiator and target node
- Optional authentication information using CHAPS

You can use the `iscsiadm` command to retrieve the relevant iSCSI information as shown in the next diagram.

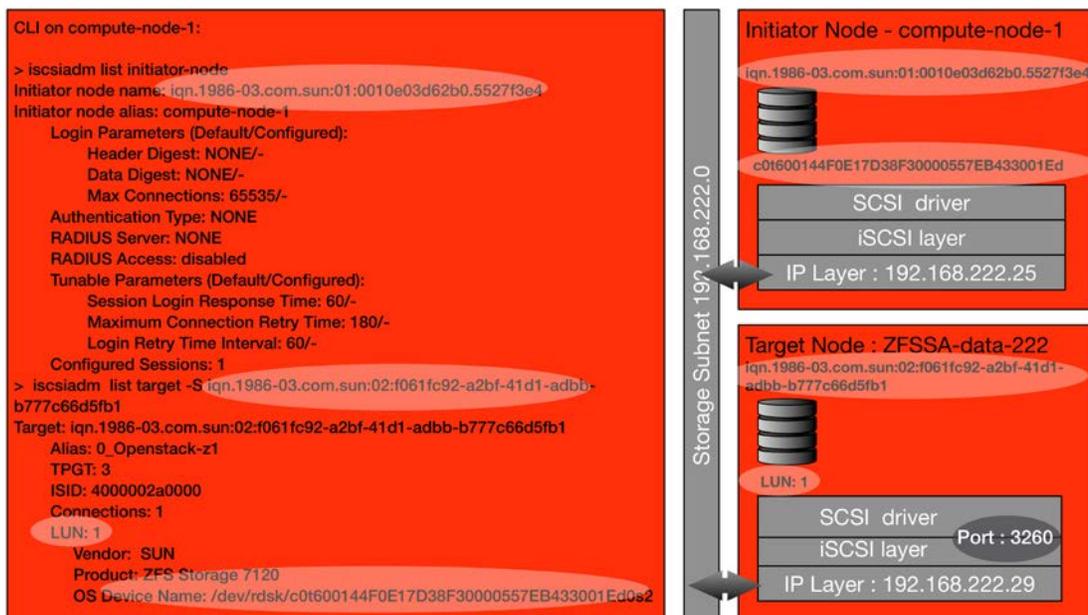


Figure 8. Relevant iSCSI info

The example shows the retrieval of the node iSCSI initiator IQN and LUNs visible from ZFSSA-data-222 to node compute-node-1 using the target IQN value used by the zfssa-data-222 storage subsystem to export the LUNs.

## Understanding the Role of the Oracle OpenStack ZFSSA Cinder Driver

The Oracle OpenStack ZFSSA Cinder driver creates and manages volumes by creating the related LUNs on the Oracle ZFS Storage Appliance, setting the appropriate properties on the Appliance, and managing the visibility of those LUNs to the appropriate OpenStack Compute nodes and OpenStack guest instances on which the volumes are going to be used.

To understand the various configuration parameters and their relationships, consider the lifecycle of an OpenStack Cinder volume. An OpenStack Cinder volume has two states: attached to an instance, or not attached (free).

When an OpenStack Cinder volume is created, the Oracle OpenStack ZFSSA Cinder driver creates a related LUN on the Oracle ZFS Storage Appliance. During this process, the Oracle OpenStack ZFSSA Cinder driver sets the following iSCSI properties of the LUN on the Appliance:

- Alias for a target IQN – The Appliance automatically creates a value for the IQN. For each target, Cinder sets the following attributes:
  - IP interfaces to be used for data access to the LUN as specified in the Cinder configuration file
  - Optional CHAP authentication for the target side as specified in the Cinder configuration file
- Target group as specified in the Cinder configuration file containing the target as created in the previous step
- Aliases for the initiator IQNs as specified in the Cinder configuration file
- Initiator group to contain initiator IQNs as processed in the previous step

The Oracle ZFS Storage Appliance can limit an iSCSI LUN's visibility by granting access on an initiator IQN basis. The Oracle OpenStack ZFSSA Cinder driver uses this capability to manipulate the volume's visibility depending on its state. In the free state, a LUN is assigned to a hidden initiator group in the Appliance that prevents the LUN from being accessible by any initiator; that is, any OpenStack node in the network.

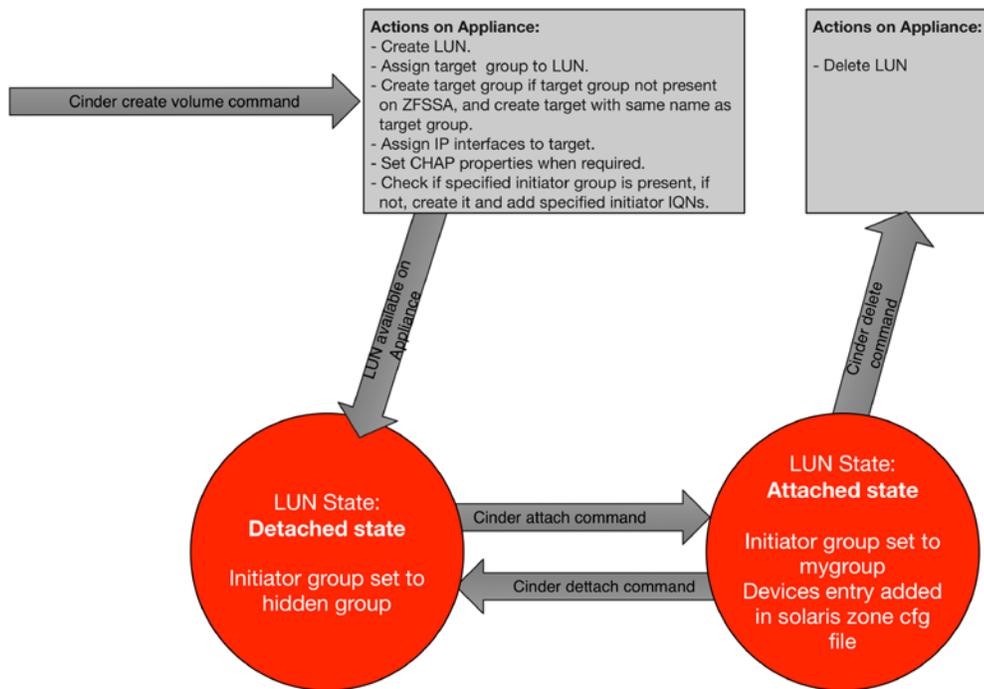


Figure 9. Cinder volume and initiator group relationship

When attaching a volume to an OpenStack guest instance, the related LUN on the Oracle ZFS Storage Appliance is made 'visible' to the related OpenStack guest instance. The Oracle OpenStack ZFSSA Cinder driver does this by changing the LUN initiator group property of the LUN on the Oracle ZFS Storage Appliance to the initiator group as specified in the Cinder configuration file. The initiator group contains the initiator IQN(s) of the OpenStack Compute node(s) as specified in the Cinder configuration file.

The Oracle OpenStack ZFSSA Cinder driver also adds a device definition entry to the guest (Solaris zone) configuration file on the Compute node on which the guest instance is running using the URL of the LUN to be attached.

```
device:
  match not specified
  storage: iscsi://zfssa-1-data222:3260/target.iqn.1986-
03.com.sun:02:f061fc92-a2bf-41d1-adbb-b777c66d5fb1,lun.1
  id: 1
  bootpri not specified
```

The guest instance picks the LUN up as a local LUN in the form of `/dev/(r) dsk/cxy`. On the guest instance, when a LUN is not used as a raw device, the LUN must be partitioned and a file system created on it according to the guest OS and/or application requirements before the LUN can be used.

The Oracle OpenStack ZFSSA Cinder driver uses the Oracle ZFS Storage Appliance REST service to monitor and administer the Oracle ZFS Storage Appliance. It is recommended to reserve a dedicated admin IP interface on the Oracle ZFS Storage Appliance for the Oracle OpenStack ZFSSA Cinder driver administrative functions, to avoid deteriorating response times caused by high volumes of data traffic.

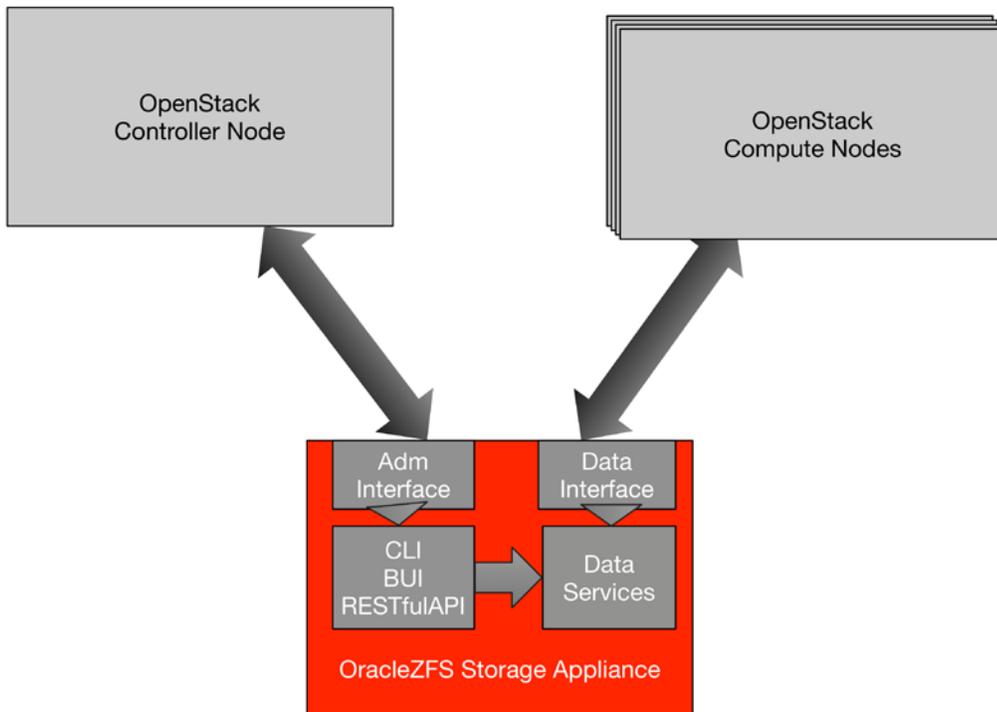


Figure 10. Communication channels between OpenStack nodes and Oracle ZFS Storage Appliance



When using the Oracle ZFS Storage Appliance in a cluster configuration, setup of the communication IP interface requires special attention. See the section "Setting Up Cinder Service with Oracle ZFS Storage Appliance Cluster Configurations" for further details.

## Installing and Configuring the Oracle OpenStack ZFSSA Cinder Driver

The Oracle OpenStack ZFSSA Cinder driver is installed as part of the Oracle OpenStack on Oracle Solaris software bundle. Edit the Cinder configuration file `/etc/cinder/cinder.conf` to specify the details of the Cinder storage back end that will be used to communicate with its related Oracle ZFS Storage Appliance and to prepare the Appliance itself to accept communication requests from the Cinder driver.

Details of the installation process and a description of all parameter options used by the Oracle OpenStack ZFSSA Cinder driver are listed in its accompanying `ZFSSACinderDriver.README` document. See the References section for a link to this document.

**IMPORTANT:** Before the installation process can be started, the Oracle ZFS Storage Appliance must contain at least one storage pool, and the iSCSI initiator IQN of each OpenStack Compute node must be identified.

The following tasks form the installation and configuration process for the OpenStack ZFSSA Cinder driver:

- Install and execute the Cinder workflow (`cinder.akwf`) on the Oracle ZFS Storage Appliance. For details, see the `ZFSSACinderDriver.README` document listed in the References section. The workflow creates an admin user to be used by the Oracle OpenStack ZFSSA Cinder driver and enables the Oracle ZFS Storage Appliance REST service.
- Verify and set up the admin and data network interfaces on the Oracle ZFS Storage Appliance.
- Verify that all storage network connections on all OpenStack nodes are configured and connections to the Oracle ZFS Storage Appliance can be established for both the admin and data networks.
- Set up the Cinder configuration file and restart the Cinder services on the OpenStack controller node.
- (Re)start the OpenStack Cinder volume service using `svcadm`:  

```
root@controller-node:~# svcadm enable cinder-volume:default cinder-volume:setup
```

## Setting Up Network Interfaces on the Oracle ZFS Storage Appliance

In the architecture used in this paper, two storage network interfaces are used, each connected to a different subnet. To provide complete isolation between the subnets, VLAN tagging is used. This creates two independent data paths between the OpenStack nodes and the Oracle ZFS Storage Appliance. Next to these, an administrative interface is created on the Oracle ZFS Storage Appliance for the Cinder driver.

The following datalinks and interfaces have been created as illustrated in the following Oracle ZFS Storage Appliance CLI session output:

```
zfssa-cinderadm:configuration net datalinks> show
Datalinks:

DATALINK      CLASS      LINKS      STATE  ID      LABEL
igb0          device    igb0       up     -       dl-igb0
igb222002     vlan      igb2       up     222     vlan222
igb223003     vlan      igb3       up     223     vlan223

zfssa-cinderadm:configuration net interfaces> show
Interfaces:

INTERFACE     STATE     CLASS  LINKS      ADDRS      LABEL
igb0          up       ip     igb0       192.168.0.29/24  zfssa-cinderadm
igb222002     up       ip     igb222002  192.168.222.29/24  storage222
igb223003     up       ip     igb223003  192.168.223.29/24  storage223
```

## Setting Up Storage Connections on Oracle Solaris OpenStack Nodes

When setting up storage connections on Oracle Solaris OpenStack nodes, it is highly recommended to use the same network device on each node for the same configured subnet. Also use the same ‘number’ in the IP address (xx.yy.zz.number) for each subnet.

## Setting Up the Cinder Configuration File

The Cinder service is located on the OpenStack Controller node. As part of the Cinder service initialization, it reads its configuration information from the file `/etc/cinder/cinder.conf`. This very large configuration file contains template sections for drivers from various storage vendors as well as multiple back-end definitions for different (or the same) storage subsystems. The Oracle OpenStack ZFSSA Cinder driver retrieves information from this file on how to reach and configure the Oracle ZFS Storage Appliance.

Each back-end definition defines various characteristics of the volume and storage back end so that different volume types with different I/O characteristics can be configured. See the Best Practices section of this paper for further details.

In the global section, named [DEFAULT], active back ends and the default back end must be specified. Each back end starts with a [`<NAME>`] line.

The following table shows the mandatory properties required in each back-end definition that must be specified for the Oracle OpenStack ZFSSA Cinder driver.

**TABLE 1. REQUIRED PROPERTIES FOR EACH BACK-END DEFINITION**

Properties	Required setup on Oracle ZFS Storage Appliance	Description
<code>volume_driver</code>	no	Specify the ZFSSA Cinder driver; <code>cinder.volume.drivers.zfssa.zfssaiscsi.ZFSSAISCSIDriver</code>
<code>san_ip</code>	yes	The Appliance admin IP interface to be used by Cinder.
<code>san_login</code>	yes	A user account setup on the Appliance with privileges to perform all Appliance admin functions by the Cinder driver.
<code>san_password</code>	yes	The password of the user account on the Appliance.
<code>zfssa_pool</code>	yes	The storage pool on the Appliance to be used to create the OpenStack Cinder volumes.
<code>zfssa_project</code>	no	Project to be used to store the volume properties as defined in the back-end definition section and hold all the volumes created with this back-end definition.
<code>zfssa_target_portal</code>	yes	The IP address and port number used by the OpenStack nodes to connect to the volumes using the iSCSI protocol.
<code>zfssa_target_interfaces</code>	yes	The name of one or more IP interfaces on the Appliance to expose the volume LUNs to.
<code>zfssa_initiator_group</code>	no	Name of the initiator group to group the initiator IQNs as specified in the next property.
<code>zfssa_initiator(s)</code>	no	The IQNs of all OpenStack node initiators to be allowed to attach the volumes to their VMs created with this back end.

When one of the properties in the table is listed as ‘does not need to be set up,’ it means the Cinder driver will create the property on the Oracle ZFS Storage Appliance if it is not already present. Properties that need to be present on the Appliance for the back-end properties must be configured on the Appliance before the Cinder volume services are (re)started on the OpenStack Controller Node.

Note: The `san_ip` and `zfssa_target_portal` can use the same IP interface of the Oracle ZFS Storage Appliance. However, using two different IP interfaces is strongly recommended to avoid long response times and risk of timeout errors on commands to the Appliance REST service due to high data traffic on the same interface.

The next diagram shows the relationships between the various Oracle OpenStack Cinder ZFSSA driver properties and the iSCSI and network properties as set up on the OpenStack nodes and the Oracle ZFS Storage Appliance.

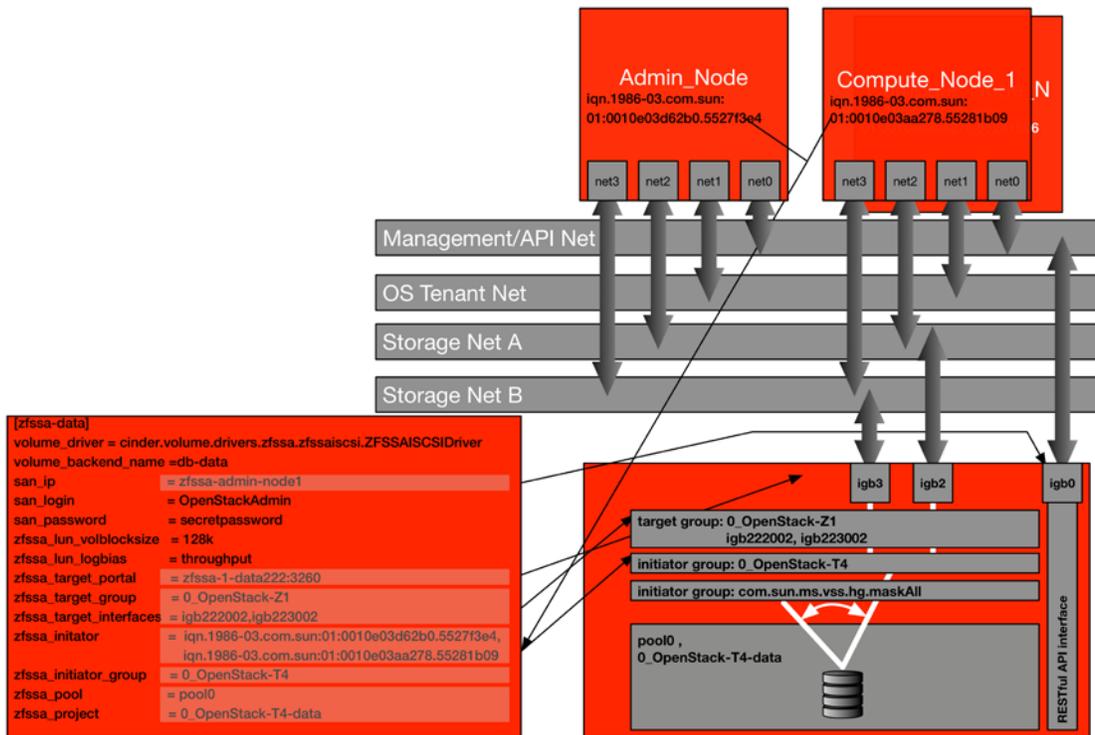


Figure 11. Relevant iSCSI info for `cinder.conf` file

Note: As mentioned earlier, when a volume is created on the Oracle ZFS Storage Appliance by the Oracle OpenStack Cinder ZFSSA driver, the volume is assigned to the already present, hidden initiator group, `com.sun.ms.vss.hg.maskAll`, on the Appliance. As soon as a volume is attached to a guest instance, the volume is moved to the initiator group as specified in the back-end definition in the `cinder.conf` file.

After the required changes to the `cinder.conf` file on the OpenStack Controller node have been completed, restart the Cinder volume services using the `svcadm` command.

## Setting Up Cinder Service with Oracle ZFS Storage Appliance Cluster Configurations

Using an Oracle ZFS Storage Appliance in a clustered configuration is highly recommended when high availability and performance scaling features are important. A clustered configuration consists of two Oracle ZFS Storage Appliance nodes. When specifying storage back-end definitions in the Cinder service involving a cluster configuration, specific operating characteristics around the Oracle ZFS Storage Appliance storage pools and network resources require special attention.

Network interfaces and storage pools are part of cluster resources. These resources contain objects that are in most cases active on one cluster node at a time. In an active-active cluster configuration, a cluster node owns the resources that it operates and manages. When a cluster node fails, the other node takes over the resources of the failed node.

In the case of storage pools, all LUN definition and property management operations need to be executed on the node that owns the storage pool that contain the LUNs. The Appliance IP interface objects used

to serve out the LUNs are also cluster resource objects and need to be owned by the same node that owns the storage pool.

Therefore, both Oracle ZFS Storage Appliance IP interfaces as specified in the `san_ip` and `zfssa_target_portal` need to be owned as resources by the Appliance node that owns the related pool in the Cinder back-end definition. They need to be of the type 'SINGLETON'. Refer to either the Oracle ZFS Storage Appliance online help or the *Oracle ZFS Storage Appliance Administration Guide* for a detailed description of types of cluster resources and how to configure them.

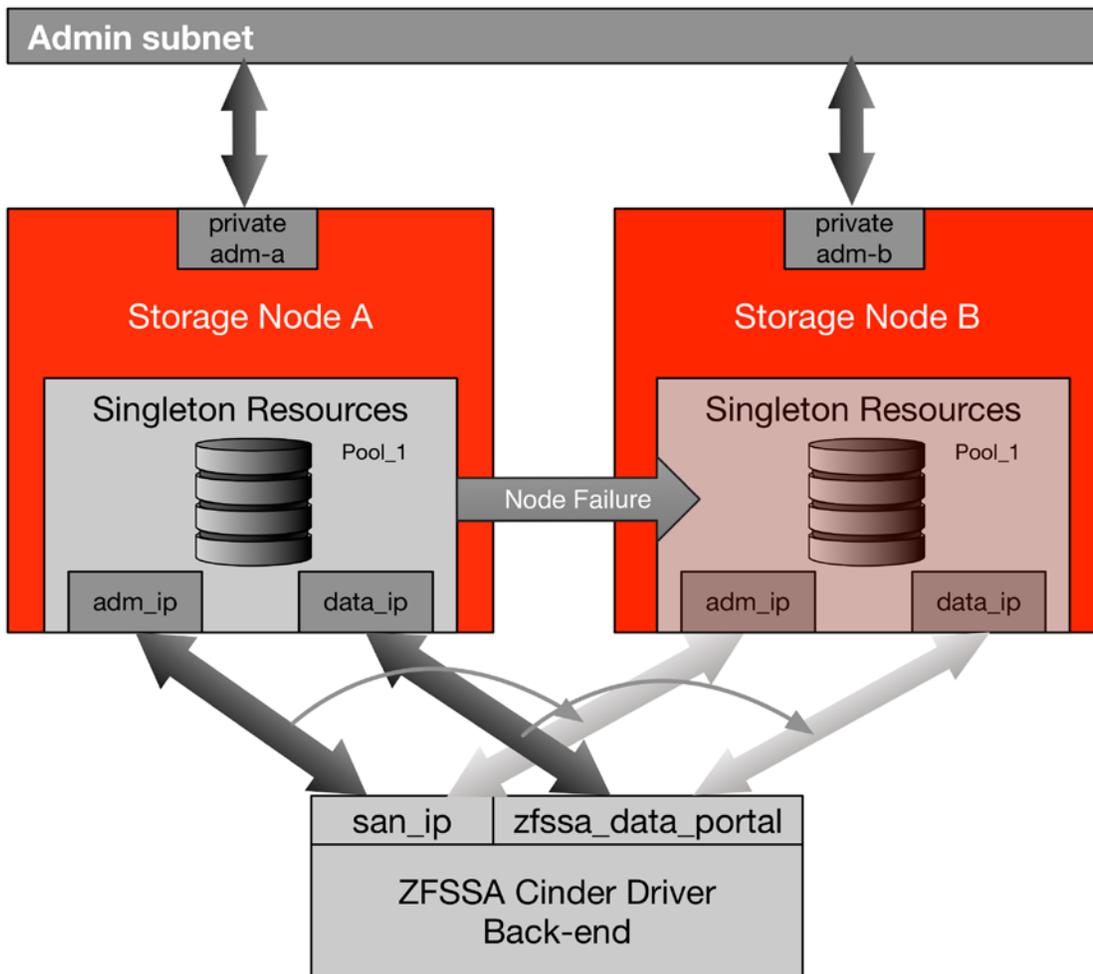


Figure 12. Configuring an Oracle ZFS Storage Appliance cluster for use with Cinder

Often a separate IP interface is reserved on each Appliance cluster node to maintain access to the node for administrative actions, even if the other node has taken over all resources. To ensure access, the related IP interface object is locked to a node in a 'PRIVATE' state.

Those interfaces are not suitable to be used as OpenStack management interfaces as specified in the `san_ip` property. More information on the Oracle ZFS Storage Appliance networking functions and features can be found in the white paper "Networking Best Practices with the Oracle ZFS Storage

Appliance" on Oracle's NAS Storage Documentation, White Papers and Solution Briefs web page noted in the Reference section.

## Using Multiple Cinder Storage Back-end Definitions

The `cinder.conf` file can be used to set up multiple storage back ends, each with its own property definitions and a related Cinder driver. Multiple back-end definitions are useful for creating volume definitions with different I/O characteristics, like volume blocksize and cache behavior settings. These customized back-end definitions enable OpenStack administrators to provision volumes based upon storage provisioning type requirements, such as highly available, low latency, and high security storage, or generic workloads requiring high capacity density, or volumes for database redo log files, and so forth. Additionally, OpenStack administrators do not need to know the specifics of the storage subsystem as provided by the Oracle ZFS Storage Appliance in order to create volumes that behave optimally for their particular environments.

Besides using multiple back ends to define volumes with specific I/O characteristics, the definitions can also be used to control the placement of volumes on different physical disks by using different pools or place volumes on different Oracle ZFS Storage Appliances.

The following table gives an overview of Oracle ZFS Storage Appliance properties that can be used in Oracle OpenStack Cinder ZFSSA back-end definitions to specify characteristics for creating the customized volumes.

**TABLE 2. REQUIRED CINDER PROPERTIES FOR VOLUME CREATION**

Cinder Property	Description	I/O Characteristics
<code>zfssa_pool</code>	Name of pool in the Appliance to be used to create the volume.	Each pool is created according to the specified RAID layout; select the RAID layout optimal to the I/O profile of applications using this pool.
<code>zfssa_project</code>	Name of the project to be used in the specified pool of the Appliance. If the project does not exist, it will be created.	Use a project to logically group volumes together and to use volume data encryption when required.
<code>zfssa_lun_volblocksize</code>	Volume block size. Default is 8k. Size can be 512, 1k, 2k, 4k, 8k, 16k, 32k, 64k, 128k.	Select the volblock size that matches the I/O profile requirements of the volumes.
<code>zfssa_lun_sparse</code>	Flag to enable sparse (thin-provisioned). Default=False.	Enable sparse option to over provision capacity and the average capacity of a number of LUNs will not exceed the total available capacity.
<code>zfssa_lun_compression</code>	Data compression. Default is off. Value can be 'off, lzjb, gzip-2, gzip, gzip-9.'	Use data compression when disk space utilization is important and the data in the volumes are suitable for compression.
<code>zfssa_lun_logbias</code>	Synchronous write bias. Value can be 'latency, throughput'.	Select logbias value matching write I/O characteristics of a volume, latency for transactional type I/O, throughput for large sequential type I/O.

Note that when directly creating a rproject on the Oracle ZFS Storage Appliance, you have the option to use data encryption. The encryption property will be applied to all volumes created within that project. In order to use encryption, you must set up the Oracle ZFS Storage Appliance to use encryption keys, either local on the Appliance or under the control of an Oracle Key Management (OKM) server.

Each back-end definition starts with a header containing the configuration group name of the back end and ends as soon as a non-back-end property definition statement is encountered. The back-end group name is not related to the property `volume_backend_name`. The `volume_backend_name` can be used to point the Cinder volume type to the back-end definition to use when creating a volume.

```
[DEFAULT]
..
enabled_backends=zfssa-data,zfssa-log
default_volume_type=zfssa-data
..
[zfssa-data]
volume_backend_name=db-data
..
[zfssa-log]
volume_backend_name=db-log
..
```

When using multiple back-end definitions, the property `enabled_backends` must be set in the global section of the `cinder.conf` file:

```
enabled_backends=zfssa-data,zfssa-log
```

The default volume type used is specified in the global section by:

```
default_volume_type=zfssa-data
```

The available back-end services and their status are listed as follows:

```
Controller-adm-node:# cinder service-list
+-----+-----+-----+-----+-----+-----~/
| Binary | Host | Zone | Status | State | |
+-----+-----+-----+-----+-----+-----~/
| cinder-scheduler | controller-adm-node | nova | enabled | up | ~/
| cinder-volume | controller-adm-node@zfssa-data | nova | enabled | up | ~/
| cinder-volume | controller-adm-node@zfssa-log | nova | enabled | up | ~/
+-----+-----+-----+-----+-----+-----~/
```

Different back-end group definitions can have the same name in the property `volume_backend_name`. The Cinder scheduler determines which back end is used based on scheduling properties set in the `cinder.conf` file. Details can be found in the *OpenStack Cloud Administrator Guide* in the section "Configure multiple-storage back ends" from docs.openstack.org.

When creating a volume, you can control which back end is used by specifying the volume type. For each back-end definition, a relation to the volume type must be set up by creating a new volume type and a

type-key within the new type pointing to the back end. The property `volume_back_end_name` of the back-end section is used for the type key.

```

controller-node:~# cinder type-create zfssa-data
+-----+-----+
|          ID          |   Name   |
+-----+-----+
| 19bde848-f7d8-4ae8-9679-ac2e84f09839 | zfssa-data |
+-----+-----+
controller-node:~# cinder type-key zfssa-data set volume_backend_name=db-data
controller-node:~# cinder extra-specs-list
+-----+-----+-----+
|          ID          |   Name   |   extra_specs   |
+-----+-----+-----+
| 19bde848-f7d8-4ae8-9679-ac2e84f09839 | zfssa-data | {u'volume_backend_name': u'db-data'} |
| 2ea3183e-be1b-407d-9318-c2e2fe0720fe | zfssa-log  | {u'volume_backend_name': u'db-log'}  |
+-----+-----+-----+
controller-node:~# cinder create --volume-type zfssa-data --display-name my_volume1 2
+-----+-----+-----+
|   Property   |   Value   |
+-----+-----+-----+
| attachments  |          []          |
| availability_zone | nova      |
| bootable     | false     |
| created_at   | 2015-07-22T10:36:18.225519 |
| display_description | None     |
| display_name | my_volume1 |
| encrypted    | False     |
| id           | d0627a16-72dd-4069-aad7-c23892919379 |
| metadata     |          {}          |
| size         | 2         |
| snapshot_id  | None      |
| source_valid  | None      |
| status       | creating  |
| volume_type  | zfssa-data |
+-----+-----+-----+
controller-node:~#

```

## Administering Oracle ZFS Storage Appliance Volumes in the OpenStack Environment

To create and manage volume types in the OpenStack environment, use either the Cinder and Nova command line utilities or the OpenStack Horizon network BUI interface.

Existing and available volumes and their status can be viewed from the Horizon volumes pane seen in the following figure.

## Volumes

<input type="checkbox"/>	Project	Host	Name	Size	Status	Type	Attached To	Bootable	Encrypted	Actions
<input type="checkbox"/>	demo	controller-adm-node@zfssa-mpath#mpath-test	pb-sales-data-test-no-mpath	30GB	In-Use	zfssa-mpath	Attached to pbsales0406-2 on c1d1	No	No	Update Volume Status
<input type="checkbox"/>	demo	controller-adm-node@zfssa-mpath#mpath-test	pb-sales-data-test-mpath	30GB	Available	zfssa-mpath	Attached to pbsales0406-2 on c1d2	No	No	Delete Volume ▾
<input type="checkbox"/>	demo	controller-adm-node@zfssa-data#db-data	qqq-rootzpool	10GB	In-Use	zfssa-data	Attached to qqq on c1d0	No	No	Update Volume Status
<input type="checkbox"/>	service	controller-adm-node@zfssa-data#db-data	pbEng4-rootzpool	20GB	In-Use	zfssa-data	Attached to pbEng4 on c1d0	No	No	Update Volume Status
<input type="checkbox"/>	demo	controller-adm-node@zfssa-data#db-data	pbsales0406-2-rootzpool	20GB	Error	zfssa-data	Attached to pbsales0406-2 on c1d0	No	No	Delete Volume ▾
<input type="checkbox"/>	service	controller-adm-node@zfssa-images#os-images	pb-data3	3GB	Available	zfssa-images		No	No	Delete Volume ▾
<input type="checkbox"/>	service	controller-adm-node@zfssa-data#db-data	p3-rootzpool	20GB	Available	zfssa-data		No	No	Delete Volume ▾

Displaying 7 items

Figure 13. Horizon pane showing volume status information

You can also create and delete volumes in the Horizon interface. When creating a new volume, specify its type and size in the Horizon Create Volume dialog window, as seen in the following figure.

**Create Volume**

Volume Name \*  
MyDataVolume

Description  
Generic Data Volume

Volume Source  
No source, empty volume

Type  
zfs-data

Size (GB) \*  
10

Availability Zone  
Any Availability Zone

**Description:**  
Volumes are block devices that can be attached to instances.

**Volume Limits**

Total Gigabytes (137 GB) 1,000 GB Available

Number of Volumes (2) 10 Available

Cancel Create Volume

Figure 14. Horizon pane to create a volume

The ZFSSA Cinder iSCSI driver supports the following additional Cinder volume operations:

- Extend Volume
- Create Snapshot
- Delete Snapshot
- Create Volume from Snapshot
- Attach Volume to Guest Instance
- Detach Volume from Guest Instance
- Create Bootable Volume – Creating a bootable volume requires the existence of a Glance image volume. When creating a new OpenStack guest instance, the contents of this image volume is used to download the boot image from the Glance volume repository onto the newly created volume in the Cinder volume repository.
- A typical `cinder` CLI command for this task would take the following format:  

```
cinder create -image-id <glance-im-id> --display-name=mytest --volume-type <myvolumetype>
```



The Cinder snapshot-related commands fully utilize the Oracle ZFS Storage Appliance snapshot and clone functions, providing a virtual instantaneous access to volumes created using the Cinder snapshot and 'create volume from snapshot' commands.

Note: Cinder volume Attach and Detach operations always require a reboot of the related guest instance.

The Cinder CLI application provides extensive information on its available CLI options; invoke it by using help as an argument.

## Best Practices for Deploying Oracle OpenStack on Oracle Solaris with Oracle ZFS Storage Appliance

The following recommendations provide best practices and considerations for designing and configuring your OpenStack Cinder service architecture using Oracle Solaris and Oracle ZFS Storage Appliance.

### Planning for Growth with a Multiple Network Architecture

An OpenStack configuration can be started with just one hardware node or a single controller node and a couple of compute nodes using a single network. However it make sense to plan for a larger scale deployment and plan to use multiple networks as described in this document. OpenStack uses virtual network elements to build a logical network infrastructure. This abstracts the network architecture from the physical network transport elements, like host network connections and physical cabling. Care must be taken to not oversubscribe the physical network elements by allocation to many logical data streams on a single physical element. In deployments involving multiple physical OpenStack nodes, it is important to plan to use multiple physical network connections. Other reasons for using multiple network connections and/or subnets are data security, availability, and reliability.

Many factors need to be considered before starting a network design incorporating storage subsystem data traffic. Key elements to scope according to customer and application(s) requirement are:

- Reliability/Availability
- Scalability/Performance
- Security
- Manageability
- Available budget for both capital investment and operational costs

#### *Reliability/Availability*

The first step to take is to determine the business requirements and translate those into IT-specific requirements. Reliability and Availability information can be retrieved from a Business Continuity (BC) plan and derived from the Service Level Agreements (SLAs) between the IT organization and the various business departments in the company.



Use the information from scoping the listed requirements to find the right balance between solution costs and the cost impact of loss of services that is acceptable for the business. It is equally important to keep all involved groups in the loop so that agreed-upon expectations are maintained and verified.

Avoid making designs too complex, or meant to solve problems that do not exist. Put yourself in the position of support engineers who have been called out of bed at 4 a.m. on a weekend to deal with a major downtime escalation. They must be able to understand the design in order to properly troubleshoot it.

### ***Scalability/Performance***

As in all virtual network environments, the physical network underneath the virtual network must be able to cope with the required connections and data bandwidth requirements on the virtual network architecture. The physical network design must be planned for further expansion and care must be taken to avoid bottlenecks on any of the components in the physical network design. In this regard, the key components to be considered are the OpenStack node the network service is running on and its related network HBAs.

### ***Security***

Customer IT security requirements, legal and/or (Government) regulatory requirements may dictate the use of different (physical) networks for data and administrative purposes. Both the OpenStack software architecture and the Oracle ZFS Storage Appliance offer various options to separate various data streams. The ILOM (Integrated Lights Out Manager) access feature of the Solaris SPARC servers and the Oracle ZFS Storage Appliance node ILOM feature offers strict and secure separation between administrative access and normal data production access of the hardware. It requires a separate network connection into the network used for strict administrative access.

Access control and authentication to volumes located on the Oracle ZFS Storage Appliance can be realized using the iSCSI initiator and target group properties in the Cinder configuration file and using the iSCSI CHAP authentication function.

### ***Manageability***

Separating different types of network traffic makes it easier to manage, and monitor network utilisation and identify potential bottlenecks in the infrastructure.

## **Incorporating Uniformity into the Network Design**

The Oracle Solaris Operating system enumerates network adapters as netxx in order of detection during the kernel boot process. Link names used by the kernel are based on type of network HBAs used.

Use the same type of network adapters in each hardware node. Be sure to place them in the same option slots on the motherboard so that the interface port numeration in the kernel and device link-names are always the same on each OpenStack node.



Configure each network adapter port on each server to the same subnet, keeping a uniform network configuration over all OpenStack nodes. This way it is much easier to configure the Oracle Solaris EVS virtual switch setup.

## Using Logical Hostnames

During the OpenStack installation process, many of its components' configuration files have to be modified. Quite often the network access information needs to be specified. In many examples and demo scripts the node's IP address is used. It is highly recommended that you do not use IP addresses wherever possible. Use the node-name instead and use either network DNS or `/etc/hosts` for name resolution. This makes configuration files easier to read and avoids having to administrate IP addresses in many places in the infrastructure.

Before starting the OpenStack installation process, make sure IP name resolution is set up and properly functioning.

Also make sure to specify the node-name associated with the right subnet for the required configuration information. Most networking access information for OpenStack nodes needed in their configuration files are node-names associated with the administrative subnet.

Many OpenStack processes store critical resource information in a central database structure on the Controller node. Granted privileges of all OpenStack services is also stored in that database as part of the OpenStack installation process. Use the OpenStack controller admin node-name for this. Using an IP address will make it very hard to change the configuration information in that database without detailed knowledge of the database layout and related commands to change the info.

## Using NTP to Synchronize OpenStack Nodes Time

It is important to keep system time clocks on each OpenStack node in sync with each other. For instance, for monitoring and debugging purposes it is essential that the time stamps used for logging events on one node match similar event loggings on another host.

Use the Oracle Solaris NTP client functionality to ensure time synchronization among all OpenStack nodes. Setting up the NTP client is described in the *'Installing and Configuring OpenStack in Oracle Solaris'* manual.

## Employing a Uniform Symmetrical Hardware Design

Where possible, use the same type of server for all nodes, each with exactly the same hardware configuration. Using the same Host Bus Adapters (HBAs) in the same HBA slots offers multiple advantages:

- Simplifies swapping servers in case of a complete server failure.
- Requires only one type of server hardware to be kept as spare stock.
- Eases troubleshooting software problems, as references to various objects like network ports and local disk(s) will always be the same on each OpenStack node.

## Using Configuration to Manage Storage Capacity for Various Cinder Guests

When managing storage capacity for the various guest instances in the OpenStack environment, follow the OpenStack Project/Tenant and User management model. Align the Appliance project property with the OpenStack project property to logically separate capacity used by different groups in the organization. Do likewise with the Appliance target-group property to create different target groups for different projects.

For larger configurations, use a combination of Appliance pool and project properties, to logically organize and separate volumes on the Oracle ZFS Storage Appliance. Use the pool RAID properties to select the appropriate data availability level and I/O characteristics required for a specific group of OpenStack guests.

If OpenStack compute nodes are separated in different groups for different types of OpenStack guest environments, use the Appliance target group and initiator group to restrict the visibility of volumes to the OpenStack nodes within just that type of group.

## Use Multiple Backend Volume Definitions

As previously discussed, Cinder OpenStack back-end definition specifies the Oracle ZFS Storage Appliance pool, project, iSCSI access properties, and specific volume characteristic properties that are used. By using multiple back-end definitions, you can logically organize volumes by workgroups, I/O characteristics, and restricted visibility (among other constructs).

Consider the following for each logical organizational function.

### *Organize by OpenStack user/organizational workgroups.*

Use the Oracle ZFS Storage Appliance pool properties to physically separate OpenStack volumes within the Appliance into a certain disk group. Use the raid type option of the pool to match the data availability requirements for a specific workgroup.

To further control access restrictions and volume visibility, use the Appliance `targetgroup` and `initiatorgroup` properties.

Further data set separation can be implemented by using multiple Oracle ZFS Storage Appliances. Separation per Appliance can be used to separate different workgroup storage capacity by specifying a different Appliance in the back-end `san_ip` and `zfssa_target_portal` property for each back end.

Note that the use of multiple pools must be carefully considered. The Oracle ZFS Storage Appliance uses SSD drives as a secondary data cache system. They are allocated on a per-pool basis. This implies that the highest cache utilization is achieved when as many disks in the subsystem as possible share the same cache devices. Spreading the SSD devices over different pools reduces overall secondary cache utilization.

### *Organize by I/O characteristics.*

Different types of I/O application profiles benefit from tuning the volume block size and the Oracle ZFS Storage Appliance cache behavior for a volume by tuning the Oracle OpenStack Cinder ZFSSA driver properties `zfssa_lun_volumeblocksize` and `zfssa_lun_logbias`.

For instance, random transactional type I/O profiles requiring low latency response benefit from smaller volume (LUN) block sizes and cache optimization set for latency. Logging type I/O profiles benefit from volumes using a larger block size and cache optimization set for throughput. By creating different back-end definitions for each, the preferred type of characteristics can be selected when creating a volume by choosing a specific volume-type related to the required back-end.

### *Organize by access restrictions and/or data separation.*

The Oracle OpenStack Cinder ZFSSA `zfssa_target_group` and `zfssa_initiator_group` properties can be used to restrict access to certain OpenStack Compute nodes. Note that at least one iSCSI initiator value must be specified in the `zfssa_initiator` property for the `zfssa_initiator_group` to be created on the Appliance by the Cinder driver.

For information on how to configure multiple Cinder back ends, see the section "Using Multiple Cinder Storage Back-end Definitions" elsewhere in this paper.

## **Increase Block Storage API Service Throughput**

The OpenStack Cinder API service by default runs as one process. This default behavior limits the number of API requests that the Cinder Block Storage service can process at any given time. To increase the API request throughput, increase the number of Cinder API service processes. In the `/etc/cinder/cinder.conf` file, change the following property to the required number of threads:

```
# Number of workers for OpenStack Volume API service. The
# default is equal to the number of CPUs available. (integer
# value)
osapi_volume_workers=4
```

The OpenStack Cinder service is one of a number of OpenStack services for which a number of so-called worker threads – including Keystone, Nova, Neutron, Heat and Glance – can be specified. When using the default for each (not specifying any value in the configuration file of the service) the related service spawns as many instances as there are virtual CPU cores on the server. This is not a workable situation. So the total number of worker threads of all OpenStack services needs to be carefully balanced to prevent oversubscribing the total amount of available CPU resources.

Use the command `psrinfo -pv` to determine the number of CPU cores and virtual processors supported by the Oracle server.

```
Controller-adm-node:~# psrinfo -pv
The physical processor has 8 cores and 64 virtual processors (0-63)
```

....

```
SPARC-T4 (chipid 0, clock 2848 MHz)
```

Note that the Cinder API service only deals with administrative data traffic between the Oracle OpenStack ZFSSA Cinder driver and the Oracle ZFS Storage Appliance and not with the actual data traffic. The value for the `osapi_volume_workers` is influenced by the number of Oracle ZFS Storage Appliances used in the architecture and the expected level of Cinder volume administrative commands. A value between 2 for small environments and 8 for a very large environment is recommended.

The OpenStack service Keystone is probably the most important service when tuning the number of worker threads. Note that increasing the number of OpenStack services worker processes also increases the number of connections to the SQL database used by OpenStack to store status and configuration information. When experiencing any related SQL connect errors in OpenStack log files, check the `max_connection` parameter in the SQL environment setup. Currently its default value is 151. You might need to increase this value if you increase the number of worker threads.

```
mysql> show variables like '%connections%';
```

```
+-----+-----+
| Variable_name | Value |
+-----+-----+
| max_connections | 151 |
| max_user_connections | 0 |
+-----+-----+
```

## Managing Volumes

The OpenStack Nova Service creates and manages guest instances. When Nova creates a new instance, it uses an ephemeral disk to create a guest VM OS boot and root partition. An ephemeral disk is purged and deleted when the instance is terminated. This also includes all application data on that volume. So create separate volumes for important application data that needs to be kept or reused by other instances.

## Use Volume Labels

After a volume is created and attached to an OpenStack guest instance, it still needs to be initialized. When volumes are used as raw device by an application on the guest instance, the application takes care of that step.

When the volume is going to be used as a block device, you need to initialize the volume by partitioning it, typically in two steps:

- Creating one or more partitions on the volume (LUN from the operating system perspective) using the Solaris `format` command
- Creating an initial file-system structure on those partitions.



Special consideration should be given to the alignment of the partition to the logical block size (volblocksize of a LUN on the appliance) and the alignment of the file system structure to the partition. See Oracle's white paper on this topic, 'Aligning Partitions to Maximize Storage Performance'.

When creating a ZFS file-system, the `zpool create` function can take care of the creating a properly aligned partition and the file-system structure in one step. Specify the LUNs raw device name(s) in the CLI `zpool create` command.

When creating partitions on a LUN it is good practice to create a unique label on each LUN to make it easier to identify then.

The `format` utility shows the device name, its capacity and the associated label with the LUN for easy identification:

```
compute-node1: #format
0. c0t5000CCA016B14590d0 <HITACHI-H109030SESUN300G-A31A-279.40GB>  solaris
    /scsi_vhci/disk@g5000cca016b14590
    /dev/chassis/SYS/HDD4/disk
1. c0t600144F0AE323ECA0000559881E30043d0 <SUN-ZFS Storage 7330-1.0-4.00GB>  sales-boot
    /scsi_vhci/ssd@g600144f0ae323eca0000559881e30043
3. c0t600144F0E17D38F300005570337B000Dd0 <SUN-ZFS Storage 7120-1.0-20.00GB> sales-data
    /scsi_vhci/ssd@g600144f0e17d38f300005570337b000d
4. c0t600144F0E17D38F3000055801909001Fd0 <SUN-ZFS Storage 7120-1.0-34.00GB> eng-boot
    /scsi_vhci/ssd@g600144f0e17d38f3000055801909001f
5. c0t600144F0E17D38F30000558152A10020d0 <SUN-ZFS Storage 7120-1.0-35.00GB> eng-data
    /scsi_vhci/ssd@g600144f0e17d38f30000558152a10020
6. c0t600144F0E17D38F30000555CA6D4001Dd0 <SUN-ZFS Storage 7120-1.0-20.00GB> mytst
    /scsi_vhci/ssd@g600144f0e17d38f30000555ca6d4001d
```

## Troubleshooting Configurations

The OpenStack environment is complex and difficult to troubleshoot. Many OpenStack components log events and errors into a log file. Normally, log files are located in `/var/svc/log` and `/var/log/<openstackcomponent>`. Note, however, that you can specify a different location for log files in their respective component's configuration files.

Start with checking the Oracle Solaris Service Management status of the OpenStack component services by using the `svcs -xv` command. If a service failed to start, its corresponding log file is listed and should be checked for further details.

You can find detailed OpenStack troubleshooting information in the *Installing and Configuring OpenStack in Oracle Solaris* document. Also check the '[Known Limitations](#)' chapter in that document.

Access of OpenStack Guest instances to the OpenStack volumes is directly handled by the Oracle Solaris Zoning functions and the iSCSI stack in the Oracle Solaris global zone. The architecture in this document uses a network setup to access the storage subsystems that bypasses the OpenStack Neutron gateway to external network function to avoid any I/O bottlenecks.

## Troubleshooting iSCSI Connectivity

As described earlier, each OpenStack node needs to be configured to connect to the storage IP subnets. In cases of volume access issues, verify the storage IP network connectivity on the related OpenStack compute node by using the `ping` and `traceroute` commands. For example:

```
compute-adm-1:~# ping 192.168.222.29
192.168.222.29 is alive

compute-adm-1:~# traceroute 192.168.222.29
traceroute: Warning: Multiple interfaces found; using 192.168.222.25 @ net222002
traceroute to 192.168.222.29 (192.168.222.29), 30 hops max, 40 byte packets
 1 zfssa-1-data222 (192.168.222.29)  0.171 ms  0.096 ms  0.089 ms
compute-adm-1:~#
```

Note that the output of `traceroute` also shows the route used to the storage subsystem and should be a route within the 192.168.222.0 subnet.

If the command output indicates network connectivity issues, use the `dladm` command to check if the network port is physically connected to a network:

```
Compute-adm-1:~# dladm
```

LINK	CLASS	MTU	STATE	OVER
net0	phys	1500	up	--
net1	phys	1500	unknown	--
net2	phys	9000	up	--
net3	phys	9000	up	--
net222002	vnic	9000	up	net2
net223003	vnic	9000	up	net3

Note that the ports used for the storage subnets are set up to use an MTU size of 9000. To verify that large packets travel through the network without being broken up, use the `ping` command to specify a packet size, like `ping -s -v 192.168.222.29 9000`.

Next, verify what iSCSI targets and target devices have been detected using the `iscsiadm` command. In large environments, this may show many iSCSI target IQNs and LUNs. To limit the iSCSI LUNs to the Oracle ZFS Storage Appliance you want to check, specify the configured iSCSI with the `iscsiadm -S` option.

The Appliance's Edit iSCSI Target window lists the target name and target IQN. For example:

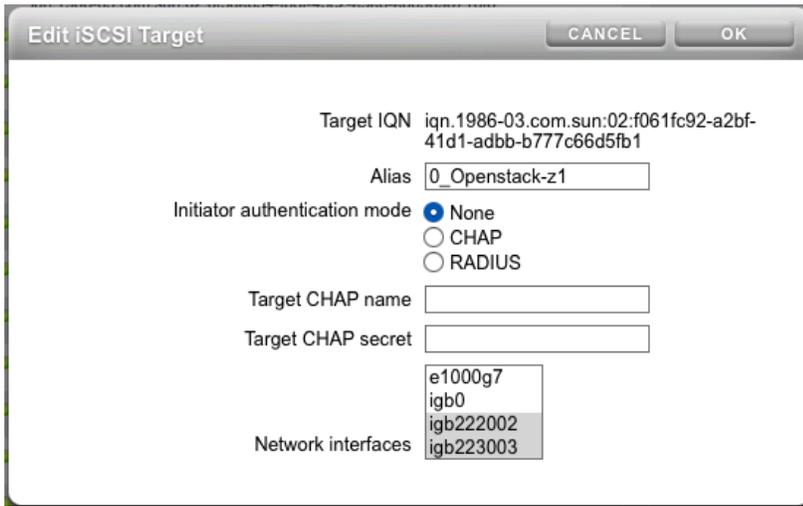


Figure 15. iSCSI target definitions shown in Oracle ZFS Storage Appliance iSCSI target window

All LUNs exposed through this target are visible over the two network interfaces `igb222002` and `igb223003` on the Appliance that have been connected to the two storage subnets. The `iscsiadm` command confirms the LUN is visible on the `compute-node-1` over the two storage subnets.

```
Compute-adm-1:~# iscsiadm list target -S iqn.1986-03.com.sun:02:f061fc92-a2bf-41d1-adbb-b777c66d5fb1
```

```
Target: iqn.1986-03.com.sun:02:f061fc92-a2bf-41d1-adbb-b777c66d5fb1
  Alias: 0_Openstack-z1
  TPGT: 3
  ISID: 4000002a0000
  Connections: 1
  LUN: 0
    Vendor: SUN
    Product: ZFS Storage 7120
    OS Device Name: /dev/rdisk/c0t600144F0E17D38F30000557EB40A001Dd0s2
Target: iqn.1986-03.com.sun:02:f061fc92-a2bf-41d1-adbb-b777c66d5fb1
  Alias: 0_Openstack-z1
  TPGT: 2
  ISID: 4000002a0000
  Connections: 1
  LUN: 0
    Vendor: SUN
    Product: ZFS Storage 7120
    OS Device Name: /dev/rdisk/c0t600144F0E17D38F30000557EB40A001Dd0s2
```

Use the command `mpathadm` to retrieve further details on the multipath configuration of the LUNs.

## Troubleshooting Using Analytics

The Analytics function from the Oracle ZFS Storage Appliance can be used to:

- Check and analyze the network traffic between the Appliance and the OpenStack nodes.
- Check and understand the nature of the I/O load in terms of block sizes used and type of I/O patterns, such as sequential versus random. Understanding I/O load profiles helps to determine the settings of back-end properties like `zfssa_lun_logbias` and `zfssa_lun_volblocksize`.
- When using multiple pools, check on load distribution among pools.

The following figure shows a screen capture of analytics displaying the iSCSI I/O load, broken down by LUN, by client, and by network device. Multipathing has been used and Analytics shows how the I/O load is distributed over the two Oracle ZFS Storage Appliance interfaces and the two initiators from the client.



Figure 16. Oracle ZFS Storage Appliance Analytics function showing LUN load balancing over two iSCSI network path

## Appendix A: Cinder Configuration File

The following listing shows part of the `/etc/cinder/cinder.conf` file that is relevant to the ZFSSA Cinder driver as used for the architecture described in this paper. For a detailed description of the ZFSSA Cinder driver properties and its syntax, see the [ZFSSACinderDriver.README](#) file.

```
# In cinder.conf default section:
[DEFAULT]
# Default volume type to use (string value)
default_volume_type=zfssa-data
#
# A list of backend names to use. These backend names should
# be backed by a unique [CONFIG] group with its options (list
# value)
enabled_backends=zfssa-log,zfssa-data,zfssa-images,zfssa-mpath,zfssa-test
# Always good to set the 'global' value of the cinder driver to ZFSSA too
volume_driver = cinder.volume.drivers.zfssa.zfssaiscsi.ZFSSAISCSIDriver
# ~~~~~
# Start of Cinder backends used on Oracle ZFS Storage Appliance
# Options defined in Cinder.volume.drivers.zfssa.zfssaiscsi
#
[zfssa-images]
volume_backend_name=os-images
volume_driver = cinder.volume.drivers.zfssa.zfssaiscsi.ZFSSAISCSIDriver
san_ip = zfssa-admin-node1
san_login = OpenStackAdmin
san_password = verysecret
zfssa_pool = pool0
zfssa_project = 0_OpenStack-images
zfssa_initiator_group = 0_Openstack-compute-group1
zfssa_initiator=iqn.1986-03.com.sun:01:0010e03d62b0.5527f3e4,iqn.1986-
03.com.sun:01:0010e03aa278.55281b09
zfssa_target_group = 0_Openstack-z1
zfssa_target_portal = zfssa-1-data222:3260
zfssa_target_interfaces = igb222002,igb223002
zfssa_lun_volblocksize = 8k
zfssa_lun_logbias = latency

[zfssa-log]
volume_backend_name=db-log
volume_driver = cinder.volume.drivers.zfssa.zfssaiscsi.ZFSSAISCSIDriver
san_ip = zfssa-admin-node1
san_login = OpenStackAdmin
san_password = verysecret
zfssa_pool = pool0
zfssa_project = 0_OpenStack-log
zfssa_initiator_group = 0_Openstack-compute-group2
zfssa_initiator=iqn.1986-03.com.sun:01:0010e03d62b0.5527f3e4,iqn.1986-
03.com.sun:01:0010e03aa278.55281b09
zfssa_target_interfaces = igb222002,igb223002
zfssa_target_group = 0_Openstack-z2
zfssa_target_portal = zfssa-1-data222:3260
zfssa_lun_volblocksize = 128k
zfssa_lun_logbias = latency

[zfssa-data]
volume_backend_name=db-data
volume_driver = cinder.volume.drivers.zfssa.zfssaiscsi.ZFSSAISCSIDriver
san_ip = zfssa-admin-node1
san_login = OpenStackAdmin
san_password = verysecret
zfssa_pool = pool0
zfssa_project = 0_OpenStack-1j-data
zfssa_initiator=iqn.1986-03.com.sun:01:0010e03d62b0.5527f3e4,iqn.1986-
03.com.sun:01:0010e03aa278.55281b09
zfssa_initiator_group = 0_OpenStack-compute-group3
zfssa_target_group = 0_OpenStack-z3
zfssa_target_interfaces = igb222002,igb223002
zfssa_target_portal = zfssa-1-data222:3260
zfssa_lun_volblocksize = 8k
zfssa_lun_logbias = throughput
```



```
# End ZFSSA ++++++
```

## Appendix B: Oracle OpenStack on Oracle Solaris Open Issues

As of the time of this whitepaper's publish, the following bugs were open. Each of these bugs can be dealt with as explained for each mentioned bug. Software versions used in this document:

- Oracle Solaris 11.3 and Oracle Solaris development version 12
- Oracle OpenStack Juno release
- Oracle OpenStack ZFSSA Cinder driver version 1.0 as delivered with Oracle OpenStack Cinder package for Oracle Solaris

**TABLE 3. BUGS ENCOUNTERED IN ORACLE OPENSTACK FOR ORACLE SOLARIS 11.3**

Bug Number	Issue	Impact	Workaround
21952273	Default value in <code>cinder.conf</code> file of <code>volume_dd_blocksize=1M</code> not supported by Oracle Solaris <code>dd</code> utility.	Issue causes direct failure of cinder command to create a cinder volume from a Glance image.	Use <code>'volume_dd_blocksize=1048576'</code> instead of <code>'volume_dd_blocksize=1M</code> in the <code>cinder.conf</code> file.
21909435	<code>dd</code> utility fails on not having write privileges on target device	When issuing a cinder create volume from image command, the command fails because <code>dd</code> utility does not have the proper privileges to write to the created cinder volume.	Use <code>cinder-volume:solaris:cmd:RO::/usr/bin/dd:privs=file_dac_read;uid=0</code> in the <code>/etc/security/exec_attr.d/cloud:openstack:cinder</code> file.

**TABLE 4. BUGS ENCOUNTERED IN ORACLE ZFS STORAGE APPLIANCE CINDER DRIVER VERSION 1.0**

Bug Number	Issue	Impact	Workaround
21951996	ZFSSA property <code>zfssa_target_portal</code> does not always accept logical node name	Impact: Using IP addresses in OpenStack config files is very undesirable as <code>target_portal</code> info is stored as part of metadata of volumes, making it impossible to change IP addresses of nodes at a later stage.	Specify IP address of Appliance data port instead of node/host name in <code>zfssa_target_portal</code> property for each back-end definition in <code>cinder.conf</code> file <code>zfssa_target_portal = 192.168.222.29:3260</code>
21894294	No initiator group created on appliance when no <code>zfssa_initiator</code> specified or contains syntax error	Impact: Creating undesired effect of a volume assigned to the Appliance 'default' initiator group during volume attach operation	Always specify a <code>'zfssa_initiator'</code> property for each cinder back-end in the <code>cinder.conf</code> file
21909542	error when using same initiator in different back-end definitions on same Appliance	Impact: Same initiator value for initiator property cannot be used in different back-end definitions with different initiator group names on same Appliance.	Manually create the required initiator groups on the Appliance and manually populate them with the required initiators

## References

- Oracle ZFS Storage Appliance Product Information  
<http://www.oracle.com/us/products/servers-storage/storage/nas/overview/index.html>
- Oracle ZFS Storage Appliance White Papers and Subject-Specific Resources  
<http://www.oracle.com/technetwork/server-storage/sun-unified-storage/documentation/index.html>

Recommended related whitepapers:

- "Networking Best Practices with the Oracle ZFS Storage Appliance"
- "Aligning Partitions to Maximize Storage Performance"
- Oracle ZFS Storage Appliance Document library  
[http://docs.oracle.com/cd/E51475\\_01/index.html](http://docs.oracle.com/cd/E51475_01/index.html)
- The *Oracle ZFS Storage Appliance Administration Guide* is also available through the Oracle ZFS Storage Appliance help context.  
The Help function in Oracle ZFS Storage Appliance can be accessed through the browser user interface.
- README file for Oracle OpenStack ZFSSA Cinder driver for Oracle ZFS Storage Appliance  
<https://openstack.java.net/ZFSSACinderDriver.README>
- OpenStack Configuration Reference Manual, Oracle ZFSSA iSCSI Driver in Volume Drivers Section  
<http://docs.openstack.org/juno/config-reference/content/zfssa-volume-driver.html>
- Installing and Configuring OpenStack in Oracle Solaris 11.2  
[http://docs.oracle.com/cd/E36784\\_01/html/E54155/index.html](http://docs.oracle.com/cd/E36784_01/html/E54155/index.html)
- Oracle Solaris OpenStack on OTN  
<http://www.oracle.com/technetwork/server-storage/solaris11/technologies/openstack-2135773.html>
- Getting started with OpenStack on Oracle Solaris 11  
<http://www.oracle.com/technetwork/articles/servers-storage-admin/getting-started-openstack-os11-2-2195380.html>
- Download OpenStack Unified Archive  
<http://www.oracle.com/technetwork/server-storage/solaris11/downloads/unified-archives-2245488.html>
- Download Oracle Solaris 11  
<http://www.oracle.com/technetwork/server-storage/solaris11/downloads/>
- OpenStack documentation repository  
<http://docs.openstack.org>
- The NIST Definition of Cloud computing, Publication 800-145  
<http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf>

- 
- NIST Cloud Computing Reference Architecture  
[http://www.nist.gov/customcf/get\\_pdf.cfm?pub\\_id=909505](http://www.nist.gov/customcf/get_pdf.cfm?pub_id=909505)



Oracle Corporation, World Headquarters  
500 Oracle Parkway  
Redwood Shores, CA 94065, USA

Worldwide Inquiries  
Phone: +1.650.506.7000  
Fax: +1.650.506.7200

---

CONNECT WITH US

-  [blogs.oracle.com/oracle](https://blogs.oracle.com/oracle)
-  [facebook.com/oracle](https://facebook.com/oracle)
-  [twitter.com/oracle](https://twitter.com/oracle)
-  [oracle.com](https://oracle.com)

### Integrated Cloud Applications & Platform Services

Copyright © 2015, Oracle and/or its affiliates. All rights reserved. This document is provided *for* information purposes only, and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document, and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group. 0615

Using the Oracle ZFS Storage Appliance as Storage Back End for OpenStack Cinder  
December 2015

Author: Peter Brouwer, Application Integration Engineering