Service Providers: Future-Proof Your Cloud Infrastructure

Service providers must choose their path carefully to remain competitive in the outsourced IT market. The rapid pace of public cloud innovation combined with aggressive pricing leads to only a few viable alternatives. OpenStack is the platform of choice for your next generation software-defined infrastructure data center.

Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>How Service Providers Can Thrive in the Era of Hyperscale</td>
<td>2</td>
</tr>
<tr>
<td>Services Implementation</td>
<td>3</td>
</tr>
<tr>
<td>OpenStack Components</td>
<td>5</td>
</tr>
<tr>
<td>Other Considerations</td>
<td>13</td>
</tr>
<tr>
<td>Conclusion</td>
<td>15</td>
</tr>
</tbody>
</table>
How Service Providers Can Thrive in the Era of Hyperscale

The success of cloud and managed service providers is dependent on the underlying platform selected for their cloud infrastructure. To remain viable, this infrastructure must be scalable, flexible, integrate well with existing hardware/software investments, support open API’s, and offer end-to-end automation.

In today’s competitive environment, cost is probably the highest concern as you choose your next generation platform (See: The Total Economic Impact of SUSE OpenStack Cloud¹). Competing with hyperscale cloud providers requires an evolving platform with extensive development support to remain competitive with the rapid advances in capabilities and feature sets while avoiding vendor lock-in. OpenStack is the choice for an increasing number of service providers.

Cloud Positioning

Today, almost every organization understands the convenience factors associated with utilizing resources—compute, network and storage—available in a cloud setting. Depending upon the local expertise, organizations have quite a few cloud computing options available. Some may simply let individuals utilize public cloud utilities for development or production workloads. Others may choose to rely only on internal cloud instances, given enough local support and expertise to maintain and operate the internal cloud infrastructure. Finally, many organizations will utilize both public and private cloud options with a hybrid cloud approach.

Many recent studies have shown a preference for customers to place development workloads in public clouds while keeping production workloads on premise due to concerns on security, performance, and data sovereignty (in some cases). Hyperscale cloud vendors typically provide inexpensive capacity, but often do not have the expertise to help customers with specific applications or security requirements. Local cloud service providers often step in to manage the overall environment and help with applications expertise with a higher level of service.

Business Opportunities

Currently the dominant non-proprietary technology choice to provide cloud computing functionality is OpenStack. Started as a collaboration between NASA (for the compute scheduling, now referred to as Nova) and Rackspace (for the object store, now referred to as Swift), this project has grown to thousands of contributors, hundreds of supporting companies, tens of sub-projects with a supporting OpenStack foundation. Anyone is welcome to participate in the code and documentation generation plus there is a large directory of how the technology is being used in various verticals and industry segments. Due to the open source philosophy, it is easy to find peers and join a community of like-minded individuals to exchange information, experience and knowledge.

For service providers utilizing OpenStack technology to operate a cloud offering, there are several options for deployment.

Since OpenStack is a collection of open source projects, based upon the Apache 2.0 License, CSP’s could track the semi-annual release cycles and directly download and use the source code. This “do it yourself” option would require dedicated professionals very familiar with interacting with the open source community and being very adept at deploying such Python-based projects on top of their desired Linux distribution. Another option is to utilize Linux packages for the OpenStack projects that are prepared for some of the more popular Linux distributions. Like the first option, a fair amount of experience, rigor and skill is needed to deploy OpenStack with this approach since some deployment details are still left to the implementer. Both of the first two options also leave the maintenance tasks, like tracking of updates and security concerns, to the administrator along with staying current with upstream offerings over time. As another alternative, several companies, including SUSE, offer a complete, ready-to-use OpenStack distribution. Typically, these solutions offer a deployment framework, access to updates over time, an upgrade process to move forward across versions and a defined life cycle for support requests. Further, the vendors of hardware and software already in place in your data center will typically test their drivers against popular distributions vs. individual upstream packages.

The remainder of this document presents the choices and implementation details for an OpenStack based cloud service, relevant to a service provider plus other technical considerations that weigh into keeping a viable offering. Each of the sections will contain a reference to SUSE’s OpenStack Cloud and the relevant process, technology and options it provides that are applicable to the topic.

Tip
For general information on the SUSE OpenStack Cloud solution, please refer to the SUSE OpenStack Cloud 8 documentation website.²

Services Implementation
Multi-Deployment Topologies
As a service provider, one of the first considerations is how distributed your collection of cooperating cloud instances will be. There are differing concepts and terms to describe this distribution, some of which are based upon the actual OpenStack implementation approach itself. These architectures (summarized in Table 1: Use Cases for Different OpenStack Topologies, page 5) are labeled regions, availability zones, host aggregates, and cells.

Figure 1. Cloud Topologies

2  SUSE OpenStack Cloud 8 Documentation: www.suse.com/documentation/suse-openstack-cloud-8/
REGIONS
Often meant to describe the geographic partitioning of resources, a region is typically a collection of all the necessary OpenStack services in a given locale. Different regions can share a few of the available services, for example to provide access control and the user interface. This does imply that users need to explicitly select a target region. Of course the granularity of the physical boundary distance can be defined by the service provider as they begin to do multi-region or multi-site deployments.

As one extrapolates out from a centralized data center approach, service providers may offer many, yet typically small footprint, cloud instances. This may be appropriate for Fog, Edge\(^3\), and sometimes even retail types of deployments. Depending upon the use case, these may be completely standalone or coupled and scheduled in ways like regions or in the other OpenStack-specific fashions mentioned in the following sections.

AVAILABILITY ZONES
Within a region, compute resource nodes can be logically grouped into an Availability Zone (AZ). This is to ensure the launching of a new workload happens where it is needed and often provides a level of availability across zones if the workloads are so implemented. Availability Zones are often isolated in different data centers within close proximity to each other (this distance is limited by latency requirements for synchronous replication). A host can only be a member of one AZ at a time. Typically, end-users have visibility into which AZ they place their workloads.

HOST AGGREGATES
Host Aggregates can be thought of as method to further partition availability zones. Based upon admin-defined meta-data tags, compute resources can be logically grouped with common characteristics, say those with a certain hypervisor type, possessing a graphics-processor unit or having a particular storage or network capability. Membership of these compute nodes can straddle both host aggregates and availability zones (see Figure 2).

---


---

**Figure 2. Host Aggregates**
CELLS
Cells are a strategy for scaling compute nodes to large numbers within a region. In a monolithic implementation, message queues and databases can become saturated at high compute node counts. Based upon a tree structure, the top-level cell only has an API functionality, but no compute resources (see Figure 3). This allows control across multiple cloud instances while distributing/scaling the message queue and database in a hierarchical structure.

SUSE OPENSTACK CLOUD CONSIDERATIONS
- All of the above deployment technology types are supported. However, due to scalability concerns, only a finite number of compute nodes are supported in any given instance and the control plane nodes need to be managed and scaled accordingly. The networking infrastructure needs to be planned carefully to provide the necessary bandwidth and performance for the expected workloads.
- In the next sections of this paper, specific OpenStack services will be described, and as applicable, noted where functionality is enhanced or impacted by the distribution topologies selected.

OpenStack Components
An OpenStack implementation consists of many services that are integrated together to provide the overall solution (see Figure 4 on the following page). The OpenStack APIs are key to the extensibility of the platform and enable integration with other systems as well as automation. The OpenStack specific components include identity authentication (Keystone) and the compute nodes (Nova). The storage section contains the pieces required to supply storage to each instance (Object, Block, or File) and the catalog of instance images (Glance). Networking is provided by the Neutron module which gives OpenStack its

Table 1. Use Cases for Different OpenStack Topologies

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Considerations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regions</td>
<td>Linked via REST APIs</td>
</tr>
<tr>
<td></td>
<td>API endpoints are unique to each region</td>
</tr>
<tr>
<td></td>
<td>Failure domains can be server rack, datacenter, or geography</td>
</tr>
<tr>
<td></td>
<td>A host can only be a member of one AZ</td>
</tr>
<tr>
<td></td>
<td>End-user visible/selectable</td>
</tr>
<tr>
<td>Availability Zones</td>
<td>Compute nodes can be assigned to more than one host aggregate group</td>
</tr>
<tr>
<td></td>
<td>Requires enabling of host aggregate filtering in nova.conf</td>
</tr>
<tr>
<td></td>
<td>Not typically exposed to end-users</td>
</tr>
<tr>
<td>Host Aggregates</td>
<td>Single compute endpoint</td>
</tr>
<tr>
<td></td>
<td>Single region</td>
</tr>
<tr>
<td></td>
<td>Additional scheduling layer</td>
</tr>
<tr>
<td></td>
<td>Linked via RPC</td>
</tr>
<tr>
<td>Cells</td>
<td>Single compute endpoint</td>
</tr>
<tr>
<td></td>
<td>Single region</td>
</tr>
<tr>
<td></td>
<td>Additional scheduling layer</td>
</tr>
<tr>
<td></td>
<td>Linked via RPC</td>
</tr>
</tbody>
</table>

4 Cells Documentation: [https://docs.openstack.org/nova/latest/user/cells.html](https://docs.openstack.org/nova/latest/user/cells.html)
Identity Service (Keystone)
Likely the first consideration of OpenStack services is how to manage, authenticate and authorize the use of a cloud instance. For both local and service providers, defining the source of the authentication and the scaling across potentially multiple instances, projects, and deployment topologies is an up-front design decision.

Fortunately, OpenStack’s Identity Service, commonly referred to as Keystone project, provides such credential validation and returns a token. This validation can be federated, sourced, and even combined from any number of authoritative backend services. This driver-based, federated service retrieves the various attributes associated with the token such as role, user, group, domain and project which then get related to the various resource-usage quotas allocated. Keystone is deployed within the controller node of the OpenStack architecture.

Tip
- It is highly recommended that Keystone is presented to users via web-servers with an SSL terminator.

CASE 1: “ALL-IN-ONE” KEYSTONE DEPLOYMENT
In discrete cloud instances, a Keystone deployment is required and can be standalone and self-contained. SUSE recommends that the entire control plane, including Keystone, should be

---

5 Keystone, the OpenStack Identity Service: https://docs.openstack.org/keystone/latest/
deployed in a highly available architecture to avoid Single Points of Failure (SPOF). In order of increasing complexity, the following approaches may be utilized, with the noted tradeoffs:

In the simplest case, one Keystone centralized instance exists and other Keystone instances can access the content from the centralized one in a federated configuration6 (see Figure 5). In this example, Keystone uses tokens to coordinate with a Service Provider (e.g. ISP, Application SP, Storage SP) and/or an Identity Provider (e.g. online service that authenticates users—Google, Facebook, etc.).

**CASE 2: GEOGRAPHICALLY SEPARATED KEYSTONE DEPLOYMENT**

In more geographically separated regions, a Keystone database approach, where one is the primary and the others are secondary and synced in an asynchronous fashion might be the preferred method (see Figure 6).

**Tradeoffs:**
- Downtime of centralized instance limits changes
- Propagation of writes to central source

For low-latency, nearby regions, Keystone deployments, based upon localized SQL databases can be shared in a synchronous, HA—clustered fashion (see Figure 7).

**Tradeoffs:**
- Increased complexity
- Limited geographic scaling

---

6 Federated Identity: [https://docs.openstack.org/keystone/pike/admin/federated-identity.html](https://docs.openstack.org/keystone/pike/admin/federated-identity.html)
CASE 3: KEYSTONE INTEGRATION WITH AN EXISTING BACKEND SERVICE

As many established service providers may already have a centralized backend service readily available, each Keystone instance can leverage this source to scale the OpenStack Identity Service across as many OpenStack instances as desired (see Figure 8). Such leveraging addresses some of the above tradeoffs and also that of the normal ebb and flow of active customers utilizing the service.

When leveraging an existing centralized backend service, Keystone instances in each region can be pointed to a common or specific backend. This deployment choice still has ramifications to consider.

**Advantages:**
- Tighter access control
- Adherence to locality constraints
- Single-source of credentials, unified access
- Reduced maintenance efforts of accounts

**Trade-offs:**
- Single point of failure with connectivity from each instance to central authority
- Higher admin maintenance and complexity due to less uniform deployments

**SUSE OpenStack Cloud Considerations:**
Deploying Keystone in a standalone, local fashion or in a clustered, high-availability manner are supported as well as the use of existing LDAP-based back-ends (details are available in Installing with Cloud Lifecycle Manager: [www.suse.com/documentation/suse-openstack-cloud-8/book_install/data/book_install.html](https://www.suse.com/documentation/suse-openstack-cloud-8/book_install/data/book_install.html)).

**Dashboard Service (Horizon)**
A close second in terms of importance to consider for OpenStack services is how to present and manage user interaction with a cloud instance. Requesting resources, utilizing the various services and tracking the status of requested workloads is all core to the total customer experience.

Fortunately, OpenStack’s Dashboard Service project commonly referred to as Horizon, provides a web-based user interface to all the underlying services, which are abstracted per the Infrastructure-as-a-Service instance across compute, networking and storage resources. Resources can be requested, started, paused, stopped and deleted upon demand. Further, the Dashboard Service only allows modifications and requests based upon the user’s role, privileges and allotted quota.

Consistent with many other service provider offerings, such a dashboard service may already be a common practice. A cloud admin is able to see the entire set of resources, making certain

---

7 [Integrate Identity with LDAP](https://docs.openstack.org/keystone/pike/admin/identity-integrate-with-ldap.html)
8 [Horizon: The OpenStack Dashboard Project](https://docs.openstack.org/horizon/latest/)
things publicly visible to all users and configuring quotas for each set of users. A non-admin user has a restricted view to anything public and to those resources within their own project and is subject to the quota limits for consumption.

**Tips**
- If so configured, a user can also download a configuration file to interact with the cloud instance in a command-line fashion for cases where additional automation is desired.
- When scaling across multiple cloud instances, different regions may have a distinct Horizon deployment. Unless this is a purposely separated region, it is advisable to have a shared Identity Service.
- Of course, the Horizon deployment can also make multiple regions visible without having to directly access a given region’s dashboard. Since Horizon heavily relies upon the OpenStack Identity Service, it is necessary to:
  - Have a shared Keystone authentication scheme in each region
  - Enable multi-domain support in both services
  - Explicitly configure each of the desired “AVAILABLE_REGIONS” or “OPENSTACK_KEYSTONE_REGION_CHOICES” API endpoints in the Horizon configuration

**SUSE OpenStack Cloud Considerations:**
Instances of the Dashboard Service can be deployed on dedicated hosts or in a clustered, high availability manner. Visibility of multiple regions can also be configured as per the product documentation.

**Image Service (Glance)**
Another OpenStack service worth considering as a shared resource is the repository for workload images. For consistency of workload deployments, having the same image visible from the Horizon Dashboard Service across distributed cloud instances helps users utilize all shared resources.

Fortunately, the OpenStack Image Service project, commonly referred to as Glance, is based upon an object store for the repository of images. These are then visible in the Dashboard Service, along with the respective properties and meta-data. Shared or public images can be visible to all users, to allow a consistent, known starting point for workload deployment. If quotas and role membership allow, images specific to a given project or tenancy can be uploaded, updated and deleted, either from the Horizon web interface or from the command-line client.

**Tips**
- If so configured, a user can also download a configuration file to interact with the cloud instance in a command-line fashion for cases where additional automation is desired.
- Glance can be based on a number of backend storage repository plug-ins (see Figure 10). Local, direct-attached file-based storage can be used. Other OpenStack services, like Swift or Cinder, can be leveraged. RADOS block devices from a Ceph-based storage are also an option as is VMware.

---

![Glance Diagram](https://docs.openstack.org/glance/latest/)

**Figure 10. Glance Image Service**

If a service provider already has one of the previous storage technologies in place across the distributed deployments, this can be leveraged for Glance to make images available. As distances

---

9 Glance Documentation: [https://docs.openstack.org/glance/latest/](https://docs.openstack.org/glance/latest/)
and latencies increase, this approach becomes more difficult and mirroring or synchronization technologies would have to be employed to offer the same level of consistency. Such efforts would be further complicated by not only the image replication, but also the corresponding properties and meta-data needing to be transported for any publicly visible images at the very least.

**SUSE OpenStack Cloud Considerations:**
- Files and meta-data in the Image Service can be managed across the above cited storage backends.
- A dedicated host can provide this service or it can be deployed in a clustered, high available manner.

---

**Network Service (Neutron)**

One of the core OpenStack services requiring upfront consideration, not only for a stand-alone cloud instance, but even more so for distributed deployments is the Network Service. Multiple aspects of networking technology are fundamental in the OpenStack landscape. The OpenStack Network Service project, commonly referred to as Neutron, provides all of the abstraction over the physical infrastructure. Designed in a modular, plug-in based architecture, there are a large number of plug-in types, and layer 2 mechanism drivers across a wide range of driver types. Neutron enables OpenStack to integrate with existing vendor networking hardware/software providing a flexible way to extend the network. The overall software architecture is shown in Figure 11 below:

---

**neutron-server**
- Controller node(s)
- Services networking API
- Enforces network model
- IP addressing for each port

**neutron-*plugin-agent**
- Provides access to a persistent database
- Utilizes AMQP protocol

**Software Defined Networking Service**
- Provides additional network services to tenant networks
- Utilizes REST APIs

**neutron-*-plugin-agent**
- Compute nodes
- Manage local vswitch

**neutron-L3-agent**
- L3/NAT forwarding for external network access

**neutron-DHCP-agent**
- DHCP resources for tenant networks

**Message Queue**
- Utilizes RabbitMQ for SUSE OpenStack Cloud
- Facilitates command and control functions

**Figure 11. Network Service Architecture**

- **neutron-server**: Accepts API requests and routes to the appropriate neutron-plugin-agent for execution.
- **neutron plugins**: Performs orchestration of the backend devices such as creating ports, networks, subnets, and IP addressing. Agents and plugins differ depending on the network virtual/physical switch technologies being used (e.g. Cisco, Juniper, OpenSwitch, OpenFlow, Linux Bridging, etc.)
- **neutron agents**: Run on the Compute and/or Network Nodes. Receive commands from plugins on the neutron-server and execute them on the individual nodes.
In any OpenStack deployment, multiple network subnets are in use. Some subnets are for intra-cluster communication and do not need to be visible outside the cluster. Other subnets are only used for deployment aspects, again not needing to be visible beyond the cluster. Additional subnets might be leveraged for management interfaces, like Baseboard Management Controllers (BMC) and may or may not need to exist outside the cluster. Finally there may be one or more subnets set aside for the workloads to have a public access to present their deployed services. For each of these, the service provider needs to assess how distributed cloud instances might need consistency, say for the public access networks, and how many exist in a truly private, intra-cluster sense within a region. The challenge is then around provisioning a single network range across the distributed, physical sites.

The next network consideration is around availability. To reduce the impact of a component or network interface outage, Linux provides the ability to bond or team multiple connections together. This can be done in a number of modes, to effectively provide availability and failover or to leverage multiple interfaces’ bandwidth in an aggregated fashion. As with the previous consideration, once bonding is utilized, this adds to the complexity of scaling this across distributed, physical sites. In addition, the increased bandwidth requirements become harder to scale over wide-area network locations.

**Tips**
- Due to the complexity and wide variety of deployment configuration options, it is advisable to spend some time researching each of the alternatives and doing a proof-of-concept deployment to assess how the selected options fit into the overall architectural decisions.

**SUSE OpenStack Cloud Considerations:**
- The Neutron role can be deployed on a dedicated host or in a clustered, high availability fashion. As the scale of the cloud instance increases, the suggested deployment is to move the Neutron role either to dedicated cluster nodes or to possibly utilize the Distributed Virtual Router (DVR) methodology to reduce potential centralized bottlenecks.
- Nearly all of the upstream options are supported in the Neutron role. This also includes integration with vendor-specific Software-Defined-Network options which will likely already be in place in service provider data centers.
- Refer to the product documentation which details the number of networks required and the various bonding and teaming modes that might be applicable.
- Containers managed by Kubernetes (via Magnum or SUSE Containers as a Service Platform) can be made resilient by deploying the underlying worker nodes on multiple compute nodes.

**Compute Service (Nova)**
One of the core services abstracted by OpenStack is that of compute resource scheduling. Workloads can be provisioned as virtual machines, bare-metal instances and even system containers. User interaction can be via the Dashboard Service or via the...
OpenStack command-line client. Fortunately, OpenStack’s Compute Service project\(^\text{11}\), commonly referred to as Nova, provides a set of daemons to facilitate the scheduling and management of such workloads. As detailed in the Multi-Deployment Topologies section, compute resources can be scaled across regions, availability zones, host aggregates and cells. The Nova architecture is shown in Figure 13 at right:

The following OpenStack Services are dependencies:

- Identity (Keystone)
- Image (Glance)
- Network (Neutron)
- Bare-metal (Ironic)
- Magnum (Container)

In addition, various allocation ratios of virtual to physical resources for RAM, CPU and Disk plus amounts of reserved memory for compute hosts are all configurable. Each of these is important for a service provider to balance the expected workloads across the physical resource nodes. Another important consideration is whether live-migration of instance is part of the service provider offering. While simple to enable within a region, this feature also has a dependency on a shared storage element (accessible from each compute node) to expedite migration.

By utilizing these varying topology distinctions, categories of compute resources can be made available from general to very specific use cases.

\(^{11}\) OpenStack Compute (nova): [https://docs.openstack.org/nova/latest/](https://docs.openstack.org/nova/latest/)
Depending upon the latency and distances across multiple regions or sites, by utilizing these varying topology distinctions, categories of compute resources can be made available from general to very specific use cases.

**SUSE OpenStack Cloud Considerations:**
- Supports Ceph deployments in the same fashion as other OpenStack services, with some limiting assumptions. Also supports the integration of an existing, external Ceph cluster to the various OpenStack services which expect persistent data stores.
- Information on SUSE’s Ceph-based product can be found on SUSE’s website.

**Other OpenStack Services**

As mentioned in the introduction section, OpenStack is a collection of many components or sub-projects. Each provides a certain set of functionality that can ultimately be combined to address distinct target customer use cases and needs. To help guide administrators and service providers, the OpenStack Project Navigator provides links to each sub-project’s descriptions and details. Further, the Sample Configurations section starts with a functional, use-case driven area and then highlights which components should likely be used for such a deployment. Details around each component’s adoption percentage, maturity and project age are also valuable assessments to a service provider to consider.

**SUSE OpenStack Cloud Considerations:**
- In the product’s Deployment Guide those OpenStack services that are supported, and even those that are optional or in technical preview status are delineated with descriptions on their implementation specifics. The OpenStack.org website contains additional content on the current status of all active projects as well as sample configurations based on case studies and real-world reference architectures.

**Other Considerations**

This section describes further considerations around the operational aspects of OpenStack deployments. A core reference on this topic is the OpenStack Administrator Guide which covers each of the services around architecture, installation and configuration considerations.

**Security**

Regardless of the approach used to deploy the OpenStack technology, a paramount concern is how to ensure security of the components over time and that of the overall solution. The key differentiator in obtaining the security updates for the components, however, does implicitly lie in how the technology is obtained. Unless one uses the OpenStack vendor distribution approach, this effort lands squarely on the service provider to interact with upstream developers, watching for security updates in the code stream and monitoring typical security venues like the Common Vulnerabilities and Exposures (CVE) database. Then the application of such security updates must be manipulated and applied to the OpenStack infrastructure. Should the service provider choose to follow the OpenStack vendor distribution approach (e.g. SUSE OpenStack Cloud), only the latter step of applying updates must be part of the cloud admin process. Further, by taking advantage of either the built-in resiliency of certain services, like Swift or Ceph, or by deploying the various OpenStack services on clustered, high availability nodes, security updates can be iteratively applied to the infrastructure with little or no service disruption.

A second, and equally important security consideration is that of the overall infrastructure from a usage standpoint. By design, OpenStack is inherently multi-tenant capable, providing isolation for users in different project groups. Also, for each project group and their workloads, security group rules by default are quite restrictive to limit access to the workloads (unless arbitrarily opened), good isolation paradigms are already present. Some proprietary tools exist and can also be used to assess normal network connectivity workflows, identify possible anomalies, and help to tighten down the security group rules.

---

12 SUSE Enterprise Storage: www.suse.com/products/suse-enterprise-storage/
13 OpenStack Overview: www.openstack.org/software/
15 OpenStack Project Navigator: www.openstack.org/software/project-navigator/
16 OpenStack Sample Configurations: www.openstack.org/software/sample-configs#web-applications
17 OpenStack Administrator Guides: https://docs.openstack.org/pike/admin/
By following many of the tips outlined in the previous OpenStack service sections, the authorization aspect is easy to administer and secure. In almost all cases, any public access API should be configured to use HTTPS. Further detailed information on security can be found in the OpenStack Security Guide18.

**Building an Architecture for High-Availability**

Creating an OpenStack architecture that is highly resilient requires many of the standard best practices you would employ in any enterprise class system. The first step is identifying the Single Points of Failure (SPOF) in both hardware (e.g. network infrastructure, power supplies, storage, etc.) and software (including infrastructure and applications)19. For OpenStack, your planning should identify stateless vs. stateful services to better understand critical areas of attention (see Table 2). Stateless services can achieve resiliency through load balancing while stateful services require either active/active or active/passive clustering.

![Figure 14. Example of an OpenStack HA Configuration](image)

### Table 2. Examples of Stateless vs. Stateful Services

<table>
<thead>
<tr>
<th>Stateless</th>
<th>Stateful</th>
</tr>
</thead>
<tbody>
<tr>
<td>nova-api</td>
<td>OpenStack Configuration Database</td>
</tr>
<tr>
<td>nova-conductor</td>
<td>Message Queue</td>
</tr>
<tr>
<td>glance-api</td>
<td>Cinder Volume</td>
</tr>
<tr>
<td>keystone-api</td>
<td>Ceilometer Agent</td>
</tr>
<tr>
<td>neutron-api</td>
<td>neutron-L3-agent</td>
</tr>
<tr>
<td>nova-scheduler</td>
<td>neutron-DCHP-Agent</td>
</tr>
<tr>
<td>Apache webserver</td>
<td></td>
</tr>
<tr>
<td>Cinder Scheduler</td>
<td></td>
</tr>
</tbody>
</table>

An example of a HA architecture can be found in the guide “Planning an Installation with Cloud Lifecycle Manager”20 shown in Figure 14 below.

Get the latest SUSE OpenStack service provider content at: [www.suse.com/csp](http://www.suse.com/csp)

---

18 OpenStack Security Guide: [https://docs.openstack.org/security-guide](https://docs.openstack.org/security-guide)
19 Introduction to OpenStack High Availability: [https://docs.openstack.org/ha-guide/intro-ha.html](https://docs.openstack.org/ha-guide/intro-ha.html)
In this architecture, stateless API requests are sent to the virtual IP address and load balanced. These requests are then sent on to a controller node for processing. The stateful configuration database is clustered with Galera while the message queue has automatic failover clustering.

**Conclusion**

Creating a strategy to evolve your datacenter into software defined infrastructure is time-consuming, but necessary for the long term viability of your business. OpenStack bridges the gap between what you have now and where you need to be to remain competitive. As new technologies are created, they are often introduced on OpenStack first as this is one of the few open cloud technologies with wide support. OpenStack currently supports multiple hypervisors, most of the software defined networking and storage initiatives, upcoming containers/Kubernetes, and Cloud Foundry PaaS to name a few. Future-proof your architecture by starting your OpenStack journey now.

“SUSE OpenStack Cloud gives our customers the power to do anything they want,” says Kankaala. “We are seeing rapid growth in customers building their own revenue-generating services on our platform: they focus on their business opportunity, and we provide the robust cloud service that underpins it.”

KAI KANKAALA
Board Advisor, Sales
Datalounges