Learning OpenStack High Availability

OpenStack is one of the most popular open source cloud computing platforms, mostly used for deploying Infrastructure as a Service (IaaS) solutions. Enabling high availability in OpenStack is a required skill for cloud administrators and cloud engineers in today's world.

This book helps you to achieve high availability and resiliency with OpenStack. This means clustering, fencing, load-balancing, distributed networking, leveraging shared storage, automatic failover, and replication. We start with a basic understanding of what a highly available design is meant to achieve in OpenStack and various ways to achieve high availability in OpenStack, through simple step-by-step procedures.

Through hands-on examples, you will develop a solid knowledge of horizontally-scalable, fault-resistant, and highly-available OpenStack clusters.

Who this book is written for

This book is for OpenStack administrator, cloud administrator, cloud engineer or cloud developer with some real-time understanding of cloud computing, OpenStack and familiarity with Linux command is essential to start with this book.

What you will learn from this book

- Develop an understanding of what a highly available design should be using RTO, MTTR, and SLA concepts
- Set up and configure Galera and RabbitMQ clusters to build an efficient and reliable messaging cluster
- Get to know the advanced areas of network load balancing using configurations of HAproxy and keepalived
- Control active/passive OpenStack services by setting up a cluster for two or more nodes
- Gain a deeper understanding of how OpenStack services work in cooperation with a stateless mode to offer a scalable cloud framework
- Configure and run a distributed neutron installation based on openvswitch and different tunneling protocols

In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 1 'An Introduction to High Availability Concepts'
- A synopsis of the book’s content
- More information on Learning OpenStack High Availability
**About the Author**

**Rishabh Sharma** is currently working as a chief technology officer (CTO) at JOB Forward, Singapore. Prior to working for JOB Forward, he worked for Wipro Technologies, Bangalore, as a solution delivery analyst. He was involved in research projects of cloud computing, proof of concepts (PoC), infrastructure automation, big data solutions, and various giant customer projects related to cloud infrastructure and application migration.

In a short span of time, he has worked on various technologies and tools such as Java/J2EE, SAP(ABAP), AWS, OpenStack, DevOps, big data, and Hadoop. He has also authored many research papers in international journals and IEEE journals on a variety of issues related to cloud computing.

He has authored five technical books until now. He recently published two books with international publications:


He is also an open source enthusiast and writes for the Open Source For You (OSFY) magazine. You can get in touch with him at **ater.rishabh.sharma@gmail.com**.
Preface

OpenStack is a set of software tools and packages used to build private and hybrid cloud computing platforms. It is one of the most popular and widely adopted open source software managed by OpenStack Foundation. Since its introduction in 2010, a huge number of reputed industries, including Red Hat, Intel, HP, IBM, AMD, Canonical, and many others, support OpenStack.

The most promising feature of OpenStack is that it provides a DIY (Do-It-Yourself) approach to cloud computing, and it can easily embrace the new development features. Therefore, companies such as Google and Facebook created their own data storage and cloud services using OpenStack.

High availability typically means achieving 99.99% availability, and basically this can be possible by removing all single point of failure (SPOF), which is also applicable for OpenStack. This book covers all the basic and advance approaches related to achieving high availability in OpenStack with detailed step-by-step explanations, hands-on exercises, and screenshots. These provide you with a real-time practical understanding of implementing high availability in OpenStack. Some customer case studies are also included:


What this book covers

Chapter 1, *An Introduction to High Availability Concepts*, introduces the reader to a basic understanding of high availability in a production environment. The requirements and common design patterns will be explained in details; architectural choices to maximize availability have been critically discussed.

Chapter 2, *Database and Messaging Services*, explains how to obtain resilient and available OpenStack supporting services, such as database (MariaDB with Galera clustering) and RabbitMQ with replicated queues.

Chapter 3, *Load Balancing for Active/Active Services*, delves into the basic and advanced topics of network load balancing and explains with detailed examples, the configurations of HAProxy and keepalived.

Chapter 4, *Clustering, Fencing, and Active/Passive Services*, describe the services in OpenStack that are still not fully stateless, and thus require classical clustering methods such as Pacemaker. This chapter will analyze in depth the construction of a cluster of services with their resource agents and dependencies.

Chapter 5, *Highly Available OpenStack Services*, deals with the scaling and resiliency of all the basic OpenStack services: compute, image, storage, object storage, and dashboard.

Chapter 6, *Distributed Networking*, delves into the details of Neutron Distributed Virtual Routers, multiple L3 agents in active/passive configuration, and third-party networking drivers that offer high availability options. The OpenStack Networking services are the most difficult to scale and render highly available.

Chapter 7, *Shared Storage*, describes how shared storage is a fundamental requirement for high availability and for a quick recovery from a failure (evacuating failed nodes with live migration). This chapter presents different options to provide shared storage to OpenStack and explains their configuration and setup to some extent.

Chapter 8, *Failure Scenario and Disaster Recovery*, focuses on how for any OpenStack operator who strives to bring the cloud they manage to higher availability, understanding the different ways that a failure can occur and how these can impact the performance and availability of the service is a fundamental skill. This chapter analyzes different failure scenarios and proposes solutions to provide a swift and effective recovery to a normal operational level.
Chapter 9, *The Principles of Design for Highly Available Applications*, explains how having a highly available cloud might not be enough, if the application running on top of it doesn't take advantage of the principles and concepts of a resilient design. This chapter mainly explains how correct application design can help improve reliability and uptime of end user services; particular focus has been dedicated to microservice architectures and distributed web applications.

Chapter 10, *Monitoring for High Availability*, covers how the control and maintenance of a cloud is of the utmost importance; visibility in operations and alerting on failures are the basis for correct functioning and quick recovery in case of unexpected outages or planned maintenance windows. The chapter introduces a few key concepts and tools to correctly measure and control the operations of an OpenStack cloud.

Chapter 11, *Use Cases and Real-World Examples*, covers a number of real-world examples of HA deployments. The relevant lessons for a design of similar resilient clouds have been distilled and are presented to the reader.
An Introduction to High Availability Concepts

Over the past couple of years, cloud computing has made a significant impact in transforming IT from a niche skill to a key element of enterprise production environments. From an Infrastructure as a Service (IaaS) point of view, cloud computing is much more advanced than mere virtualization; various industries and online businesses have started moving test, staging, and productions scenarios roles to IaaS and started replacing traditional dedicated resources with on-demand resource models.

OpenStack is one of the most popular and commonly used open source cloud computing platforms, and it is mainly used to deploy infrastructure as a service solution. Enabling high availability in OpenStack is a required skill for cloud administrators and cloud engineers. This chapter will introduce you to high availability concepts, a way of measuring and achieving high availability through architectural design in OpenStack.

In this chapter, we will cover the following topics:

- What does high availability mean?
- How to measure high availability
- Architecture design for high availability
- High availability in OpenStack
What does High Availability (HA) mean?
The basic understanding of high availability in the IT world is when any system continuously operates (100 percent operational) without any down time despite occurrences of failure in hardware, software, and application.

How to measure high availability
In order to measure a system’s high availability, we usually check the total duration of the uptime of the system. For example if system availability is 99 percent, it means that the system is operational for 8672.4 hours throughout the year because the total hours in a year is 8760.

The following formula is used to calculate the total availability, and availability can be increased using high availability techniques to increase MTTF and decrease MTTR that we are going to discuss in this book with respect to OpenStack. Let’s understand these term in more detail:

- **Mean time between failures** (MTBF): MTBF tells us the estimated time between two frequent failures within a process or a component, which can be repairable.
- **Mean time to failure** (MTTF): MTTF is the total estimated time of a system where repairing is not possible.
- **Mean time to repair or replace** (MTTR): MTTR is the average estimated time to repair a failed component or the total replacement time of a failed component.

The formula to calculate availability is this:

\[
\text{Availability} = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}}
\]

The following table represents the **Service Level Agreement (SLA)** for high availability between a consumer and a provider:

<table>
<thead>
<tr>
<th>The level of availability</th>
<th>Downtime/day</th>
<th>Downtime/month</th>
<th>Downtime/year</th>
</tr>
</thead>
<tbody>
<tr>
<td>one 9s(90%)</td>
<td>144.00 minutes</td>
<td>72 hours</td>
<td>36.5 days</td>
</tr>
<tr>
<td>two 9s(99%)</td>
<td>14.40 minutes</td>
<td>7 hours</td>
<td>3.65 days</td>
</tr>
<tr>
<td>three 9s(99.9%)</td>
<td>86.40 seconds</td>
<td>43 minutes</td>
<td>8.77 hours</td>
</tr>
<tr>
<td>four 9s(99.99%)</td>
<td>8.64 seconds</td>
<td>4 minutes</td>
<td>52.60 minutes</td>
</tr>
<tr>
<td>five 9s(99.999%)</td>
<td>0.86 seconds</td>
<td>26 seconds</td>
<td>5.26 minutes</td>
</tr>
</tbody>
</table>
The following are some commonly used terms that are used to understand and measure high availability:

- **Single point of failure (SPOF):** It reflects a part or a particular component of a system that brings the entire system to a halt; when it fails, the whole system will stop working. The consideration for a possibly single point of failure identifies the critical components of a complex system that would cause a total system failure in case of any breakdown.

- **Recovery time objective (RTO):** It is a very important matrix to measure business continuity. RTO determines the tolerance of a business process when a system is not available. For high user traffic e-commerce websites and mission critical businesses, RTO should be zero or near to zero.

- **Recovery point objective (RPO):** RPO analysis is a very critical metric for any business. It is calculated as a total amount of data loss when a system is not available. It is measured in terms of time.

- **Service level agreement (SLA):** SLA is a mutual agreement between consumers and a service provider that defines detailed service offerings, delivery time, the quality of service (QoS), and the scope or constraints of the offered services.

### Common content in the contract

Metrics related to performance assurance depend on the following components:

- RTO and RPO
- Uptime and downtime ratio
- System throughput
- Response time

### How to achieve high availability

High availability in a production environment is achieved by removing single point of failures, enhancing replication, redundancy, and fail-over capability of the system, and quickly detecting failures as they occur. High availability can be achieved at many different levels including the infrastructure level, data center level, geographic redundancy level, and even application level. The basic understanding of high availability in the infrastructure level includes the following things:

- Multiple web servers
- Multiple database servers
An Introduction to High Availability Concepts

- Multiple load balancers
- Leveraging different storage systems

Here, multiple web servers means multiple web nodes, multiple load balancers means active/passive load balancers, multiple database servers means replicated DB servers, and leveraging different storage system means the maximum utilization of all possible storage solutions with a backup plan that provides redundancy at each layer of the configuration.

For advanced HA configurations, we also need to plan about automatic failover and geo-replication, in case of disaster recovery, and also need to design our application for high availability. In short, we can say that there are various advanced techniques to achieve high availability, which we will discuss in detail in the following chapters, but the prime objective of each technique is to obtain the following five principals of high availability as described in this figure:

![High availability principles]

**Architecture design for high availability**

The following are some common design patterns used to design a high availability architecture. Let's discuss this in detail:

- **Design for failure consideration**: This must be the first and foremost concern of a cloud architect. Whenever any company or organization decides to move to a cloud infrastructure solution, it has to plan for failure. If the failure conditions are planned properly, then this will be of little or no consequence to the HA services or the resources, and the system will always be available.
• **Decouple your components**: All the components in a cloud infrastructure should be decoupled and isolated from each other; for example, whenever we have set upload balancing applications, traditionally web servers have been tightly coupled with the application server, but this is not the best practice. Isolation and decoupling are necessary between components. The following figure shows that each web server and application server is highly coupled:

![High coupling diagram](image)

Web servers and application servers can be loosely coupled and isolated by putting a load balancer between them, as shown in the following figure. Any web or application server can be easily scaled up or scaled down without any dependency.

![Decoupled applications diagram](image)
• **Build security in every layer:** While designing a cloud infrastructure, building security in each layer is recommended. As security is the shared responsibility of the consumer and the cloud provider, the following are some steps that a consumer must take to ensure security:

  - Enforce the principle of least privileges when developing applications
  - Encrypt data during transitions state
  - Try to use a multiway and multifactor authentication

• **Design parallel:** While designing a cloud infrastructure, a parallel architecture that is fast and efficient should be implemented. Two possible approaches in parallel architecture could be as follows:

  - A server works on a job sequentially for 4 h
  - Four servers work on a job in parallel for 4 h

There is no difference cost-wise, but the second approach is likely to be four times faster; therefore, a parallel design is highly recommended.

• **Automate and test everything:** When a production environment is enabled to handle cloud service failure, you should test your automated processes for the occurrence of hardware, software, and application failure. This kind of automation is provided by the cloud infrastructure service provider, which allows you to implement failover automated processes across instances, availability zones, regions, and other clouds. You should automate the data backups as well because in case of any outage or disaster, your data must be immediately ready.

Any disaster recovery plan can only be successful if it is tested properly to make sure it works. It can be tested by disabling your various cloud servers and associated services and transferring high loads to your production servers so that you can test the actual potential of your existing infrastructure. Significantly, cloud infrastructure solutions provide easily configurable disaster recovery and backup services nowadays. Despite highly popular services, organizations cannot effectively enable high availability services in the cloud until they implement a proper architecture and use the right management tools.
High availability in OpenStack

Many organizations go with the OpenStack cloud nowadays to provide an infrastructure as a service environment since OpenStack has excellent hybrid cloud features and can leverage public cloud scale out feature through APIs. Therefore, OpenStack is quite capable of handling the workload of mission critical applications.

OpenStack provides an easy-to-control dashboard to manage networking, storage, and computing resources and facilitates users to have a provision for computing resources. The following figure illustrates the architecture of OpenStack:

![OpenStack Architecture Diagram]

Organizations that run their production applications over OpenStack, generally try to achieve five 9s (99.999) of availability. Hence, the failure of any running controller node or service should not create any kind of disruption on any application running on the resources provided or managed by OpenStack.
In short, achieving high availability in OpenStack means removing all single points of failure, implementing redundancy, and replicating component failover capability and automation. Also, workload should be scaled out or scaled in according to real-time workload. We can achieve high availability in OpenStack in the following ways:

- Enabling multimaster database replication and building an efficient and reliable messaging cluster that forms the basis of any highly available OpenStack deployment
- Setting up open source load balancing software such as HAProxy and keepalived and load balancing of HTTP REST API's, MySQL, and AMQP clusters
- Enabling some classical clustering methods such as a pacemaker with two or more nodes to control active/passive OpenStack services
- Enabling OpenStack services in stateless mode to offer a scalable cloud framework with scaling and resiliency of all the basic OpenStack services: compute, image, storage, object storage, and dashboard
- Leveraging third-party networking drivers that offer high availability options
- Enabling distributed networking with Neutron Distributed Virtual Routers and configuring multiple L3 agents in active/passive configuration and third-party networking drivers that offer high availability options
- Leveraging software-defined storage such as GlusterFS, Ceph, traditional enterprise storage such as NFS, iSCSI for quick recovery after a failure, and backup services
- Enabling automatic failover and geo-replication using swift and effective recovery technique such as network partitioning split brain

**Summary**

In this chapter, we learned the fundamentals of high availability and what a highly available design is meant to achieve in a production environment.

We have also learned about SPOF, RTO, RPO, MTTR, MTTF, the concept of SLA, and the general architectural design of a highly available system with an overview of OpenStack high availability requirements and the various ways of achieving high availability in OpenStack.

In the next chapter, we will take a deep dive into database replication and how to build an efficient and reliable messaging cluster to enable high availability in OpenStack.
Where to buy this book

You can buy Learning OpenStack High Availability from the Packt Publishing website. Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers. Click here for ordering and shipping details.