

PROFESSIONAL TRAINING REPORT

AT

Sathyabama Institute of Science and Technology

(Deemed to be University)

Submitted in partial fulfillment of the requirements for the award

Of Bachelor of Technology Degree in Information Technology by

RAJALASHMI.R

REG NO : 40120072



DEPARTMENT OF INFORMATION TECHNOLOGY

SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade “A” by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

DECLARATION

I **RAJALASHMI R** hereby declare that the Project Report entitled "**BARBICAN IN OPENSTACK**" done by me under the guidance of **Dr.L.MARY GLADENCE** at Sathyabama Institute of Science and Technology (Deemed to be University) Jeippar Nagar, Rajiv Gandhi Salai, Chennai - 600119 is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology degree in Information Technology

DATE: 29.04.2023

PLACE: Chennai

R Rajalashmi

SIGNATURE OF THE CANDIDATE

**SATHYABAMA**

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the Bonafide work of **RAJALASHMI R (40120072)** who carried out the project entitled "PRIDE COURSE "CDAC" under my supervision from January 2023 to April 2023.

Internal Guide

(DR.L.MARY GLADENCE M.E.,Ph.D.,)

Head of the Department

(DR.R.SUBHASHINI M.E.,Ph.D.,)

Submitted for Viva voce Examination held on 29/04/2023

Internal Examiner

External Examiner

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. SASIKALA M.E., Ph.D, Dean**, School of Computing **Dr. R. Subhashini M.E.Ph.D.**, Head of the Department of Information Technology for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.L.Mary Gladence M.E.Ph.D.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Information Technology** who were helpful in many ways for the completion of the project

TRAINING CERTIFICATE

Ref.No.: 64 /FSPCC/BC001/04/2023



इलेक्ट्रॉनिक्स एवं
सूचना प्रौद्योगिकी मंत्रालय
MINISTRY OF
ELECTRONICS AND
INFORMATION TECHNOLOGY
सत्यमेव जयते



COURSE COMPLETION LETTER

It is certified that

Rajalashmi R

Sathyabama Institute of Science and Technology

has successfully undergone the Bridge Course Training Program on

Cloud Computing Technology

Under "FutureSkills PRIME" Programme

commenced from 11 Mar, 2023

V.S.M.

Shri. V. Sridharan
Chief Investigator
Cloud Computing Technology
Lead Resource Centre

L R PRAKASH Digitally signed by L R PRAKASH
Date: 2023.04.26 02:24:42
+05'30'

Cdr.(Retd.) L.R. Prakash
Senior Director
Centre Head
C-DAC, Chennai

TABLE OF CONTENT

CHAPTER	TITLE	PAGE NO
1	ABSTRACT	8
2	INTRODUCTION	9
3	BARBICAN IN OPENSTACK	10
4	AIM AND SCOPE	11
5	OPENSTACK 5.1. ADVANTAGES OF OPENSTACK 5.2. KEY FEATURES OF OPENSTACK 5.3 GOALS OF OPENSTACK	12
6	TOOLS OF OPENSTACK	23
7	USECASE OF OPENSTACK	31
8	INSTALL AND CONFIGURATION OF BARBICAN	32
9	KEY MANAGER SERVICE OVERVIEW	37
10	BARBICAN FOR DEVELOPERS	37

11	RESULT & DISSCUSION	39
12	CONCLUSION	47
13	REFERENCE	47

CHAPTER 1

ABSTRACT

In this study, we are explaining one of the key manager service BARBICAN . It provides secure storage, provisioning and management of secret data. This includes keying material such as Symmetric Keys, Asymmetric Keys. OpenStack is an open source platform that uses pooled virtual resources to build and manage private and public clouds. The tools that comprise the OpenStack platform, called "projects," handle the core cloud-computing services of compute, networking, storage, identity, and image services. OpenStack allows you to create both virtual machines & containers. And the benefits of virtual machines are countless! From lowering hardware costs and saving time to customizability, flexibility, and easier & quicker provisioning. OpenStack can also help you create containers, which are an alternative to VMs. Barbican is the OpenStack Key Manager service. It provides secure storage, provisioning and management of secret data. This includes keying material such as Symmetric Keys, Asymmetric Keys, Certificates and raw binary data. Barbican is an OpenStack key management service offering secure storage, provisioning, and management of key data. The barbican service provides management of secrets, keys and certificates via multiple storage back-ends. Barbican is a ReST API designed for the secure storage

CHAPTER 2

INTRODUCTION

OpenStack does not assume the existence of an operating data center; it provides the tools to operate a data center and make its resources available to consumers. There is no required layer underneath OpenStack as a whole. It provides the abstractions needed to deal with external systems like compute, storage, and networking hardware, the Domain Name System, and identity management systems. The OpenStack APIs provide a consistent interface to these systems, which may each potentially be implemented by a variety of vendors and Open Source projects.

This broad base provides an abstraction that can host more specialised services - both those within OpenStack itself and those from third-parties.

Barbican is an OpenStack key management service offering secure storage, provisioning, and management of key data. The barbican service provides management of secrets, keys and certificates via multiple storage back-ends.

Barbican is a ReST API designed for the secure storage

CHAPTER 3

BARBICAN IN OPENSTACK

In simple terms, OpenStack Barbican is an open source Key Management service that provides safe storage, provisioning, and management of sensitive or secret data. This data could be of various content types anything from symmetric or asymmetric keys or certificates or raw binary data.

With Barbican, users can secure their data seamlessly and maintain its accessibility without personally managing their keys. Barbican also addresses concerns about privacy or misuse of data among users.

As mentioned above, users can utilize OpenStack Barbican for safe and secure storage, provisioning, and management of sensitive data. Barbican boasts of a plug-in based architecture that allows users to store their data in multiple secret stores. These stores can be software-based, like a software token, or hardware-device-based, like a Hardware Security Module (HSM). The tool allows users to securely store and manage anything from passwords to encryption keys to X.509 certificates.

Another advantage of OpenStack Barbican is its ability to integrate seamlessly with all other enterprise-grade cloud services from OpenStack. With a simplified integration with block storage, the key management service stores keys for encrypted volumes. With object storage, Barbican enables encryption of data at rest. Similarly, seamless integration happens with Keystone, providing identity authentication and complete role-based access control. With OpenStack's image storage, the integration allows users to verify signed images, to check whether an uploaded image is altered or not.

CHAPTER 4

AIM AND SCOPE

AIM

Aim of this project / study is to securely store data using barbican manager in open stack. As, data security and encryption are always priorities for cloud users.

SCOPE

OpenStack is an open source software platform for private clouds and public clouds. As an Infrastructure as a Service (IaaS) platform, OpenStack enables companies to easily and efficiently add servers, storage and networking components to their cloud

CHAPTER 5

OPENSTACK

OpenStack is a cloud OS that is used to control the large pools of computing, storage, and networking resources within a data center. OpenStack is an open-source and free software platform. This is essentially used and implemented as an IaaS for cloud computing.

We can call the OpenStack a software platform that uses pooled virtual resources to create and manage private and public cloud. OpenStack offers many cloud-related services (such as networking, storage, image services, identity, etc.) by default. This can be handled by users through a web-based dashboard, a RESTful API, or command-line tools. OpenStack manages a lot of virtual machines; this permits the usage of physical resources to be reduced.

Basic Principles of OpenStack

Open Source: Under the Apache 2.0 license, OpenStack is coded and published. Apache allows the community to use it for free.

Open Design: For the forthcoming update, the development group holds a Design Summit every 6 months.

Open Development: The developers maintain a source code repository that is freely accessible through projects like the Ubuntu Linux distribution via entig100s.

Open Community: OpenStack allows open and transparent documentation for the community.

5.1. ADVANTAGES OF OPENSTACK

OpenStack cloud solutions provide several benefits for businesses that have or are seeking to virtualize elements of their infrastructure or services. Some of these include:

1. Improving business agility
2. Increasing the efficiency of core processes
3. Enhancing the availability of resources
4. Adding flexibility and adaptability to your infrastructure

Instead of relying on these services, you can use open-source tools to create your own cloud within your organization's data center. These capabilities make the following benefits possible:

Shorter time-to-market. With OpenStack, you do not have to wait for the networking services you need. OpenStack puts them all at your fingertips.

Faster innovation. With OpenStack, you have the power to orchestrate and take advantage of self-service tools as you wish. Using this flexibility, you can adjust your workflow to what works best for your team—and custom-design it to make better solutions faster.

Comply with regulations easier. OpenStack gives you the power to make your own private clouds within your data center. This makes it easy to control your access privileges, ensuring that only the eyes that should see assets are able to. When you have granular control over your network policies, you do not have to worry about whether sensitive data ended up being errantly categorized or you have run afoul of another aspect of a regulation.

Need not commit to a single vendor. Because you are in control of provisioning your own cloud resources, there is no need to commit to a vendor for the long term. If you do not like one, you can just move on to the next option.

5.2. KEY FEATURES OF OPENSTACK

Compute (Nova): Compute is a controller that is used to manage resources in virtualized environments. It handles several virtual machines and other instances that perform computing tasks.

Object Storage (Swift): To store and retrieve arbitrary data in the cloud, object storage is used. In Swift, it is possible to store the files, objects, backups, images, videos, virtual machines, and other unstructured data. Developers may use a special identifier for referring the file and objects in place of the path, which directly points to a file and allows the OpenStack to manage where to store the files.

Block Storage (Cinder): This works in the traditional way of attaching and detaching an external hard drive to the OS for its local use. Cinder manages to add, remove, create new disk space in the server. This component provides the virtual storage for the virtual machines in the system.

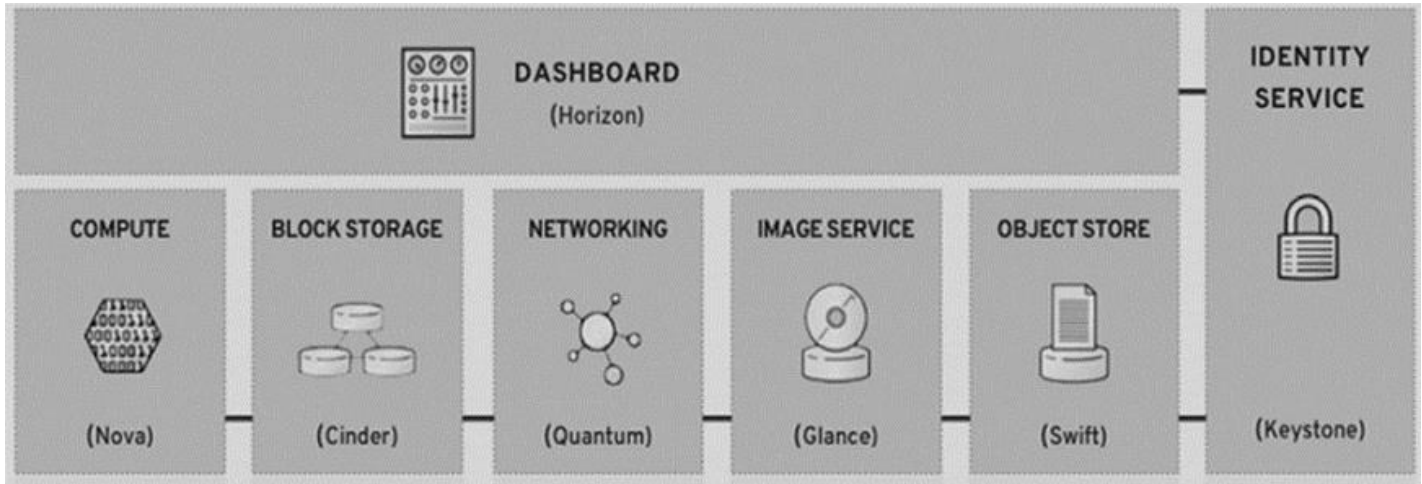
Networking (Neutron): This component is used for networking in OpenStack. Neutron manages all the network-related queries, such as IP address management, routers, subnets, firewalls, VPNs, etc. It confirms that all the other components are well connected with the OpenStack.

Dashboard (Horizon): This is the first component that the user sees in the OpenStack. Horizon is the web UI (user interface) component used to access the other back-end services. Through individual API (Application programming interface), developers can access the OpenStack's components, but through the dashboard, system administrators can look at what is going on in the cloud and manage it as per their need.

Identity Service (Keystone): It is the central repository of all the users and their permissions for the OpenStack services they use. This component is used to manage identity services like authorization, authentication, AWS Styles (Amazon Web Services) logins, token-based systems, and checking the other credentials (username & password).

Image Service (Glance): The glance component is used to provide the image services to OpenStack. Here, image service means the images or virtual copies of hard disks.

When we plan to deploy a new virtual machine instance, then glance allows us to use these images as templates. Glance allows virtual box (VDI), VMware (VMDK, OVF), Raw, Hyper-V (VHD) and KVM (qcow2) virtual images.



Telemetry (Ceilometer): It is used to meter the usage and report it to OpenStack's individual users. So basically, Telemetry provides billing services to OpenStack's individual users.

Orchestration (Heat): It allows the developers to store the cloud application's necessities as a file so that all-important resources are available in handy. This component organizes many complex applications of the cloud through the templates, via both the local OpenStack REST API and Query API.

Shared File System (Manila): It offers storage of the file to a virtual machine. This component gives an infrastructure for managing and provisioning file shares.

Elastic Map-reduce (Sahara): The Sahara component offers a simple method to the users to preplanned Hadoop clusters by referring to the multiple options such as the Hadoop version, cluster topology and hardware details of nodes and some more.

5.3 GOALS OF OPENSTACK

We use these OpenStack-wide goals to achieve visible common changes, push for basic levels of consistency and user experience, and efficiently improve certain areas where technical debt payments have become too high – across all OpenStack projects. From a contributor perspective, the goals help unify the community's work as a whole, with some shared objectives. We know this work is worthwhile, because these goals are how we achieve the OpenStack mission together. We will build these goals based on our shared values.

We know this process is working when:

- Users and operators see OpenStack working well as a platform, rather than a collection of loosely-related, different projects.
- We make progress on solving long-standing issues that would otherwise have lingered indefinitely at different stages of implementation in various projects.
- All contributors and projects know they are valued members of the successful and thriving OpenStack community.
- We facilitate marketing communication around release time due to successful completion of community-wide goals during the cycle.

Process Details:

Identifying Goals:

The goal process enables the community of OpenStack projects to surface common concerns and work out specific technical strategies for addressing these concerns.

The first step in the process is to build a backlog of potential goals. This helps us coalesce feedback received from deployers, users, contributors, and PTLs. It is the reference used as a base for goal discussion during Forum events. Finally, it serves as inspiration for prospective goal champions.

Defining Goals:

Goals are defined here in the governance documentation to ensure that we establish a common understanding of the expectations being set.

Once champions have volunteered to propose and drive a specific goal, they should iterate through goal definition in the **/goals/proposed/** directory. This allows to keep the goal selection process separate from the goal definition process. Goal proposals should have the **goal-proposal** gerrit topic, whose voting rules are specified in the house rules.

Goal definitions should use the provided template so they are all formatted consistently. The goal definition does not take the place of any blueprints, spec documents, or other planning tools used within a project to track its work, but can be referenced from those documents.

This separates the discussion for each goal, and allows authors to gradually refine and improve their proposal through multiple incremental changes. Goals should be discussed on the mailing list to collect feedback on their feasibility, and consensus on whether they have been completely and clearly described. Below checklist will help to know if goal is ready to be selected or need more pre-work.

Goal Checklist

Below checklist provide a high-level view for goal readiness and the same is mandatory to fill while proposing the goal.

Item	Requirements	Catagory
Design	Is design finalized?	Mandatory
Implementation	Are implementation details finalized?	Mandatory
Dependency & Blocker	Is there any dependency or blocker in OpenStack or External software?	Mandatory
Current state	Any project completed this goal?	Optional

Item	Requirements	Catagory
Meeting discussion	Is there any regular meeting or discussion planned where projects can ask questions and get help?	Optional

Selecting goals:

The TC will consider proposed goals from the **/goals/proposed/** directory and select a set of OpenStack-wide goals to allow planning and other discussion at the PTG event at the start of each cycle.

OpenStack-wide goals are not tied to any release, and they can have target dates which can be a particular cycle release date or multi-cycle or even in between a release cycle. Goal completion target dates will be selected by considering the amount of work needed and the community (project and champions) bandwidth. If a goal requires a large amount of work, it can be divided into different milestones, with each milestone targetting a set of things to complete. For example, if goal 'A' needs two big things to do, it can be divided into two milestones, say Milestone 1: Yoga cycle release, Milestone 2: Z cycle release.

There is no minimum or maximum limit on the number of goals to be selected at a time. It can be zero or more depends on the complexity and work required per goal. For example, if three groups of contributors would like to start three different goals and projects are also ok with that, then it is fine to select all three goals at a time and implement those in parallel. Also, it is completely fine to have no goal for a particular time with a valid reason, for example, if any previous OpenStack-wide goal is still pending and needs more time to complete, or project team has less bandwidth and working on other priority items like popup team work etc.

To select a goal, a TC member can pick one or more goals from the **/goals/proposed/** directory and move them to the **/goals/selected/** directory.

This allows to consider the proposed series goals as a group, and take into account how feasible they are together, considering the timing and amount of work required. Champions for selected goals should +1 this patch to confirm they are ready to work on their goal.

Tracking Goal Progress:

After goals are approved, the goal champions can create one story per goal and one task per project per goal or setup an etherpad to track progress on completion.

Note

It is expected that some projects may not always be able to complete a goal within the targeted cycle. If this is the case, existing projects are encouraged to continue to track their progress once they are able to further work on the goals.

New projects do not need to retroactively track past goals as part of the created stories or in pre-StoryBoard documentation, whether the goals were completed before or after becoming an official project.

Team Acknowledgment of Goals:

Each PTL is responsible for updating the progress on their tracking item (etherpad, story, or blueprint). A status indicating progress is being made should include a link to a planning artifact (such as a spec). The planning artifact is likely to be a link to a spec or bug, and the completion artifact is likely to be a link to one or more committed patches.

This step is also the indication that a project team is signing up as agreeing to the goal and committing to do the work to complete the goal, within their project. That commitment may mean doing it themselves or may mean simply prioritizing reviews submitted by others. Either way, the commitment is to have the goal completed, and all teams are expected to commit the necessary resources to ensure that it is finished.

All project teams are expected to prioritize these goals above other work.

If the project has already met the goal, skip to the “Completing Goals” step.

If the goal does not apply to a project, the PTL should explain why that is the case, instead of linking to planning artifacts.

Completing Goals:

After a goal is completed, each PTL is responsible for updating the storyboard task or etherpad used to track the goal progress for their project to set the status to “Merged” and to add links to all completion artifacts, before the final deadline of the goal.

If a goal is not completed, that information should be added, along with any references needed to track when that work will be completed.

Champion or TC member needs to move the completed goal from **/goals/selected/** directory to **/goals/completed/** directory.

Goal Champion Responsibilities:

The “goal champion” is responsible for guiding teams to implement a goal, but is not responsible for doing the work themselves.

The champion is typically somewhat familiar with the technical issues involved in a goal, at least enough to help teams find the information they need. The champion does not need to be an expert with the tools or libraries used for the goal, but some experience does help.

The champion is responsible for configuring StoryBoard to track progress on their goal, and then ensuring that it is kept up to date as work progresses.

The champion should check in with project teams regularly to ensure they are able to make progress on the work and have not become stuck.

At the end of the release, there may be open tasks and these can remain open in case someone comes along later and completes the task. As goal champion you are not responsible for completing any incomplete tasks for which you are not assigned, i.e. the project assigned to the task is responsible for completing it. It is also encouraged to send a retrospective email to the openstack-discuss mailing list with a summary of the goal including things such as how many projects completed the goal, reasons behind some projects that did not complete the goal, anything notable that came up during the goal implementation phase, and next steps if there are any.

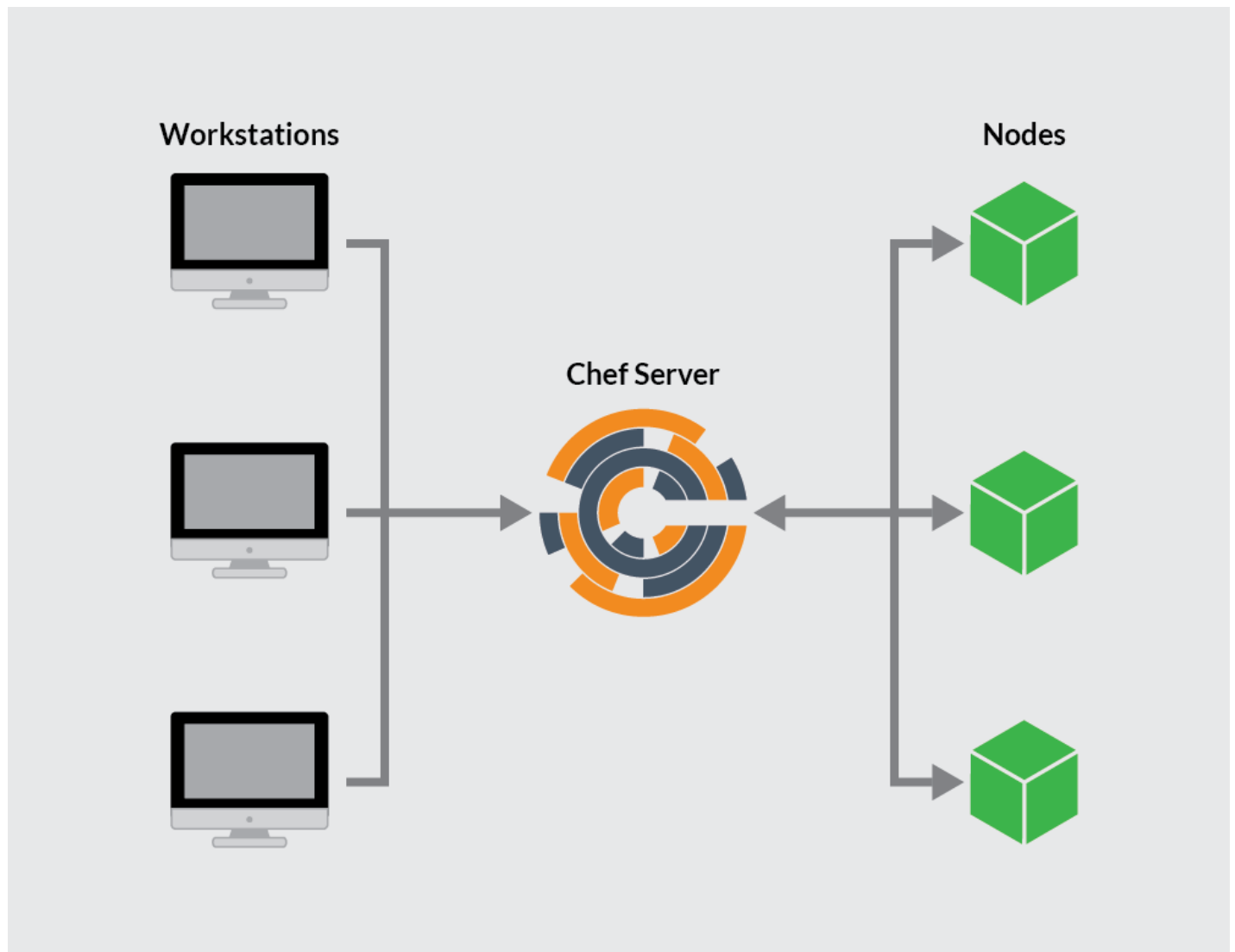
CHAPTER 6

TOOLS OF OPENSTACK

Many open-source tools are available to install, manage, and run the OpenStack cloud. The most important question is how to choose the best one.

Chef

Chef is a great automation framework that makes it simple to set up servers and applications to any real, virtual, or cloud location, no matter the extent of the infrastructure. Chef is a product simply focussed on its developer user base. Chef includes cookbooks for engaging with different preferences of OpenStack.



It makes use of Ruby as a programming language and consists of a searchable portal where one can get community-contributed techniques and cookbooks. Chef automates the handling of managing configurations, making sure that systems are configured properly and consistently.

Chef has an agent-based architecture that involves a client on each VM or server instance while being led by a central master agent. Chef also enjoys broad support in the form of a wealth of cookbooks and documentation. Some of its features include:

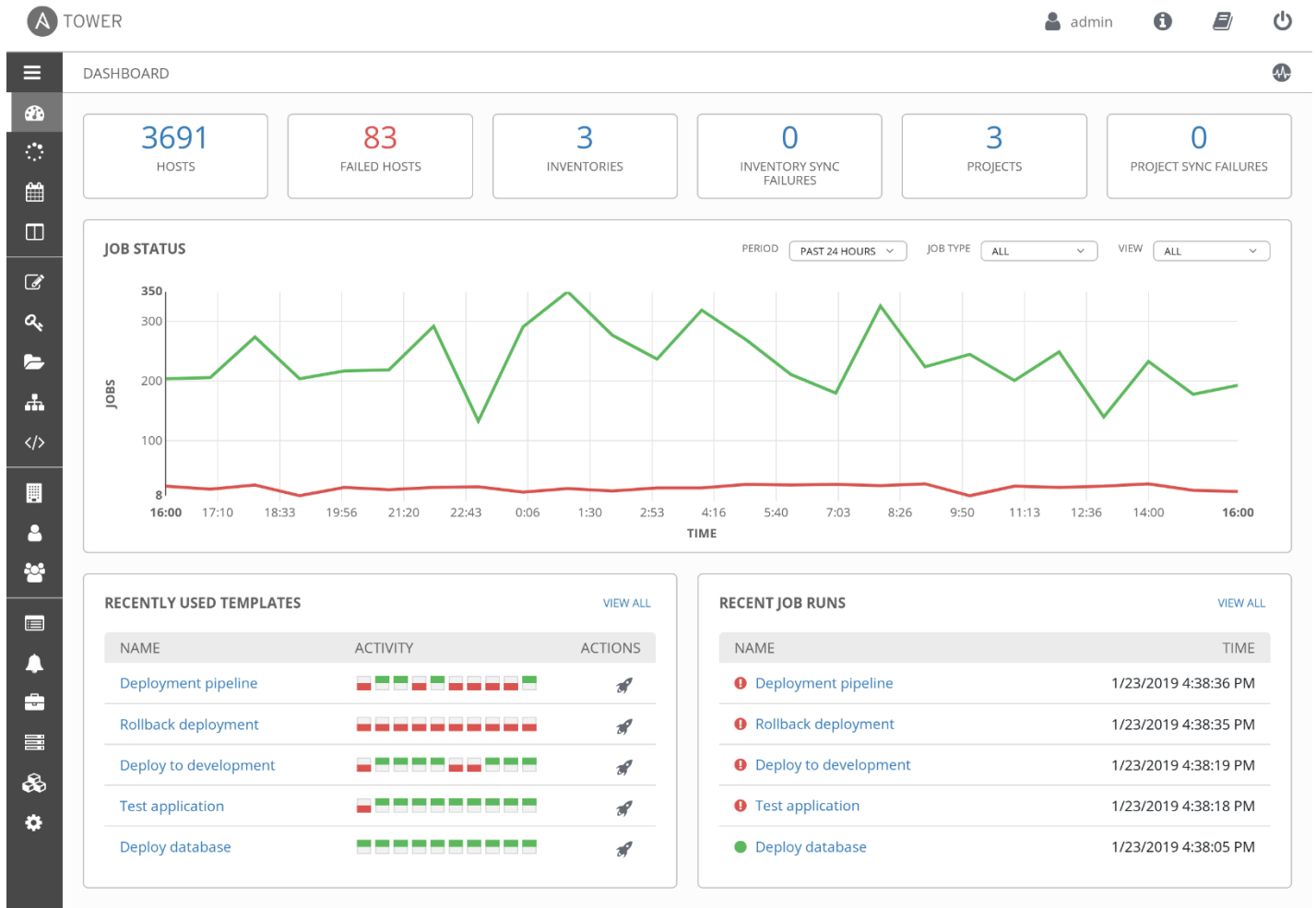
- Use simple declarative definitions for simple administrative tasks.
- Configure a variety of cloud-based SaaS services, and integrate cloud provisioning APIs
- It presents a single process to manage on-premises and cloud estates.
- makes the infrastructure highly flexible, versionable and accurately testable
- Ensures consistent configurations across the development lifecycle
- Automatically correct configuration drift
- It comprises of the Chef Development Kit, which has robust testing tools for validating infrastructure changes

This IT automation and configuration tool is well appropriated to development centric Infrastructure projects. It provides configuration management and much infrastructure management tasks using Recipes.

Ansible

Ansible is an infrastructure automation tool, which gives support to configure systems, deploy software, and set up more advanced IT-intensive tasks like consistent deployments and zero downtime while installing updates. Ansible OpenStack modules can be utilized to manage all sorts of cloud operations.

Ansible goes beyond more than simple deployment.



Ansible's main goals are simplicity and ease-of-use. It further has a substantial focus on security and reliability. Ansible provides powerful tools for setting up and managing OpenStack — to provision, configure and deploy applications and works on top of the cloud.

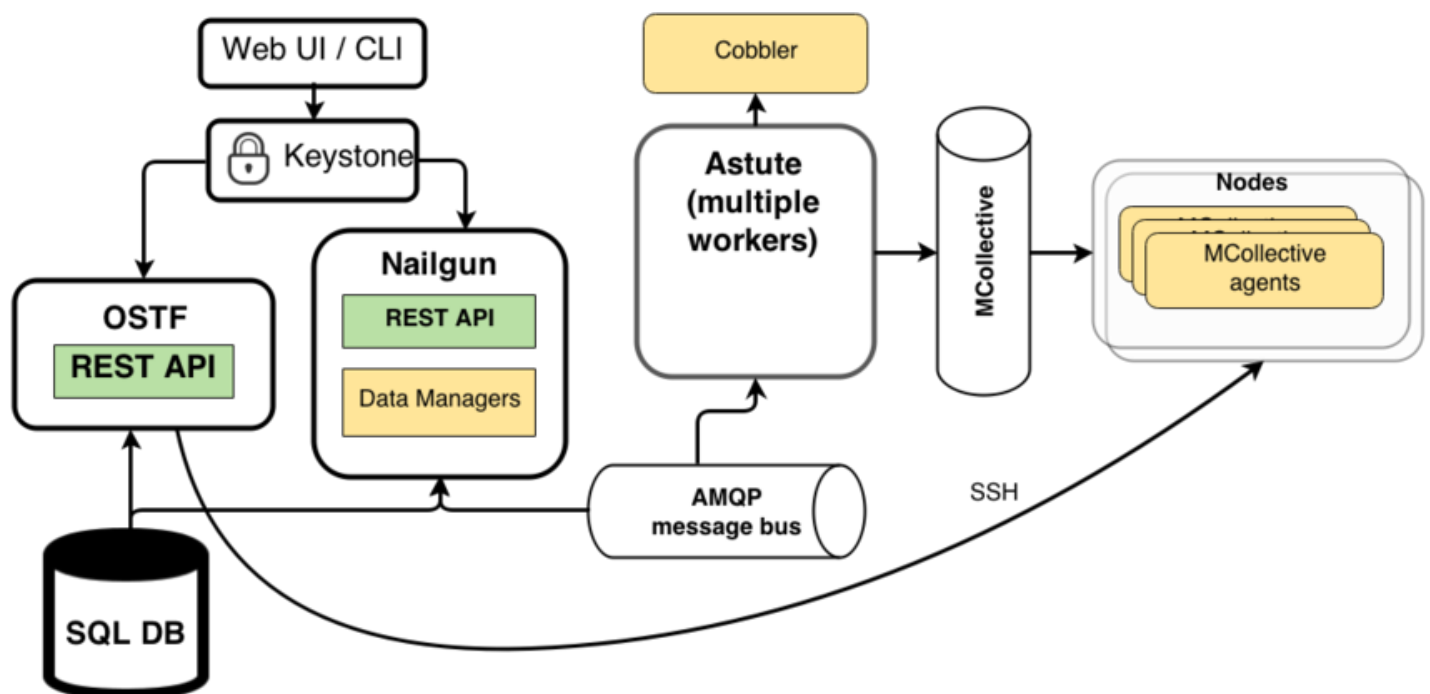
It is based on a simple, agentless architecture, so there is no need to configure VMs or workstations before installation. Ansible can normally work with them through the command line. Some of its features include:

- It provides simple solutions for all configuration management issues.
- Allows reliable and repeatable IT infrastructure configuration
- It is designed to easy-to-learn for administrators and developers.
- Provides systems administrators a control language that uses modules or routines to produce all sorts of tasks on nodesMost secure remote configuration management system

- It consists of 1300+ modules with an active community for support and development.

Fuel

Fuel is a GUI based tool for systems administrators to set up and manage OpenStack Cloud. Fuel is an upstream OpenStack project that concentrates on automating the deployment and testing of OpenStack and a range of third-party options.



Fuel automatically detects all virtual nodes connected from the grid. It consists of several separate components. Some components can also be reused separately from the Fuel without any modifications, and some involve little tweaks. The following are some of its features.

- It performs post-deployment checks and also performs tests for validating the OpenStack Cloud deployed.
- It can spin up and manage multiple OpenStack clusters
- It includes support for CentOS and Ubuntu
- It helps the sysadmin to track real-time logs via the GUI of OpenStack Cloud.

Puppet

Puppet is a Declarative Language for 'write once deploy many' packages for on-demand Openstack configuration and version management. Puppet is deployed in a client/server setup or in the serverless mode, where clients periodically poll the server for the needed state and deliver back status records to the server.

Events

Get insight into the failures and changes across your Puppet runs. Data refreshed: a few seconds ago.

Show:

Summary Classes with corrective changes

Classes with events

Class	Count
All	25
Failures	5
Corrective changes	18
Intentional changes	2
Corrective no-ops	0
Intentional no-ops	0
Skips	0

Nodes with events

Node	Count
All	13
Failures	7
Corrective changes	3
Intentional changes	4
Corrective no-ops	0
Intentional no-ops	0
Skips	0

Resources with events

Resource	Count
All	6
Failures	1
Corrective changes	2
Intentional changes	3
Corrective no-ops	0
Intentional no-ops	0
Skips	0

Filter by class name: Apply

Classes with corrective changes: 18

Table shows nodes affected per class. [Export data](#)

Class	Failures	Corrective changes	Intentional changes	Corrective no-ops	Intentional no-ops	Skips
apache	-	2	2	-	-	-
apache::vhost	-	4	-	-	-	-
app	-	1	1	-	-	-
base::linux	2	7	-	-	-	-
base::unix	-	1	-	-	-	-
chocolatey	-	1	-	-	-	-
chocolatey::install.pp	-	2	-	-	-	-
chocolatey::config.pp	-	1	-	-	-	-
docker	2	4	1	-	-	-
java	-	3	-	-	-	-
java::config	-	1	-	-	-	-
java::params	1	1	-	-	-	-
postgresql::globals	-	1	1	-	-	-

Puppet can plan, upgrade, and manage nodes throughout their life cycle. All modules and configurations are produced with a Puppet-specific language based on Ruby or Ruby itself, and therefore, will require programmatic expertise in extension to system administration skills. It involves a Web UI and reporting tools.

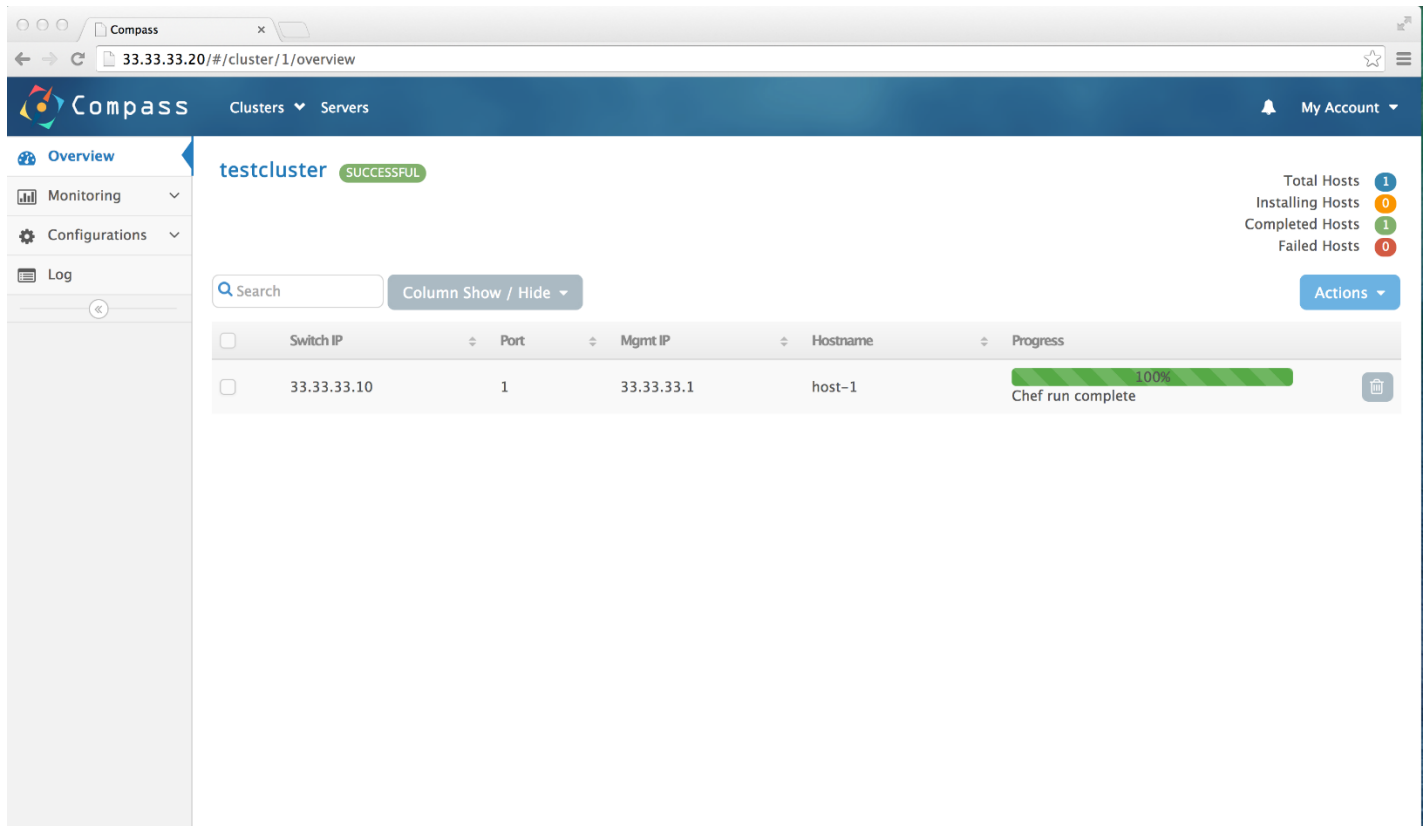
Puppet Enterprise allows for real-time management of nodes using prebuilt modules present on the master servers. The reporting tools are fully developed, giving in-depth details on how agents are performing and what changes have been made. Some of its features include:

- It supports full-scale automation with specific reporting and compliance factors.
- It has integrations and modules from the Forge
- Makes patching and updating stress-free.
- Creates hardened baseline configurations to meet regulatory standards
- Automate and enforce patch management and reduce potential vulnerabilities.

Compass

Compass is for the automated deployment and management of OpenStack.

It reduces complexity, manages time, and controls all sorts of blunders in the data center server management. It supports in bootstrapping the server pool correlated with any cloud platform from exposed metal nodes.



Compass assists systems administrators in determining hardware, deploying the OS and hypervisor, and also gives comprehensive configuration management. Some more features as below.

- Allows implementation of different configuration flavors through metadata.
- Assists in the infrastructure bootstrapping prepare and also provides programmability for operators to do this.
- Implements extensibility through the assimilation of several tools OpenStack cluster configuration.
- The Compass core blends with many other tools for resource discovery, OS planning, and package deployment.

CHAPTER 7

USECASE OF OPENSTACK

OpenStack Use Case: Telecom

Suppose a telecom is using OpenStack to provide communication services to end-users. What is OpenStack cloud in the context of a telecom company? Let's say the telecom starts out with a relatively small customer base, servicing around 5,000 total users. Every time these people connect with each other, their messages and calls have to be sent through the telecom's network.

Word starts spreading about the great service the telecom is providing, and more users start signing up. With a traditional platform, the telecom company may have to heavily invest in new equipment and staff to make sure their network can handle the new load. But because they are using the cloud, they can gain access to what they need as they need it.

OpenStack forms the bridge between the telecom's needs and the resources that help them meet them. They can add additional virtual processors (vCPUs), for example, to manage the computational load it takes to add new users as they sign on. OpenStack gives them the ability to provision these anywhere in their network, as well as see how they are functioning.

CHAPTER 8

INSTALL AND CONFIGURATION OF BARBICAN

Install and configure for Ubuntu

This section describes how to install and configure the Key Manager service for Ubuntu 14.04 (LTS).

Prerequisites:

Before you install and configure the Key Manager service, you must create a database, service credentials, and API endpoints.

1.To create the database, complete these steps:

Use the database access client to connect to the database server as the **root** user:

- # mysql

- Create the **barbican** database:

- CREATE DATABASE barbican;

- Grant proper access to the **barbican** database:

- GRANT ALL PRIVILEGES ON barbican.* TO 'barbican'@'localhost' \
- IDENTIFIED BY 'BARBICAN_DBPASS';
- GRANT ALL PRIVILEGES ON barbican.* TO 'barbican'@'%' \
- IDENTIFIED BY 'BARBICAN_DBPASS';

Replace **BARBICAN_DBPASS** with a suitable password.

- Exit the database access client.

- `exit;`

2. Source the **admin** credentials to gain access to admin-only CLI commands:

3. `$ source admin-openrc`

4. To create the service credentials, complete these steps:

- Create the **barbican** user:

- `$ openstack user create --domain default --password-prompt barbican`

- Add the **admin** role to the **barbican** user:

- `$ openstack role add --project service --user barbican admin`

- Create the **creator** role:

- `$ openstack role create creator`

- Add the **creator** role to the **barbican** user:

- `$ openstack role add --project service --user barbican creator`

- Create the barbican service entities:

- `$ openstack service create --name barbican --description "Key Manager" key-manager`

5. Create the Key Manager service API endpoints:

```
5. $ openstack endpoint create --region RegionOne \  
6.   key-manager public http://controller:9311 \  
7. $ openstack endpoint create --region RegionOne \  
8.   key-manager internal http://controller:9311 \  
9. $ openstack endpoint create --region RegionOne \  
10.  key-manager admin http://controller:9311
```

Install and configure components:

1. Install the packages:

```
2. # apt-get update \  
3. \  
4. # apt-get install barbican-api barbican-keystone-listener barbican-worker
```

2. Edit the **/etc/barbican/barbican.conf** file and complete the following actions:

- In the **[DEFAULT]** section, configure database access:

```
◦ [DEFAULT] \  
◦ ... \  
◦ sql_connection = mysql+pymysql://barbican:BARBICAN_DBPASS@controller/barbican
```

Replace **BARBICAN_DBPASS** with the password you chose for the Key Manager service database.

- In the **[DEFAULT]** section, configure **RabbitMQ** message queue access:

- **[DEFAULT]**
- ...
- `transport_url = rabbit://openstack:RABBIT_PASS@controller`

Replace **RABBIT_PASS** with the password you chose for the **openstack** account in **RabbitMQ**.

- In the **[keystone_authtoken]** section, configure Identity service access:

- **[keystone_authtoken]**
- ...
- `www_authenticate_uri = http://controller:5000`
- `auth_url = http://controller:5000`
- `memcached_servers = controller:11211`
- `auth_type = password`
- `project_domain_name = default`
- `user_domain_name = default`
- `project_name = service`
- `username = barbican`
- `password = BARBICAN_PASS`

Replace **BARBICAN_PASS** with the password you chose for the **barbican** user in the Identity service.



Populate the Key Manager service database:

If you wish the Key Manager service to automatically populate the database when the service is first started, set `db_auto_create` to `True` in the [DEFAULT] section. By default this will not be active and you can populate the database manually as below:

```
$ su -s /bin/sh -c "barbican-manage db upgrade" barbican
```

3. Barbican has a plugin architecture which allows the deployer to store secrets in a number of different back-end secret stores. By default, Barbican is configured to store secrets in a basic file-based keystore. This key store is NOT safe for production use.

For a list of supported plugins and detailed instructions on how to configure them, see [Configure Secret Store Back-end](#)

Finalize installation:

Restart the Key Manager services:

```
# service barbican-keystone-listener restart
# service barbican-worker restart
# service apache2 restart
```

CHAPTER 9

KEY MANAGER SERVICE OVERVIEW

The Key Manager service provides secure storage, provisioning and management of secrets, such as passwords, encryption keys, etc.

The Key Manager service consists of the following components:

Barbican-api service

Provides an OpenStack-native RESTful API that supports provisioning and managing Barbican secrets.

Barbican-worker service

Provides an Openstack RPC interface that interacts with barbican-api and reads from the barbican message queue. Supports the fulfillment of Barbican orders.

Barbican-keystone-listener service

Listens to messages from the Keystone notification service. Used to manage the representation of Keystone projects in the Barbican database when projects are deleted.

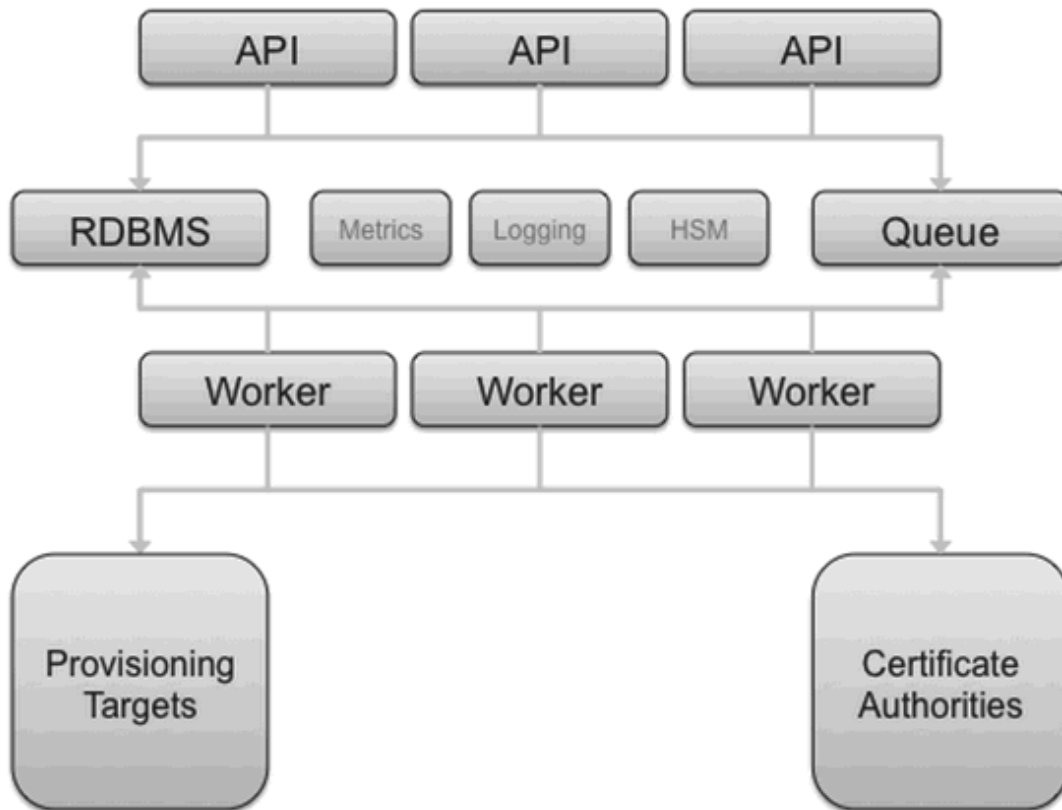
CHAPTER 10

BARBICAN FOR DEVELOPERS

It describes the architecture and technology selections for Barbican. In general, a goal is to utilize the OpenStack architecture and technology selections as much as possible. An overall architecture is presented first, followed by technology selection details to implement the system.

Overall Architecture

The next figure presents an overall logical diagram for Barbican.



CHAPTER 11

USING KEYSTONE MIDDLEWARE WITH BARBICAN

To enable Keystone integration with Barbican you'll need a relatively current version of Keystone. It is sufficient if you are installing an OpenStack cloud where all services including Keystone and Barbican are from the same release. If you don't have an

instance of Keystone available, you can use one of the following ways to setup your own.

1. Simple Dockerized Keystone
2. Installing Keystone
3. An OpenStack cloud with Keystone (Devstack in the simplest case)

Hooking up Barbican to Keystone:

Assuming that you've already setup your Keystone instance, connecting Barbican to Keystone is quite simple. When completed, Barbican should require a valid X-Auth-Token to be provided with all API calls except the get version call.

1. Turn off any active instances of Barbican
2. Edit **/etc/barbican/barbican-api-paste.ini**
 - a. Change the pipeline **/v1** value from unauthenticated **barbican_api** to the authenticated **barbican-api-keystone**. This step will not be necessary on barbican from OpenStack Newton or higher, since barbican will default to using Keystone authentication as of OpenStack Newton.

- b. **[composite:main]**
- c. `use = egg:Paste#urlmap`
- d. `/: barbican_version`
- e. `/v1: barbican-api-keystone`

- f. Replace **authtoken** filter values to match your Keystone setup

- g. **[filter:authtoken]**
- h. `paste.filter_factory = keystonemiddleware.auth_token:filter_factory`

```
i. auth_plugin = password
j. username = {YOUR_KEYSTONE_USERNAME}
k. password = {YOUR_KEYSTONE_PASSWORD}
l. user_domain_id = {YOUR_KEYSTONE_USER_DOMAIN}
m. project_name = {YOUR_KEYSTONE_PROJECT}
n. project_domain_id = {YOUR_KEYSTONE_PROJECT_DOMAIN}
o. www_authenticate_uri = http://{YOUR_KEYSTONE_ENDPOINT}:5000/v3
p. auth_url = http://{YOUR_KEYSTONE_ENDPOINT}:5000/v3
```

Alternatively, you can shorten this to

```
[filter:authtoken]
```

```
paste.filter_factory = keystonemiddleware.auth_token:filter_factory
```

and store Barbican's Keystone credentials in the **[keystone_authtoken]** section of **/etc/barbican/barbican.conf**

```
[keystone_authtoken]
```

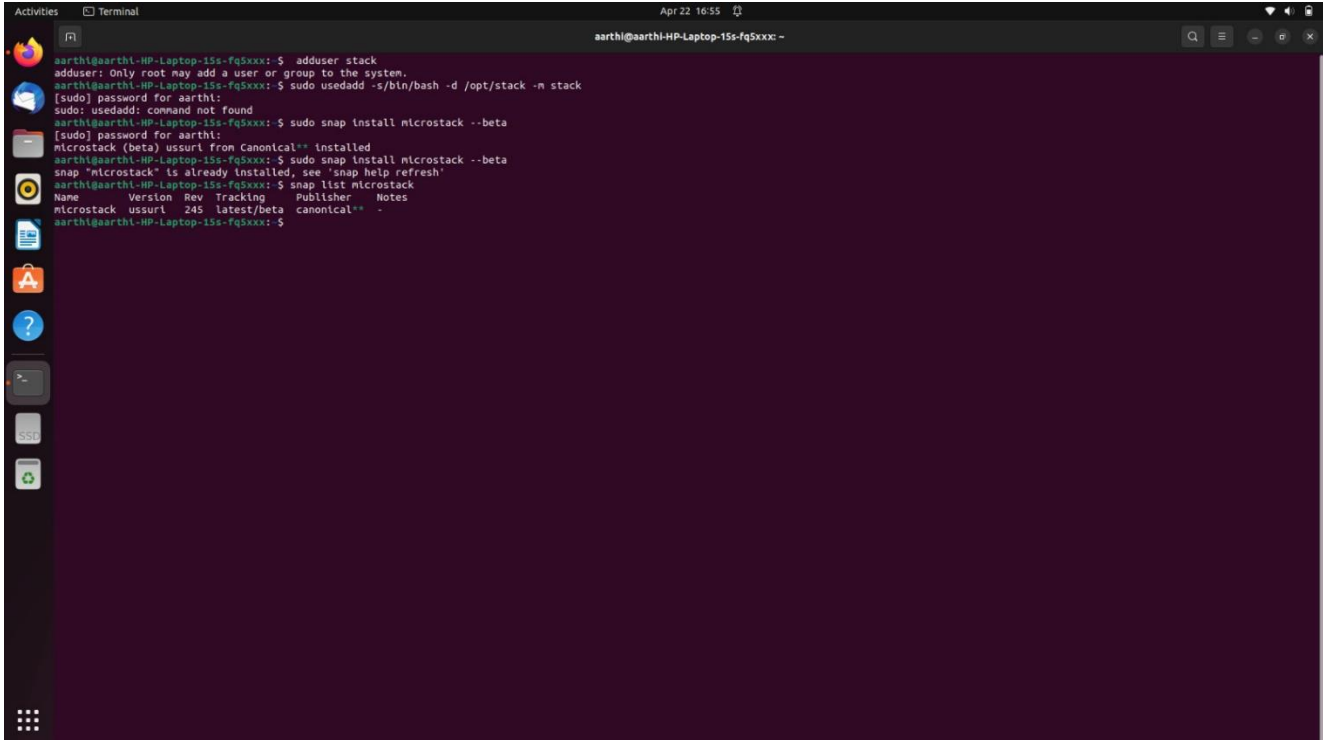
```
auth_plugin = password
username = {YOUR_KEYSTONE_USERNAME}
password = {YOUR_KEYSTONE_PASSWORD}
user_domain_id = {YOUR_KEYSTONE_USER_DOMAIN}
project_name = {YOUR_KEYSTONE_PROJECT}
project_domain_id = {YOUR_KEYSTONE_PROJECT_DOMAIN}
www_authenticate_uri = http://{YOUR_KEYSTONE_ENDPOINT}:5000/v3
auth_url = http://{YOUR_KEYSTONE_ENDPOINT}:5000/v3
```

3. Start Barbican **{barbican_home}/bin/barbican.sh start**

CHAPTER 12

RESULT & DISCUSSION

As a result, we installed the openstack and the components of openstack are enabled in our ubuntu OS. The barbican local.config file and plugin file is installed

A terminal window titled 'Terminal' with the address bar 'aarthi@aarthi-HP-Laptop-15s-fq5xxx: ~'. The terminal shows the following commands and output:

```
aarthi@aarthi-HP-Laptop-15s-fq5xxx: $ adduser stack
adduser: Only root may add a user or group to the system.
aarthi@aarthi-HP-Laptop-15s-fq5xxx: $ sudo usedadd -s/bin/bash -d /opt/stack -m stack
[sudo] password for aarthi:
sudo: usedadd: command not found
aarthi@aarthi-HP-Laptop-15s-fq5xxx: $ sudo snap install microstack --beta
[sudo] password for aarthi:
microstack (beta) ussurl from Canonical** installed
aarthi@aarthi-HP-Laptop-15s-fq5xxx: $ sudo snap install microstack --beta
snap 'microstack' is already installed, see 'snap help refresh'
aarthi@aarthi-HP-Laptop-15s-fq5xxx: $ snap list microstack
Name      Version Rev Tracking Publisher Notes
microstack ussurl  245  latest/beta canonical** -
aarthi@aarthi-HP-Laptop-15s-fq5xxx: $
```

Fig 12.1

```
Activities Terminal Apr 22 17:01 aarth@aarthi-HP-Laptop-15s-fq5xxx -

aarth@aarthi-HP-Laptop-15s-fq5xxx:~$ adduser stack
adduser: Only root may add a user or group to the system.
aarth@aarthi-HP-Laptop-15s-fq5xxx:~$ sudo usedadd -s/bin/bash -d /opt/stack -m stack
[sudo] password for aarth:
sudo: usedadd: command not found
aarth@aarthi-HP-Laptop-15s-fq5xxx:~$ sudo snap install microstack --beta
[sudo] password for aarth:
microstack (beta) ussurl from Canonical++ installed
aarth@aarthi-HP-Laptop-15s-fq5xxx:~$ sudo snap install microstack --beta
snap "microstack" is already installed, see 'snap help refresh'
aarth@aarthi-HP-Laptop-15s-fq5xxx:~$ snap list microstack
Name Version Rev Tracking Publisher Notes
microstack ussurl 245 latest/beta canonical+ -
aarth@aarthi-HP-Laptop-15s-fq5xxx:~$ sudo microstack init --auto --control

2023-04-22 16:56:28,921 - microstack_init - INFO - Configuring clustering ...
2023-04-22 16:56:29,106 - microstack_init - INFO - Setting up as a control node.
2023-04-22 16:56:31,009 - microstack_init - INFO - Generating TLS Certificate and Key
2023-04-22 16:56:31,000 - microstack_init - INFO - Configuring networking ...
2023-04-22 16:56:36,152 - microstack_init - INFO - Opening horizon dashboard up to *
2023-04-22 16:56:36,772 - microstack_init - INFO - Waiting for RabbitMQ to start ...
2023-04-22 16:56:42,080 - microstack_init - INFO - RabbitMQ started!
2023-04-22 16:56:42,081 - microstack_init - INFO - Configuring RabbitMQ ...
2023-04-22 16:56:42,726 - microstack_init - INFO - RabbitMQ Configured!
2023-04-22 16:56:42,743 - microstack_init - INFO - Waiting for MySQL server to start ...
2023-04-22 16:56:48,043 - microstack_init - INFO - MySQL server started! Creating databases ...
2023-04-22 16:56:49,214 - microstack_init - INFO - Configuring Keystone Fernet Keys ...
2023-04-22 16:57:01,999 - microstack_init - INFO - Bootstrapping Keystone ...
2023-04-22 16:57:08,530 - microstack_init - INFO - Creating service project ...
2023-04-22 16:57:12,112 - microstack_init - INFO - Keystone configured!
2023-04-22 16:57:12,132 - microstack_init - INFO - Configuring the Placement service...
2023-04-22 16:57:24,837 - microstack_init - INFO - Running Placement DB migrations...
2023-04-22 16:57:27,448 - microstack_init - INFO - Configuring nova control plane services ...
2023-04-22 16:57:35,968 - microstack_init - INFO - Running Nova API DB migrations (this may take a lot of time)...
2023-04-22 16:57:53,138 - microstack_init - INFO - Running Nova DB migrations (this may take a lot of time)...
2023-04-22 16:58:45,017 - microstack_init - INFO - Creating default flavors...
2023-04-22 16:59:01,455 - microstack_init - INFO - Configuring nova compute hypervisor ...
2023-04-22 16:59:01,455 - microstack_init - INFO - Checking virtualization extensions presence on the host
2023-04-22 16:59:01,474 - microstack_init - INFO - Hardware virtualization is supported - KVM will be used for Nova instances
2023-04-22 16:59:03,173 - microstack_init - INFO - Configuring the Spice HTML5 console service...
2023-04-22 16:59:03,597 - microstack_init - INFO - Configuring Neutron
2023-04-22 16:59:06,283 - microstack_init - INFO - Configuring Glance ...
2023-04-22 17:00:06,283 - microstack_init - INFO - Adding cirros image ...
2023-04-22 17:00:35,612 - microstack_init - INFO - Creating security group rules ...
2023-04-22 17:00:42,022 - microstack_init - INFO - Configuring the Cinder services...
2023-04-22 17:01:01,962 - microstack_init - INFO - Running cinder DB migrations...
2023-04-22 17:01:30,483 - microstack_init - INFO - restarting libvirt and virtlogd ...
2023-04-22 17:01:33,756 - microstack_init - INFO - Complete. Marked microstack as initialized!
aarth@aarthi-HP-Laptop-15s-fq5xxx:~$
```

Fig 12.2

```
Activities Terminal Apr 22 17:01 aarth@aarthi-HP-Laptop-15s-fq5xxx -

aarth@aarthi-HP-Laptop-15s-fq5xxx:~$ adduser stack
adduser: Only root may add a user or group to the system.
aarth@aarthi-HP-Laptop-15s-fq5xxx:~$ sudo usedadd -s/bin/bash -d /opt/stack -m stack
[sudo] password for aarth:
sudo: usedadd: command not found
aarth@aarthi-HP-Laptop-15s-fq5xxx:~$ sudo snap install microstack --beta
[sudo] password for aarth:
microstack (beta) ussurl from Canonical++ installed
aarth@aarthi-HP-Laptop-15s-fq5xxx:~$ sudo snap install microstack --beta
snap "microstack" is already installed, see 'snap help refresh'
aarth@aarthi-HP-Laptop-15s-fq5xxx:~$ snap list microstack
Name Version Rev Tracking Publisher Notes
microstack ussurl 245 latest/beta canonical+ -
aarth@aarthi-HP-Laptop-15s-fq5xxx:~$ sudo microstack init --auto --control

2023-04-22 16:56:28,921 - microstack_init - INFO - Configuring clustering ...
2023-04-22 16:56:29,106 - microstack_init - INFO - Setting up as a control node.
2023-04-22 16:56:31,009 - microstack_init - INFO - Generating TLS Certificate and Key
2023-04-22 16:56:31,000 - microstack_init - INFO - Configuring networking ...
2023-04-22 16:56:36,152 - microstack_init - INFO - Opening horizon dashboard up to *
2023-04-22 16:56:36,772 - microstack_init - INFO - Waiting for RabbitMQ to start ...
2023-04-22 16:56:42,080 - microstack_init - INFO - RabbitMQ started!
2023-04-22 16:56:42,081 - microstack_init - INFO - Configuring RabbitMQ ...
2023-04-22 16:56:42,726 - microstack_init - INFO - RabbitMQ Configured!
2023-04-22 16:56:42,743 - microstack_init - INFO - Waiting for MySQL server to start ...
2023-04-22 16:56:48,043 - microstack_init - INFO - MySQL server started! Creating databases ...
2023-04-22 16:56:49,214 - microstack_init - INFO - Configuring Keystone Fernet Keys ...
2023-04-22 16:57:01,999 - microstack_init - INFO - Bootstrapping Keystone ...
2023-04-22 16:57:08,530 - microstack_init - INFO - Creating service project ...
2023-04-22 16:57:12,112 - microstack_init - INFO - Keystone configured!
2023-04-22 16:57:12,132 - microstack_init - INFO - Configuring the Placement service...
2023-04-22 16:57:24,837 - microstack_init - INFO - Running Placement DB migrations...
2023-04-22 16:57:27,448 - microstack_init - INFO - Configuring nova control plane services ...
2023-04-22 16:57:35,968 - microstack_init - INFO - Running Nova API DB migrations (this may take a lot of time)...
2023-04-22 16:57:53,138 - microstack_init - INFO - Running Nova DB migrations (this may take a lot of time)...
2023-04-22 16:58:45,017 - microstack_init - INFO - Creating default flavors...
2023-04-22 16:59:01,455 - microstack_init - INFO - Configuring nova compute hypervisor ...
2023-04-22 16:59:01,455 - microstack_init - INFO - Checking virtualization extensions presence on the host
2023-04-22 16:59:01,474 - microstack_init - INFO - Hardware virtualization is supported - KVM will be used for Nova instances
2023-04-22 16:59:03,173 - microstack_init - INFO - Configuring the Spice HTML5 console service...
2023-04-22 16:59:03,597 - microstack_init - INFO - Configuring Neutron
2023-04-22 16:59:06,283 - microstack_init - INFO - Configuring Glance ...
2023-04-22 17:00:06,283 - microstack_init - INFO - Adding cirros image ...
2023-04-22 17:00:35,612 - microstack_init - INFO - Creating security group rules ...
2023-04-22 17:00:42,022 - microstack_init - INFO - Configuring the Cinder services...
2023-04-22 17:01:01,962 - microstack_init - INFO - Running cinder DB migrations...
2023-04-22 17:01:30,483 - microstack_init - INFO - restarting libvirt and virtlogd ...
2023-04-22 17:01:33,756 - microstack_init - INFO - Complete. Marked microstack as initialized!
aarth@aarthi-HP-Laptop-15s-fq5xxx:~$
```

Fig 12.3

```

Python-3.12.0a1/Objects/funcobject.c.h
Python-3.12.0a1/Objects/clinic/descrobject.c.h
Python-3.12.0a1/Objects/clinic/enumobject.c.h
Python-3.12.0a1/Objects/clinic/listobject.c.h
Python-3.12.0a1/Objects/clinic/structseq.c.h
Python-3.12.0a1/Objects/clinic/longobject.c.h
Python-3.12.0a1/Objects/clinic/classobject.c.h
Python-3.12.0a1/Objects/clinic/bytearrayobject.c.h
Python-3.12.0a1/Objects/clinic/bytearrayobject.c
Python-3.12.0a1/Objects/clinic/dictobject.c.h
Python-3.12.0a1/Objects/clinic/complexobject.c.h
Python-3.12.0a1/Objects/clinic/memoryobject.c.h
Python-3.12.0a1/Objects/clinic/codeobject.c.h
Python-3.12.0a1/Objects/clinic/tupleobject.c.h
Python-3.12.0a1/Objects/capsule.c
Python-3.12.0a1/Objects/iterobject.c
Python-3.12.0a1/Objects/bytearrayobject.c
Python-3.12.0a1/Objects/dictobject.c
Python-3.12.0a1/Objects/dictnotes.txt
Python-3.12.0a1/Objects/interpreteridobject.c
Python-3.12.0a1/Objects/boolobject.c
Python-3.12.0a1/Objects/call.c
Python-3.12.0a1/Objects/genericaliasobject.c
Python-3.12.0a1/Objects/funcobject.c
Python-3.12.0a1/Objects/setobject.c
Python-3.12.0a1/Objects/floatobject.c
Python-3.12.0a1/Objects/_asn1_trampoline.S
Python-3.12.0a1/Objects/cellobject.c
Python-3.12.0a1/Objects/picklebufferobject.c
Python-3.12.0a1/Objects/_opcode.c
Python-3.12.0a1/Objects/_nodeobject.c
Python-3.12.0a1/Objects/exceptions.c
Python-3.12.0a1/Objects/object_layout.md
Python-3.12.0a1/Objects/listsort.txt
Python-3.12.0a1/Objects/complexobject.c
Python-3.12.0a1/Objects/sliceobject.c
Python-3.12.0a1/Objects/listobject.c
Python-3.12.0a1/Objects/_perf_trampoline.c
Python-3.12.0a1/Objects/_object.c
Python-3.12.0a1/Objects/_lnotab_notes.txt
Python-3.12.0a1/Objects/_typeobject.c
Python-3.12.0a1/Objects/_exception_handling_notes.txt
Python-3.12.0a1/Objects/_object_layout_312.gv
Python-3.12.0a1/Objects/_memoryobject.c
Python-3.12.0a1/Objects/_fileobject.c
Python-3.12.0a1/Objects/_unicodetype_db.h
Python-3.12.0a1/Objects/_namespaceobject.c
Python-3.12.0a1/Objects/_locations.md
Python-3.12.0a1/Objects/_tupleobject.c
Python-3.12.0a1/Objects/_install.sh
aarth@aarth-HP-Laptop-15s-fq5xxx:~$ cd python-3.12.0a1
bash: cd: python-3.12.0a1: No such file or directory
aarth@aarth-HP-Laptop-15s-fq5xxx:~$ ./configure --enable-optimizations
bash: ./configure: No such file or directory
aarth@aarth-HP-Laptop-15s-fq5xxx:~$

```

Fig 12.4

```

aarth@aarth-HP-Laptop-15s-fq5xxx:~$ sudo snap install openstack
aarth@aarth-HP-Laptop-15s-fq5xxx:~$ snap get openstack config.credentials
aarth@aarth-HP-Laptop-15s-fq5xxx:~$

```

Fig 12.5

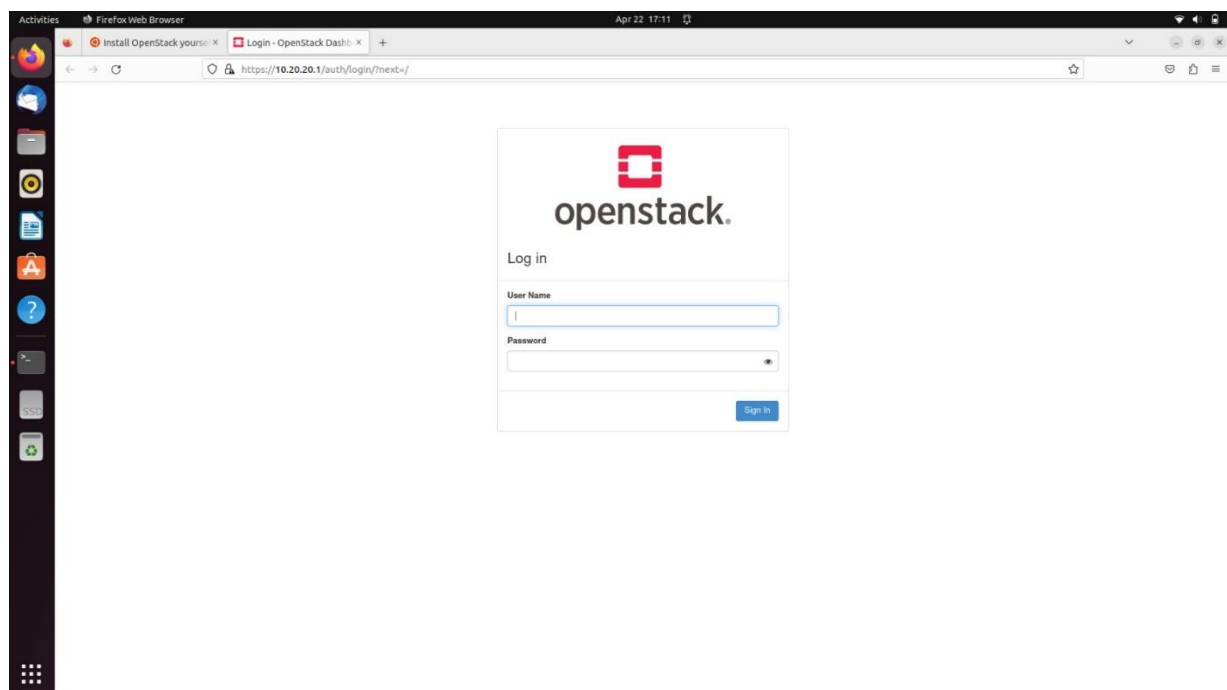


Fig 12.6

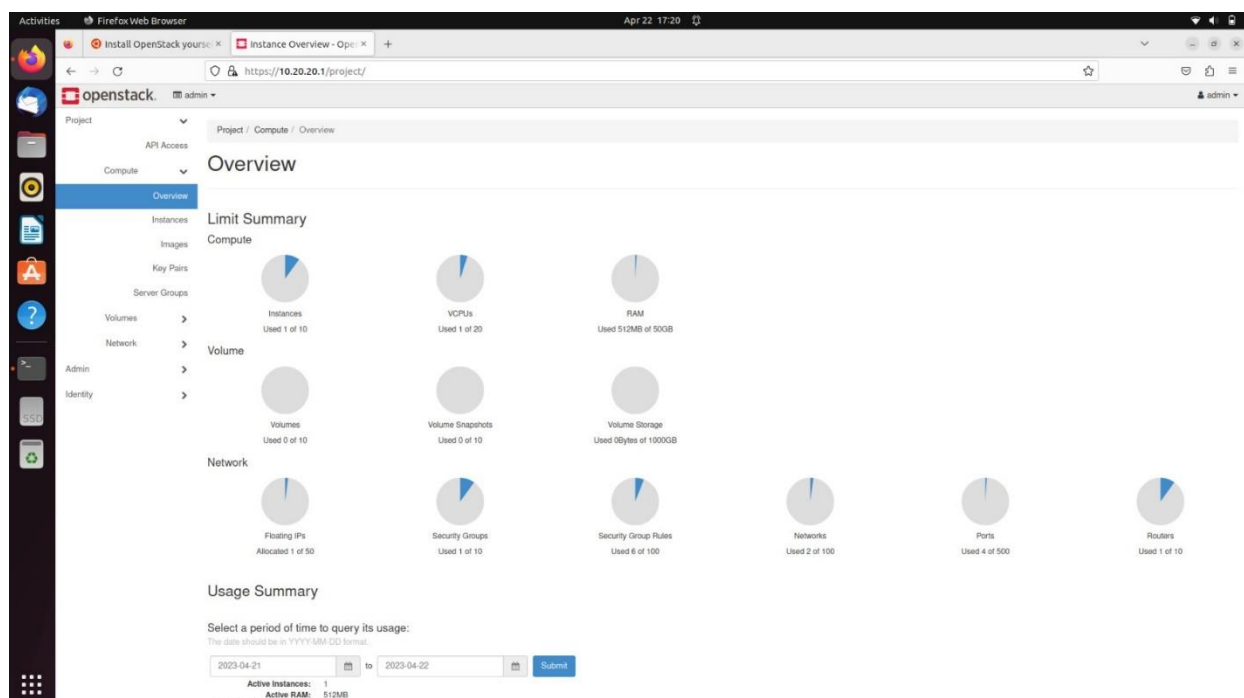


Fig 12.7

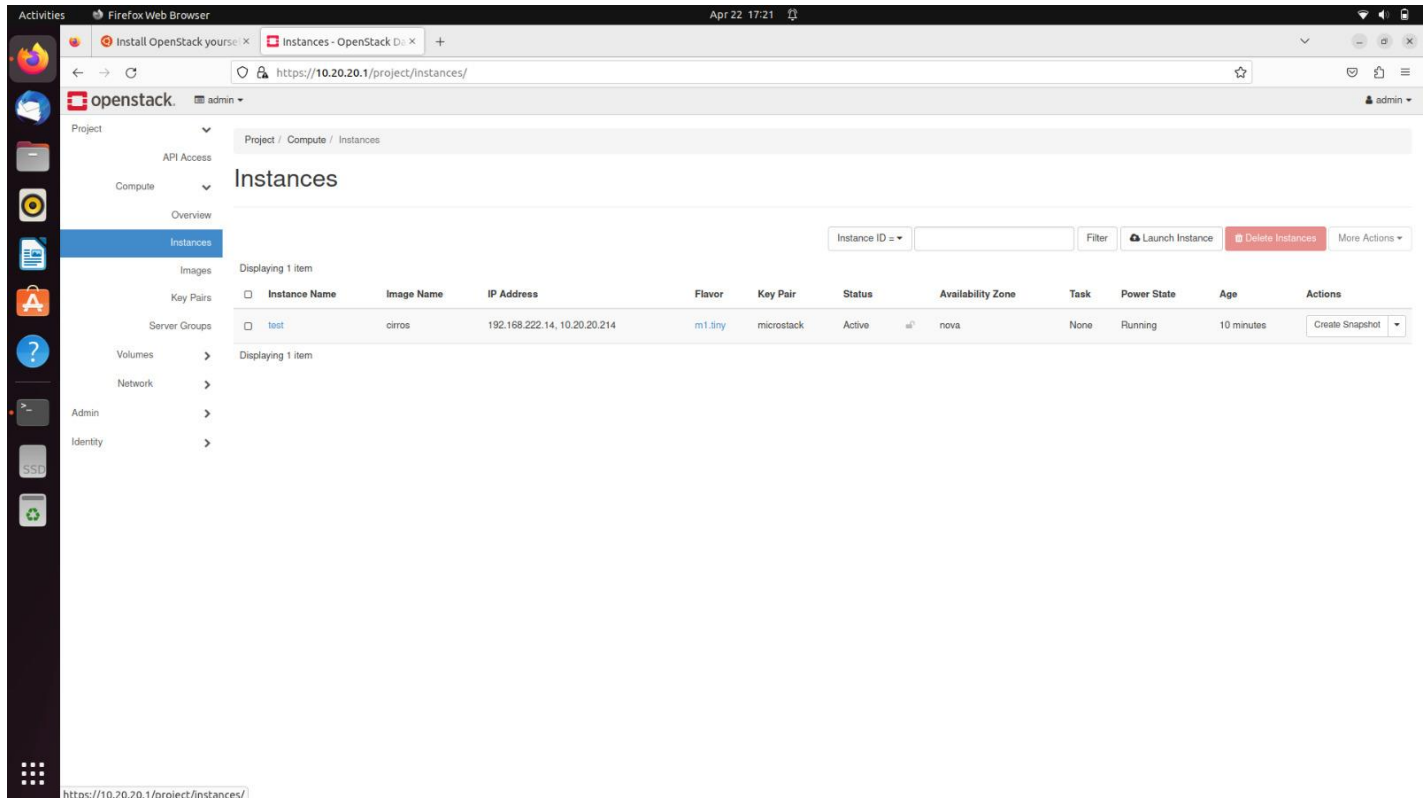


Fig 12.8

Choosing the secret storage behind Barbican is an important decision that will affect the security, performance, scalability, and cost of your secret management system. There are many factors to consider that will be specific to your needs.

Users with regulatory requirements or existing HSMs will gravitate toward the KMIP or PKCS#11 plugin. Users with tight budgets will look closely at software-only options. Those with high performance needs will look to crypto plugins.

Luckily, Barbican supports multiple plugins, which can be useful if your needs change later. It can also allow different secrets with different requirements to be stored by different plugins.

CHAPTER 13

CONCLUSION

- ❖ The recommended way to securely store and manage secrets in OpenStack is to use Barbican.
- ❖ Barbican is an OpenStack service that is multi-tenant aware and that uses Keystone tokens for authentication. This means that access to secrets is controlled via OpenStack policies for tenants and RBAC roles.
- ❖ Secrets (such as certificates, API keys, and passwords) can either be stored as an encrypted blob in the Barbican database, directly in a secure storage system, including a Hardware Security Module (HSM) appliance.

CHAPTER 14

REFERENCE

- ▶ Ding, J. 2008. MSc Project in A web 2.0 approach to a fashionista website. Heriot-Watt University
- ▶ Jazayeri, M. 2007 Some trends in web application development. Future of Software Engineering.
- ▶ Lu, K. M. and Ab Hamid, S.H.2007, Conceptual Design of Web-Based Appointment Management System using Object
- ▶ Alex Chaffee (2000-08-17). "What is a web application (or "webapp")?". Retrieved 2008-07-27.

