

PROFESSIONAL TRAINING REPORT
Sathyabama Institute of Science and Technology
(Deemed to be University)

Submitted in partial fulfilment of the requirements for the award
of Bachelor of Technology Degree in Information Technology
By

Sowmiya RK
(40120087)



DEPARTMENT OF INFORMATION TECHNOLOGY

SCHOOL OF COMPUTING

**SATHYABAMA INSTITUTE OF SCIENCE AND
TECHNOLOGY (DEEMED TO BE UNIVERSITY)**
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,

CHENNAI, TAMILNADU –

600119 **April 2023**





SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

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

(www.sathyabama.com)

DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Sowmiya RK (Reg. No. 40120087)** who carried out the project as a "**Openstack Installation with Binaries on Ubuntu**" under our supervision from January 2023 to April 2024.

Internal Guide

Dr. P Jeyanthi, M.E., Ph.D.

Head of the Department

Dr. R Subhashini, M.E., Ph.D.

Internal Examiner

External Examiner

DECLARATION

I Sowmiya(Reg. No. 40120087) hereby declare that the Project Report entitled "Openstack Installation with Binaries on Ubuntu" is done by me under the guidance of Dr. P Jeyanthi, M.E., Ph.D., and is submitted in partial fulfilment of the requirements for the award of Bachelor of Technology degree in Information Technology.

DATE: 29/4/23

PLACE: Chennai

Sowmiya

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to the **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. P Jeyanthi, M.E., Ph.D.**, Head of the Department of Information Technology for providing me with 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. P Subhashini, M.E., Ph.D.** for her valuable guidance, suggestions and constant encouragement that 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.

CERTIFICATE OF COMPLETION



ABSTRACT

OpenStack is a popular open-source software platform used for managing cloud computing infrastructure. It is a set of software tools for building and managing cloud computing platforms for public and private clouds. OpenStack is widely used by organizations of all sizes to deploy cloud services and infrastructure.

One of the important aspects of OpenStack deployment is the installation process. Installing OpenStack with binaries on Ubuntu is a common method that allows users to get started with OpenStack quickly and easily. In this approach, pre-built binary packages are used to install OpenStack components rather than building from source.

This method simplifies the installation process and makes it possible for users to get up and running with OpenStack in a short amount of time. In this context, we will explore the process of installing OpenStack with binaries on Ubuntu and the benefits it provides.

TABLE OF CONTENT

Chapter No.	Title	Page No.
1	Introduction	1
2	Problem Statement	2
3	Literature Survey	3
4	Technological Overview	5
4.1	Libraries used	6
4.2	Tools/ Service Used	11
5	Architecture & Methodology	17
5.1	Service Configuration	18
5.2	Openstack Dashboard	19
6	Results & Discussion	20
6.1	Configure Keystone	20
6.2	Configure Glance	20
6.3	Configure Nova	21
6.4	Configure Neutron	23
6.5	Configure Horizon	24
7	References	25

List of Images

Sl. No	Figure Number	Figure Description

1	5.1	Configuration Flow
2	6.1	Configuration Keystone
3	6.2	Configuration Glance
4	6.3.1	Configuration Nova part1
5	6.3.2	Configuration Nova part2
6	6.3.3	Configuration Nova part3
7	6.4.1	Configuration Neutron part1
8	6.4.2	Configuration Neutron part2
9	6.4.3	Configuration Neutron part3
10	6.4.4	Configuration Neutron part4
11	6.5	Configuration Horizon
12	6.8	Configuration Glance
13	6.9	Configuration Glance

List of Tables

Sl. No	Table Number	Table Description
1	4.1	List of modules and technologies used

Chapter 1 INTRODUCTION

OpenStack is an open-source cloud computing platform that is widely used for building and managing cloud infrastructure. Installing OpenStack with binaries on Ubuntu is a popular method that simplifies the installation process for users.

This approach involves using pre-built binary packages to install OpenStack components, which makes it easier for users to get started with OpenStack. With this method, users can quickly deploy OpenStack and begin using its powerful cloud management features.

In this context, we will explore the process of installing OpenStack with binaries on Ubuntu and the benefits it offers. The ETL phase in this work involved understanding the data, anomaly, nature, and quality of data. And knowledge extraction involved understanding the business and business needs.

Chapter 2 PROBLEM STATEMENT

Despite the growing popularity of OpenStack as a cloud computing platform, many users encounter difficulties when installing and configuring it on Ubuntu.

The installation process can be complex and time-consuming, especially for those with limited experience in cloud computing and Linux administration.

This presents a challenge for organizations and individuals who want to leverage the benefits of OpenStack but lack the technical expertise to set it up effectively.

To address this problem, a straightforward and reliable installation method using pre-built binaries is needed to simplify the installation process and make it more accessible to a wider range of users.

2.2. OBJECTIVES

The objective of this project is to simplify the installation of OpenStack on Ubuntu using pre-built binaries, making it more accessible to a wider range of users.

This will involve developing a step-by-step guide that provides clear and concise instructions for setting up an OpenStack environment using prebuilt binaries.

The guide will include the necessary steps for configuring various OpenStack services such as Keystone, Nova, Neutron, and Glance, and will be designed to be easy to follow, even for users with limited experience in cloud computing and Linux administration.

The overall goal is to enable more organizations and individuals to benefit from the power and flexibility of OpenStack by simplifying the installation process and reducing the barriers to adoption.

Chapter 3

LITERATURE SURVEY

"OpenStack Cloud Computing Cookbook" by Kevin Jackson and Cody Bunch: This book provides a comprehensive guide to installing, configuring, and managing an OpenStack cloud environment. It includes a chapter on installing OpenStack using pre-built binaries, which can be helpful for users who are new to OpenStack and want a straightforward installation process.

"The OpenStack Operations Guide" by Tom Fifield, Diane Fleming, Anne Gentle, Lorin Hochstein, Jonathan Proulx, and Everett Toews: This guide provides an in-depth look at the best practices for deploying and operating an OpenStack cloud. It includes a section on installing OpenStack using pre-built packages, which can help users get up and running quickly and easily.

"OpenStack Installation Guide for Ubuntu" by the OpenStack Foundation: This guide provides step-by-step instructions for installing OpenStack on Ubuntu, including using pre-built packages. It includes detailed information on configuring various OpenStack services, such as Keystone, Nova, Neutron, and Glance.

"OpenStack Cloud Computing: Architecture Guide" by John Rhoton: This book provides an overview of the OpenStack cloud computing platform, including its architecture and components. It includes a section on installing OpenStack using pre-built packages, which can be helpful for users who are new to OpenStack and want a simplified installation process.

"OpenStack Essentials" by Dan Radez: This book provides an introduction to the OpenStack cloud computing platform, including its features and capabilities. It includes a chapter on installing OpenStack using pre-built packages, which can be useful for users who are new to OpenStack and want a simplified installation process.

Overall, the literature survey shows that there are several resources available for installing OpenStack on Ubuntu using pre-built binaries. These resources can be helpful for users who are new to OpenStack or who want a simplified installation process. However, it is important to follow best practices and ensure that the installation is properly configured to ensure a stable and reliable OpenStack environment.

Chapter 4

TECHNOLOGICAL OVERVIEW

This whole work is based on Python 3.8 For working on different modules, different Python libraries and tools have been brought to use.

The list of the libraries and tools used are as follows:

Table 4.1 List of Modules Used

S. no.	Name	Purpose
1	Python	Proraming Language
2	Libffi	Calling library dinamically
3	Libssl	To implement SSL
4	Libxml2	Parse and validate xml
5	Libxslt1	Perform XSLT transfermation
6	Libjpeg	Handling jpeg image files
7	Libpq	slq client library
8	Libmysqlclient	mysql client library

9	Libyaml	handle yaml files
---	---------	-------------------

4.1. LIBRARIES USED

4.1.1. *sudo apt*

The command "sudo apt" is used in Ubuntu and other Debianbased operating systems as a package manager.

"sudo" gives the user superuser privileges for executing the command, allowing for the installation of packages that require elevated permissions.

"apt" stands for "Advanced Packaging Tool" and is used to manage the installation, updating, and removal of software packages on the system. Command for Installation:

```
sudo apt
```

4.1.2. *python-pip*

"python-pip" is a package manager for the Python programming language.

It is used for installing, upgrading, and managing Python packages and their dependencies.

"python-pip" is a crucial tool for managing the various Python dependencies required for OpenStack.

It simplifies the process of installing and upgrading Python packages, and it can also be used to uninstall packages when necessary.

By using "python-pip", users can easily manage the Python packages required for OpenStack without having to worry about manual installation and dependency resolution.

Command for Installation:

```
sudo apt python-pip
```

4.1.3. *python-dev*

"python-dev" is a package that contains header files and a static library for developing Python extensions in C or C++.

It is used for developing Python modules that require C or C++ code, such as modules for scientific computing, machine learning, and database connectivity.

"python-dev" also includes the Python.h header file, which is required for building C or C++ extensions that interface with the Python interpreter.

Without "python-dev", it would not be possible to compile and install Python packages that require C or C++ extensions.

It is recommended to install "python-dev" alongside "pythonpip" when developing Python applications or working with packages that have C or C++ dependencies.

Command for Installation:

```
sudo apt python-dev
```

4.1.4. *libxml2-dev*

"libffi-dev" is a library that provides a portable interface for calling C functions from other programming languages.

It is required by some software packages that use dynamic code generation and require a just-in-time compiler (JIT) to generate machine code at runtime.

"libffi-dev" also provides a set of tools and headers for building and testing C code that uses the Foreign Function Interface (FFI).

FFI is used to enable cross-language interoperability, allowing code written in different programming languages to communicate with each other.

Installing "libffi-dev" is necessary for building and running software that uses the FFI, such as the PyPy Python interpreter and the LuaJIT just-in-time compiler.

Command for Installation:

```
sudo apt libxml2-dev
```

4.1.5. *libxslt1-dev*

"libxslt1-dev" is a library that provides an implementation of the XSLT (eXtensible Stylesheet Language Transformations) standard.

It is used for processing and transforming XML documents, such as converting XML data into HTML or other formats suitable for display on the web.

"libxslt1-dev" includes a set of tools and headers for building and testing applications that use XSLT.

It also provides a command-line tool called "xsltproc" that can be used for testing and debugging XSLT stylesheets. "libxslt1-

dev" is a prerequisite for building and running software that uses XSLT, such as web frameworks that use XML templates to generate HTML pages. Command for Installation:

```
sudo apt libxslt1-dev
```

4.1.6. *libjpeg-dev*

"libjpeg-dev" is a library that provides support for the JPEG image format.

It is used by many software applications for reading, writing, and manipulating JPEG images.

"libjpeg-dev" includes a set of tools and headers for building and testing applications that use the JPEG image format.

It also provides a command-line tool called "djpeg" that can be used for testing and debugging JPEG images.

"libjpeg-dev" is a prerequisite for building and running software that uses JPEG images, such as image editing software or image processing libraries.

Command for Installation:

```
sudo apt libjpeg-dev
```

4.1.7. *libpq-dev*

"libpq-dev" is a library that provides support for the PostgreSQL database system.

It is used by many software applications for interacting with PostgreSQL databases.

"libpq-dev" includes a set of tools and headers for building and testing applications that use PostgreSQL databases.

It also provides a command-line tool called "psql" that can be used for testing and debugging PostgreSQL queries.

"libpq-dev" is a prerequisite for building and running software that uses PostgreSQL databases, such as web applications or data analytics tools. Command for Installation:

```
sudo apt libpq-dev
```

4.1.8. *libmysqlclient-dev*

P "libmysqlclient-dev" is a library that provides support for the MySQL database system. It is used by many software applications for interacting with MySQL databases.

"libmysqlclient-dev" includes a set of tools and headers for building and testing applications that use MySQL databases.

It also provides a command-line tool called "mysql" that can be used for testing and debugging MySQL queries.

"libmysqlclient-dev" is a prerequisite for building and running software that uses MySQL databases, such as web applications or data analytics tools. Command for Installation:

```
sudo apt libmysqlclient-dev
```

4.1.9. *libyaml-dev*

"libyaml-dev" is a library that provides support for the YAML (YAML Ain't Markup Language) serialization format.

It is used by many software applications for reading and writing YAML files.

"libyaml-dev" includes a set of tools and headers for building and testing applications that use the YAML format.

"git" is a version control system that is used for tracking changes in code and collaborating on software development projects.

It is often used for downloading and managing code repositories, including those related to OpenStack.

Command for Installation:

```
do apt libyaml -dev
```

4.2. TOOLS / SERVICE USED

4.2.1. KEYSTONE

Keystone is an identity service in OpenStack that provides authentication and authorization for all OpenStack services. It acts as a central repository for users, roles, and permissions. Keystone supports different types of authentication methods, including password-based, token-based, and federated identity systems. Keystone also integrates with external identity providers, such as LDAP and Active Directory.

Keystone manages the assignment of roles and permissions to users and groups, allowing administrators to control access to resources across the OpenStack cloud. It supports role-based access control (RBAC) and attribute-based access control (ABAC). Keystone provides a simple and scalable architecture that is highly available and fault-tolerant. It uses a database to store its configuration and policies.

Keystone is a critical component in OpenStack, as it enables secure communication and collaboration between different

OpenStack services. It provides a unified and consistent identity management framework, making it easier for users to manage their cloud resources. Keystone also provides audit trails and logging capabilities, helping administrators track and monitor user activities.

4.2.2. NOVA

Nova is the compute service in OpenStack that provides virtual machine (VM) and instance management. It is responsible for launching and scheduling VM instances on physical hosts, managing their lifecycles, and providing network and storage connectivity. Nova enables users to create, start, stop, and terminate VMs, as well as scale up or down their computing resources.

Nova architecture is designed to be modular and flexible, allowing it to integrate with different hypervisors, such as KVM, VMware, and Hyper-V. It also supports multiple storage backends, including local disk, NFS, and iSCSI. Nova uses a messaging system to communicate with other OpenStack services, such as Keystone and Neutron, enabling secure and reliable communication.

Nova provides advanced features, such as live migration, which allows users to move running VMs between hosts without downtime, and instance snapshots, which enable users to capture the state of their VMs for backup or recovery purposes. It also supports policy-based scheduling, enabling administrators to allocate resources based on user-defined policies.

Nova is a critical component in OpenStack, as it provides the foundation for creating and managing VMs in the cloud. It enables users to leverage the power of virtualization to run their workloads efficiently and cost-effectively. Nova is highly scalable and can handle large-scale deployments, making it suitable for enterprises and service providers.

4.2.3. NEUTRON

Neutron is a networking service in OpenStack that provides network connectivity as a service to virtual machines and other instances created within an OpenStack cloud. It enables the creation and management of virtual networks and network services such as routers, subnets, and floating IPs.

Neutron also supports advanced network capabilities such as network segmentation, quality of service (QoS), and network security groups.

It allows for the integration of third-party software-defined networking (SDN) solutions, enabling greater flexibility and customization.

Neutron's modular architecture enables the integration of plugins for various networking technologies, including virtual and physical switches, routers, and firewalls.

Overall, Neutron simplifies the management of networking in OpenStack, providing a centralized platform for creating and managing networks and network services.

4.2.4. NEUTRON

Glance is an OpenStack service that provides a registry for virtual machine images, which can be used to quickly deploy new virtual machines within an OpenStack cloud environment.

Through Glance, administrators can store, manage, and share images of virtual machines and other instances, making it easy to deploy new instances on-demand.

Glance supports a wide range of image formats, including popular Linux distributions such as Ubuntu and CentOS, as well as Windows and other operating systems.

Glance also provides a metadata service that allows administrators to associate additional information with each image, such as its operating system version, patch level, and other details.

Glance integrates with other OpenStack services, such as Nova and Cinder, to provide seamless image deployment and management across the cloud environment.

Overall, Glance is a critical component of the OpenStack ecosystem, allowing administrators to quickly and easily deploy

new instances, manage image lifecycles, and maintain consistency across their cloud environment.

4.2.5. HORIZON

Horizon is the web-based user interface for OpenStack. It allows administrators and users to access and manage OpenStack services and resources through a graphical interface. With Horizon, users can create, modify, and delete instances, manage networks and routers, create and manage users and projects, and more.

Horizon is extensible and can be customized to meet specific needs. Users can add or remove panels, modify workflows, and create new views to match their requirements. Horizon also provides a dashboard feature that allows users to create custom dashboards for monitoring and managing their resources.

Horizon is built using the Django web framework, which provides a robust and secure platform for web development. It supports multiple authentication backends, including Keystone, LDAP, and OAuth. Horizon also includes a REST API that can be used for automation and integration with other systems.

Horizon provides a consistent and intuitive user experience across all OpenStack services. It provides a unified view of resources and services, making it easier for users to manage their cloud infrastructure. With Horizon, users can monitor usage

and performance, troubleshoot issues, and perform other tasks quickly and efficiently.

Horizon is continuously evolving and improving, with new features and enhancements being added with each OpenStack release. It is a critical component of OpenStack, providing a userfriendly interface that makes it easier to manage and use the cloud platform.

Chapter 5 METHODOLOGY & ARCHITECTURE

The architecture of OpenStack installation with binaries on Ubuntu is relatively straightforward. It involves several components that work together to create a functional cloud infrastructure. The architecture typically includes:

1. **Ubuntu Operating System** - This is the base operating system on which OpenStack is installed. Ubuntu is a popular choice for OpenStack installations due to its stability and ease of use.

2. **Binary Packages** - Pre-built binary packages for OpenStack components are downloaded and installed on the Ubuntu system. These packages include Keystone, Glance, Nova, Neutron, Cinder, and Swift, among others.

3. **OpenStack Services** - Once the binary packages are installed, OpenStack services are started. These services include the Identity service (Keystone), Image service (Glance), Compute service (Nova), Networking service (Neutron), Block Storage service (Cinder), and Object Storage service (Swift).

4. **OpenStack Dashboard** - Finally, the OpenStack Dashboard (Horizon) is installed and configured to allow users to manage their cloud infrastructure through a web-based interface.

Overall, the architecture of OpenStack installation with binaries on Ubuntu is designed to be simple and straightforward, making it easy for users to get started with OpenStack quickly and efficiently.

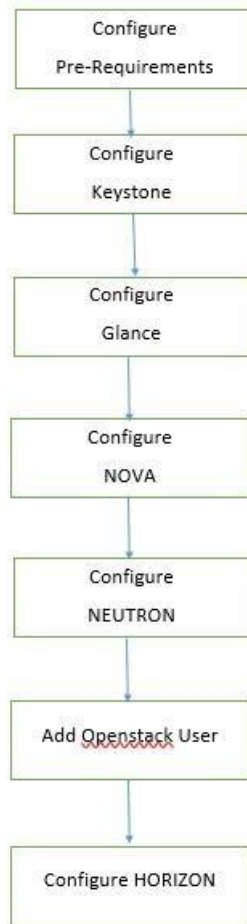


Fig. 5.1 Configuration Flow

5.1. SERVICE CONFIGURATION

Service configuration refers to the process of setting up and customizing various services in a system to meet specific requirements. In the context of OpenStack, service configuration involves configuring different services such as Nova, Neutron, Keystone, Glance, and Horizon to work together seamlessly.

This includes setting up the database, configuring communication between the services, defining network topologies, and configuring access control policies. Service configuration can be performed

manually through configuration files or automated using configuration management tools such as Ansible or Puppet.

Proper service configuration is critical to ensuring the stability and reliability of an OpenStack deployment.

5.2. Openstack Dashboard

The OpenStack Dashboard, also known as Horizon, is a webbased graphical user interface that allows users to interact with various OpenStack services, including compute, networking, and storage.

It provides a unified and easy-to-use interface for managing and monitoring OpenStack resources, enabling users to create and manage instances, networks, and volumes, among other things.

The dashboard also provides a customizable dashboard view, role-based access control, and support for various authentication methods, including LDAP, Keystone, and OAuth.

Chapter 6

RESULTS & DISCUSSION

6.1 CONFIGURING KEYSTONE

```
root@d1p:~# vi /etc/keystone/keystone.conf

# line 430: uncomment and specify Memcache Server
memcache_servers = 10.0.0.30:11211

# line 557: change to MariaDB connection info
connection = mysql+pymysql://keystone:password@10.0.0.30/keystone

# line 2467: uncomment
provider = fernet

root@d1p:~# su -s /bin/bash keystone -c "keystone-manage db_sync"

# initialize Fernet key
root@d1p:~# keystone-manage fernet_setup --keystone-user keystone --keystone-group keystone
root@d1p:~# keystone-manage credential_setup --keystone-user keystone --keystone-group keystone

# define keystone API Host
root@d1p:~# export controller=10.0.0.30

# bootstrap keystone
# set any password for [adminpassword] section
root@d1p:~# keystone-manage bootstrap --bootstrap-password adminpassword \
--bootstrap-admin-url http://$controller:5000/v3/ \
--bootstrap-internal-url http://$controller:5000/v3/ \
--bootstrap-public-url http://$controller:5000/v3/ \
--bootstrap-region-id RegionOne
```

Fig. 6.1 Configuring keystone

6.2 CONFIGURING GLANCE

```
root@d1p ~(keystone)# mv /etc/glance/glance-api.conf /etc/glance/glance-api.conf.org
root@d1p ~(keystone)# vi /etc/glance/glance-api.conf

# create new
[DEFAULT]
bind_host = 0.0.0.0

[glance_store]
stores = file,http
default_store = file
filesystem_store_datadir = /var/lib/glance/images/

[database]
# MariaDB connection info
connection = mysql+pymysql://glance:password@10.0.0.30/glance

# keystone auth info
[keystone_authtoken]
www_authenticate_uri = http://10.0.0.30:5000
auth_url = http://10.0.0.30:5000
memcached_servers = 10.0.0.30:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = glance
password = servicepassword

[paste_deploy]
flavor = keystone

root@d1p ~(keystone)# chmod 640 /etc/glance/glance-api.conf
root@d1p ~(keystone)# chown root:glance /etc/glance/glance-api.conf

root@d1p ~(keystone)# su -s /bin/bash glance -c "glance-manage db_sync"
root@d1p ~(keystone)# systemctl restart glance-api
```

Fig. 6.2 Configuring Glance

6.3. CONFIGURE NOVA

```
root@d1p ~(keystone)# mv /etc/nova/nova.conf /etc/nova/nova.conf.org
root@d1p ~(keystone)# vi /etc/nova/nova.conf

# create new
[DEFAULT]
# define own IP address
my_ip = 10.0.0.30
state_path = /var/lib/nova
enabled_apis = osapi_compute,metadata
log_dir = /var/log/nova
# RabbitMQ connection info
transport_url = rabbit://openstack:password@10.0.0.30

[api]
auth_strategy = keystone

# Glance connection info
[glance]
api_servers = http://10.0.0.30:9292

[oslo_concurrency]
lock_path = $state_path/tmp

# MariaDB connection info
[api_database]
connection = mysql+pymysql://nova:password@10.0.0.30/nova_api

[database]
connection = mysql+pymysql://nova:password@10.0.0.30/nova
```

Fig. 6.3.1 Configuring Nova part1

```
# Keystone auth info
[keystone_authtoken]
www_authenticate_uri = http://10.0.0.30:5000
auth_url = http://10.0.0.30:5000
memcached_servers = 10.0.0.30:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = nova
password = servicepassword

[placement]
auth_url = http://10.0.0.30:5000
os_region_name = RegionOne
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = placement
password = servicepassword

[wsgi]
api_paste_config = /etc/nova/api-paste.ini

root@d1p ~(keystone)# chmod 640 /etc/nova/nova.conf
root@d1p ~(keystone)# chgrp nova /etc/nova/nova.conf

root@d1p ~(keystone)# mv /etc/placement/placement.conf /etc/placement/placement.conf.org
root@d1p ~(keystone)# vi /etc/placement/placement.conf
```

Fig. 6.3.2 Configuring Nova part2

```
# create new
[DEFAULT]
debug = false

[api]
auth_strategy = keystone

[keystone_authtoken]
www_authenticate_uri = http://10.0.0.30:5000
auth_url = http://10.0.0.30:5000
memcached_servers = 10.0.0.30:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = placement
password = servicepassword

[placement_database]
connection = mysql+pymysql://placement:password@10.0.0.30/placement

root@d1p ~(keystone)# chmod 640 /etc/placement/placement.conf
root@d1p ~(keystone)# chgrp placement /etc/placement/placement.conf
```

Fig. 6.3.3 Configuring Nova part3

6.4 CONFIGURE NEUTRON

```
root@d1p ~(keystone)# mv /etc/neutron/neutron.conf /etc/neutron/neutron.conf.org
root@d1p ~(keystone)# vi /etc/neutron/neutron.conf

# create new
[DEFAULT]
core_plugin = ml2
service_plugins = router
auth_strategy = keystone
state_path = /var/lib/neutron
dhcp_agent_notification = True
allow_overlapping_ips = True
notify_nova_on_port_status_changes = True
notify_nova_on_port_data_changes = True
# RabbitMQ connection info
transport_url = rabbit://openstack:password@10.0.0.30

[agent]
root_helper = sudo /usr/bin/neutron-rootwrap /etc/neutron/rootwrap.conf

# Keystone auth info
[keystone_authtoken]
www_authenticate_uri = http://10.0.0.30:5000
auth_url = http://10.0.0.30:5000
memcached_servers = 10.0.0.30:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = neutron
password = servicepassword

# MariaDB connection info
[database]
connection = mysql+pymysql://neutron:password@10.0.0.30/neutron_ml2
```

Fig. 6.4.1 Configuring Neutron part1

```

# Nova auth info
[nova]
auth_url = http://10.0.0.30:5000
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = nova
password = servicepassword

[oslo_concurrency]
lock_path = $state_path/tmp

root@d1p ~(keystone)# chmod 640 /etc/neutron/neutron.conf
root@d1p ~(keystone)# chgrp neutron /etc/neutron/neutron.conf
root@d1p ~(keystone)# vi /etc/neutron/l3_agent.ini

# line 21: add
interface_driver = linuxbridge

root@d1p ~(keystone)# vi /etc/neutron/dhcp_agent.ini

# line 21: add
interface_driver = linuxbridge

# line 43: uncomment
dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq

# line 52: uncomment and change
enable_isolated_metadata = true

root@d1p ~(keystone)# vi /etc/neutron/metadata_agent.ini

# line 22: uncomment and specify Nova API server
nova_metadata_host = 10.0.0.30

```

Fig. 6.4.2 Configuring Neutron part2

```

# line 34: uncomment and specify any secret key you like
metadata_proxy_shared_secret = metadata_secret

# line 300: uncomment and specify Memcache Server
memcache_servers = 10.0.0.30:11211

root@d1p ~(keystone)# vi /etc/neutron/plugins/ml2/ml2_conf.ini

# line 154: add
# OK with no value for [tenant_network_types] now (set later if need)
[ml2]
type_drivers = flat,vlan,vxlan
tenant_network_types =
mechanism_drivers = linuxbridge
extension_drivers = port_security

root@d1p ~(keystone)# vi /etc/neutron/plugins/ml2/linuxbridge_agent.ini

# line 218: add
[securitygroup]
enable_security_group = True
firewall_driver = iptables
enable_ipset = True

# line 277: add own IP address
local_ip = 10.0.0.30

root@d1p ~(keystone)# vi /etc/nova/nova.conf

# add follows into [DEFAULT] section
use_neutron = True
linuxnet_interface_driver = nova.network.linux_net.LinuxBridgeInterfaceDriver
firewall_driver = nova.virt.firewall.NoopFirewallDriver
vif_plugging_is_fatal = True
vif_plugging_timeout = 300

```

Fig. 6.4.3 Configuring Neutron part3

```
# add follows to the end : Neutron auth info
# the value of [metadata_proxy_shared_secret] is the same with the one in [metadata_agent.ini]
[neutron]
auth_url = http://10.0.0.30:5000
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = neutron
password = servicepassword
service_metadata_proxy = True
metadata_proxy_shared_secret = metadata_secret
```

Fig. 6.4.4 Configuring Neutron part4

6.5 CONFIGURE HORIZON

```
root@d1p ~(keystone)# vi /etc/openstack-dashboard/local_settings.py

# line 99: change Memcache server
CACHES = {
    'default': {
        'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',
        'LOCATION': '10.0.0.30:11211',
    },
}

# line 113: add
SESSION_ENGINE = "django.contrib.sessions.backends.cache"

# line 126: set Openstack Host
# line 127: comment out and add a line to specify URL of Keystone Host
OPENSTACK_HOST = "10.0.0.30"
#OPENSTACK_KEYSTONE_URL = "http://%s/identity/v3" % OPENSTACK_HOST
OPENSTACK_KEYSTONE_URL = "http://10.0.0.30:5000/v3"

# line 131: set your timezone
TIME_ZONE = "Asia/Tokyo"

# add to the end
OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = True
OPENSTACK_KEYSTONE_DEFAULT_DOMAIN = 'Default'
OPENSTACK_API_VERSIONS = {
    "identity": 3,
    "volume": 3,
    "compute": 2,
}

root@d1p:~# systemctl restart apache2
```

Fig. 6.5 Configure Horizon

Chapter 7

REFERENCES

1. https://www.serverworld.info/en/note?os=Ubuntu_20.04&p=openstack_ussuri&f=1
2. <https://docs.openstack.org/install-guide/environmentpackagesubuntu.html>
3. <https://linuxhint.com/install-openstack-ubuntu/>
4. <https://www.digitalocean.com/community/tutorials/installopenstackubuntu-devstack>
5. <https://ubuntu.com/openstack/install>
6. <https://www.linuxfordevices.com/tutorials/ubuntu/installopenstackon-ubuntu>