eman ta zabal zazu

Universidad del País Vasco Euskal Herriko Unibertsitatea

BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

ILLINOIS INSTITUTE OF TECHNOLOGY

MASTER'S IN TELECOMMUNICATION TECHNOLOGY ENGINEERING
&
MASTER'S IN INFORMATION TECHNOLOGY AND MANAGEMENT

MASTER'S THESIS

<DESIGN AND BUILD OF A CUSTOM PRIVATE CLOUD>

Student: <Baranda Barron, Jon Andoni>
Professor: <Hajek, Jeremy>

Acadmeic year: <2021-2022>

Date: <Chicago, 13, August, 2022>

Universidad del País Vasco Euskal Herriko Unibertsitatea | BILBOKO INGENIARITZA ESKOLA ESCUELA DE INGENIERÍA DE BILBAO

ILLINOIS INSTITUTE OF TECHNOLOGY

1. Index

1. *Index* 1
2. *List of figures* 3
3. *List of tables* 5
4. *Introduction* 6
5. *Context* 7

5.1 2.1 Research Group Mystic 7

6. *Objectives and scope of work* 8
7. *Benefits of the research* 9

7.1 Social benefits 9
7.2 Economic benefits 9
7.3 Technical benefits 10

8. *State of the art* 11

8.1 What is cloud computing? 11
8.2 Types of clouds 13
8.2.1 According to the development 13
8.2.2 According to service model 14

9. *Analysis of alternatives* 16

9.1 Platform 16
9.2 Hardware 17
9.3 Working method 19

10. *Risk Analysis* 20

10.1 Possible risks 20
10.2 Risk Impact 21
10.3 Contingency plan 22

11. *Description of the system: Design* 24

11.1 Services 24
11.3 Architecture Nodes 26

12. *Installation and configuration process* 29

12.1 Identity service. Keystone 29
12.1.1 Prerequisites 29
12.1.2 Installation and configuration 30

12.2 Image service. Glance 31
12.2.1 Prerequisites 31

12.2.2 Installation and configuration 33

12.3 Placement service 34

12.3.1 Prerequisites 34

12.3.2 Installation and configuration 36

12.4 Compute service. Nova 37

12.4.1 Prerequisites 37

12.4.2 Installation and configuration 39

12.5 Networking service. Neutron 41

12.5.1 Prerequisites 41

12.5.2 Installation and configuration 43

12.6 Dashboard service. Horizon 46

12.6.1 Installation and configuration 46

13. Cloud Management. Result 47

13.1 Tools to manage the cloud 47

13.2 Openstack dashboard capabilities 48

13.3 Dashboard features 48

13.3.1 Log in 49

13.3.2 Overview panel 50

13.3.3 API Access 50

13.3.4 Instances 51

13.3.5 Images 53

13.3.6 Key Pairs 54

13.3.7 Volumes 55

13.3.8 Snapshots 56

13.3.9 Network Topology 56

13.3.10 Network security groups 57

13.3.11 Admin panel 58

14. Conclusions 60

2. List of figures

Figure 1 Illinois Institute of technology logo 7
Figure 2 Elements that make up a computer system 11
Figure 3 Types of clouds according to the development 14
Figure 4 Types of clouds according to service model 15
Figure 5 Conceptual architecture of Openstack 24
Figure 6 Architecture Nodes 27
Figure 7: Keystone prerequisites 29
Figure 8: Keyston installation and configuration commands 30
Figure 9:Creating glance database 31
Figure 10Creating glance service credentials 31
Figure 11Creating glance service entity 32
Figure 12: Creating glance service API Endpoints 32
Figure 13Intsallation and configuration of glance service 33
Figure 14: Prerequisites of placement service 34
Figure 15 Creating placement service API endpoints 35
Figure 16 Installation and configuration of placement service 36
Figure 17 Creating compute service novas databases 37
Figure 18 Creating compute service credentials 37
Figure 19: Creating compute service API Endpoints 38
Figure 20 Installing Nova service packages 39
Figure 21 Configuration of Nova service 40
Figure 22 Population and register of Nova databases 40
Figure 23Creating Neutron database 41
Figure 24Creating neutron service API endpoints, register and credentials 42
Figure 25 Installing components for neutron option 1 43
Figure 26 Editing neutron config file for option 1 44
Figure 27 Configuring neutron ML2 plug-in 44
Figure 28 Configuring Neutron service Linux bridge agent 44
Figure 29 Configuring Neutron DHCP agent 45
Figure 30 Configuring Neutron metadata agent 45
Figure 31 Installing Horizon required packages 46
Figure 32 Configuration of Horizon service config file 46
Figure 33 Instruction to stablish connection to OpenStack cloud 49
Figure 34 OpenStack Log In page 49
Figure 35 OpenStack Overview Panel 50
Figure 36 OpenStack API Access 50
Figure 37 Instances management panel 51
Figure 38 How to create and configure and instance 52

Figure 39Management panels of a created instance in OpenStack 52
Figure 40 image managemnt in OpenStack 53
Figure 41 Keypairs in OpenStack 54
Figure 42 Volume configuration panel 55
Figure 43 Volumen overview panel 55
Figure 44: Snapshots panel 56
Figure 45 Cloud network topology example 57
Figure 46: Network security groups overview 57
Figure 47 Projects, users, groups and roles configuration panels 59

3. List of tables

Table 1 Cloud platforms compared 17
Table 2 Risk impact probability ratio 22

4. Introduction

To understand what we know today as cloud computing it is fundamental to know the origins of it. There are two widely used and researched computing paradigms that must be understood, such as clusters and networks. A cluster can be defined as a type of parallel system that consists of interconnecting individual computers so that they function as a single unit. A personal computer is considered a stand-alone computer as one that can operate independently, also called a node. Thus, a cluster is a composition of a group of nodes, which appear on the network as a single machine and function as such.

We can think of the network (TCP/IP) as a cluster specialization, with these cluster systems research centers and universities used to provide computing and storage services to third parties, through standard protocols, giving rise to what is now known as grid (clusters + networking) computing.

The next step after cluster and grids is virtualization. This new technology involves the virtual creation of resources, which can be operating systems, servers, storage devices, network resources, etc. This has many advantages, since it is possible to clone an environment without having to install and configure all the software required by the application. Virtualization and cloud computing have a lot in common, they are frequently used in the same context. Many private cloud-computing applications are made accessible through virtual machines, which all work together to keep a cloud infrastructure running. Cloud is virtualization but all virtualization is not cloud.

Cloud computing platforms have the characteristics of both technologies, clusters, and grids, but has its own additional advantages. Cloud computing is a model that offers private users, companies, and public organizations a new way of providing information processing services that, as we have said, are valid for any type of user. Cloud is all controlled via API over HTTP, that is what brings it all together. Cloud solutions and services offer new advantages that traditional IT systems cannot offer. Above all, we were able to highlight cost savings and easily expand the resources available to the company.

The goal of this project is to create a private cloud that can be customized to the needs of the costumer. Giving the users the benefits of improving speed during deployment (industry tells us that speed to deployment is the largest factor relating to cloud adoption), efficiency, cost reduction, data security, scalability, mobility and so much more.

5. Context

Every project arises for a reason, which in this case is the need to create a personalized private cloud in a simple and fast way. In this section we will present the context of the subject, i.e., the set of circumstances for which this study has been considered necessary.

5.1 2.1 Research Group Mystic

During the research project I have been working in collaboration with a research group called Mystic. The group consists of a NSF-funded testbed to study system re-configurability across the entire computing stack, from the processor to memory, storage, and the network. This system is aimed at enabling low-level experimentation and reconfiguration at the level of networks-on-chip (NoC), universal memory, and the network interconnect with multidimensional network topologies.

The reason to mention them in this section is to thank them and give them credit for all the help with the Openstack system design, install and configuration. The group is well-experienced on cloud network installations and especially on Openstack Open-Source software.

Figure 1 Illinois Institute of technology logo

6. Objectives and scope of work

The main objective of this research is the design and build of a custom private cloud. It is based on different points, which in turn define the scope of the work:

- **Research and choose the best way to create a private cloud:** Before starting any project, it is important to learn as much as possible about the technology and the state of the art. This will introduce the researcher on the topic to study, as well as give the bases to start the hands-on part of the project. The goal of this part it is going to be to decide the best way to create the private cloud, considering requirements, goals, existing service as well as some other factors.

- **Design and configure the private cloud:** After understanding the bases and deciding the way and requirements to build the cloud it is time to start working on it. This is the step to pass from the theory to the practice. The goal of this step is to end up having a fully functional private cloud. During this step it is fundamental to document every step taken, this can help in a future to recreate the project or to continue the research.

- **Test and verify in multiple ways the cloud designed:** The last step of the research it is going to be to test the cloud created in the previous step in different ways. This includes testing how system monitoring works, file storage, processing, overall user experience and so on.

7. Benefits of the research

There is no doubt that work is carried out with the objective of providing some benefit, and this is no different. This section analyzes the benefits from three different points of view: social, economic and technical.

7.1 Social benefits

As it has been previously mentioned, the purpose of this research project is to find a way to create a custom private cloud. In recent years, we have had to deal with the outsourcing and globalization of significant corporations. Their computational and data processing requirements have increased more quickly than personal computers' computing power. Nowadays technology has a huge role on how the society evolves and grows, that's the reason why this kind of projects are so important. Being able to create a cloud in a custom and easy way allows companies to keep growing without having infrastructure limits and focusing more on the development side. If a company evolves, the society does so too, offering new and advanced services and products to the end users. Taking all these into account we can say that this Project has a big social impact.

7.2 Economic benefits

As far as the economic benefits are concerned, different sections of this final master work help to give a greater profitability. Starting from the fact that the installation of this cloud is going to be fully customizable, the end user is going to be able to decide which services wants to install and which not, this way there are no unnecessary resources and therefore less cost. Because it is well known how expensive it can be maintaining servers (storage, compute power, infrastructure…). This will not only save space, but also, thanks to the proper use of the equipment, it could be used for a longer time.

7.3 Technical benefits

The technical benefits are those from which the main project will benefit, i.e. giving the end user a simple, fast and reliable way to create a custom cloud. This of course includes multiple of services such as, virtualization, storage, database management, monitoring, access management to resources. Any user that wants to have these kind of services is going to benefit from this research project.

8.State of the art

Recently, a new paradigm for hosting and providing services over the Internet called "cloud computing" has emerged. The ability to start small and add resources only when there is an increase in service demand makes cloud computing appealing to business owners because it removes the need for users to plan ahead for provisioning. This is a great way to provide user requirements in pay as you go principle and increase the resources only when it is necessary, when the service demand increases.

8.1 What is cloud computing?

Cloud computing is a new way of delivering services. But before defining cloud computing, it is necessary to know some concepts. A computer system is a set of interconnected parts that store and process information. The parts that make up a computer system are hardware, software and users.

- The hardware provides the computing resources of the system.
- Software can therefore be divided into operating systems and application software.
- Users are all administrators, developers, etc., as well as end users.

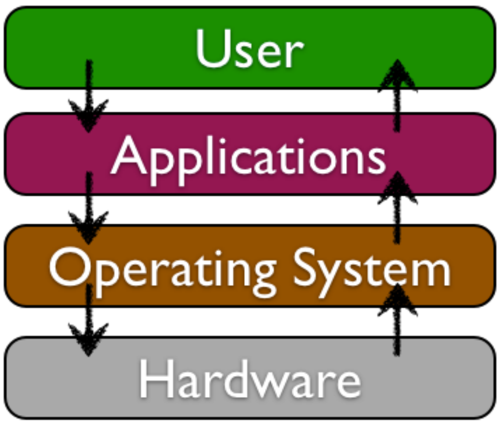


Figure 2 Elements that make up a computer system

According to the National Institute of Standards and Technology (NIST), cloud computing is defined as a technology model that provides pervasive, adaptive, on-demand network access to a pool of accessible computing resources. Shared configurations, such as networks and servers, can rapidly provision and release devices, applications and storage services with little administrative effort or interaction with service providers. A cloud is a type of parallel distributed system that consists of a group of interconnected devices virtualized in a way that represents a queue of machines or a uniform pool of computing resources. The characteristics are determined between the service provider and the consumer.

The main characteristics of cloud computing are:

- Pay-per-use service, which allows users to expand or reduce the resources they need quickly, automatically and without having to negotiate with service providers. This translates into lower costs, since you only pay for the services used.
- The ability to access resources over the network, either from standard mechanisms or from heterogeneous platforms such as mobile devices, laptops, tablets, etc.
- Scalable and flexible, resources can be released or acquired quickly on demand, increasing or decreasing automatically. It can be said that we can have unlimited resources.
- Metering service. With these systems, it is possible to automatically monitor and optimize the use of resources, such as bandwidth, storage and more.
- Resource pooling, which means that the provider's resources are shared so that some consumers can access them. This access will be according to the individualization of each consumer according to their request. Consumers do not know the actual location of the resource they are using, but can select a region, such as a country, city or data center.

8.2 Types of clouds

Existing cloud computing solutions can be categorized into several groups.

8.2.1 According to the development

One of the ways in which ratings can be made is based on privacy. Thus, there are four models for the development of cloud computing: private, public, community and hybrid clouds.

- Private Cloud: A single organization that manages and operates its cloud services, which can be the same service provider, contracting company or a third party. In the case of a third party, it will act according to the needs of the organization. This type of cloud is usually chosen when the information is important, it is necessary to focus computing resources and we want to obtain flexibility in its management. This is possible because the procurement solution adapts to the needs of the contracting company and information protection, security monitoring and surveillance can be implemented.
- Public Cloud: In this type of cloud, the location of the information is not known in advance. The service provider offers its resources to various entities free of charge. A public cloud should not be considered an insecure cloud. In this type of cloud, the service delivery time is short. In addition, the cost of using it is very low, if any.
- Community cloud: services are shared in a closed community of entities with the same objective, collaborating with each other. In this case, the cloud is managed and operated by one or more community entities.
- Mixed clouds: These clouds may include two or more of the cloud types discussed before: public, private or community. Some services are publicly available, such as development tools, and others are private, such as infrastructure. An entity that opts for these solutions alone can take advantage of the benefits that the rest of the cloud has. For example, there is a lot of flexibility in purchasing services, but also the ability to control them.

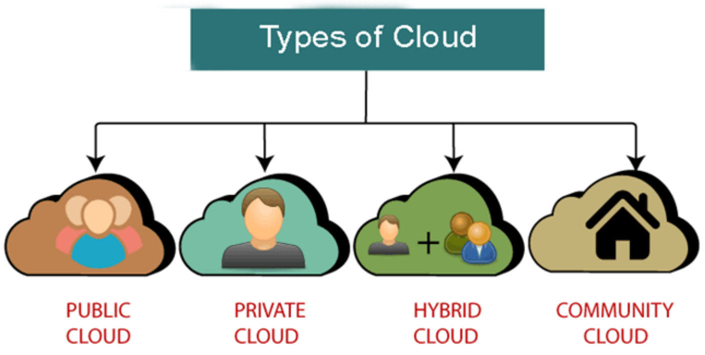


Figure 3 Types of clouds according to the development

8.2.2 According to service model

Analyzing the proposed service model, it is possible to notice that there are three families, today called XaaS, which means everything as a service, or simply a service. Although the solutions offered today are generally mixed, i.e., combining several service models, the following classification can be made to better understand the concepts.

- Software as a Service (SaaS): In this group what is provided are end products as a service, so that the entity can use these applications to develop its own operations. The applications can be accessed from different devices and several clients can view them at the same time. The client receives the necessary licenses to use the requested software. In addition, this can be done in two ways: the provider has applications on its server and the customer accesses them through a browser or application installed on the customer's system and the licenses expire at the end of the contract period.

- Platform as a Service (PaaS): this service model consists of providing the user with tools with which they can develop, test, deploy, maintain and host their IT applications without installing them on their local clusters. The great advantage of this model is that users do not have to pay for tool licenses and get free maintenance and updates.

- Infrastructure as a Service (IaaS): provides storage and computing resources that users will use to develop their own software. In other words, customers are provided with the use of IT infrastructure as a service. This method was chosen as an alternative to avoid having to occupy all the servers, storage space and network equipment necessary to carry out the customer's operations.

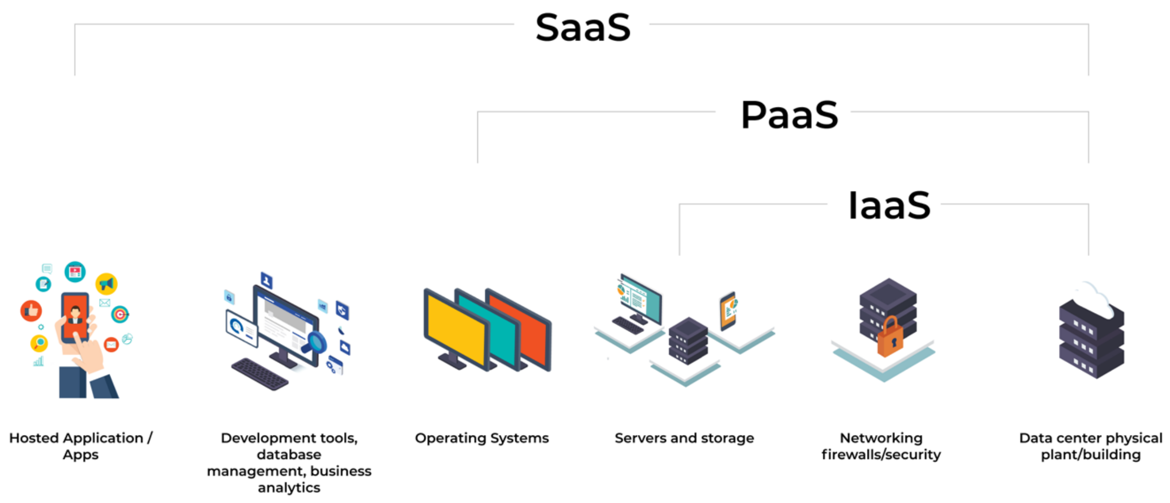

SaaS
PaaS
IaaS
Hosted Application / Apps
Development tools, database management, business analytics
Operating Systems
Servers and storage
Networking firewalls/security
Data center physical plant/building

Figure 4 Types of clouds according to service model

9. Analysis of alternatives

For the elaboration and development of this final master work, several resources are used. In this section the different alternatives are presented, analyzed and the decision taken is argued. The points to be analyzed are hardware used for the different servers, software used for the management of the virtual cloud and finally, the working method.

9.1 Platform

The operation of the system used throughout the project is explained below. The characteristics and operation of each of the processes are explained, however, the computer oversees performing these operations. For this reason, it is necessary to create files called scripts, which are nothing more than documents containing instructions written in programming code. These scripts are going to form what we call the "platform of the cloud". Here are some of the most popular cloud platforms:

- **Openstack**: OpenStack is an open-source cloud platform that manages distributed compute, network and storage resources, aggregates them into pools, and allows on-demand provisioning of virtual resources through a self-service portal. OpenStack is a cost-effective extension of the existing public cloud infrastructure, enabling organizations to optimize their cloud maintenance costs and service providers to build an infrastructure competitive to hyperscalers.

- **VMware:** VMware Cloud delivers a new model of cloud operations. Create a more efficient and automated data center, along with hybrid operations that leverage the same tools, processes, skills and teams to multiple public clouds and the edge.

- **Ganeti:** Open source software for managing clusters of virtual servers from Google. Ganeti provides OS installation for virtual machine instances as well as startup, shutdown and failover of servers running Xen or KVM virtualization software.

The following table shows a comparison between some of the main features of the mentioned platforms.

	VMware	OpenStack	Ganeti
Hypervisor	ESXi	Many hypervisor (KVM, LXC, ESXi, Hyper-V…)	KVM, Xen
Customer and Operations access	Windows client, VCloud Director	OpenStack native dashboard, 3rd parties dashboards, CLI	Ganeti Web manager, Synnefo, CLI
Storage	VMFSover SAN and iSCSI	Default non-persistent image. Pluggable Cinder: block volumes, Ceph, several vendor SAN	Native: local disks, DRBD, C
Eph. Pluggable external storage	Traditional switching infrastructure, SDN with additional products	Traditional switching and Software Defined Network	Traditional switching, pluggable extensions to Software Defined Network

Table 1 Cloud platforms compared

One thing to keep in mind when choosing the platform is that it should not be the one that limits the project, i.e., that it should offer the necessary tools to be able to carry out tasks both in the present and in the future. The features shown in the table are some of the most used tasks involving decision making. After carefully reviewing all the features OpenStack has been the chosen platform, the wide compatibility with third party software-hardware and the huge community on the internet behind the project have been the key defining factors. In addition, projects in the same field have been carried out in the university laboratory with the OpenStack platform, so learning may become easier.

9.2 Hardware

The hardware is a key element of this entire project, it is going to affect the design as well as the final performance of the cloud. There are multiple elements that affect the decision of the hardware:

- **Compatibility:** In the previous section the platform that is going to be used has been chosen, therefore the hardware that is going to be used must be compatible with it.

- **<u>Performance:</u>** The project has some goals that want to be achieved, this involves speed, features, power consumption, future proofing and so much more.

- **<u>Budget and availability:</u>** Everyone would wish to have unlimited budget to carry out a project but sadly in most cases this does not happen. That's why it is important to take in to account the resources that are available.

In the case of this research project the platform chosen has been OpenStack which is widely used and therefore, compatible with most of the common hardware. When it comes to performance, this project is the initial phase to create a private cloud which means that in order to do basic tests it is not required a lot of compute power or storage. Finally, thanks to the Illinois Institute of Technology 4 servers have been provided to carry out the research, these machines were already built and ready to use at the facilities of the university.

9.3 Working method

Especially due to the situation in which the world has found itself in recent years (at the time of this work), i.e., the confinement due to the Covid-19 virus, it is necessary to establish alternative ways of working. The chosen method directly affects the work in different aspects: time, performance, accessibility... Fortunately, the situation has not prevented the project from being carried out and different possibilities have been analyzed:

- <u>Local storage and processing:</u> This option is unfeasible, because having all the hardware and infrastructure to maintain the equipment in a house is not common. The main advantage of this would be not to depend on a remote device, which in the end can cause problems, either by network or otherwise.
- <u>Remote storage and processing:</u> There is no doubt that in situations like the one described above it is one of the best options, since it allows you to use devices remotely. It is not necessary to have the necessary hardware. Especially useful in this case being professional level hardware.

Analyzing the different possibilities, the final decision was to use the last of the methods, i.e., remote storage and processing. The main reason is the simplicity of the method, since even from a device as common as a cell phone it is possible to connect to the servers remotely. It is only necessary to have an Internet connection, and it doesn't even have to be very fast, since for most of the tasks, commands or text files are shared.

10. Risk Analysis

Being aware of the possible risks that a project may have, as well as knowing their origin and possible solution is of vital importance. In the unfortunate event that one of these risks occurs, it could cause delays in time. However, if the response to each risk is established beforehand, thus creating a contingency plan, the effects of the risks could be reduced. Therefore, throughout this section we first identify the potential risks. Second, the possibilities of their occurrence and their possible impact on the project are analyzed. Third and finally, an action plan is prepared.

10.1 Possible risks

These are the possible risks that may occur:

- **<u>Server/Hardware errors:</u>** There is no doubt that these tools are essential to carry out the work and it may happen that they are out of service due to a hardware or software error. In these cases, it would not be possible to make use of them and therefore there would be delays. In addition, it is sometimes necessary to go to the university to solve these problems, and if the university is closed, as was the case during the confinement of the Covid-19 virus, the problem increases.

- **<u>Corrupted data:</u>** As with any other computer, it can happen that files become corrupted, for example, due to a sudden power outage while performing some tasks. This is not the only case that can cause data corruption, so it is important to take this into account.

- **<u>Infrastructure problems:</u>** All the hardware required for the project is housed on a building of the Illinois Institute of Technology. Something that could happen is a power shortage in the area or the ISP (Internet Service Provider) cannot ensure all the time the network connection.

10.2 Risk Impact

Each of the potential risks mentioned in the previous point has a different impact. Each is discussed in detail in the following sections:

- **Errors in the servers/hardware:** In this type of devices, checks are periodically performed in order to prevent this type of problems. And as for the case mentioned, a global pandemic, it is an isolated case and the probability of it happening is minimal. However, if it were to occur, the impact would be high and serious, since these devices contain all the information of the projects carried out by the research group. It is important to mention though that the cloud paradigm expects hardware failure so it is important to plan for it. This means having spare disks, motherboards and other components.

- **Corrupted data:** The probability of this occurring is small, and its impact could be considered medium. The fact that one of the databases is corrupt means two things: either it must be obtained again or if this is not possible, a substitute must be found. In either case, extra time will be needed to fix it. Here the called MTTR (Mean Time To Recovery/Restore) can be calculated for example when a fresh install of the OS is required. In the case of Ubuntu Server the average installation time is 10-15 minutes.

- **Infrastructure problems:** Most of the buildings at the IIT are well prepared for power shortages and they have emergency systems to keep the servers working for a short time until the services are up again. If it occurs repeatedly, it can be a serious problem, but it is usually a minor setback.

Based on the analysis, the following table shows the impact of the possible risks:

		Impact				
		5%	**10%**	**20%**	**40%**	**80%**
Probability	**Not common 10%**	🟩	🟩	🟩 Corrupted data (0,02)	🟧	🟧 Server/hardware error (0,08)
	Difficult 30%	🟩	🟩	🟧	🟧	🟥
	There is a chance 50%	🟩	🟧	🟧	🟥	🟥
	Could happen 70%	🟩 Infrastructure problems (0,035)	🟧	🟧	🟥	🟥
	Most likely 90%	🟧	🟧	🟥	🟥	🟥

Table 2 Risk impact probability ratio

10.3 Contingency plan

Finally, as mentioned above, once the possibilities and impact of each of the risks have been analyzed, it is time to establish a contingency plan. The idea is to establish guidelines to be followed in the event of any of the above-mentioned risks occurring. To carry out this task, three criteria have been established: accept the risk, control the risk and reduce the risk. In the case of this project:

- **Accept the risk:** Given this situation, it would be decided how to deal with it without changing the initial project plan. This group includes the case of infrastructure problems, mainly due to their low impact. If this were to happen, we would look for another task to perform while the power or the internet is up again. It could be a good time to document the results obtained so far, for example.

- **Control the risk:** As before, the risk is accepted, however, a method to control it is established. This includes the risk of server errors. When one of these errors is detected, it must be notified immediately so that the person in charge can solve it as soon as possible. As for the continuation of the work, the scripts with which you work locally must be available beforehand, since these files are not heavy, and it is possible to have them available without the servers. In this way it is possible to work on the code while the bugs are being fixed.

- **Reduce risk:** In certain cases, it is possible to do something to reduce the impact of risks. This is the case with corrupted data. The idea here is to activate maintenance software that takes care of checking the status and integrity of the data on a regular basis. In this way the status of equipment and content is continuously monitored.

11. Description of the system: Design

This section explains more in detail and technically how the previously described cloud is going to be created. The description includes hardware, networking as well as software design.

11.1 Services

In the diagram below, the connections between the OpenStack services are depicted:

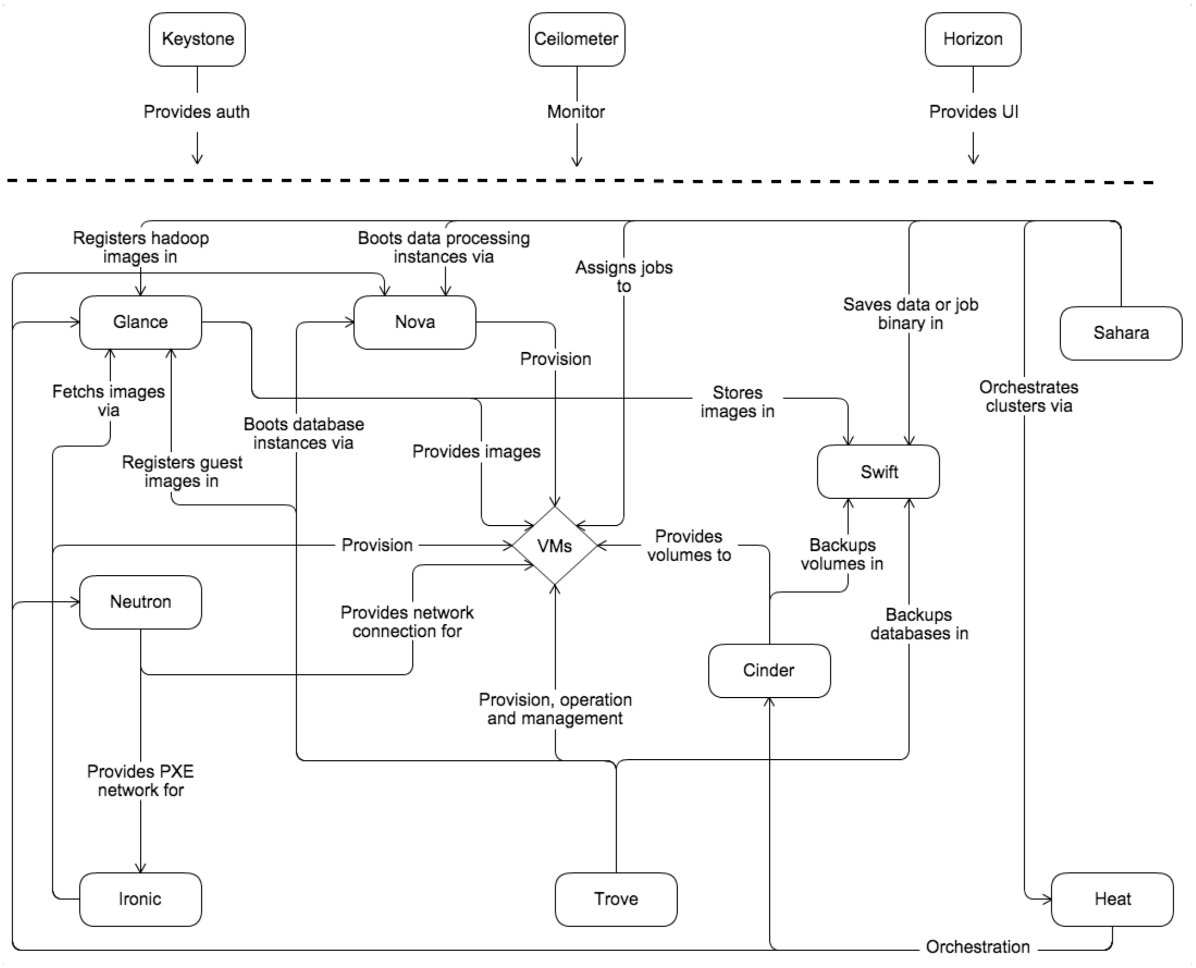


Figure 5 Conceptual architecture of Openstack

As you can see in the figure above, the architecture consists of multiple services working together. The OpenStack services are a collection of separate components that make up OpenStack. Through a common Identity service, all services authenticate one another. Public APIs are used by individual services to communicate with one another, unless privileged administrator commands are required.

An AMQP message broker is used to facilitate communication between the processes of a single service. The state of the service is kept in a database. There are a variety of message broker and database options when deploying and setting up your OpenStack cloud, including RabbitMQ, MySQL, MariaDB, and SQLite. During this project MySQL has been the one chosen, mainly because of the previous experience the author of this thesis has with it. Users can access OpenStack via the web-based user interface implemented by the Horizon Dashboard, via command-line clients and by issuing API requests through tools like browser plug-ins or curl. There are numerous SDKs available for applications. All these access techniques ultimately involve making REST API requests to the various OpenStack services.

The main and minimum required services to have the cloud up and working are keystone, glance, nova, placement, cinder, neutron, and horizon.

- **Keystone:** Keystone is an OpenStack service that provides API client authentication, service discovery, and distributed multi-tenant authorization by implementing OpenStack's Identity API. It supports LDAP, OAuth, OpenID Connect, SAML and SQL.

- **Glance:** Users can upload and find data assets intended for use with other services through the Image service (glance) project's service. Images and metadata definitions are currently included in this.

- **Nova:** Nova offers a method for provisioning compute instances (aka virtual servers). Nova has limited support for system containers and also supports the creation of virtual machines and bare metal servers (using ironic). To offer that service, Nova runs as a group of daemons on top of current Linux servers.

- **Placement:** The placement service provides an HTTP API used to track resource provider inventories and usages. Placement uses a data model and functions as a web service. The installation process entails building the required database, as well as setting up and configuring the web service.

- **Cinder:** Cinder is a Block Storage service for OpenStack. It's intended to provide end customers with storage resources that the OpenStack Compute Project can utilize. A reference implementation (LVM) or plugin drivers for other storage are used to do this. Cinder, in a nutshell, virtualizes the management of block storage devices and gives end users access to a self-service API that allows them to request and use those resources without having to know where or what kind of device their storage is deployed on.

- **Neutron:** Neutron is an OpenStack project to provide "networking as a service" between interface devices (e.g., vNICs) managed by other Openstack services (e.g., nova).

- **Horizon:** Horizon is the canonical implementation of OpenStack's Dashboard, which provides a web-based user interface to OpenStack services including Nova, Swift, Keystone, etc.

11.3 Architecture Nodes

All the services mentioned above are in charge of performing different tasks, therefore they must be installed in the appropriate machine with all the required hardware. In order to make the architecture simple and organized, different nodes are created. Two nodes (hosts) to launch a basic virtual machine or instance. Optional services such as Block Storage and Object Storage require additional nodes.

In the next figure you can see how the responsibilities are divided between nodes. In this section each of the nodes will be described as well.

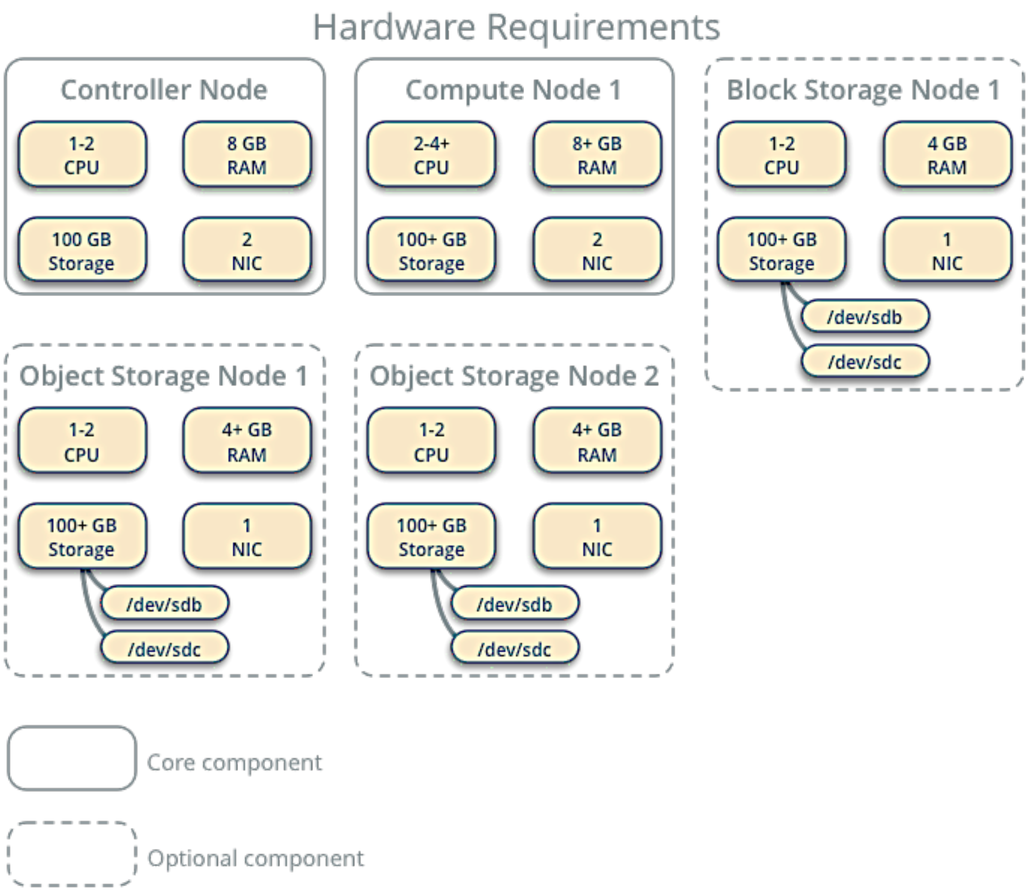


Figure 6 Architecture Nodes

- **<u>Controller node:</u>** The Identity service, Image service, Placement service, management components of Compute, management components of Networking, numerous Networking agents, and the Dashboard are all operated by the controller node. Supporting services including a SQL database, message queue, and NTP are also included. The controller node may furthermore run a portion of the Telemetry, Orchestration, Block Storage, and Object Storage services. Two network interfaces are a minimum requirement for the controller node.

- **<u>Compute node:</u>** The hypervisor portion of Compute, which manages instances, is run on the compute node. Compute use the KVM hypervisor by default. A networking service agent that connects instances to virtual networks and offers firewalling services to instances via security groups is also running on the compute node. More than one compute node can be deployed. A minimum of two network interfaces are needed for each node.

- **Block Storage node:** The disks that the Block Storage and Shared File System services provision for instances are located on the optional Block Storage node. A separate storage network should be used in production scenarios to improve performance and security. More than one block storage node can be deployed. A minimum of one network interface is needed for each node.

- **Object Storage node:** The disks used by the Object Storage service to store accounts, containers, and objects are located in the optional Object Storage node. A separate storage network should be used in production scenarios to improve performance and security. For this service, two nodes are needed. A minimum of one network interface is needed for each node. More than two object storage nodes can be deployed.

12. Installation and configuration process

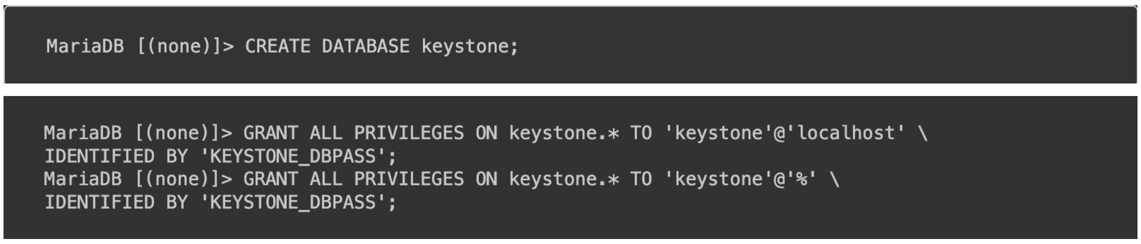
The goal of this section is to explain the procedure followed to install all the required services. This is going to be the installation process for the lates version of Openstack, Yoga version. As mentioned in previous sections, the installation will be done in Ubuntu 20.04 LTS version. This procedure is going to be done installing all the required services.

12.1 Identity service. Keystone

The OpenStack Identity service provides a single point of integration for managing authentication, authorization, and a catalog of services. The Identity service is typically the first service a user interacts with. The Identity service is also used by other OpenStack services to confirm that users are who they claim to be and locate other services within the deployment. Additionally, the Identity service is compatible with some third-party user management programs (such as LDAP). The Kystone service will be installed in the controller node.

12.1.1 Prerequisites

There are some conditions that are required before beginning the installation of Keystone. In this case, it is required to have a Database with the right root permissions.



```
MariaDB [(none)]> CREATE DATABASE keystone;
```

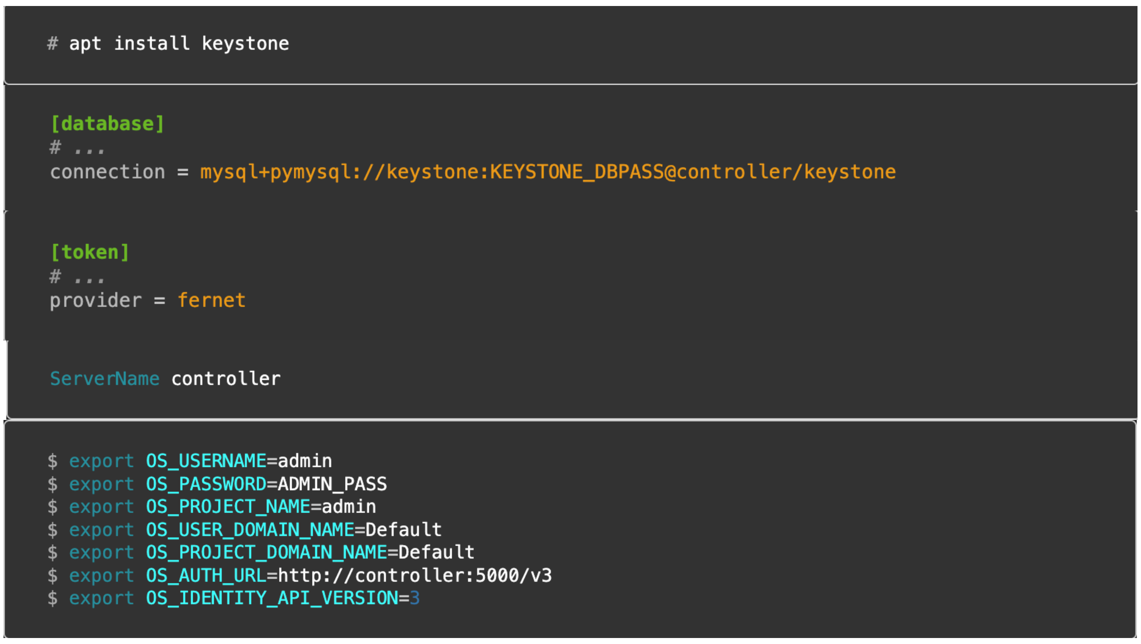
```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost' \
IDENTIFIED BY 'KEYSTONE_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%' \
IDENTIFIED BY 'KEYSTONE_DBPASS';
```

Figure 7: Keystone prerequisites

12.1.2 Installation and configuration

These are the next steps to install and configure the Identity service:

- Install the required packages.
- Edit the keystone config file to link the database from the prerequisites to the keystone service as well as setting up fernet as the token provider.
- Configure the Apache HTTP Server.
- Configure administrative accounts using environmental values.



```
# apt install keystone
```

```
[database]
# ...
connection = mysql+pymysql://keystone:KEYSTONE_DBPASS@controller/keystone
```

```
[token]
# ...
provider = fernet
```

```
ServerName controller
```

```
$ export OS_USERNAME=admin
$ export OS_PASSWORD=ADMIN_PASS
$ export OS_PROJECT_NAME=admin
$ export OS_USER_DOMAIN_NAME=Default
$ export OS_PROJECT_DOMAIN_NAME=Default
$ export OS_AUTH_URL=http://controller:5000/v3
$ export OS_IDENTITY_API_VERSION=3
```

Figure 8: Keyston installation and configuration commands

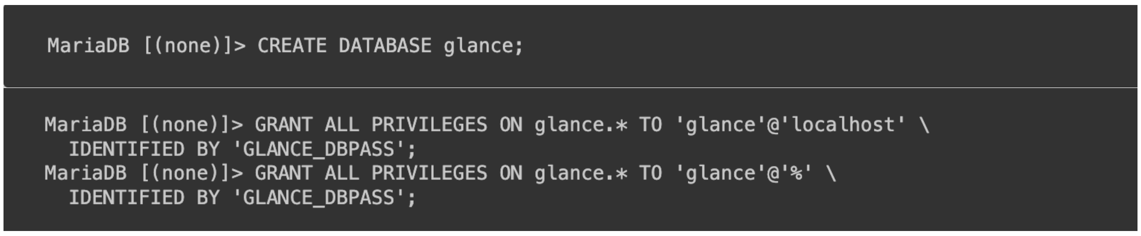
12.2 Image service. Glance

The Image service (glance) enables users to discover, register, and retrieve virtual machine images. It offers a REST API that enables you to query virtual machine image metadata and retrieve an actual image. As previously mentioned, Glance image service is going to be installed in the controller node.

12.2.1 Prerequisites

The requirements before the installation are more than the ones for the identity service. It is important to make sure all of them are fulfilled:

- It is necessary to have a database created with all the appropriate permissions.

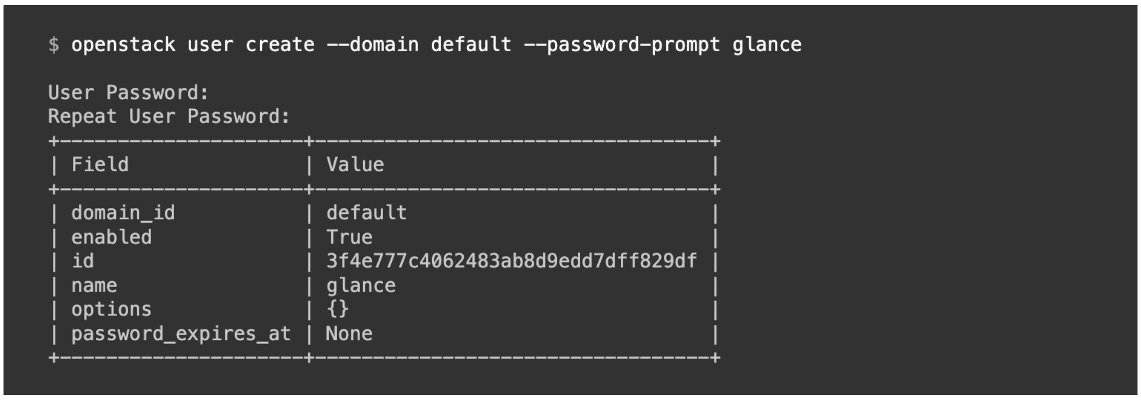


```
MariaDB [(none)]> CREATE DATABASE glance;
```

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost' \
  IDENTIFIED BY 'GLANCE_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'%' \
  IDENTIFIED BY 'GLANCE_DBPASS';
```

Figure 9:Creating glance database

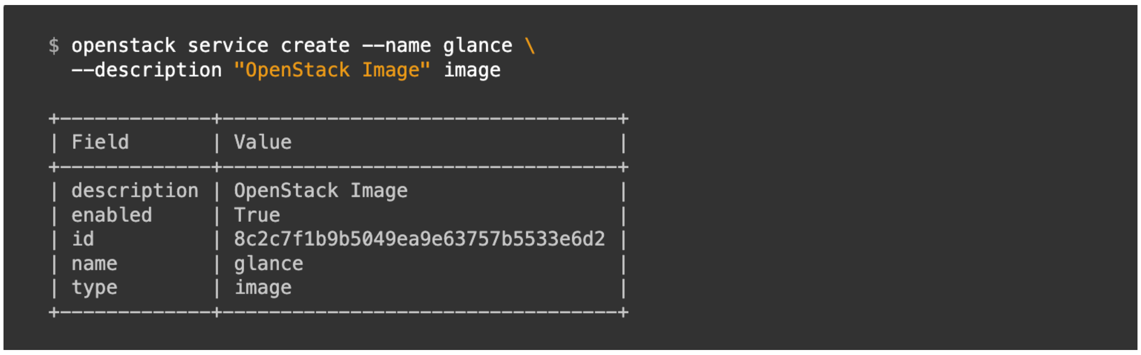
- Create service (glance) credentials.



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

User Password:
Repeat User Password:
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| domain_id           | default                          |
| enabled             | True                             |
| id                  | 3f4e777c4062483ab8d9edd7dff829df |
| name                | glance                           |
| options             | {}                               |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

Figure 10Creating glance service credentials

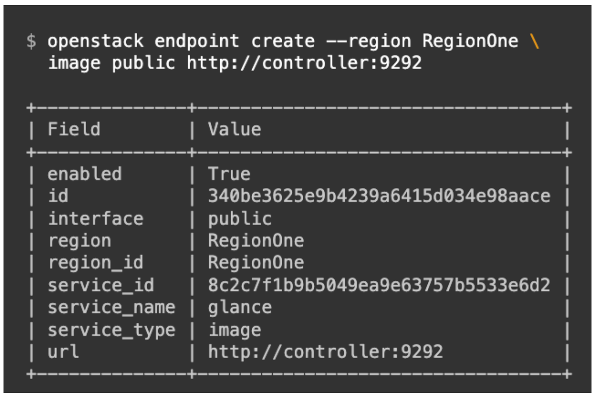
- Create the glance service entity.



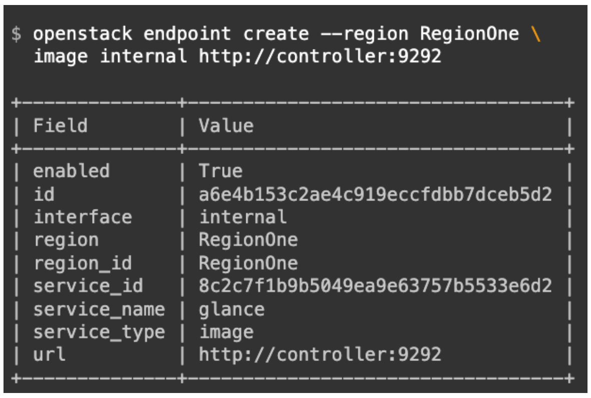
```
$ openstack service create --name glance \
  --description "OpenStack Image" image

+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Image                  |
| enabled     | True                             |
| id          | 8c2c7f1b9b5049ea9e63757b5533e6d2 |
| name        | glance                           |
| type        | image                            |
+-------------+----------------------------------+
```

Figure 11Creating glance service entity

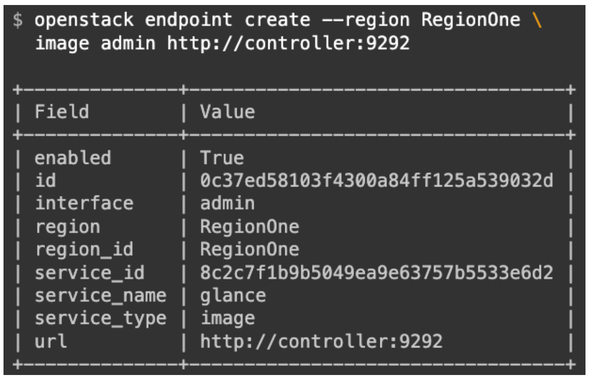
- Create the image service API endpoints.



```
$ openstack endpoint create --region RegionOne \
  image public http://controller:9292

+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 340be3625e9b4239a6415d034e98aace |
| interface    | public                           |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 8c2c7f1b9b5049ea9e63757b5533e6d2 |
| service_name | glance                           |
| service_type | image                            |
| url          | http://controller:9292           |
+--------------+----------------------------------+
```



```
$ openstack endpoint create --region RegionOne \
  image internal http://controller:9292

+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | a6e4b153c2ae4c919eccfdbb7dceb5d2 |
| interface    | internal                         |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 8c2c7f1b9b5049ea9e63757b5533e6d2 |
| service_name | glance                           |
| service_type | image                            |
| url          | http://controller:9292           |
+--------------+----------------------------------+
```



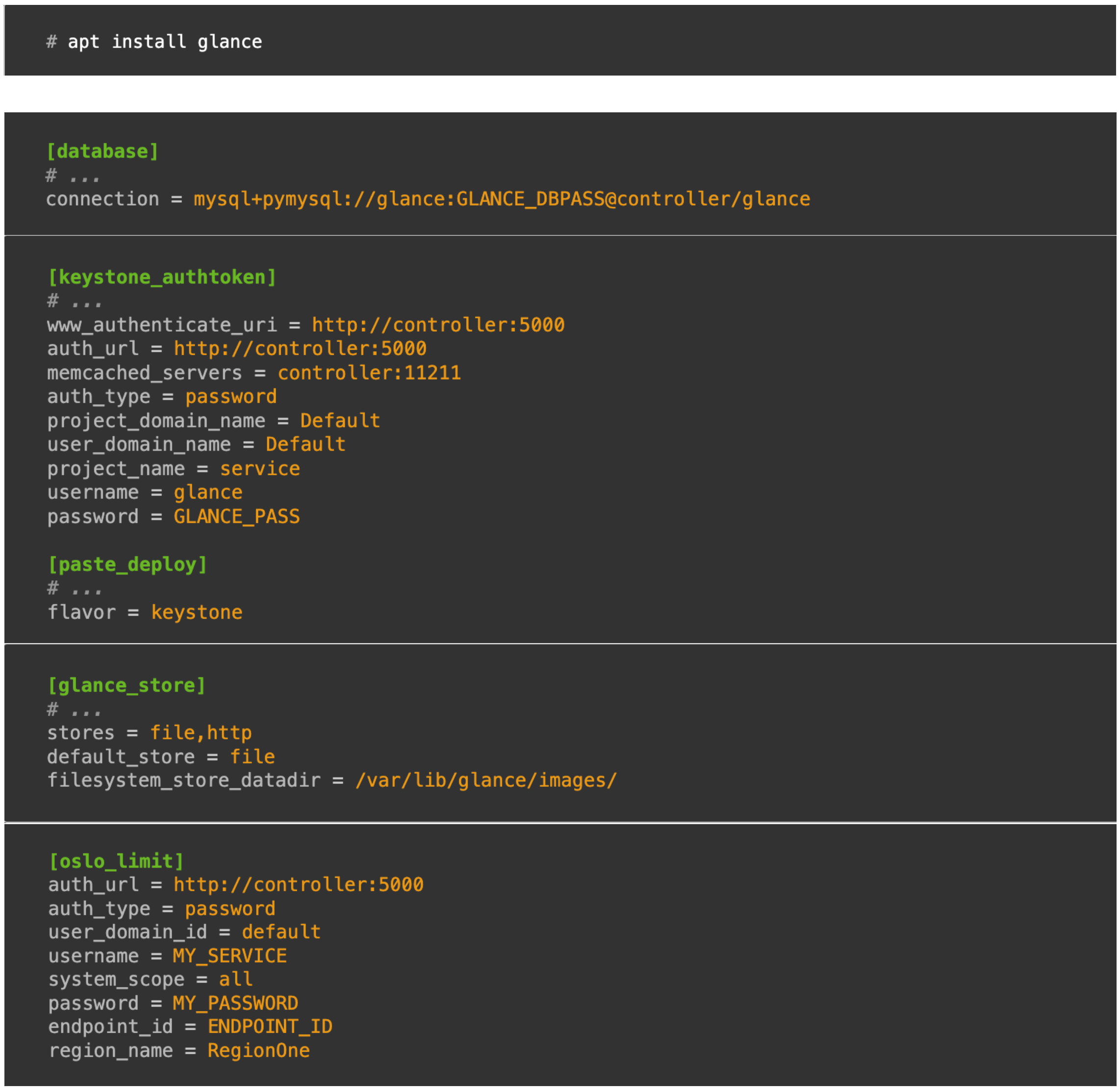
```
$ openstack endpoint create --region RegionOne \
  image admin http://controller:9292

+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 0c37ed58103f4300a84ff125a539032d |
| interface    | admin                            |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 8c2c7f1b9b5049ea9e63757b5533e6d2 |
| service_name | glance                           |
| service_type | image                            |
| url          | http://controller:9292           |
+--------------+----------------------------------+
```

Figure 12: Creating glance service API Endpoints

12.2.2 Installation and configuration

These are the next steps to install and configure the Image service:

- Install the packages.
- Edit the glance config file to link the database from the prerequisites to the placement service.



```
# apt install glance
```

```
[database]
# ...
connection = mysql+pymysql://glance:GLANCE_DBPASS@controller/glance
```

```
[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 = glance
password = GLANCE_PASS

[paste_deploy]
# ...
flavor = keystone
```

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

```
[oslo_limit]
auth_url = http://controller:5000
auth_type = password
user_domain_id = default
username = MY_SERVICE
system_scope = all
password = MY_PASSWORD
endpoint_id = ENDPOINT_ID
region_name = RegionOne
```

Figure 13Intsallation and configuration of glance service

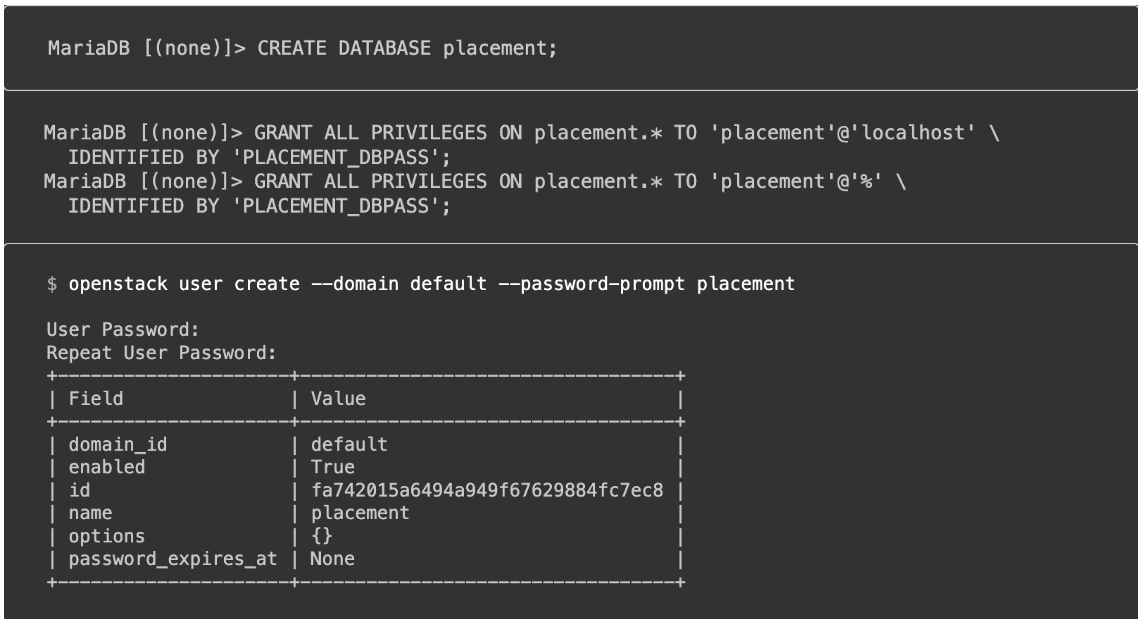
12.3 Placement service

The placement service provides an HTTP API used to track resource provider inventories and usages. Placement uses a data model and functions as a web service. The installation process entails building the required database, as well as setting up and configuring the web service.

12.3.1 Prerequisites

As always, some there are some prerequisites that are necessary to have everything working fine.

- It is necessary to have a database created with all the appropriate permissions.
- Configure User and endpoints. This includes creating a Placement service user, create and API entry for the service and create the API service endpoints.

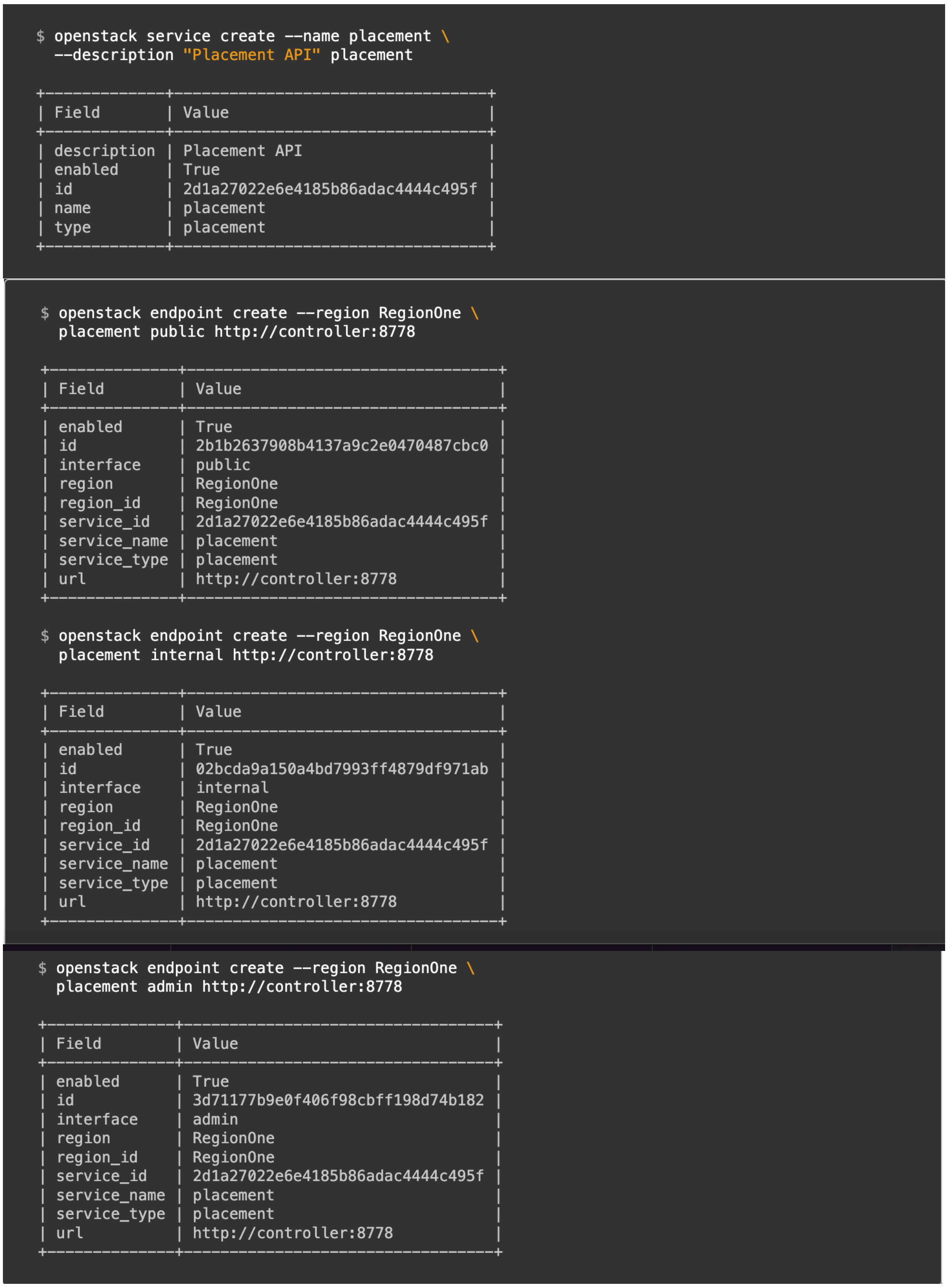


```
MariaDB [(none)]> CREATE DATABASE placement;
```

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON placement.* TO 'placement'@'localhost' \
  IDENTIFIED BY 'PLACEMENT_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON placement.* TO 'placement'@'%' \
  IDENTIFIED BY 'PLACEMENT_DBPASS';
```

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

User Password:
Repeat User Password:
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| domain_id           | default                          |
| enabled             | True                             |
| id                  | fa742015a6494a949f67629884fc7ec8 |
| name                | placement                        |
| options             | {}                               |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

Figure 14: Prerequisites of placement service



```
$ openstack service create --name placement \
  --description "Placement API" placement

+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Placement API                    |
| enabled     | True                             |
| id          | 2d1a27022e6e4185b86adac4444c495f |
| name        | placement                        |
| type        | placement                        |
+-------------+----------------------------------+
```

```
$ openstack endpoint create --region RegionOne \
  placement public http://controller:8778

+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 2b1b2637908b4137a9c2e0470487cbc0 |
| interface    | public                           |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 2d1a27022e6e4185b86adac4444c495f |
| service_name | placement                        |
| service_type | placement                        |
| url          | http://controller:8778           |
+--------------+----------------------------------+

$ openstack endpoint create --region RegionOne \
  placement internal http://controller:8778

+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 02bcda9a150a4bd7993ff4879df971ab |
| interface    | internal                         |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 2d1a27022e6e4185b86adac4444c495f |
| service_name | placement                        |
| service_type | placement                        |
| url          | http://controller:8778           |
+--------------+----------------------------------+
```

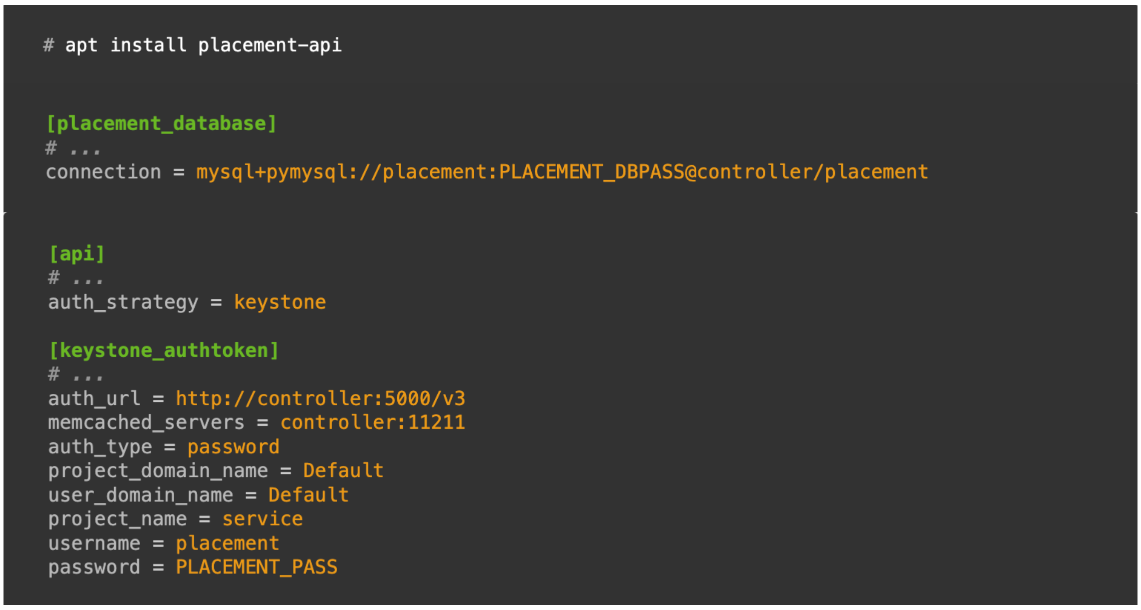
```
$ openstack endpoint create --region RegionOne \
  placement admin http://controller:8778

+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 3d71177b9e0f406f98cbff198d74b182 |
| interface    | admin                            |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 2d1a27022e6e4185b86adac4444c495f |
| service_name | placement                        |
| service_type | placement                        |
| url          | http://controller:8778           |
+--------------+----------------------------------+
```

Figure 15 Creating placement service API endpoints

12.3.2 Installation and configuration

These are the next steps to install and configure the placement service:

- Install the packages.
- Edit the placement config file to link the database from the prerequisites to the placement service.



```
# apt install placement-api


[placement_database]
# ...
connection = mysql+pymysql://placement:PLACEMENT_DBPASS@controller/placement

[api]
# ...
auth_strategy = keystone

[keystone_authtoken]
# ...
auth_url = http://controller:5000/v3
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = placement
password = PLACEMENT_PASS
```

Figure 16 Installation and configuration of placement service

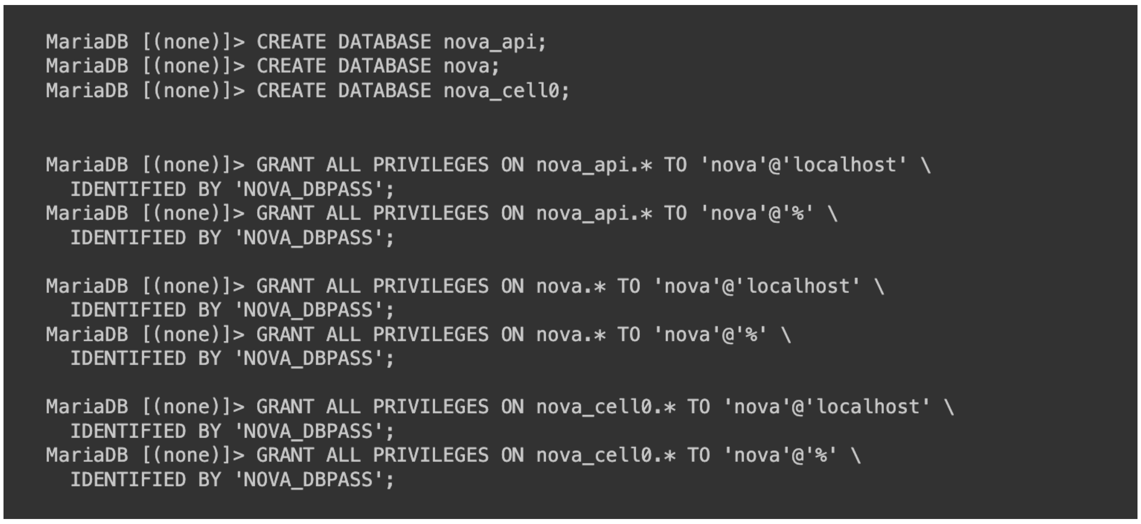
12.4 Compute service. Nova

This service helps to implement services and associated libraries to provide massively scalable, on demand, self-service access to compute resources, including bare metal, virtual machines, and containers.

12.4.1 Prerequisites

Once again, before installing a service it is fundamental to check that all the requirements are met.

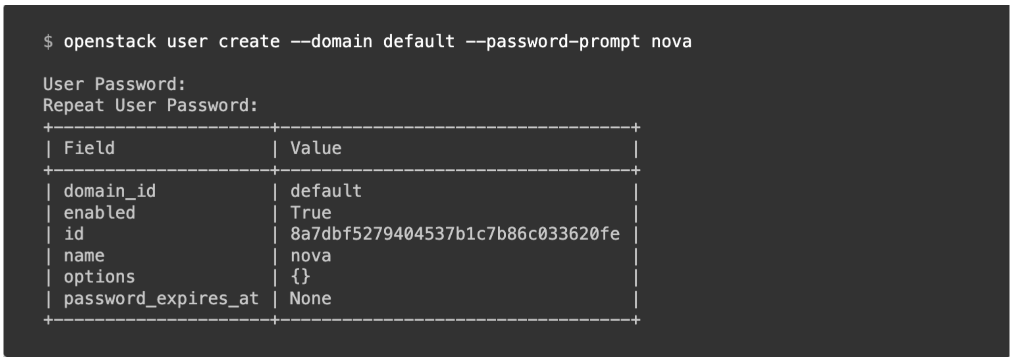
- Create three databases and make sure all of them have the right permissions.



```
MariaDB [(none)]> CREATE DATABASE nova_api;
MariaDB [(none)]> CREATE DATABASE nova;
MariaDB [(none)]> CREATE DATABASE nova_cell0;


MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'localhost' \
  IDENTIFIED BY 'NOVA_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'%' \
  IDENTIFIED BY 'NOVA_DBPASS';

MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' \
  IDENTIFIED BY 'NOVA_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%' \
  IDENTIFIED BY 'NOVA_DBPASS';

MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_cell0.* TO 'nova'@'localhost' \
  IDENTIFIED BY 'NOVA_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_cell0.* TO 'nova'@'%' \
  IDENTIFIED BY 'NOVA_DBPASS';
```

Figure 17 Creating compute service novas databases

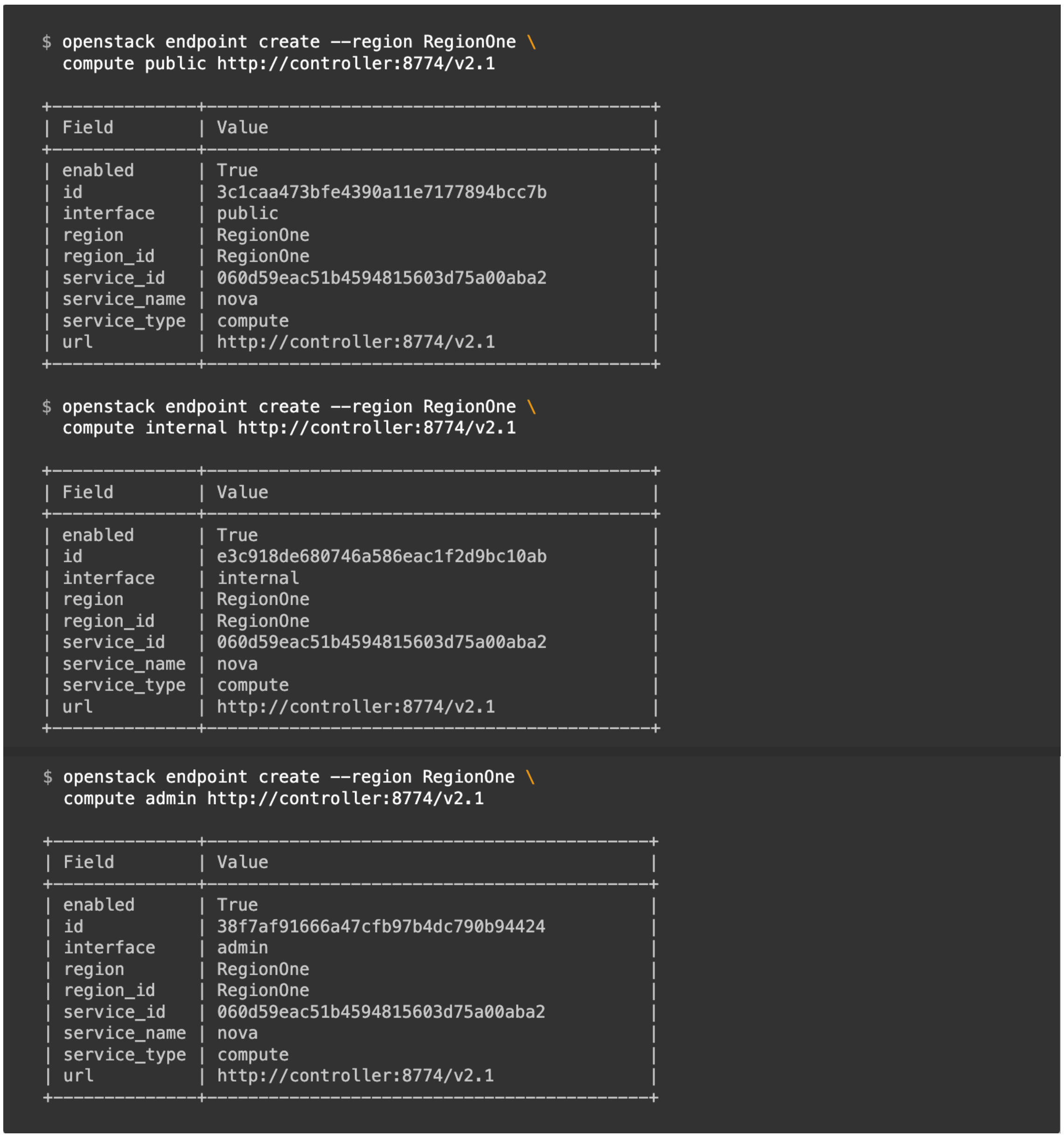
- Create the compute service credentials.



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

User Password:
Repeat User Password:
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| domain_id           | default                          |
| enabled             | True                             |
| id                  | 8a7dbf5279404537b1c7b86c033620fe |
| name                | nova                             |
| options             | {}                               |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

Figure 18 Creating compute service credentials

- Create the compute service API endpoints.



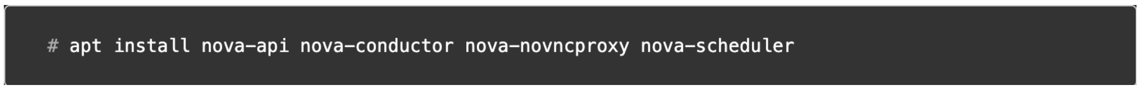
```
$ openstack endpoint create --region RegionOne \
  compute public http://controller:8774/v2.1

+--------------+-------------------------------------------+
| Field        | Value                                     |
+--------------+-------------------------------------------+
| enabled      | True                                      |
| id           | 3c1caa473bfe4390a11e7177894bcc7b          |
| interface    | public                                    |
| region       | RegionOne                                 |
| region_id    | RegionOne                                 |
| service_id   | 060d59eac51b4594815603d75a00aba2          |
| service_name | nova                                      |
| service_type | compute                                   |
| url          | http://controller:8774/v2.1               |
+--------------+-------------------------------------------+

$ openstack endpoint create --region RegionOne \
  compute internal http://controller:8774/v2.1

+--------------+-------------------------------------------+
| Field        | Value                                     |
+--------------+-------------------------------------------+
| enabled      | True                                      |
| id           | e3c918de680746a586eac1f2d9bc10ab          |
| interface    | internal                                  |
| region       | RegionOne                                 |
| region_id    | RegionOne                                 |
| service_id   | 060d59eac51b4594815603d75a00aba2          |
| service_name | nova                                      |
| service_type | compute                                   |
| url          | http://controller:8774/v2.1               |
+--------------+-------------------------------------------+

$ openstack endpoint create --region RegionOne \
  compute admin http://controller:8774/v2.1

+--------------+-------------------------------------------+
| Field        | Value                                     |
+--------------+-------------------------------------------+
| enabled      | True                                      |
| id           | 38f7af91666a47cfb97b4dc790b94424          |
| interface    | admin                                     |
| region       | RegionOne                                 |
| region_id    | RegionOne                                 |
| service_id   | 060d59eac51b4594815603d75a00aba2          |
| service_name | nova                                      |
| service_type | compute                                   |
| url          | http://controller:8774/v2.1               |
+--------------+-------------------------------------------+
```

Figure 19: Creating compute service API Endpoints

12.4.2 Installation and configuration

These are the next steps to install and configure the Nova service:

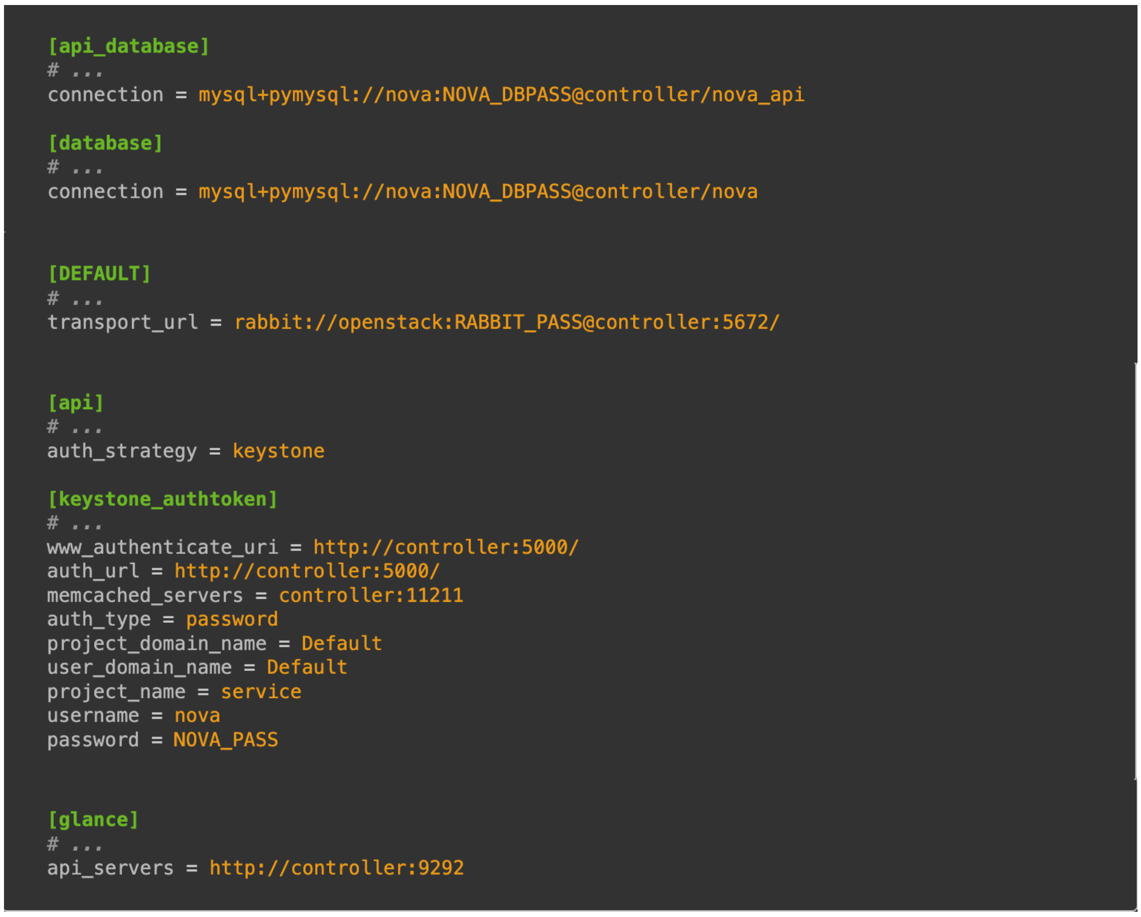
- Install the packages.



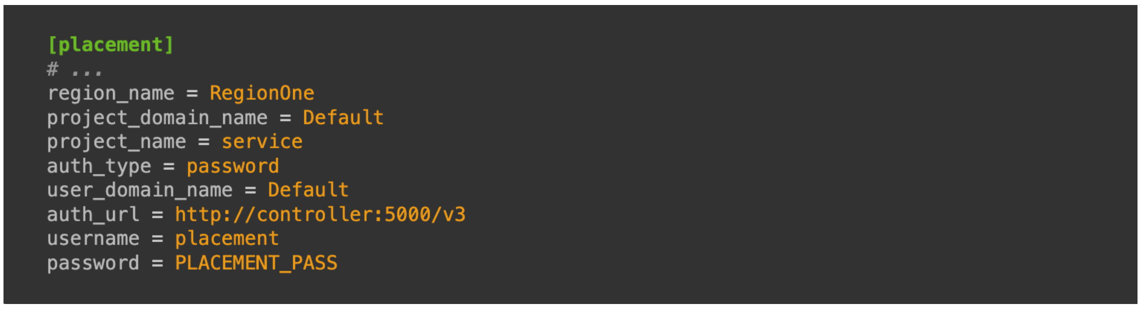
```
# apt install nova-api nova-conductor nova-novncproxy nova-scheduler
```

Figure 20 Installing Nova service packages

- Edit the Nova config file to link the database from the prerequisites to the placement service. This includes to make sure Nova works with the previously configured services.



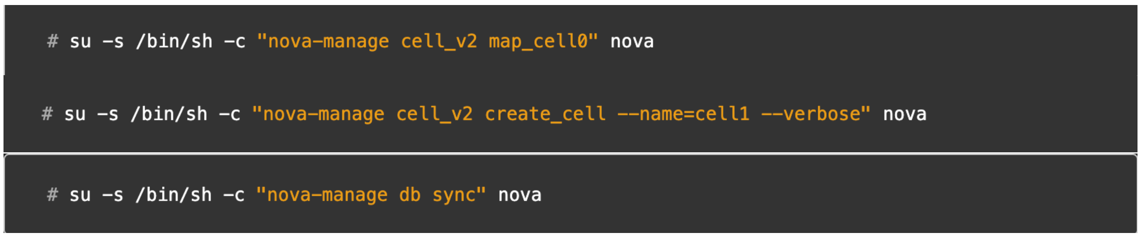
```
[api_database]
# ...
connection = mysql+pymysql://nova:NOVA_DBPASS@controller/nova_api

[database]
# ...
connection = mysql+pymysql://nova:NOVA_DBPASS@controller/nova


[DEFAULT]
# ...
transport_url = rabbit://openstack:RABBIT_PASS@controller:5672/


[api]
# ...
auth_strategy = keystone

[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 = nova
password = NOVA_PASS


[glance]
# ...
api_servers = http://controller:9292
```



```
[placement]
# ...
region_name = RegionOne
project_domain_name = Default
project_name = service
auth_type = password
user_domain_name = Default
auth_url = http://controller:5000/v3
username = placement
password = PLACEMENT_PASS
```

Figure 21 Configuration of Nova service

- Populate and register the previously created databases.



```
# su -s /bin/sh -c "nova-manage cell_v2 map_cell0" nova

# su -s /bin/sh -c "nova-manage cell_v2 create_cell --name=cell1 --verbose" nova

# su -s /bin/sh -c "nova-manage db sync" nova
```

Figure 22 Population and register of Nova databases

12.5 Networking service. Neutron.

You can build and connect interface devices to networks that are managed by other OpenStack services using OpenStack Networking (neutron). Plug-ins can be used to implement support for various networking tools and software, giving OpenStack's architecture and deployment flexibility.

12.5.1 Prerequisites

The first step is to check that we meet all the criteria.

- Create the database that neutron will use and make sure that all the permissions are right.

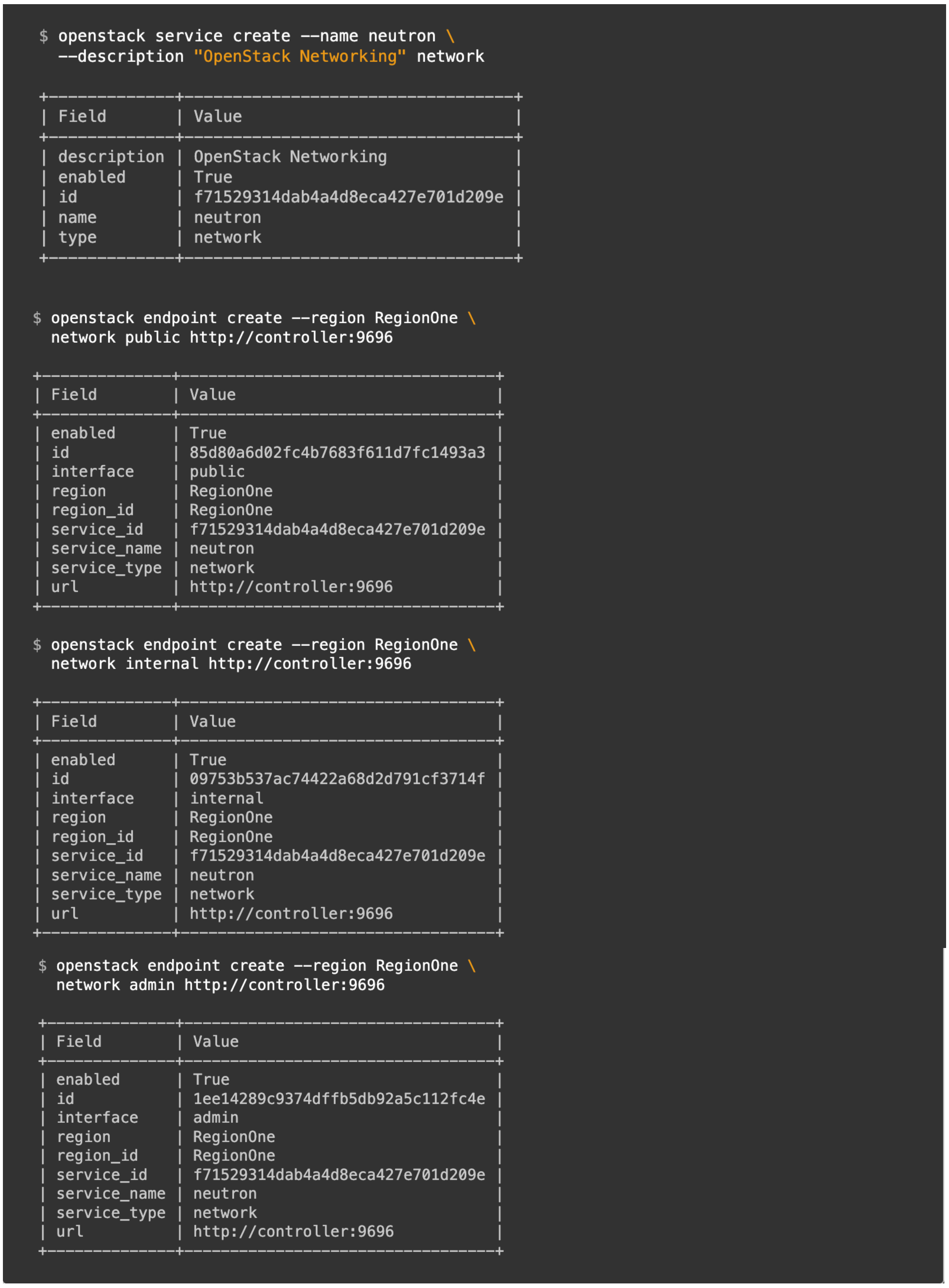
```
MariaDB [(none)] CREATE DATABASE neutron;
```

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'localhost' \
  IDENTIFIED BY 'NEUTRON_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'%' \
  IDENTIFIED BY 'NEUTRON_DBPASS';
```

Figure 23Creating Neutron database

- Create the neutron service credentials as well as the service entity and API endpoints.

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

User Password:
Repeat User Password:
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| domain_id           | default                          |
| enabled             | True                             |
| id                  | fdb0f541e28141719b6a43c8944bf1fb |
| name                | neutron                          |
| options             | {}                               |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```



```
$ openstack service create --name neutron \
  --description "OpenStack Networking" network

+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Networking             |
| enabled     | True                             |
| id          | f71529314dab4a4d8eca427e701d209e |
| name        | neutron                          |
| type        | network                          |
+-------------+----------------------------------+


$ openstack endpoint create --region RegionOne \
  network public http://controller:9696

+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 85d80a6d02fc4b7683f611d7fc1493a3 |
| interface    | public                           |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | f71529314dab4a4d8eca427e701d209e |
| service_name | neutron                          |
| service_type | network                          |
| url          | http://controller:9696           |
+--------------+----------------------------------+

$ openstack endpoint create --region RegionOne \
  network internal http://controller:9696

+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 09753b537ac74422a68d2d791cf3714f |
| interface    | internal                         |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | f71529314dab4a4d8eca427e701d209e |
| service_name | neutron                          |
| service_type | network                          |
| url          | http://controller:9696           |
+--------------+----------------------------------+
$ openstack endpoint create --region RegionOne \
  network admin http://controller:9696

+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 1ee14289c9374dffb5db92a5c112fc4e |
| interface    | admin                            |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | f71529314dab4a4d8eca427e701d209e |
| service_name | neutron                          |
| service_type | network                          |
| url          | http://controller:9696           |
+--------------+----------------------------------+
```

Figure 24Creating neutron service API endpoints, register and credentials

12.5.2 Installation and configuration

The installation and configuration can be done in two options:

- Option 1 deploys an architecture that is as basic as it can be and only allows for attaching instances to provider (external) networks. There aren't any routers, self-service (private) networks, or floating IP addresses. Provider networks can only be managed by the admin or another privileged user.

- Option 2 adds layer-3 services that enable attaching instances to self-service networks to option 1, enhancing it. Self-service networks, including routers that connect self-service and provider networks, can be managed by the demo or another non-privileged user. Additionally, instances using self-service networks can connect to external networks like the Internet via floating IP addresses.

In order to speed up the installation process and start as soon as possible testing the cloud, during these project Option 1 has been the one chosen. These are the steps followed:

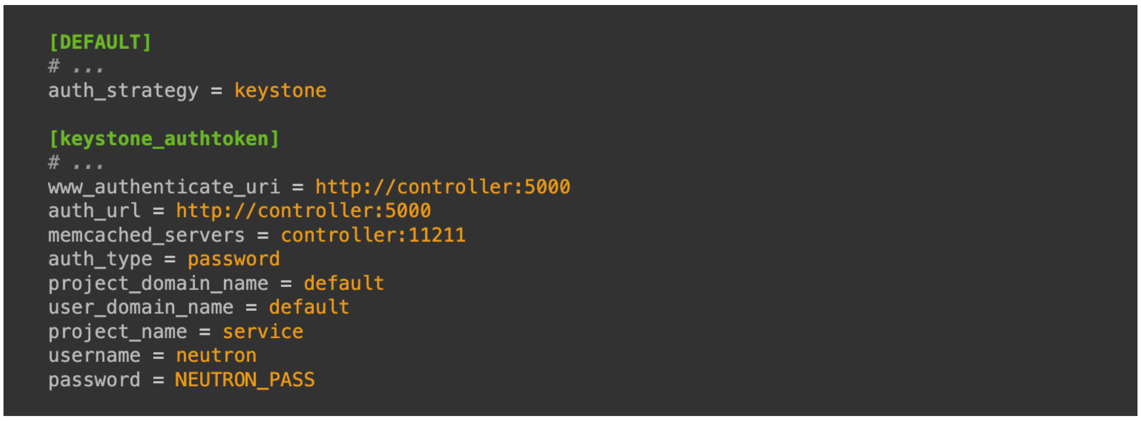
- Install required components.

```
# apt install neutron-server neutron-plugin-ml2 \
  neutron-linuxbridge-agent neutron-dhcp-agent \
  neutron-metadata-agent
```

Figure 25 Installing components for neutron option 1

- Configure the server component using the neutron config file. This process includes linking to the appropriate database as well as authenticating it using keystone.

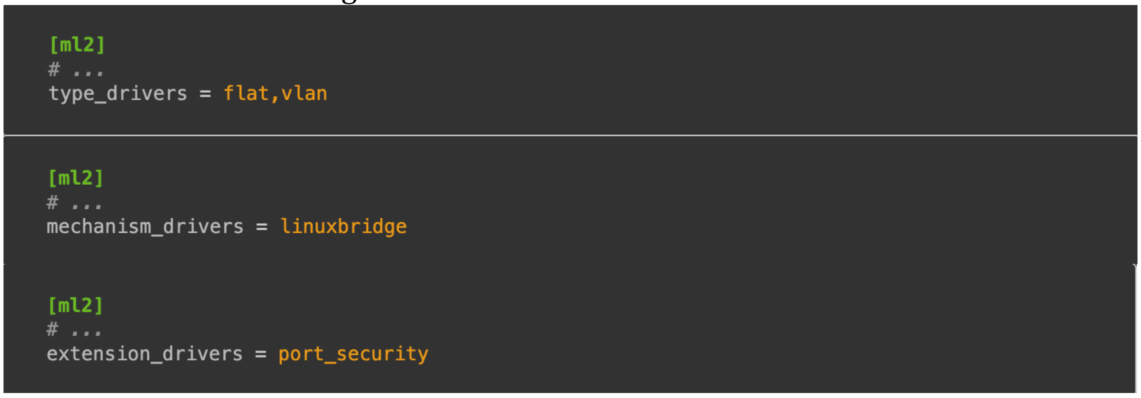
```
[database]
# ...
connection = mysql+pymysql://neutron:NEUTRON_DBPASS@controller/neutron
```



```
[DEFAULT]
# ...
auth_strategy = keystone

[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 = neutron
password = NEUTRON_PASS
```

Figure 26 Editing neutron config file for option 1

- Configure the Modular layer 2 (ML2) plugin. The ML2 plug-in uses the Linux bridge mechanism to build layer-2 (bridging and switching) virtual networking infrastructure for instances.



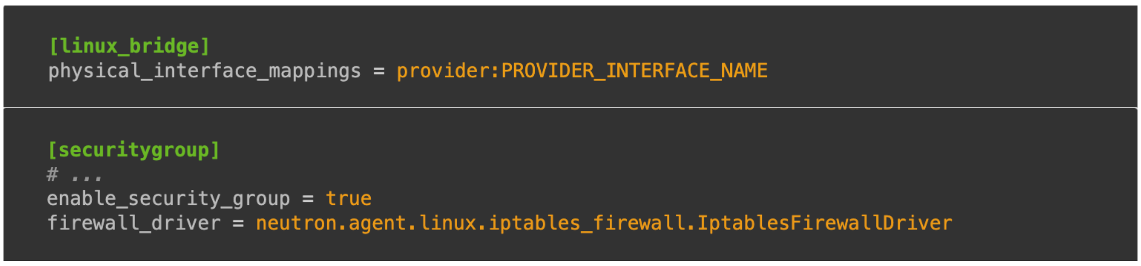
```
[ml2]
# ...
type_drivers = flat,vlan
```

```
[ml2]
# ...
mechanism_drivers = linuxbridge
```

```
[ml2]
# ...
extension_drivers = port_security
```

Figure 27 Configuring neutron ML2 plug-in

- Configure the Linux bridge agent, previously enabled in the ML2 plug-in. The Linux bridge agent builds layer-2 (bridging and switching) virtual networking infrastructure for instances and handles security groups.



```
[linux_bridge]
physical_interface_mappings = provider:PROVIDER_INTERFACE_NAME
```

```
[securitygroup]
# ...
enable_security_group = true
firewall_driver = neutron.agent.linux.iptables_firewall.IptablesFirewallDriver
```

Figure 28 Configuring Neutron service Linux bridge agent

- Configure the DHCP agent. The DHCP agent provides DHCP services for virtual networks.

```
[DEFAULT]
# ...
interface_driver = linuxbridge
dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq
enable_isolated_metadata = true
```

Figure 29 Configuring Neutron DHCP agent

- Lastly it is fundamental to configure the metadata agent. The metadata agent provides configuration information such as credentials to instances.

```
[neutron]
# ...
auth_url = http://controller:5000
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = neutron
password = NEUTRON_PASS
service_metadata_proxy = true
metadata_proxy_shared_secret = METADATA_SECRET
```

Figure 30 Configuring Neutron metadata agent

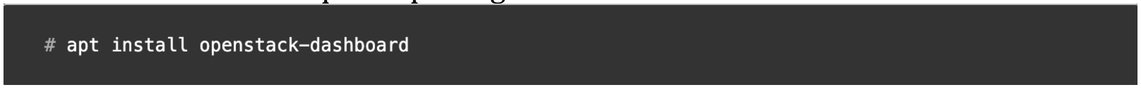
12.6 Dashboard service. Horizon

The official OpenStack Dashboard implementation is called Horizon, and it offers a web-based user interface for services like Nova, Swift, Keystone, and other OpenStack components.

12.6.1 Installation and configuration

At these points all the necessary components for the cloud should be installed, so there are no more prerequisites to install the dashboard. In order to get a more intuitive UI to control everything about the cloud the following steps have been completed:

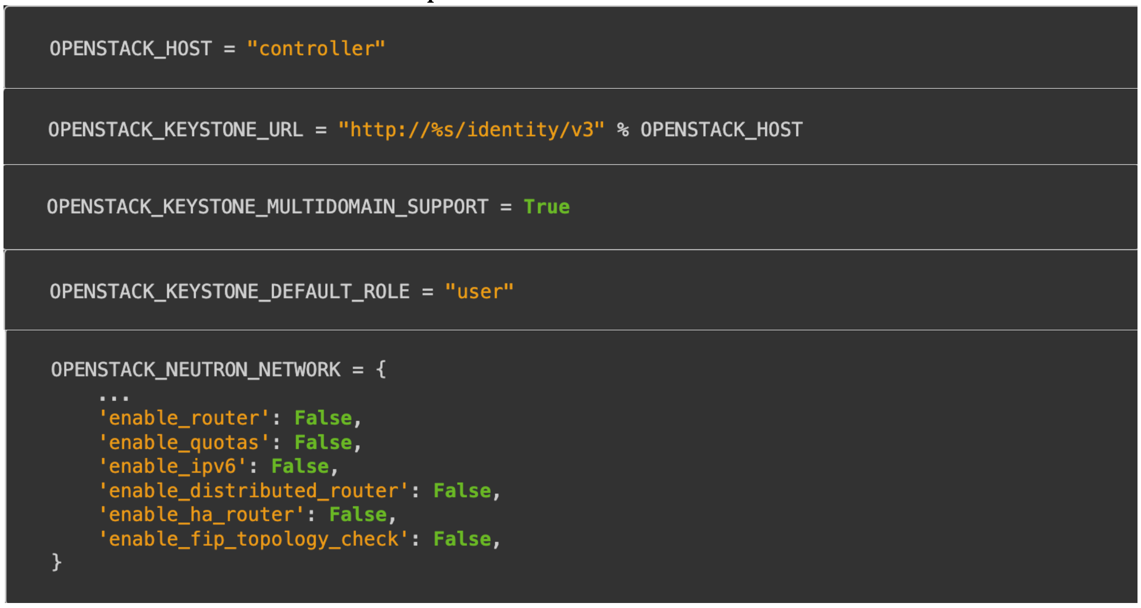
- Install all the required packages.



```
# apt install openstack-dashboard
```

Figure 31 Installing Horizon required packages

- Multiple configurations on the Horizon config file. In the next picture can be seen the most important ones.



```
OPENSTACK_HOST = "controller"
```

```
OPENSTACK_KEYSTONE_URL = "http://%s/identity/v3" % OPENSTACK_HOST
```

```
OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = True
```

```
OPENSTACK_KEYSTONE_DEFAULT_ROLE = "user"
```

```
OPENSTACK_NEUTRON_NETWORK = {
    ...
    'enable_router': False,
    'enable_quotas': False,
    'enable_ipv6': False,
    'enable_distributed_router': False,
    'enable_ha_router': False,
    'enable_fip_topology_check': False,
}
```

Figure 32 Configuration of Horizon service config file

13. Cloud Management. Result.

In this section it is going to be explained how to establish the connection to the cloud, how to manage it and all the different features and services that it offers. This includes creating instances, user management, networking and so much more.

13.1 Tools to manage the cloud

For a variety of operational tasks, administrators must execute command and control over the cloud. It's crucial to comprehend and secure these command and control infrastructures. OpenStack offers operators and tenants a variety of administration interfaces:

- **OpenStack dashboard (horizon):** Administrators and tenants can provision and use cloud-based resources using the web-based graphical interface of the OpenStack dashboard (horizon). The OpenStack API is used by the dashboard to make calls to the back-end services.
- **OpenStack API:** The OpenStack API is a RESTful web service endpoint for cloud-based resource access, provisioning, and automation. Operators and users often use third-party tools, language-specific libraries, or command-line programs (like nova or look) to access the API.
- **Secure shell (SSH):** The use of secure shell (SSH) access for Linux and Unix system administration has become standard practice. SSH communicates using safe cryptographic building blocks. Knowing the best methods for deploying SSH is crucial given the scale and significance of SSH in typical OpenStack deployments.
- **OpenStack management utilities such as nova-manage and glance-manage:** Open-source Python command-line clients that use the OpenStack Management Utilities to call APIs. Each service provided by OpenStack has a client (for example, nova, glance). Most services provide a management command-line utility that directly calls the database in addition to the normal CLI client.
- **Out-of-band management interfaces, such as IPMI:** The IPMI protocol is one of the out-of-band management interfaces that OpenStack management uses to reach nodes that are running OpenStack components. Whether the operating system is running or the system has crashed, servers can be remotely managed, diagnosed, and rebooted using the IPMI protocol, which is widely common.

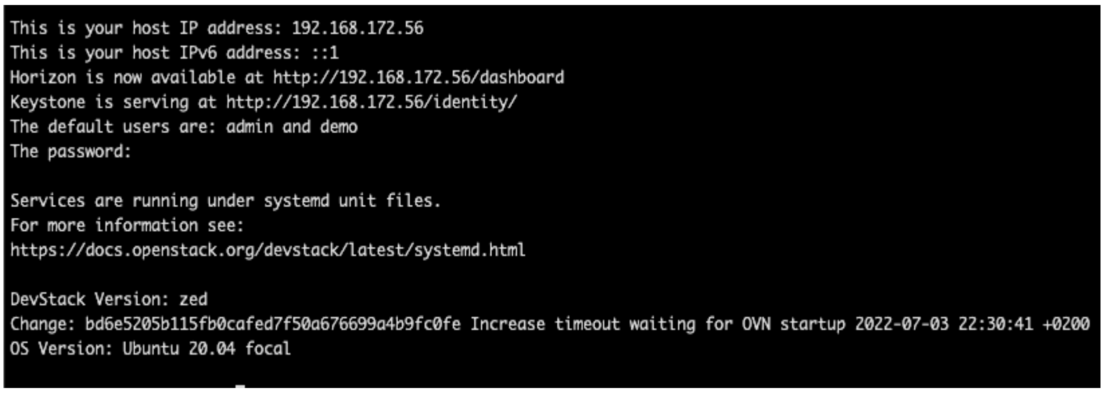
13.2 Openstack dashboard capabilities

Horizon dashboard has been the main management tool used during the elaboration of this project, mainly due to its simplicity and friendly GUI. This service has multiple capabilities:

- The dashboard gives you, as a cloud administrator, a broad overview of the size and condition of your cloud. You can create users, tenants, and projects, assign people to those entities, and impose restrictions on the resources that can be made available to them.
- Tenant users have access to a self-service portal through the dashboard where they can provide their own resources within the parameters defined by administrators.
- The dashboard offers load-balancers and routers GUI support. For instance, all of the key Networking functions are implemented in the dashboard.
- It is a Django-based expandable web application that makes it simple to integrate third-party goods and services like billing, monitoring, and extra administration tools.
- Additionally, service providers and other commercial suppliers can brand the dashboard.

13.3 Dashboard features

Previously has been mentioned that horizon provides multiple features to a cloud environment, so let’s analyze the dashboard interface and the multiple things it is possible to do with it. The instructions to connect to the dashboards are received on terminal message after finalizing the installation of Openstack. In the next image it is possible to see that it is necessary to connect to the ip address 192.168.172.56. In this section are going to be explained just the most relevant features and possibilities of the cloud infrastructure, otherwise this research document would be very extensive.



```
This is your host IP address: 192.168.172.56
This is your host IPv6 address: ::1
Horizon is now available at http://192.168.172.56/dashboard
Keystone is serving at http://192.168.172.56/identity/
The default users are: admin and demo
The password:

Services are running under systemd unit files.
For more information see:
https://docs.openstack.org/devstack/latest/systemd.html

DevStack Version: zed
Change: bd6e5205b115fb0cafed7f50a676699a4b9fc0fe Increase timeout waiting for OVN startup 2022-07-03 22:30:41 +0200
OS Version: Ubuntu 20.04 focal
```

Figure 33 Instruction to stablish connection to OpenStack cloud

13.3.1 Log in

The first thing you can see as soon as you connect to the dashboard is a login page to identify the user. This is a must have security feature in any cloud environment. Not only allows to make sure that only authorized users can access to the resources, but also allows to create different profiles and groups, allowing to each group to have permissions and access only to the features they are allowed to. For this section the user that has been used is the admin user in order to have full access to the dashboard and its features.

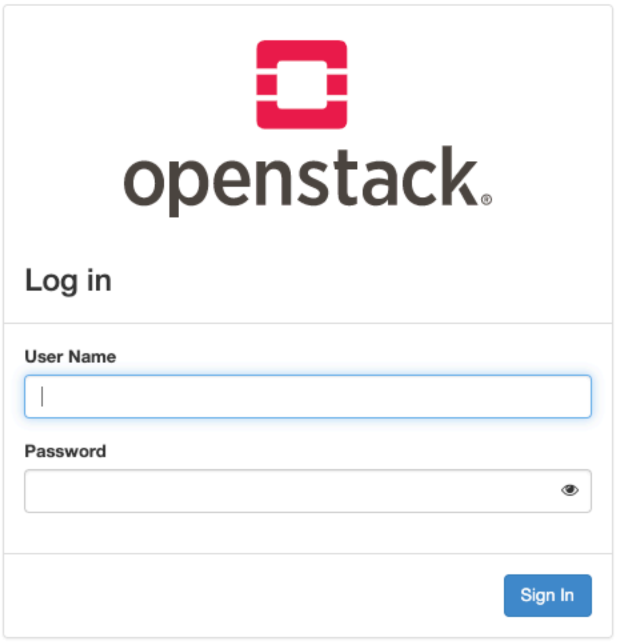
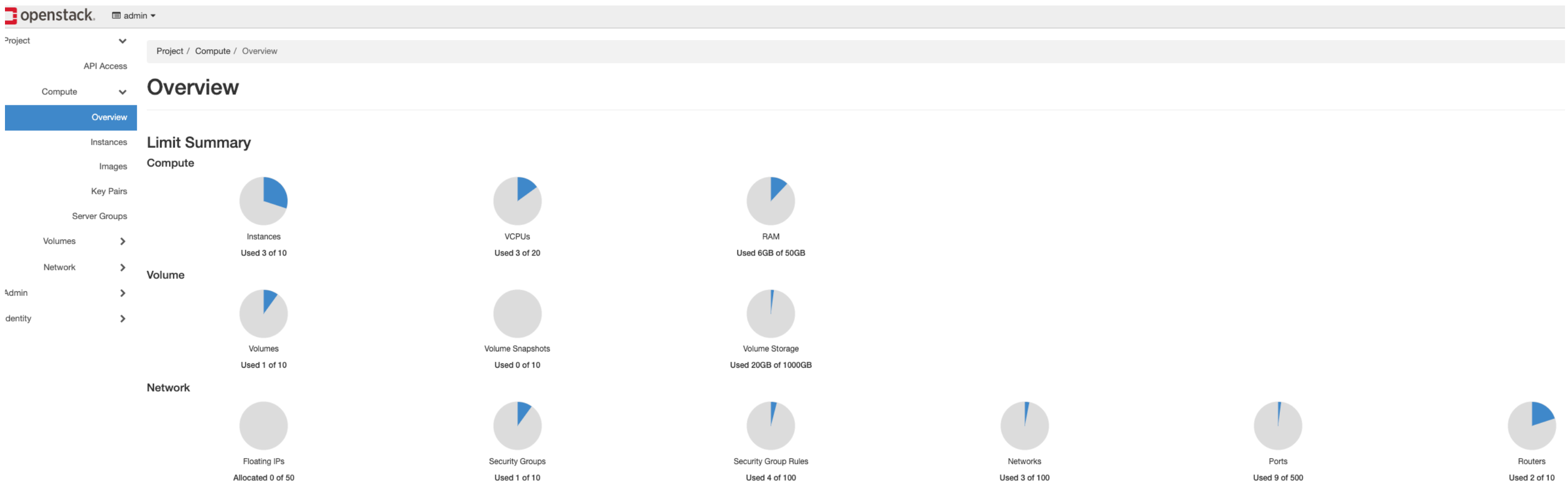


Figure 34 OpenStack Log In page

13.3.2 Overview panel

The first thing that can be seen after login in is an overview panel that shows the most relevant and basic information about the cloud. This includes the number of instances in use, the CPU, RAM and volume used and available, network configurations like floating Ips, security groups, routers, ports… From this panel there it is possible to access to the different windows that the dashboard has.



Overview

Limit Summary

Compute

Instances
Used 3 of 10

VCPUs
Used 3 of 20

RAM
Used 6GB of 50GB

Volume

Volumes
Used 1 of 10

Volume Snapshots
Used 0 of 10

Volume Storage
Used 20GB of 1000GB

Network

Floating IPs
Allocated 0 of 50

Security Groups
Used 1 of 10

Security Group Rules
Used 4 of 100

Networks
Used 3 of 100

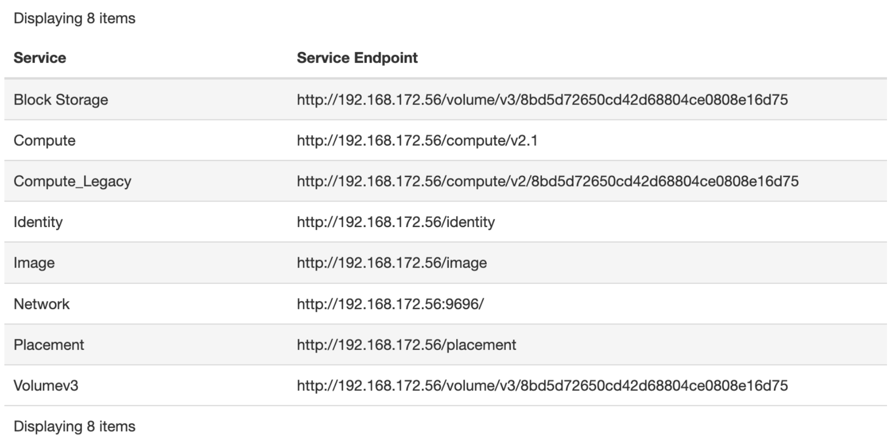
Ports
Used 9 of 500

Routers
Used 2 of 10

Figure 35 OpenStack Overview Panel

13.3.3 API Access

This panel basically shows the services that have been installed in the cloud environment as well as the service endpoints. This is basically the configuration that has been explained in the previous section.



Displaying 8 items

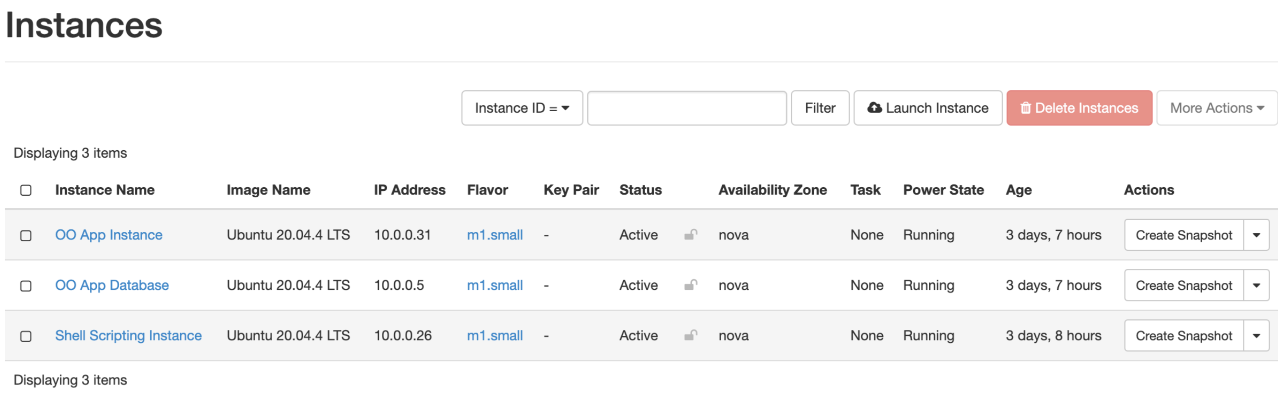
Service	Service Endpoint
Block Storage	http://192.168.172.56/volume/v3/8bd5d72650cd42d68804ce0808e16d75
Compute	http://192.168.172.56/compute/v2.1
Compute_Legacy	http://192.168.172.56/compute/v2/8bd5d72650cd42d68804ce0808e16d75
Identity	http://192.168.172.56/identity
Image	http://192.168.172.56/image
Network	http://192.168.172.56:9696/
Placement	http://192.168.172.56/placement
Volumev3	http://192.168.172.56/volume/v3/8bd5d72650cd42d68804ce0808e16d75

Displaying 8 items

Figure 36 OpenStack API Access

13.3.4 Instances

Under the compute section, it is possible to access to the instances panel. This panel shows all the running or paused instances, allows to access to the control panel of each of them as well as create new instances.



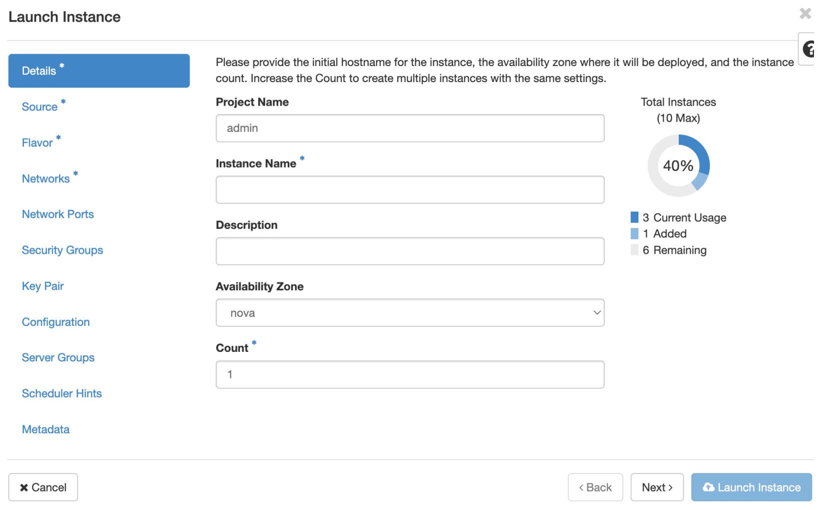
Displaying 3 items

	Instance Name	Image Name	IP Address	Flavor	Key Pair	Status		Availability Zone	Task	Power State	Age	Actions
☐	OO App Instance	Ubuntu 20.04.4 LTS	10.0.0.31	m1.small	-	Active		nova	None	Running	3 days, 7 hours	Create Snapshot
☐	OO App Database	Ubuntu 20.04.4 LTS	10.0.0.5	m1.small	-	Active		nova	None	Running	3 days, 7 hours	Create Snapshot
☐	Shell Scripting Instance	Ubuntu 20.04.4 LTS	10.0.0.26	m1.small	-	Active		nova	None	Running	3 days, 8 hours	Create Snapshot

Displaying 3 items

Figure 37 Instances management panel

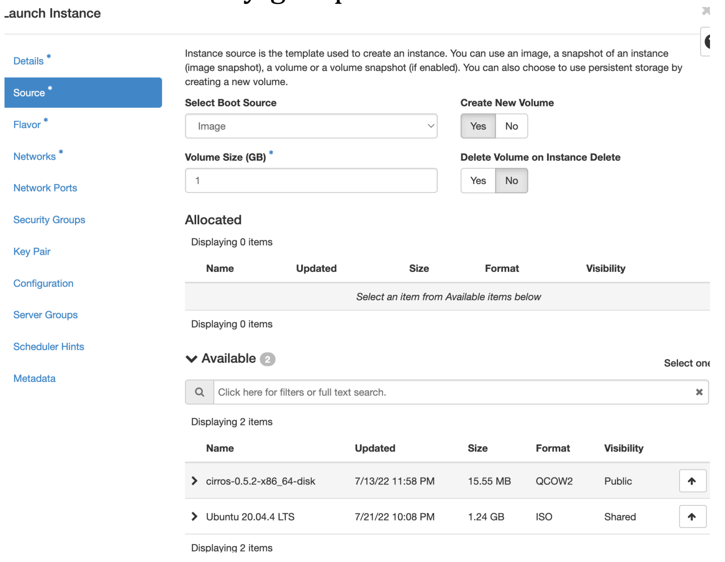
To create a new instance, it is as simple as to click in the launch instance button and access to the panel shown in the picture below. Here it is possible to configure the name, availability zone, source of instance, hardware allocated for the machine, connect it to a network, add the instance to a security group and much more.



Please provide the initial hostname for the instance, the availability zone where it will be deployed, and the instance count. Increase the Count to create multiple instances with the same settings.

Project Name	admin
Instance Name *	
Description	
Availability Zone	nova
Count *	1

▮ 3 Current Usage
▮ 1 Added
▮ 6 Remaining



Instance source is the template used to create an instance. You can use an image, a snapshot of an instance (image snapshot), a volume or a volume snapshot (if enabled). You can also choose to use persistent storage by creating a new volume.

Select Boot Source	Image
Create New Volume	[✓ Yes] No
Volume Size (GB) *	1
Delete Volume on Instance Delete	Yes [✓ No]

Allocated

Displaying 0 items

Name	Updated	Size	Format	Visibility
Select an item from Available items below				

Displaying 0 items

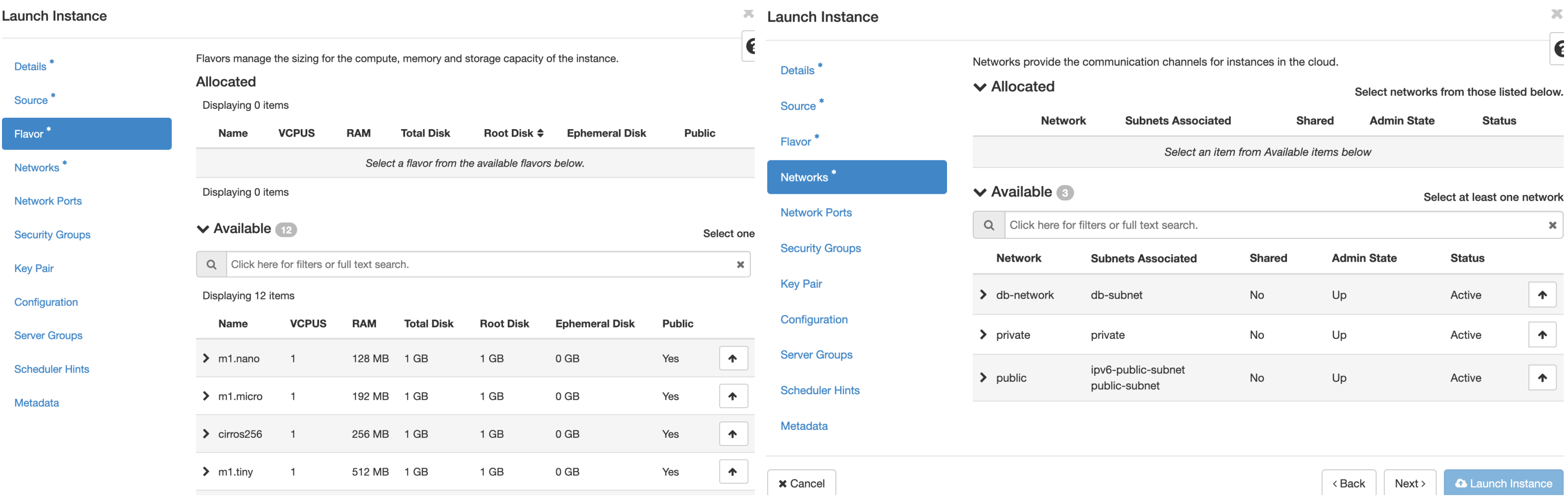
Available 2

Click here for filters or full text search.

Displaying 2 items

Name	Updated	Size	Format	Visibility
cirros-0.5.2-x86_64-disk	7/13/22 11:58 PM	15.55 MB	QCOW2	Public
Ubuntu 20.04.4 LTS	7/21/22 10:08 PM	1.24 GB	ISO	Shared

Displaying 2 items



Flavors manage the sizing for the compute, memory and storage capacity of the instance.

Allocated

Displaying 0 items

Name	VCPUS	RAM	Total Disk	Root Disk	Ephemeral Disk	Public
Select a flavor from the available flavors below.						

Displaying 0 items

Available 12 — Select one

Click here for filters or full text search.

Displaying 12 items

Name	VCPUS	RAM	Total Disk	Root Disk	Ephemeral Disk	Public
m1.nano	1	128 MB	1 GB	1 GB	0 GB	Yes
m1.micro	1	192 MB	1 GB	1 GB	0 GB	Yes
cirros256	1	256 MB	1 GB	1 GB	0 GB	Yes
m1.tiny	1	512 MB	1 GB	1 GB	0 GB	Yes

Networks provide the communication channels for instances in the cloud.

Allocated — Select networks from those listed below.

Network	Subnets Associated	Shared	Admin State	Status
Select an item from Available items below				

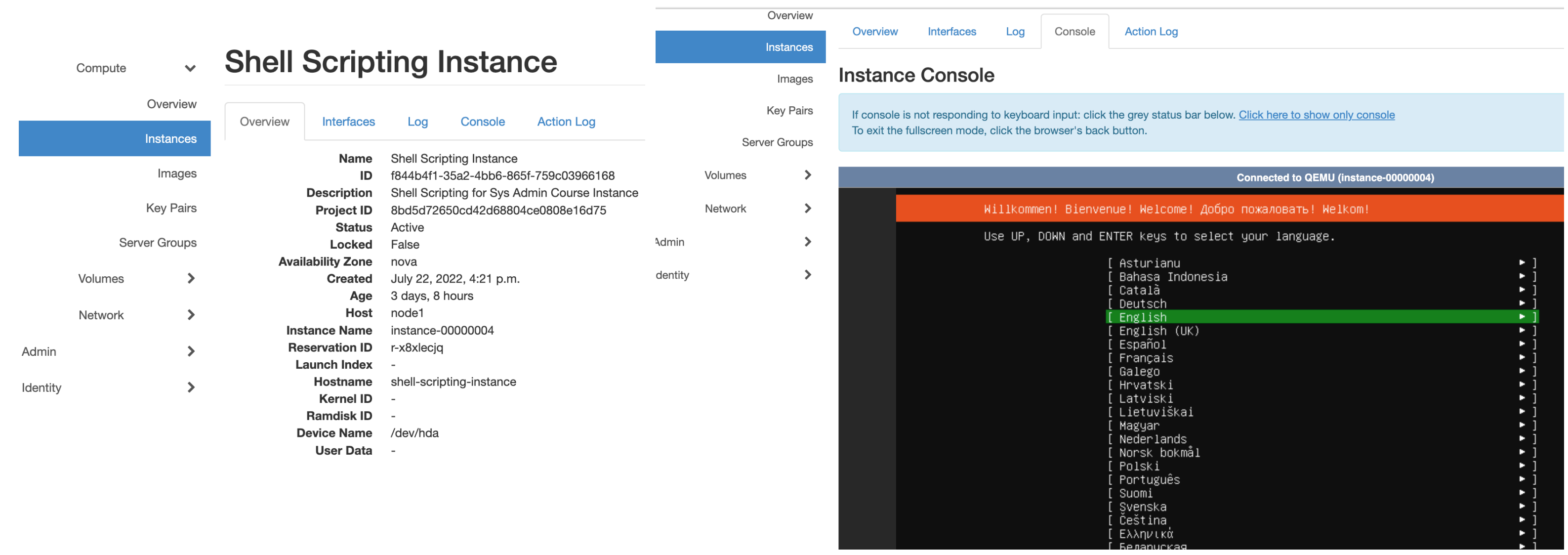
Available 3 — Select at least one network

Click here for filters or full text search.

Network	Subnets Associated	Shared	Admin State	Status
db-network	db-subnet	No	Up	Active
private	private	No	Up	Active
public	ipv6-public-subnet public-subnet	No	Up	Active

Figure 38 How to create and configure and instance

Once the instance has been created, it is possible to access to it and manage it simply by clicking on the name. This gives access to an overview of the details of the instance, a log menu of it, access to the console of the instance and more.



Shell Scripting Instance

Name	Shell Scripting Instance
ID	f844b4f1-35a2-4bb6-865f-759c03966168
Description	Shell Scripting for Sys Admin Course Instance
Project ID	8bd5d72650cd42d68804ce0808e16d75
Status	Active
Locked	False
Availability Zone	nova
Created	July 22, 2022, 4:21 p.m.
Age	3 days, 8 hours
Host	node1
Instance Name	instance-00000004
Reservation ID	r-x8xlecjq
Launch Index	-
Hostname	shell-scripting-instance
Kernel ID	-
Ramdisk ID	-
Device Name	/dev/hda
User Data	-

Instance Console

If console is not responding to keyboard input: click the grey status bar below. Click here to show only console
To exit the fullscreen mode, click the browser's back button.

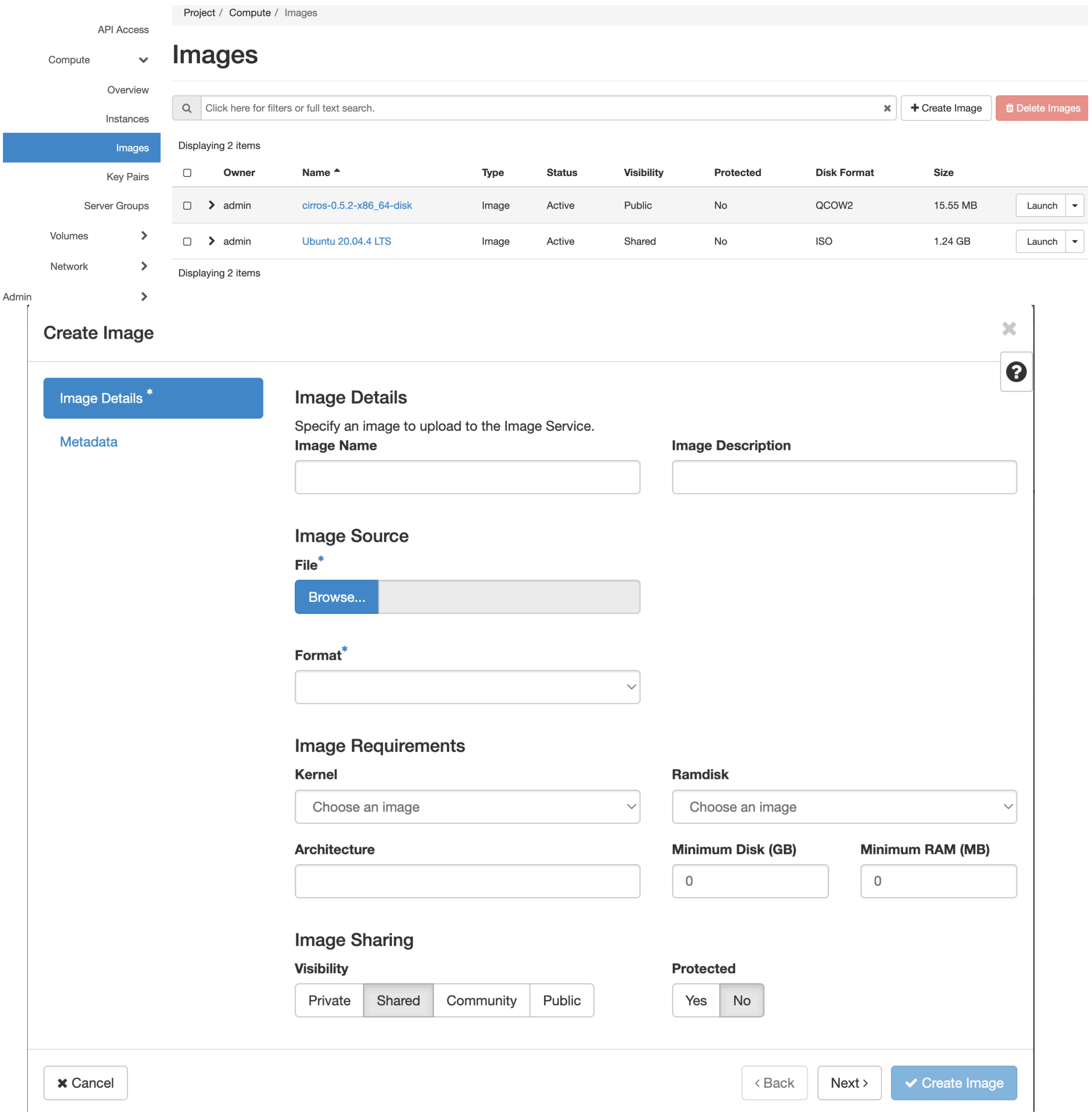
Connected to QEMU (instance-00000004)

```
Willkommen! Bienvenue! Welcome! Добро пожаловать! Welkom!

Use UP, DOWN and ENTER keys to select your language.

[ Asturianu           ▸ ]
[ Bahasa Indonesia    ▸ ]
[ Català              ▸ ]
[ Deutsch             ▸ ]
[ English             ▸ ]
[ English (UK)        ▸ ]
[ Español             ▸ ]
[ Français            ▸ ]
[ Galego              ▸ ]
[ Hrvatski            ▸ ]
[ Latviski            ▸ ]
[ Lietuviškai         ▸ ]
[ Magyar              ▸ ]
[ Nederlands          ▸ ]
[ Norsk bokmål        ▸ ]
[ Polski              ▸ ]
[ Português           ▸ ]
[ Suomi               ▸ ]
[ Svenska             ▸ ]
[ Čeština             ▸ ]
[ Ελληνικά            ▸ ]
[ Беларуская          ▸ ]
```

Figure 39Management panels of a created instance in OpenStack

13.3.5 Images

The next panel under Compute menu is images, this is where you can see the images available to create instances. This section also allows to add custom images uploading them from a local storage.



Images

Displaying 2 items

	Owner	Name	Type	Status	Visibility	Protected	Disk Format	Size
	admin	cirros-0.5.2-x86_64-disk	Image	Active	Public	No	QCOW2	15.55 MB
	admin	Ubuntu 20.04.4 LTS	Image	Active	Shared	No	ISO	1.24 GB

Displaying 2 items

Create Image

Image Details *
Metadata

Image Details

Specify an image to upload to the Image Service.

Image Name:
Image Description:

Image Source

File*: Browse...

Format*:

Image Requirements

Kernel: Choose an image
Ramdisk: Choose an image
Architecture:
Minimum Disk (GB): 0
Minimum RAM (MB): 0

Image Sharing

Visibility: Private [✓ Shared] Community Public
Protected: Yes [✓ No]

Figure 40 image managemnt in OpenStack

13.3.6 Key Pairs

An important feature in any cloud is the security, that is the reason why this section exists, it allows to create key pair and manage them. A key pair is a combination of a public key that is used to encrypt data and a private key that is used to decrypt data.

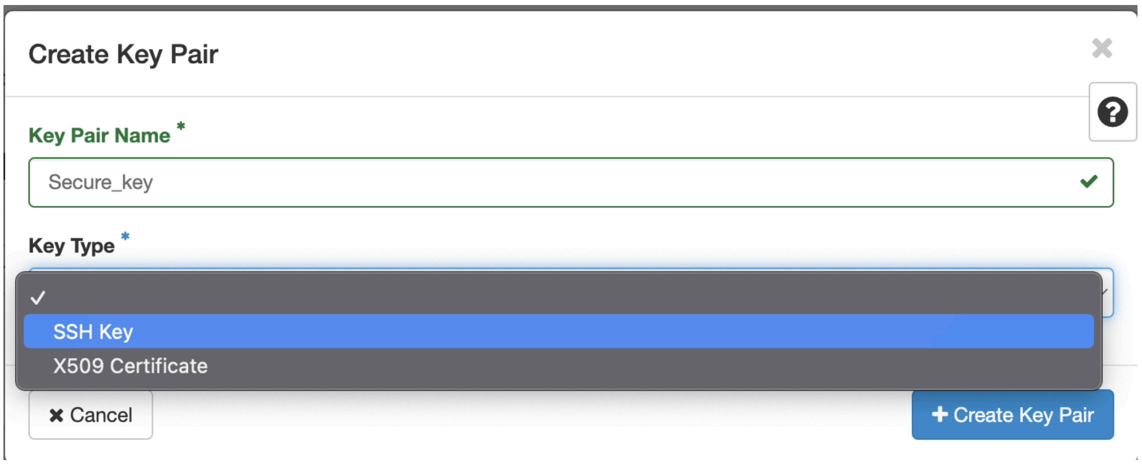
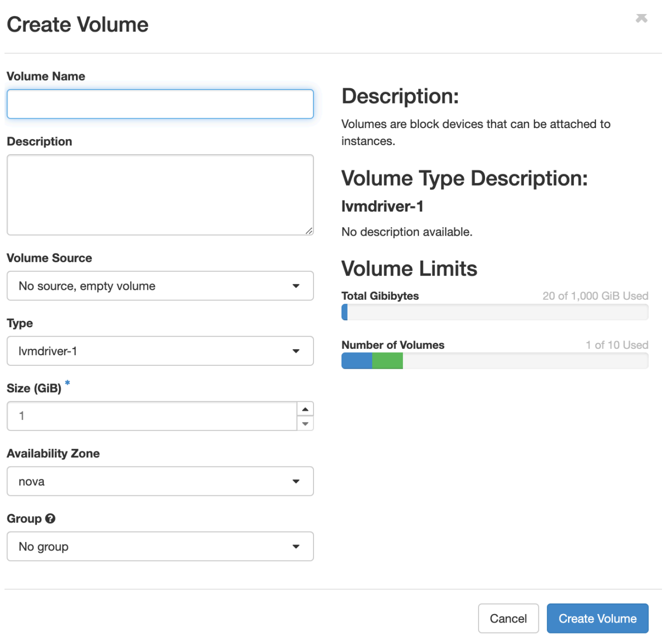


Figure 41 Keypairs in OpenStack

13.3.7 Volumes

Under the volumes tab it is possible to control anything related to storage. In A storage volume is a virtual disk that offers persistent block storage capacity for instances. Applications and data can be stored on storage volumes.

When building an instance or later, after it has started running, you can attach one or more storage volumes to it. By adding or detaching storage volumes, you can quickly scale up or scale down the instance's storage resource once it has been created. A storage volume that was associated when the instance was created, however, cannot be detached. It should be noted that data stored on a storage volume is not destroyed when the storage volume is disconnected from an instance or when the instance is deleted. Additionally, you can create snapshots of the storage volumes.


Create Volume

Volume Name:

Description:

Volume Source: No source, empty volume

Type: lvmdriver-1

Size (GiB) *: 1

Availability Zone: nova

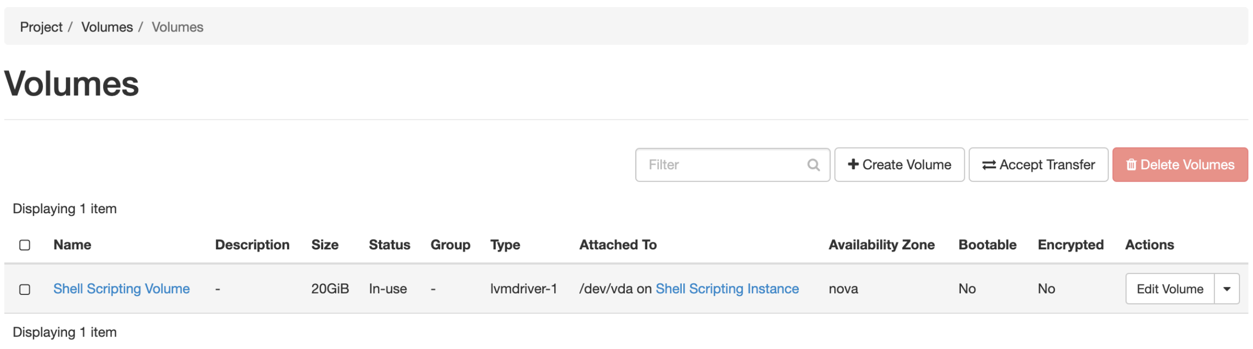
Group: No group

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

Volume Type Description:
lvmdriver-1
No description available.

Volume Limits
Total Gibibytes 20 of 1,000 GiB Used
Number of Volumes 1 of 10 Used

Figure 42 Volume configuration panel


Project / Volumes / Volumes

Volumes

Displaying 1 item

	Name	Description	Size	Status	Group	Type	Attached To	Availability Zone	Bootable	Encrypted	Actions
	Shell Scripting Volume	-	20GiB	In-use	-	lvmdriver-1	/dev/vda on Shell Scripting Instance	nova	No	No	Edit Volume

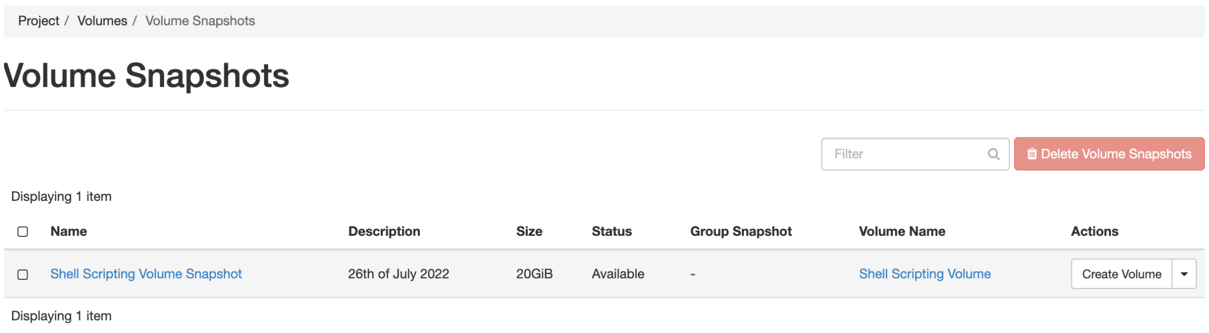
Displaying 1 item

Figure 43 Volumen overview panel

13.3.8 Snapshots

As mentioned on the volumes section, it is possible to create snapshots from volumes and this panel allows to manage all of them. A snapshot is a copy made of the storage volume at any instant. A snapshot of a storage volume is another storage volume. When you want to attach the storage volume to an running instance, you can back up the original volume by taking the snapshot of it.

When a storage volume is attached to a running instance and is running out of space. You can take a snapshot of the storage volume with an increased size of the volume. Then, you can attach the new storage volume and detach the old storage volume.


Project / Volumes / Volume Snapshots

Volume Snapshots

Displaying 1 item

	Name	Description	Size	Status	Group Snapshot	Volume Name	Actions
▢	Shell Scripting Volume Snapshot	26th of July 2022	20GiB	Available	-	Shell Scripting Volume	Create Volume

Displaying 1 item

Figure 44: Snapshots panel

13.3.9 Network Topology

A network topology shows a topological graph about devices which connect to the specific network. Also, it will return availability information for each individual device within the network as well. In the next picture it is possible to see a example topology created with a public network, a private network, a router and some instances.

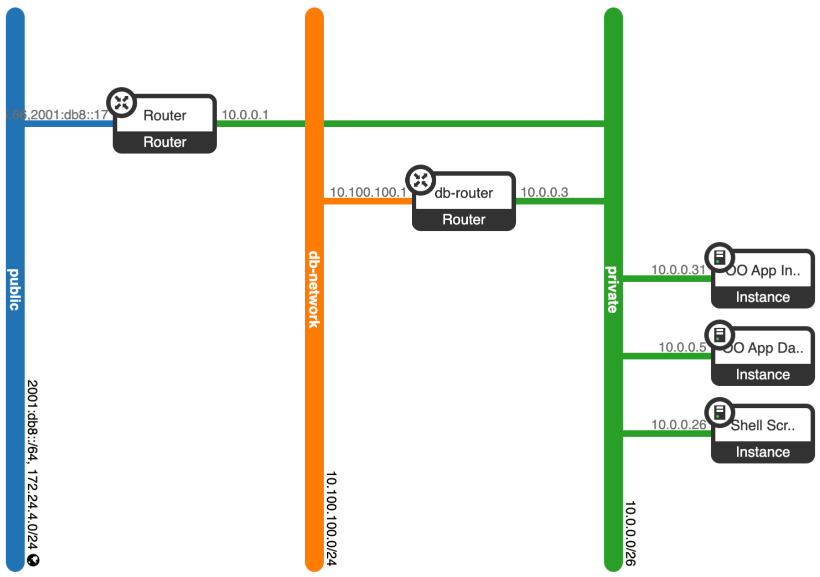
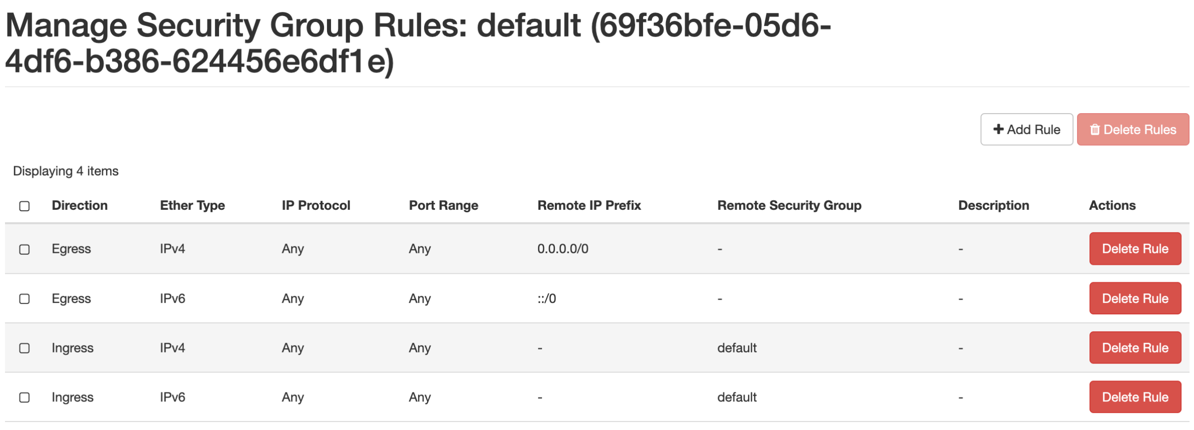


Figure 45 Cloud network topology example

13.3.10 Network security groups

Security is always a big concern in any IT infrastructure and services like network security groups allow to manage how and who can use the resources of the could designed. A security group is a container for security group rules. Security groups and their rules allow administrators and projects the ability to specify the type of traffic and direction (ingress/egress) that is allowed to pass through a virtual interface port. When a virtual interface port is created in OpenStack Networking it is associated with a security group.


Manage Security Group Rules: default (69f36bfe-05d6-4df6-b386-624456e6df1e)

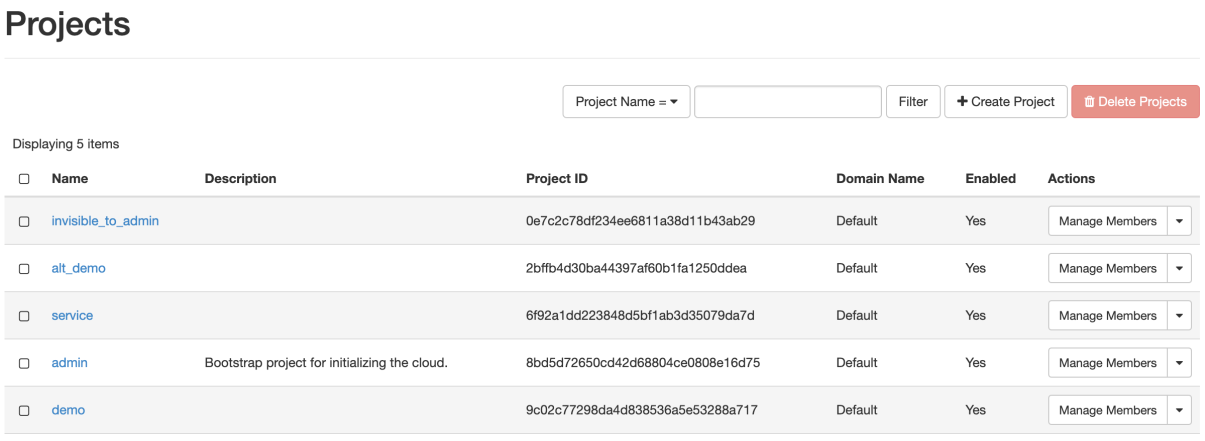
Displaying 4 items

☐	Direction	Ether Type	IP Protocol	Port Range	Remote IP Prefix	Remote Security Group	Description	Actions
☐	Egress	IPv4	Any	Any	0.0.0.0/0	-	-	Delete Rule
☐	Egress	IPv6	Any	Any	::/0	-	-	Delete Rule
☐	Ingress	IPv4	Any	Any	-	default	-	Delete Rule
☐	Ingress	IPv6	Any	Any	-	default	-	Delete Rule

Figure 46: Network security groups overview

13.3.11 Admin panel

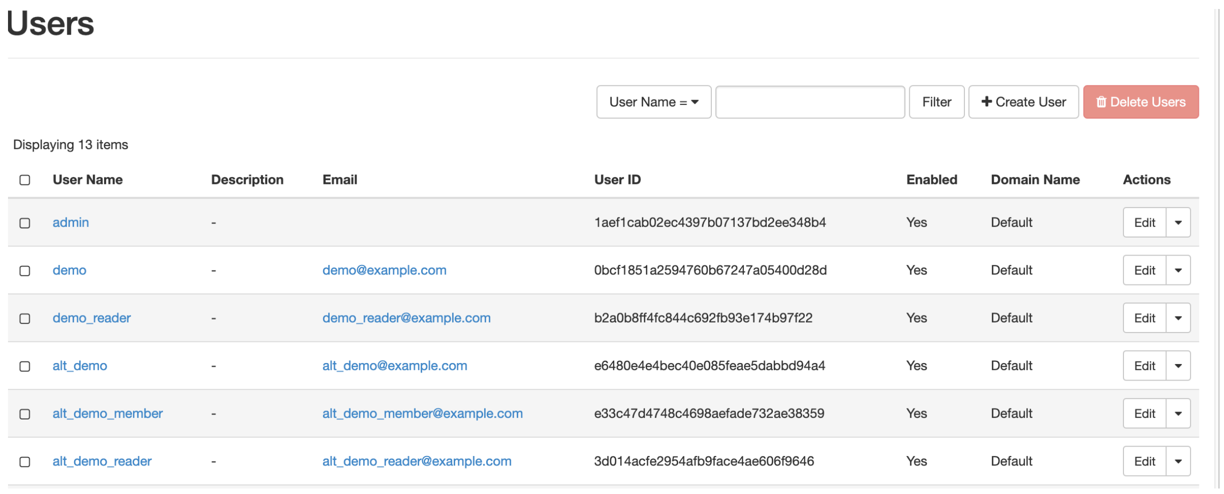
As an administrator, it is possible to manage projects, users, and roles. Projects are organizational units in the cloud to which you can assign users. Projects are also known as projects or accounts. Users can be members of one or more projects. Roles define which actions users can perform. It is possible to add, update, and delete projects and users, assign users to one or more projects, and change or remove the assignment. To enable or temporarily disable a project or user, update that project or user. Before deleting a user account, the user account must be removed from its primary project.



Projects

Displaying 5 items

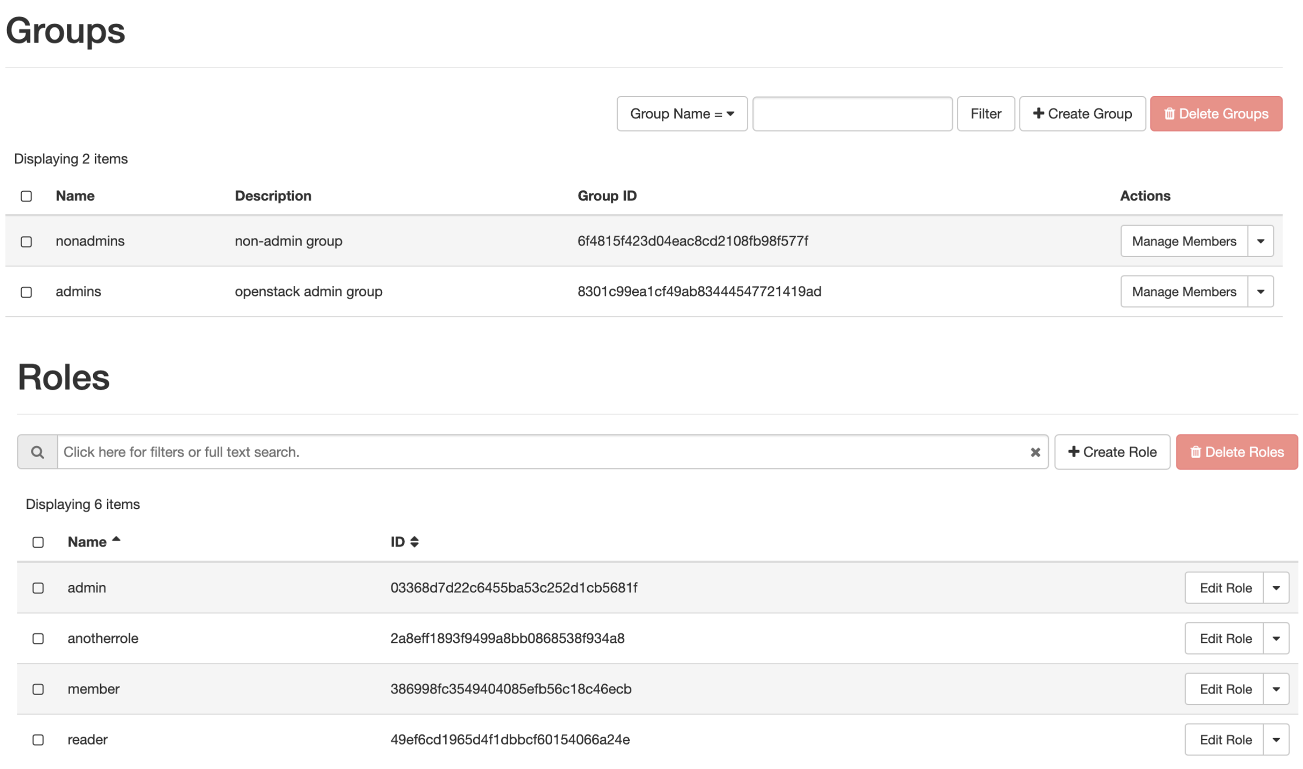
Name	Description	Project ID	Domain Name	Enabled	Actions
invisible_to_admin		0e7c2c78df234ee6811a38d11b43ab29	Default	Yes	Manage Members
alt_demo		2bffb4d30ba44397af60b1fa1250ddea	Default	Yes	Manage Members
service		6f92a1dd223848d5bf1ab3d35079da7d	Default	Yes	Manage Members
admin	Bootstrap project for initializing the cloud.	8bd5d72650cd42d68804ce0808e16d75	Default	Yes	Manage Members
demo		9c02c77298da4d838536a5e53288a717	Default	Yes	Manage Members



Users

Displaying 13 items

User Name	Description	Email	User ID	Enabled	Domain Name	Actions
admin	-		1aef1cab02ec4397b07137bd2ee348b4	Yes	Default	Edit
demo	-	demo@example.com	0bcf1851a2594760b67247a05400d28d	Yes	Default	Edit
demo_reader	-	demo_reader@example.com	b2a0b8ff4fc844c692fb93e174b97f22	Yes	Default	Edit
alt_demo	-	alt_demo@example.com	e6480e4e4bec40e085feae5dabbd94a4	Yes	Default	Edit
alt_demo_member	-	alt_demo_member@example.com	e33c47d4748c4698aefade732ae38359	Yes	Default	Edit
alt_demo_reader	-	alt_demo_reader@example.com	3d014acfe2954afb9face4ae606f9646	Yes	Default	Edit



Displaying 2 items

Name	Description	Group ID	Actions
nonadmins	non-admin group	6f4815f423d04eac8cd2108fb98f577f	Manage Members
admins	openstack admin group	8301c99ea1cf49ab83444547721419ad	Manage Members

Displaying 6 items

Name	ID	
admin	03368d7d22c6455ba53c252d1cb5681f	Edit Role
anotherrole	2a8eff1893f9499a8bb0868538f934a8	Edit Role
member	386998fc3549404085efb56c18c46ecb	Edit Role
reader	49ef6cd1965d4f1dbbcf60154066a24e	Edit Role

Figure 47 Projects, users, groups and roles configuration panels

14. Conclusions

Analyzing the results obtained during the installation, configuration and test process, this research project can be considered a success. The initial goal, designing and implementing a cloud, has been achieved. During the test section it has been possible to see the multiple features this environment offers and how easy and friendly it is to manage the entire system.

A very interesting work that could be done in the future is to do experiments with Terraform and OpenStack. Terraform is an infrastructure as code tool that lets you build, change, and version cloud and on-prem resources safely and efficiently. This means that it could be interesting to manage all the cloud with a single code script that keeps everything up to date.

Another feature that would be great to try to implement would be Kubernetes. Kubernetes, often abbreviated as "K8s", orchestrates containerized applications to run on a cluster of hosts. The K8s system automates the deployment and management of cloud native applications using on-premises infrastructure or public cloud platforms.

Finally, it must be determined how to proceed from now on. Considering the results of this work, the best path to follow would be to try to optimize the system as much as possible. During this research project the focus has been to have everything fully functional. But if a company considers implementing this design, they might be worried about response times, backup systems, security… Also, the previously mentioned Terraform as well as Kubernetes can add multiple features that companies are looking for.