

# **Automated Compute Infrastructure for Organizations using OpenStack**

A project report submitted in partial fulfillment of the degree of

## **Master of Computer Applications**

By

Sagar Suman(15MCMC09)

Himanshu Soni(15MCMC13)



School of Computer and Information Sciences  
University of Hyderabad  
Hyderabad-500046



## CERTIFICATE

This is to certify that the Project Report entitled “**Automated Compute Infrastructure for Organizations using OpenStack**” submitted by **Himanshu Soni** bearing Reg. No. 15MCMC13 and **Sagar Suman** bearing Reg. No. 15MCMC09 in partial fulfillment of the requirements for the award of Master of Computer Applications, is a bonafide work carried out by them under my/our supervision and guidance.

The project report has not been submitted previously in part or in full to this or any other University or Institution for the award of any degree or diploma.

Dr. P. Anupama  
School of Computer and Information Sciences,  
University of Hyderabad

Dean,  
School of Computer and Information Sciences,  
University of Hyderabad

# DECLARATION

We, **Himanshu Soni** and **Sagar Suman** hereby declare that this project report entitled “**Automated Compute Infrastructure for Organizations using OpenStack**” submitted by us under the guidance and supervision of **Dr. Anupama Potluri** is a bonafide work. We also declare that it has not been submitted previously in part or in full to this or any other University or Institution for the award of any degree or diploma.

Date:

Himanshu Soni

15MCMC13

Date:

Sagar Suman

15MCMC09

## Acknowledgements

It gives us an immense pleasure to express our deepest gratitude and sincere thanks to our highly respected and esteemed project supervisor Dr. Anupama Potluri for her motivation, support, guidance and above all the constant encouragement she has given us, without which it would not have been possible. Her insightful suggestions and fruitful discussions on every single aspect of this project is sincerely acknowledged.

We owe special thanks to Mr. Chandra Kesava, Director of Engineering, One Convergence Devices Pvt Ltd, whose expertise helped us to resolve the issues during the development phase of our project.

We thank Prof. Arun agarwal, Dean, SCIS, for providing excellent computing facilities and great environment for doing our project.

We are also grateful to the respected faculty members of SCIS for their constant support, valuable feedback and guidance.

We convey our thanks to the staff members of AI Lab for their help with issuing the resources.

Last but not the least, we would express our heartfelt wishes to our beloved parents and dear friends whose gracious solicitude helped us to complete this project successfully.

Himanshu Soni  
15MCMC13

Sagar Suman  
15MCMC09

# Abstract

Compute infrastructure in most organizations is maintained using Network Information Service (NIS) and Network File System (NFS). However, this requires a lot of configuration in all of the clients of NIS/NFS. Further, if the NIS/NFS server goes down, the entire system becomes inaccessible. Cloud computing allows for a centralized infrastructure that is easy to maintain, allows for fault tolerance through virtualization and requires minimal configuration compared to the NIS/NFS set up.

OpenStack is the most popular Infrastructure-as-a-Service (IaaS) platform which is also free and open source. Setting up and using OpenStack is still quite complex for both an IT administrator as well as a lay user. A lay user would like to log into his account without worrying about all the details of Virtual Machine (VM) hardware and software configuration and creating and launching VM instances. The IT administrator can shield the user from this by creating VM instances for every registered user but it makes her job cumbersome.

We have developed an application where every registered user has a profile attached to him. The profile determines the hardware and software requirements of the user as per the organization policy. The IT administrator needs to configure the profiles and the corresponding IT requirements. Our application creates VM instances for every new user automatically on user registration based on the profile of the user. When a registered user logs in, the user is directly presented with the console to their home logical volume or the standard OS GUI. Thus, the user's experience is the same as in NIS/NFS. We have also developed a resource monitoring tool using which the cloud administrator can track the users currently online and their resource usage. Whenever a user logs in, the VM of the user is launched only if there are sufficient resources available.

In conclusion, our application presents an interface to the user with which he is familiar in the NIS/NFS world by hiding all details of cloud computing from the lay user. It simplifies the job of a cloud administrator by automating the task of VM creation per user. It also helps her to monitor the usage of resources which helps in debugging problems of cloud infrastructure.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation . . . . .	1
1.2	Automated Compute Infrastructure for Organizations using OpenStack . . . . .	2
1.3	Organization of the Report . . . . .	3
<b>2</b>	<b>Technology Stack</b>	<b>4</b>
2.1	OpenStack [1] . . . . .	4
2.1.1	Components of OpenStack . . . . .	4
2.2	Python [2] . . . . .	5
2.3	PHP [3] . . . . .	5
2.4	MySQL [4] . . . . .	6
2.5	NGINX [5] . . . . .	6
2.6	VirtualBox [6] . . . . .	6
<b>3</b>	<b>Specification</b>	<b>7</b>
3.1	Introduction . . . . .	7
3.1.1	Scope . . . . .	7
3.1.2	Glossary . . . . .	8
3.2	Overall Description . . . . .	8
3.2.1	Product Functions . . . . .	8
3.2.2	User Characteristics . . . . .	9
3.2.3	Environmental and Technology Constraints . . . . .	9
3.2.4	Assumption and Dependencies . . . . .	10
3.3	Interface Requirements . . . . .	10
3.3.1	User Interface Requirements . . . . .	11
3.3.2	Administrator Interface Requirements . . . . .	11
3.4	Functional Requirements . . . . .	13
3.4.1	Use Case . . . . .	13
<b>4</b>	<b>Design and Implementation</b>	<b>14</b>
4.1	OpenStack Architecture . . . . .	14
4.1.1	Different Interfaces for accessing OpenStack Services . . . . .	14
4.2	Our application architecture . . . . .	16
4.2.1	High Level Data Flow Diagram of our application . . . . .	17
4.2.2	Entity-Relationship diagram of our Application . . . . .	18
4.3	Managing user profiles . . . . .	18
4.3.1	Creating profile using Horizon . . . . .	19
4.3.2	Creating profile using our application . . . . .	19
4.3.3	Algorithm for creating new profile . . . . .	21
4.4	User Sign-Up . . . . .	23

4.4.1	Creating account using Horizon . . . . .	23
4.4.2	User sign-up in our application . . . . .	24
4.5	Approval of user accounts . . . . .	28
4.5.1	Algorithm for user account approval . . . . .	28
4.6	Email validation . . . . .	30
4.6.1	Algorithm for email validation . . . . .	32
4.7	Sign-In . . . . .	32
4.7.1	Sign-In using Horizon . . . . .	32
4.7.2	Sign-In using our application . . . . .	33
4.8	Standard Sign-In . . . . .	33
4.8.1	Algorithm for Standard Sign-In . . . . .	36
4.9	Custom Sign-In . . . . .	36
4.9.1	Algorithm for Custom Sign-In . . . . .	37
4.10	Assigning Floating IP Addresses . . . . .	37
4.10.1	Assigning Floating IP Addresses through Horizon . . . . .	38
4.10.2	Assigning Floating IP Addresses in our Application . . . . .	39
4.10.3	Algorithm to Allocate Floating IP Addresses . . . . .	40
4.11	Resource Monitoring . . . . .	41
4.11.1	Algorithm for Resource Checking . . . . .	43
4.11.2	Algorithm for Resource Monitoring Dashboard . . . . .	43
<b>5</b>	<b>Conclusion</b>	<b>44</b>
5.1	Conclusion . . . . .	44
5.2	Future Work . . . . .	44
<b>A</b>	<b>Installation of OpenStack</b>	<b>46</b>
<b>B</b>	<b>Creating Cloud Image using VirtualBox</b>	<b>48</b>
<b>C</b>	<b>Installing Dependencies</b>	<b>50</b>

# List of Figures

3.1	Different Login Options . . . . .	10
3.2	Custom Configuration Selection . . . . .	11
3.3	Menu for uploading or Creating Profile(s) . . . . .	12
3.4	A sample configuration CSV file format to generate three profiles . . . . .	12
3.5	Menu for uploading or Creating Profile(s) . . . . .	12
4.1	OpenStack High Level Architecture . . . . .	15
4.2	Command for launching a VM . . . . .	15
4.3	Launching a VM using Horizon . . . . .	16
4.4	Application High Level Architecture . . . . .	17
4.5	Context diagram . . . . .	17
4.6	Level 0 Data Flow Diagram . . . . .	17
4.7	ER Diagram . . . . .	18
4.8	Profile creation using Horizon . . . . .	19
4.9	Profile creation form of our application . . . . .	20
4.10	Data Flow Diagram of profile creation . . . . .	20
4.11	Use Case Diagram for profile creation . . . . .	21
4.12	Activity Diagram for profile creation . . . . .	22
4.13	User creation form of Horizon . . . . .	23
4.14	Sign-up form of our application . . . . .	24
4.15	Data Flow Diagram for user Sign-Up . . . . .	25
4.16	Use Case Diagram for User Sign-Up . . . . .	25
4.17	Activity Diagram for User Sign-Up . . . . .	26
4.18	Approval of user account . . . . .	28
4.19	Activity Diagram for approval of user account . . . . .	29
4.20	DFD for approval of user account . . . . .	29
4.21	Email validation using 5 digit code . . . . .	30
4.22	DFD of email validation . . . . .	30
4.23	Use Case Diagram for email validation . . . . .	30
4.24	Activity Diagram for email validation . . . . .	31
4.25	Sign-in form of Horizon . . . . .	32
4.26	Home Screen after Sign-in in Horizon . . . . .	33
4.27	Sign-in form of our application . . . . .	34
4.28	Home Screen after Sign-in of our application . . . . .	34
4.29	Data Flow Diagram for Sign-in . . . . .	34
4.30	Activity Diagram for Sign-in . . . . .	35
4.31	Use Case Diagram for standard sign-in . . . . .	35
4.32	Use Case Diagram for custom sign-in . . . . .	36
4.33	Use Case Diagram for custom sign-in . . . . .	37
4.34	Choose “Associate Floating IP” from menu . . . . .	38
4.35	Allocate IP from public pool . . . . .	38
4.36	Associate to instance(VM) . . . . .	38



4.37 Associate IP to instance(VM) . . . . .	39
4.38 DFD of Floating IP assignment . . . . .	39
4.39 Use Case Diagram for Floating IP assignment . . . . .	39
4.40 Activity Diagram for Floating IP assignment . . . . .	40
4.41 Resource Monitoring Dashboard . . . . .	41
4.42 Data Flow Diagram for Resource Monitoring . . . . .	41
4.43 Use Case Diagram for Resource Monitoring . . . . .	42
4.44 Activity Diagram for resource monitoring . . . . .	42

# Chapter 1

## Introduction

Cloud computing is the on-demand delivery of compute power, database storage, applications, and other IT resources through a cloud services platform via the Internet [7].

Most of the organizations use a combination of **Network Information Service (NIS)** [8]/**The Lightweight Directory Access Protocol(LDAP)** [9] and **Network File System** [10] to setup their compute infrastructure. This approach is not so robust. The whole system depends on a central server. If the server goes down, none of the user directories will be accessible. Redundant servers can be set up but it involves more cost and maintenance and more configuration.

By using cloud computing the system can be made more reliable, robust, and flexible. Virtual machines(VMs) are created on the cloud server per user with desired hardware and software resources. All software needed can be bundled into an operating system image for the virtual machine. The VMs can be configured to run these pre-configured images. Thus, the IT administrator needs to configure and create the image once (or a few times depending on how many such images of different requirements need to be configured) which can be used for every user without any specific configuration per client machine. The other primary benefit of cloud infrastructure is that if the cloud server goes down, the VMs can be migrated to a different physical server transparently without any loss of information. In fact, the users will not even be aware that such a failure has occurred.

OpenStack [1] is an open source free software which provides Infrastructure-as-a-Service (IaaS), whereby virtual servers and other resources are made available to customers to set up a private cloud for their organization as required. It is one of the most popular softwares used to build cloud infrastructure. We have, therefore, chosen OpenStack as the software for use in this project.

### 1.1 Motivation

In OpenStack, when a registered user logs in, the Horizon dashboard is opened. The user can then create a Virtual Machine (VM) instance(s) of his/her choice. However, this requires knowledge of the cores, Operating System (OS) Images supported, RAM, Volumes etc., that a lay user cannot be expected to have knowledge of. To simplify the user interface, the administrator of an organization can create VM instances for every user of the organization. But, this is a cumbersome procedure.

This has motivated us to automate two processes: of creating VM instances per user type and launching them for a user as soon as the user logs in. Automate the process of creating VM instances per user based on the profile of the user, thus simplifying the job of an IT administrator.

## 1.2 Automated Compute Infrastructure for Organizations using OpenStack

Typically every organization has different types of users. Each type of user has different compute requirements. Based on this understanding, we plan to give support to the administrator to specify user types and the VM configuration needed for each type. When users register, based on the type of user, the VM instance will be automatically created. Secondly, when the user logs in, we will automatically run a VM instance of the specified configuration and present the user directly with a console/UI for the user to proceed with their work.

The features of our software are as follows:

- **Managing User Profiles:** The administrator can create user profiles using an input form or by uploading a configuration file. The inputs for profile are Profile name, OS, RAM, VCPU, HDD and Network. These values are used to create virtual networks and flavors in the OpenStack server. When a user creates an account, a profile is associated with it depending on the *User Type*. Further when the user Signs In, the same profile is used to launch a VM of the defined configuration.
- **Creating Accounts:** A user registers for an account by using a Sign-Up PHP [3] form. The input fields include *User Type*, *ID*, *Name*, *Email*, *Mobile* and *Password*. The *ID* and *Password* inputs are used for creating a user in OpenStack. The *User Type* input field is associated with the type of profile to be assigned to the user account. The profile contains configuration data - OS Image, Flavor and Network which are the mandatory fields while creating a new VM instance in OpenStack.
- **Sign-In and Authentication:** To access the user account, the user has to enter *username*, *password* and *Sign-in type* in the Sign-In form. These credentials are then authenticated by the keystone service of OpenStack. Once the user is authenticated, depending on the *Sign-In Type* – Custom or Standard – the respective operations are performed and then the user is redirected directly to the VM console.
- **Standard Sign-In:** If the user chooses the default ‘*Standard*’ Sign-In type in the Sign-In form, the default VM console is displayed on the user’s home screen. The configuration of this default VM is as per the profile defined by the administrator. These VMs are not assigned Floating IPs by default but the user can assign it using the home screen menu. The standard VMs are permanent, as the files and settings are retained even after the user Signs Out.
- **Custom Sign-In:** If the user chooses the ‘*Custom*’ Sign-In type in the Sign-In form, a hardware configuration input form is displayed on the screen. The user can choose the hardware specification as per their requirements and then sign in to get the VM console of the custom VM. These VMs are assigned Floating IPs by default. The custom VMs are temporary, as the files and settings are deleted along with the VM instance when the user Signs Out.
- **Resource Monitoring:** The resources can be monitored by the administrator using the admin panel dashboard. The resources include Number of Instances, VCPU, RAM and HDD. The dashboard displays the resources available and the resources being used in the form of pie charts. Also the the list of online users and the resources being used by them is displayed. The system also checks the set of

available resources every time a new VM is created. If the resources available are not enough for creating a new VM as per the configuration requested, the system displays an error.

- **Assigning Floating IPs:** A Floating IP is different from the IPs assigned to the VMs by OpenStack. Floating IP addresses can be instantly associated, dissociated or moved from one VM to another in the same datacenter. The IP addresses assigned by OpenStack can only be used by the VMs within the virtual network of OpenStack. However if we need to access a VM from outside the virtual network, we need to assign a floating IP to it. In our application a user can easily assign a floating IP to his/her VM by using a button on the Home Screen.

## 1.3 Organization of the Report

The rest of the report is organized as follows:

- Chapter 2 describes the Softwares and Tools we have used for developing the application.
- Chapter 3 describes the Software Specification Requirements.
- Chapter 4 describes the Detailed Design and Implementation of the application.
- Chapter 5 describes the Conclusion and the Future Work.

# Chapter 2

## Technology Stack

In this chapter, we give a brief introduction of OpenStack and its components. Then we describe the various softwares and programming languages we have used for developing our application.

### 2.1 OpenStack [1]

OpenStack is a set of software tools for building and managing cloud computing platforms for public and private clouds. OpenStack controls large pools of compute, storage, and networking resources throughout a datacenter.

#### 2.1.1 Components of OpenStack

There are several components of OpenStack. Every component provides a specific service. The components of OpenStack which we have used in our project are:

1. **Nova**

Nova is a cloud computing fabric controller, which manages pools of computer resources and works with virtualization technologies and high-performance computing configurations. Nova provides flexibility to design the cloud with no proprietary software or hardware requirements.

2. **Neutron**

Neutron manages LANs, Dynamic Host Configuration Protocol, and Internet Protocol both versions 4 and 6 in OpenStack. Users can define networks, subnets, and routers to configure their internal topology, and then allocate IP addresses and VLANs to these networks. Neutron uses Floating IP addresses to allow users to assign an external IP address to their virtual machine (VM).

3. **Glance**

Glance provides support for VM images, specifically the system disks to be used in launching VM instances. It also provides facility to take snapshots and backups of the VMs.

4. **Cinder**

Cinder manages block-level storage called volumes that VMs use. A VM can use more than one volume that are all managed by Cinder.

5. **Keystone**

Keystone provides authentication to all the OpenStack services. It currently supports token-based authN and user-service authorization.

## Project

A project (also known as tenant) comprises of a group of related users. It has a quota of resources (compute, network, storage, etc.) available for its users. The users can have different roles (profiles) within a project.

## 2.2 Python [2]

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python has many features that make it very popular like:

- Easy to learn
- Easy to read
- Easy to maintain
- A broad standard library
- Interactive Mode
- Portable

OpenStack provides Python SDK to access its services. The SDK implements Python bindings to the OpenStack API, which enables a programmer to perform automation tasks in Python by making calls on Python objects rather than making REST (refer section 3.1.2) calls directly.

We have created Python scripts using the OpenStack SDK for each of the required OpenStack services.

## 2.3 PHP [3]

PHP stands for Hypertext PreProcessor. PHP is a powerful and widely-used open source server-side scripting language to write dynamically generated web pages. PHP scripts are executed on the server and the result is sent to the client as plain HTML.

Advantage of PHP over other server-side languages:

- Easy to learn
- Open source
- Portability
- Fast performance

We have used PHP scripts for form processing and database operations.

## 2.4 MySQL [4]

MySQL is a relational database management system, which organizes data in the form of tables. MySQL is one of the many database servers based on RDBMS model, which manages a set of data that requires three specific things – data structures, data integrity and data manipulation. With MySQL co-operative server technology we can realize the benefits of open, relational systems for all the applications. MySQL makes efficient use of all system resources, on all hardware architectures to deliver unmatched performance, price performance and scalability.

MySQL is used in this project to manage database of users, pending requests, user profiles and online users.

## 2.5 NGINX [5]

NGINX is a free, open-source, high-performance HTTP Server. NGINX is known for its high performance, stability, rich feature set, simple configuration, and low resource consumption.

OpenStack does not allow hosting any scripts other than its own on its default Apache [11] HTTP Server. Therefore we have used NGINX as our web server because we have hosted our PHP scripts on the same machine on which OpenStack is installed.

## 2.6 VirtualBox [6]

VirtualBox is a powerful cross-platform Virtualization Software for x86-based systems. Using VirtualBox, we can create and run multiple Virtual Machines, each running a different operating system, on the same computer at the same time.

OpenStack creates VMs using cloud images of operating systems registered with Glance Service. There are several cloud images available on the Internet, but VMs created by those images require key-pair mechanism to login. We needed images in which a user can login using UserID-Password. So we created images of different operating systems using VirtualBox.

# Chapter 3

## Specification

This chapter describes the specifications for our application. The chapter follows the IEEE template [12] for software requirement specifications for the application.

### 3.1 Introduction

In OpenStack, when a registered user logs in, the Horizon dashboard is opened. The user will then have to create a VM instance(s) of her choice. However, this requires knowledge of the cores, OS images supported, RAM, volumes etc., that a lay user cannot be expected to have knowledge of. To simplify the user interface, the IT staff of an organization can create VM instances for every user of the organization. But, this is a cumbersome procedure.

We did a comparison between the services possible with OpenStack and Amazon Web Services (AWS) [13], the cloud service operated by Amazon. We obtained a trial version for AWS and studied the UI presented to a typical registered user by Amazon and compared it with the Horizon UI of OpenStack and found it to be similar.

This has motivated us to automate two processes: of creating VM instances per user type and launching them for a user as soon as the user logs in. Typically every organization has different profiles for users, each profile having a certain compute configuration. Based on this understanding, we have built software that requires the IT staff of an organization to simply specify user profiles and the VM configuration needed for each profile. When users register, based on the type of the user, the corresponding profile is used to create a VM instance automatically by our software. Secondly, when the user logs in, we will automatically run a VM instance of the specified configuration and present the user directly with a console/UI for the user to proceed with their work.

Thus, we are simplifying the process of VM creation and running of it for a lay user and hiding all details of cloud compute infrastructure from lay users. We are also making the task of cloud administration easier for the administrator by automating the creation of VM instances for every user based on the pre-defined profiles of the users.

#### 3.1.1 Scope

This application can be used by any organization which needs a cloud based infrastructure for its members. The application is designed to provide

- ease of administration
- ease of use to a lay user using the cloud infrastructure to access compute resources.



### 3.1.2 Glossary

Term	Description
Administrator	Person who decides roles, creates profiles based on roles, monitor the resources
Instance	Virtual machine
REST	an application program interface (API) that uses HTTP requests to GET, PUT, POST and DELETE data.
Image	The cloud version of operating system in QCOW2 format.
Floating IP	An IP address from the pool of defined range to be assigned to the virtual machines
OpenStack	A free and open-source software platform for cloud computing
OpenlabDB	The MySQL database for the application to maintain its state. It stores all the information required by the application.
VM	Virtual Machine
Standard VM	A VM generated as per the user profile setup by the administrator.
Custom VM	A VM generated as per the users' specified configuration in the login panel.

## 3.2 Overall Description

The product is an open source free software released under the GNU general Public License. It is a web based system implementing OpenStack cloud infrastructure. The product provides simple mechanism for users to create and use virtual machines.

### 3.2.1 Product Functions

The product functions include the following:

- **Registration:** Any person within that organization can register themselves to get a Standard VM based on the role of the person.
- **Login:** A user has to login to use his/her VM.
- **Custom VM:** Any registered user can create a VM of custom configuration.
- **Resource Monitoring:** The System Administrator will be able to monitor the resources. The Custom or Standard VMs will only be created if the resources are available.
- **Assigning Floating IP:** Any of the VMs created by the user can be assigned a floating IP. The Floating IP assigned can be used to SSH into the VM from an external physical host.

### 3.2.2 User Characteristics

There are two types of users who will use the system:

1. Administrator
2. Members of the organization(users)

#### User Characteristics of Administrator:

- The administrator is responsible for creating profiles, approving requests, monitoring the resources.
- The administrator will have a web interface through which he/she can easily do all the tasks without worrying about the OpenStack internals.
- The administrator should have the basic knowledge of operating systems, hardware resources to use the application.

#### User Characteristics of users:

- The users will access the application using a web browser on computer.
- An user friendly interface is provided such that a non-technical user will also be able to use a VM from his/her computer.
- They should have knowledge of using a web browser.
- Users require a valid email account so that information regarding their their account can be delivered to them.

### 3.2.3 Environmental and Technology Constraints

#### Hardware Constraints

- **Processor:** quad core or higher processor
- **Memory:** minimum 16 GB of RAM is needed.
- **HardDisk:** Minimum Requirement: 500 GB
- **Network:** Ethernet 10Gb connection with working Internet

#### Software Constraints

It is recommended to use **PYTHON 3.5.0** and **PHP 7.1.0** versions respectively for reasons of portability and flexibility to choose a variety of vendor products.

It is recommended to use OpenStack - **version Rocky** as underlying layer for the application to avoid DB inconsistencies.

- OpenStack - <https://git.openstack.org/cgit/openstack-dev/devstack>
- Python - <https://www.python.org/ftp/python/3.6.5/Python-3.6.5.tar.xz>
- PHP - [www.php.net/](http://www.php.net/)

### 3.2.4 Assumption and Dependencies

- There should be access to an OpenStack cloud.
- A physical host with web-server installed to host PHP files.
- There should be an access to MySQL database.
- A medium-bandwidth Internet connection.
- PHP-FRM (FastCGI Process Manager)
- PHP-MYSQL (PHP to Database Connectivity)
- Python OpenStack-SDK
- NGINX Web server.

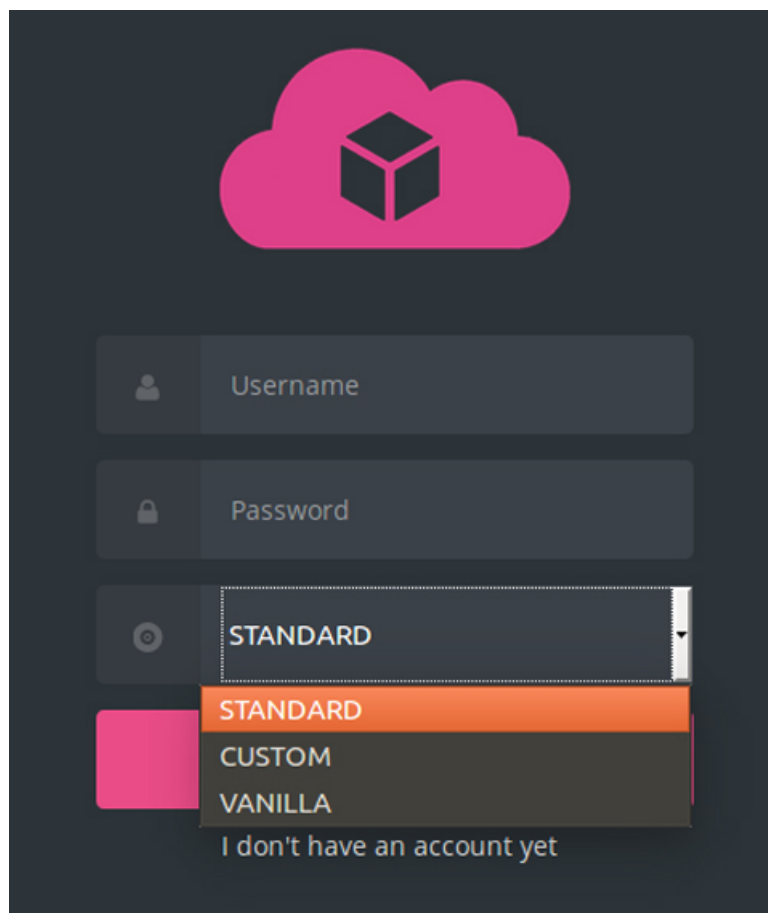


Figure 3.1: Different Login Options

## 3.3 Interface Requirements

This section lists all the interfaces for both users and administrators which are all primarily web based. Screenshots of web pages are included to demonstrate the interfaces.

### 3.3.1 User Interface Requirements

1. **Sign In:** The Sign In form consists of three Input Fields - *Username*, *Password* and *SignIn-Type*
  - **SignIn-Type:** There are three options available in the dropdown list:
    - **Standard:** The VM is launched as per the user profile.
    - **Custom:** The user is prompted to specify the resources needed for the VM using slider UI construct which makes it easy for user to specify their needs as shown in Fig. 3.2.
    - **Vanilla:** The user is redirected to the Horizon Web UI.

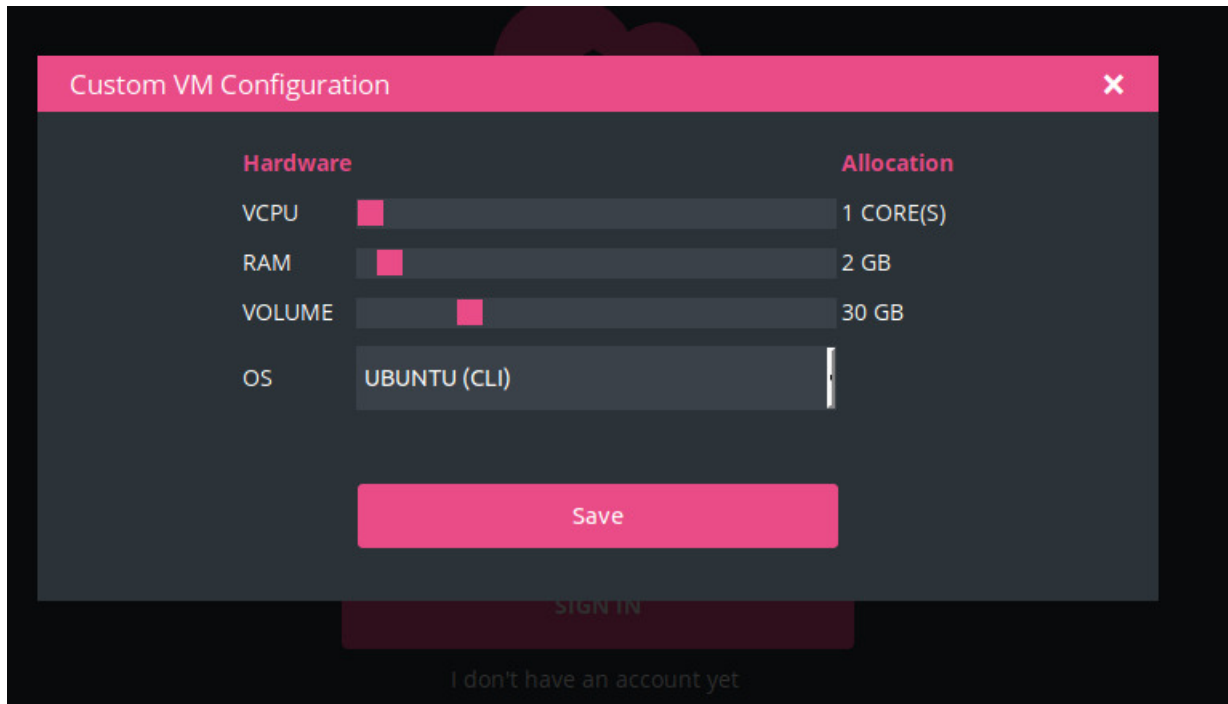


Figure 3.2: Custom Configuration Selection

2. **Sign Up:** The Sign Up form consists of 6 Input Fields - *User-Type*, *Registration ID*, *Name*, *Mobile*, *Email* and *Password*. Mobile number is optional, but rest of the inputs are mandatory.

### 3.3.2 Administrator Interface Requirements

1. **Manage Profile:** The Manage profile form allows the administrator to create profiles per user role.

Figure 3.3: Menu for uploading or Creating Profile(s)

	A	B	C	D	E	F
1	name	os	ram	cpu	hdd	network
2	student	ubuntu	5	4	10	net1
3	research_scholar	cirros	3	1	5	net1
4	faculty	centos	10	7	30	net2

Figure 3.4: A sample configuration CSV file format to generate three profiles

TYPE	OS	VCPU	RAM	HDD	DELETE
FACULTY	cirros-0.3.5-x86_64-disk	5 CORES	4096 MB	10 GB	
STUDENT	cirros-0.3.5-x86_64-disk	2 CORES	2048 MB	5 GB	

Figure 3.5: Menu for uploading or Creating Profile(s)

## 3.4 Functional Requirements

### 3.4.1 Use Case

Actor	Use Case	Goal
User	Sign up	To create an account in the system so that a VM can be created for him/her
User	Sign in (standard)	To sign into the system to use the standard VM
User	Sign in (custom)	To sign into custom mode so that the user can create VM of his specific requirements
User	Assign Floating IP	To assign a public floating IP to his VM
Admin	Validate request	To validate an account creation request
Admin	Create profile	To create resource profile
Admin	Monitor	To monitor the resources being used at any given point of time

# Chapter 4

## Design and Implementation

In this chapter we discuss about the design and implementation of our project in detail. We begin with the details of OpenStack architecture and the operations which can be performed using OpenStack Web UI and CLI.

The design and implementation aspects have been discussed under following sections:

- Data Flow Diagram
- Use Case Diagrams
- Activity Diagrams
- Database design and operations
- Algorithms

### 4.1 OpenStack Architecture

OpenStack is an open source software system to deploy cloud services mostly as Infrastructure-as-a-Service (IaaS).

OpenStack manages the hardware pool of an infrastructure using its various services - Nova, Cinder, Glance, Neutron and Keystone(discussed in Chapter 2.1.1).

OpenStack uses RESTful APIs to facilitate interaction between its components. The inputs from Horizon or CLI (discussed in the following sections) are converted into REST calls which are then authenticated by Keystone for checking user privileges. Once the REST calls are authenticated, they are redirected to the targeted services.

OpenStack uses MySQL Database to maintain system state. Information regarding instances, projects, networks, images etc. are stored in database tables.

#### 4.1.1 Different Interfaces for accessing OpenStack Services

OpenStack provides two types of user interfaces:

1. **Command Line Interface:** Every service of OpenStack can be accessed by using CLI. User can use commands to perform various operations like - creating user, uploading cloud images, creating flavors, creating networks, creating instances etc. But performing operations using these commands can be quite cumbersome and requires complete knowledge of the OpenStack components.

Typically, this is an interface used by OpenStack administrators.

A typical command to launch a VM is shown in figure 4.2

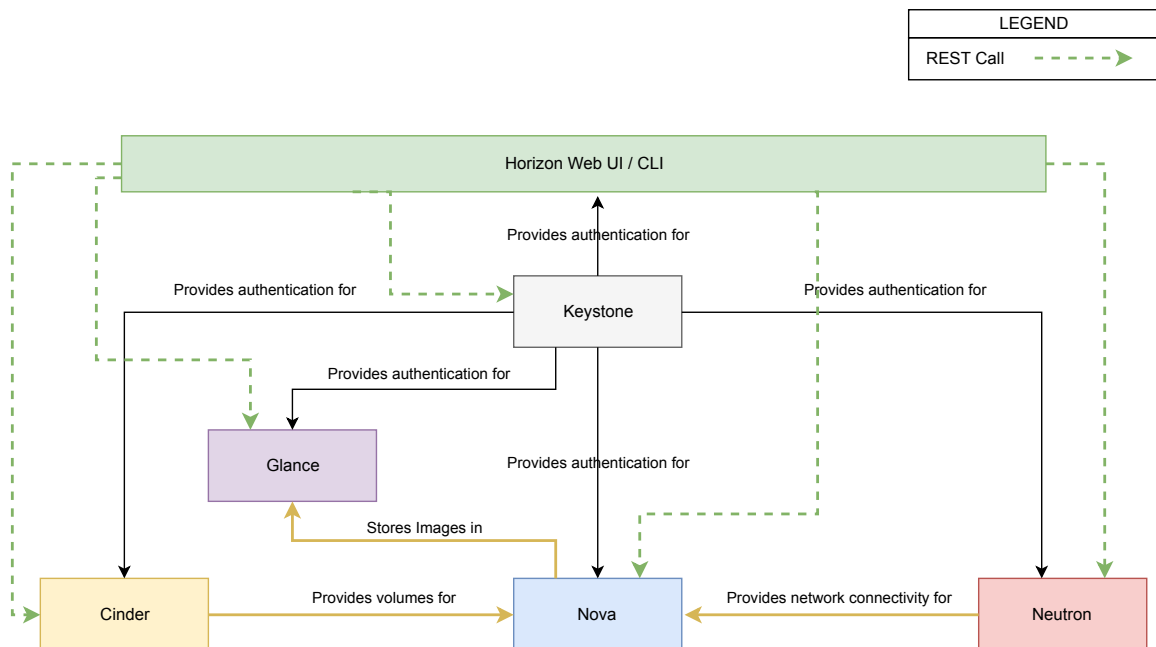


Figure 4.1: OpenStack High Level Architecture

A terminal window titled "Home" shows a user named himanshu@HS-PC running the following command to create a VM:

```
himanshu@HS-PC:~$ openstack server create --flavor m1.tiny --image $(openstack image list | grep cirros | cut -f3 -d '|') --nic net-id=$(openstack network list | grep private | cut -f2 -d '|' | -d ' ') --security-group default myVM
```

Figure 4.2: Command for launching a VM

2. **Web Interface:** OpenStack also provides a web interface known as **Horizon**. Horizon can access different services of OpenStack by using REST API. Each service provided by OpenStack supports a number of operations and a huge list of settings. To accomodate these settings and operations as an option in the UI, Horizon offers multi-step forms under seperate tabs while creating a VM. As a result the process of creating a VM using OpenStack requires detailed understanding of several components.

Fig. 4.3 shows the procedure for launching a VM using the Horizon UI. The steps involved are:

- **Instance name:** Name of the VM to be created
- **Choose image:** The operating system for the VM
- **Choose flavor:** Hardware configuration for the VM



**Launch Instance**

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**

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

**Available 3**

Select one

Click here for filters.

Name	Updated	Size	Type	Visibility
ubuntu_cli	4/24/18 10:09 AM	277.25 MB	qcow2	Public
lubuntu	4/24/18 10:09 AM	2.09 GB	qcow2	Public
cirros-0.3.5-x86_64-disk	4/24/18 9:56 AM	12.65 MB	qcow2	Public

Cancel < Back Next > Launch Instance

Figure 4.3: Launching a VM using Horizon

- **Choose network:** Virtual network created within the OpenStack to be attached with the VM
- **Choose network port:** Facility to add another network (if available) for the VM
- **Choose key-pair:** A private-public key-pair so that the VM can be accessed through SSH.
- **Choose security group:** Security group is a set of TCP/IP rules that sets the security policies for access to the VM(?).
- **Choose server group**
- **Choose metadata (not mandatory):** Metadata is useful for accessing instance-specific information from within the instance.

**Note:** Most of the fields are not frequently used.

## 4.2 Our application architecture

The architecture of our application is based on the underlying services provided by OpenStack. We are using Python OpenStack SDK for making REST Calls to the various OpenStack services. The different operations required while using OpenStack services have been divided into Python modules. Whenever an operation is required, the Python Modules are called using PHP scripts. The inputs from the users are fetched using PHP forms which are then passed into the Python modules. The Python modules generate REST Calls to interact with OpenStack Services and the values are returned to the PHP

Page. These returned values might be displayed as human readable format for the users or can be used for maintaining the state of the system.

The MySQL Server being used by OpenStack contains all the databases to maintain the state of its various services, users and resources. We have used the same MySQL database for maintaining the state of our system and storing relevant information.

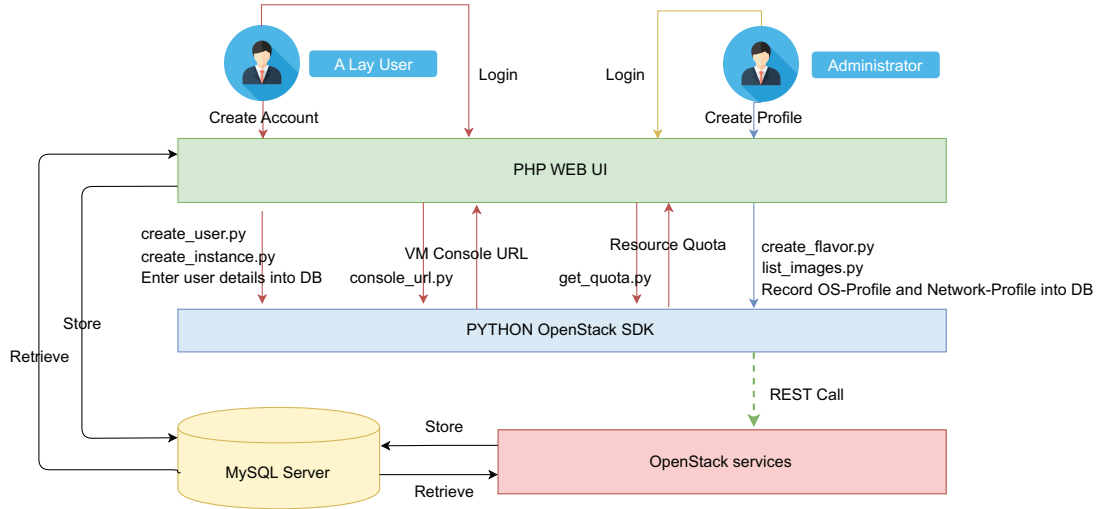


Figure 4.4: Application High Level Architecture

A High Level architecture of our application is shown in figure 4.4.

#### 4.2.1 High Level Data Flow Diagram of our application

The high level Data Flow Diagram [14] for our application is shown in Fig. 4.6 and the context diagram in Fig. 4.5.

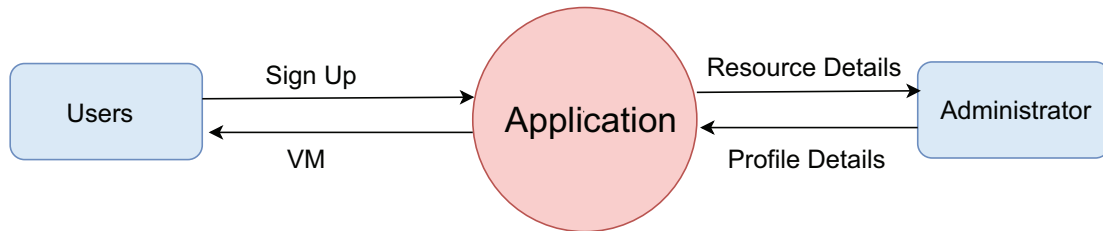


Figure 4.5: Context diagram

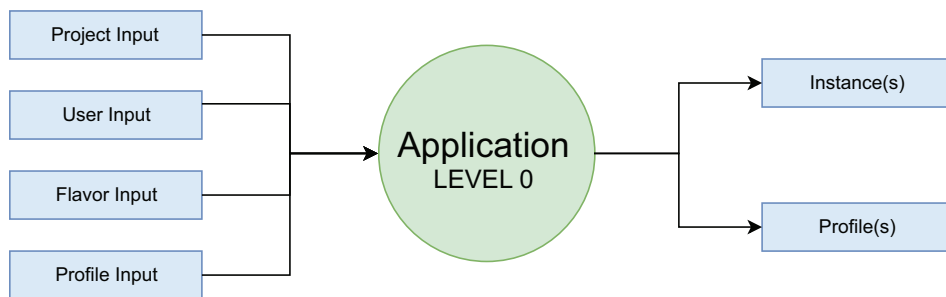


Figure 4.6: Level 0 Data Flow Diagram

## 4.2.2 Entity-Relationship diagram of our Application

Entity-Relationship (ER) [15] diagrams show the relationships between various entities in an application. The ER diagram for our application is shown in Fig. 4.7.

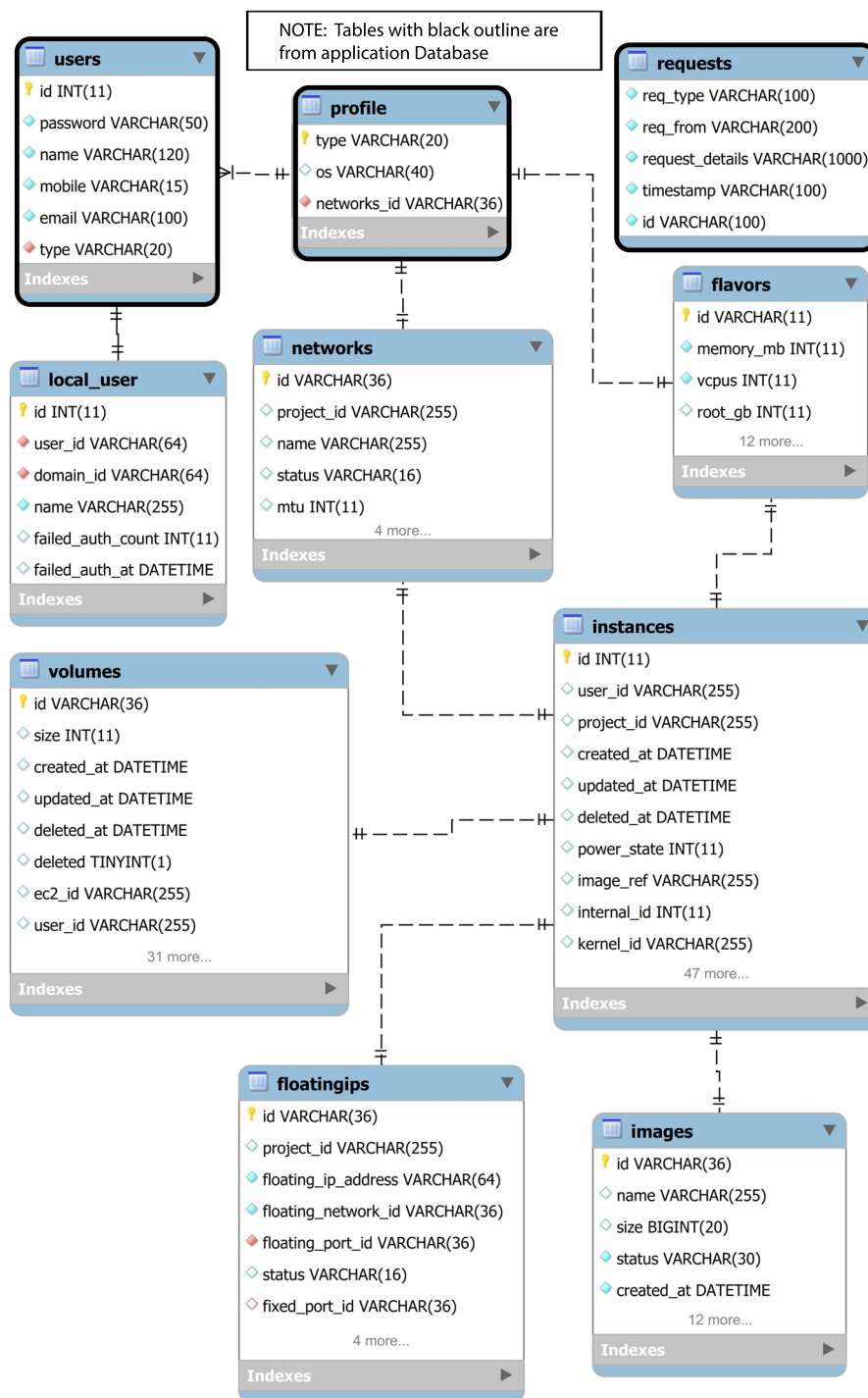


Figure 4.7: ER Diagram

## 4.3 Managing user profiles

There can be different kinds of members in an organization whose requirements may be quite different in terms of resources required. Thus, the administrator would like to create VMs for the members based on their requirements rather than use a one-size-fits-all

VM configuration. For example - in a university different kinds of users can be faculty, staff, students, researchers and administrators of the university. Each one needs different amount of resources. The organization may have a policy that decides how much resources should be allocated to a particular user type. Based on this policy document, the IT administrator can create different profiles for different users which will be used to create VMs with the corresponding resources. In OpenStack terminology, the administrator will have to create different *flavors*.

### 4.3.1 Creating profile using Horizon

While using Horizon, administrator has to create flavors one by one for every user type. The administrator cannot create multiple profiles at once. A typical profile(flavor) creation in Horizon is shown in figure 4.8.

Figure 4.8: Profile creation using Horizon

### 4.3.2 Creating profile using our application

Our application provides two ways for creating of profiles:

1. Input through a web form.
2. Upload a configuration file(CSV file) containing details of profiles.

A snapshot of profile creation GUI is shown in figure 4.8.

The DFD of the profile creation is given in Fig. 4.10 and the use case diagram in Fig. 4.11.

Figure 4.9: Profile creation form of our application

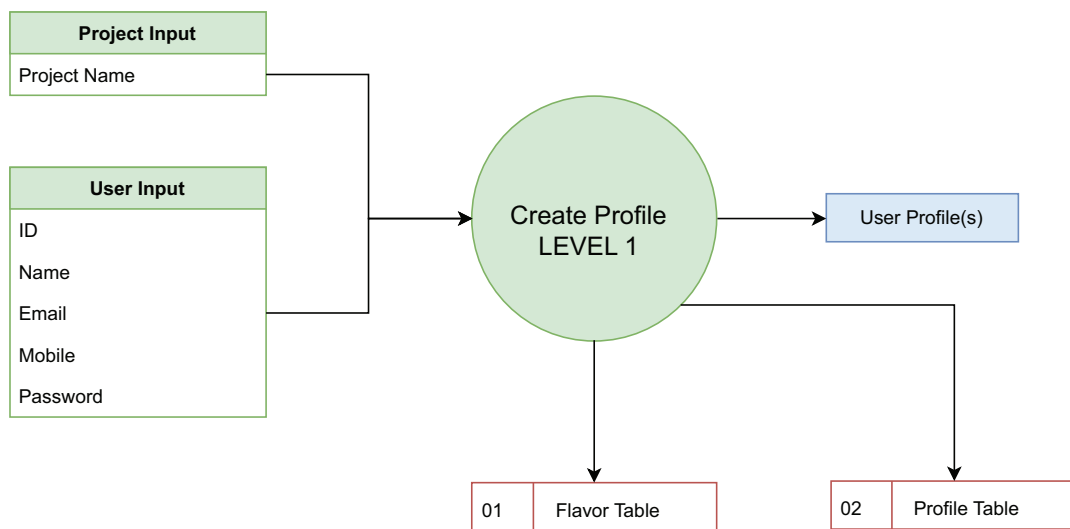


Figure 4.10: Data Flow Diagram of profile creation

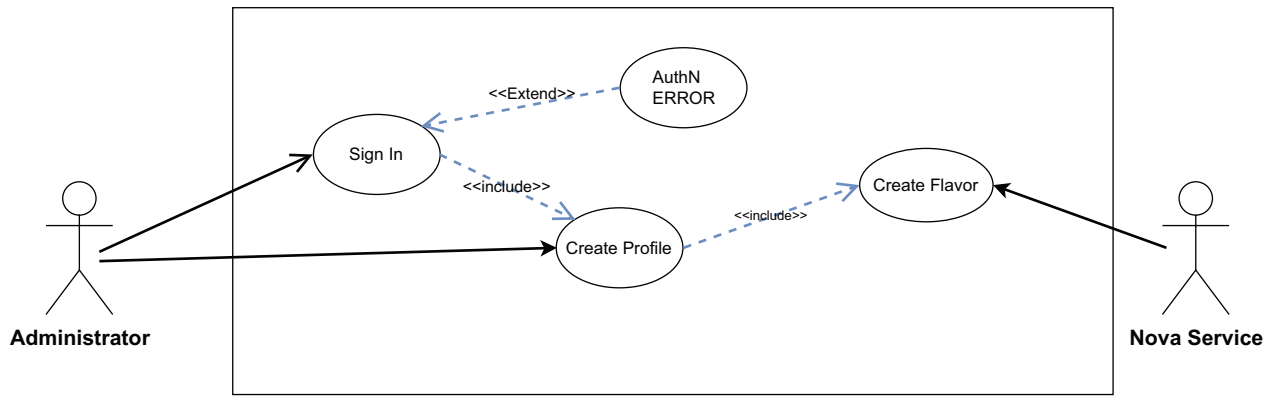


Figure 4.11: Use Case Diagram for profile creation

### 4.3.3 Algorithm for creating new profile

---

**Algorithm 1:** New profile creation

---

**Input** : Profile\_Name, OS, RAM, CPU, HDD

**Output:** New profile in database

---

- 1 Get User Input (Profile\_Name, OS, RAM, CPU, HDD)
  - 2 Get Project\_Name from application configuration file
  - 3 Run Create\_Flavor script with inputs (Name\_default, RAM, VCPU, HDD)
  - 4 **if** *Create Profile is successful* **then**
  - 5 | Save Data Inputs (Profile\_Name, OS, Network) into Profile Table
  - 6 **else**
  - 7 | display Error Message
  - 8 | exit
  - 9 **end**
-

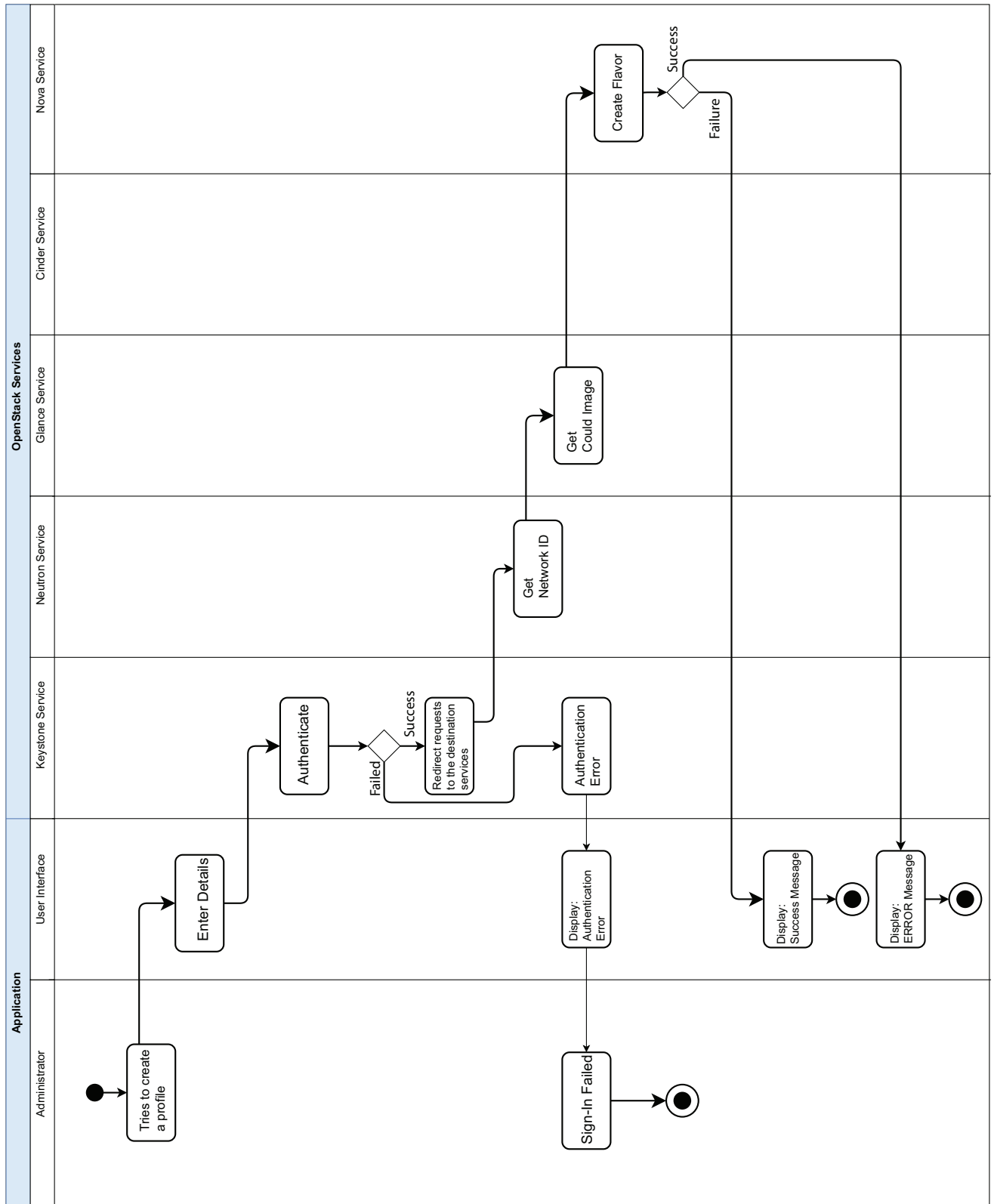


Figure 4.12: Activity Diagram for profile creation

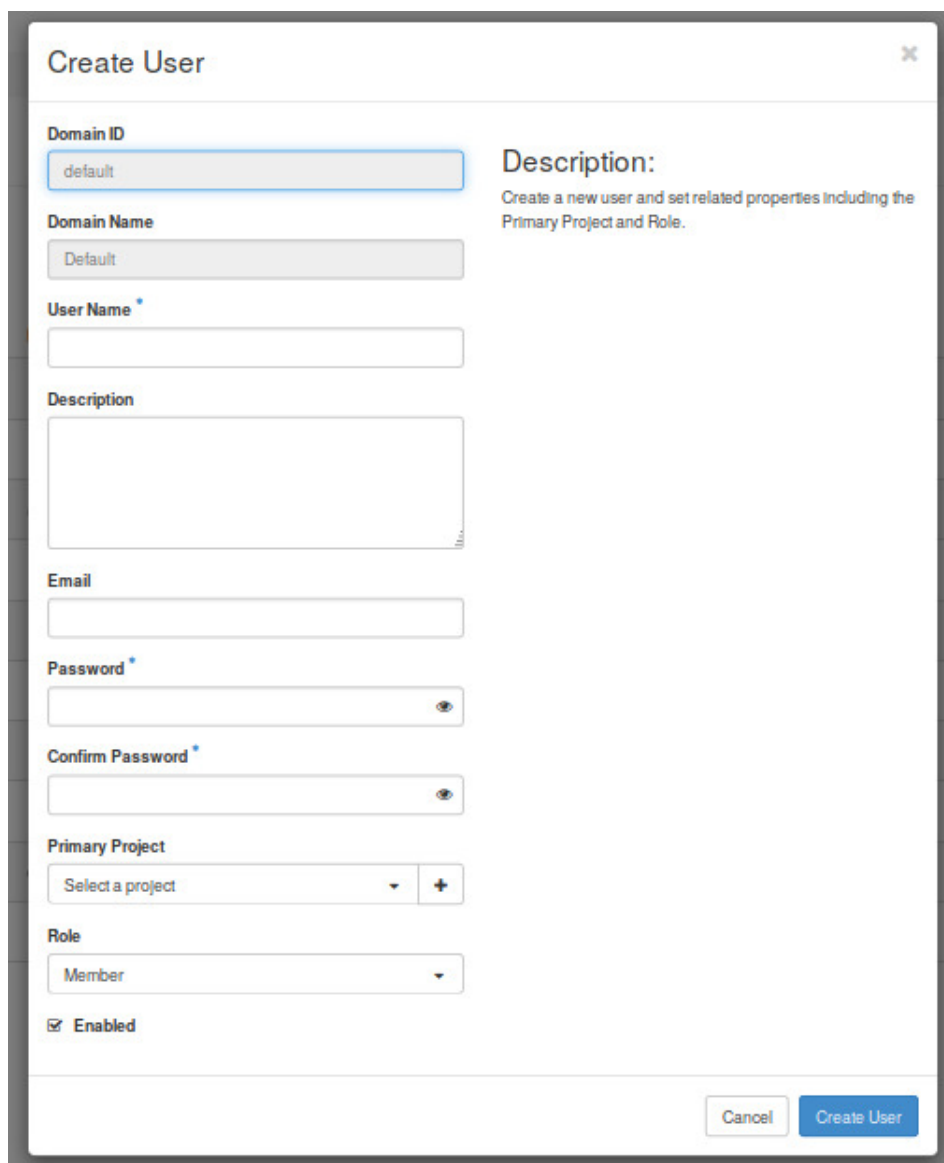
## 4.4 User Sign-Up

Any member of an organization needs to have an account on the cloud server to launch a VM. In vanilla OpenStack, the IT administrator has to create accounts for users. Then the user ID and password are given to the user by the administrator.

### 4.4.1 Creating account using Horizon

The administrator can create a user account using Horizon UI using the form shown in Fig. 4.13. This approach has two problems:

- For all the users in the organization, the administrator has to create accounts manually. It is a very tedious task.
- In Horizon form, there is a dropdown menu for adding users in a project. The Administrator can accidentally add users to another project.



The screenshot shows the 'Create User' form in the Horizon UI. The form is titled 'Create User' and has a close button (X) in the top right corner. It contains the following fields and controls:

- Domain ID:** A text input field with the value 'default'.
- Domain Name:** A text input field with the value 'Default'.
- User Name:** A text input field with an asterisk (\*) indicating it is required.
- Description:** A large text area for entering a description.
- Email:** A text input field.
- Password:** A text input field with an asterisk (\*) indicating it is required and a toggle icon for visibility.
- Confirm Password:** A text input field with an asterisk (\*) indicating it is required and a toggle icon for visibility.
- Primary Project:** A dropdown menu with the text 'Select a project' and a plus (+) button to add a new project.
- Role:** A dropdown menu with the value 'Member'.
- Enabled:** A checkbox that is checked, with the label 'Enabled'.

At the bottom right of the form, there are two buttons: 'Cancel' and 'Create User'.

Figure 4.13: User creation form of Horizon



### 4.4.2 User sign-up in our application

We have designed a simple sign-up form for creating user accounts. We solved the two problems given above when using Horizon UI in the following ways:

- The users will fill details by using sign-up form and an account creation request will be sent to administrator. The Administrator now only has to approve the request which comes from the sign-up form.  
**NOTE:** However if the user is basic (employee, student etc.), the administrator's approval will not be required.
- We have removed the *project* option from the form. Instead the project name is setup once during the installation of application and all the accounts are created within that project.

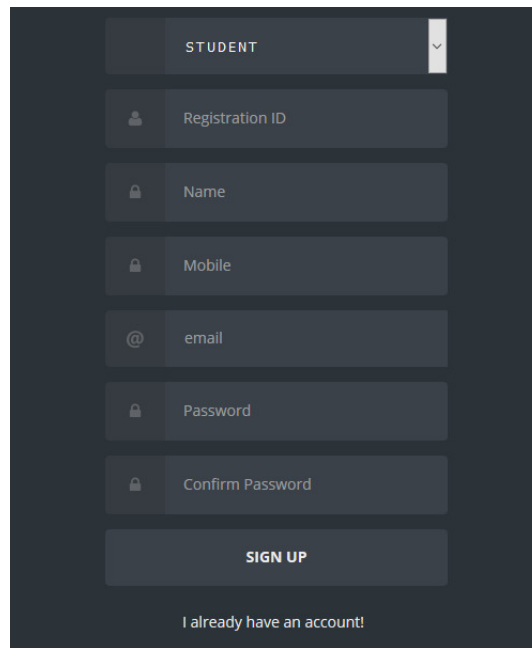
The image shows a dark-themed user sign-up form. At the top, there is a dropdown menu currently set to 'STUDENT'. Below this are several input fields, each with a small icon on the left: 'Registration ID' (person icon), 'Name' (lock icon), 'Mobile' (lock icon), 'email' (at-sign icon), 'Password' (lock icon), and 'Confirm Password' (lock icon). At the bottom of the form is a large 'SIGN UP' button. Below the button is a link that says 'I already have an account!'.

Figure 4.14: Sign-up form of our application

The sign-up form of our application is shown in figure 4.14.

The data flow diagram, use case diagram and activity diagram for the user sign-up operation are given in Fig. Fig. 4.15, Fig. 4.16 and Fig. 4.17 respectively.

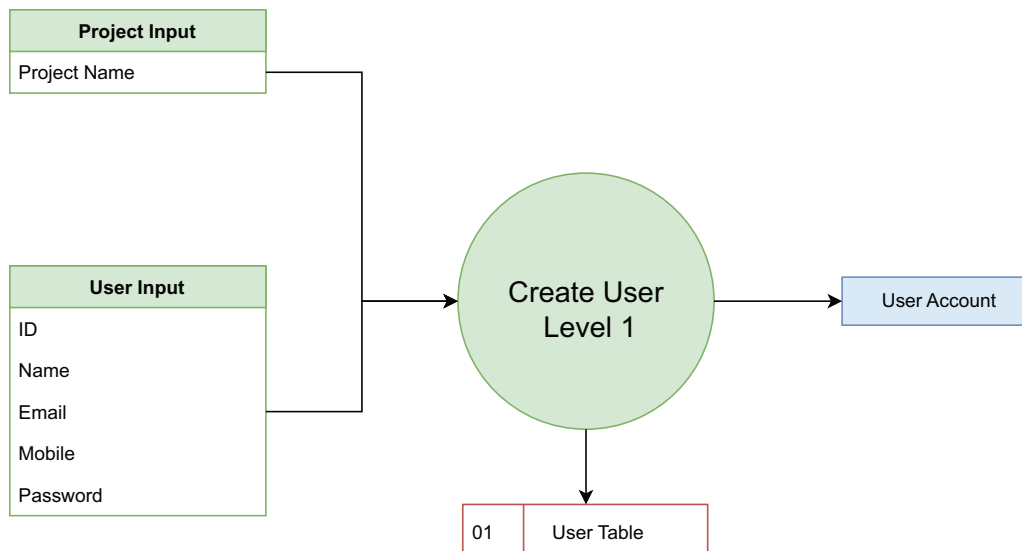


Figure 4.15: Data Flow Diagram for user Sign-Up

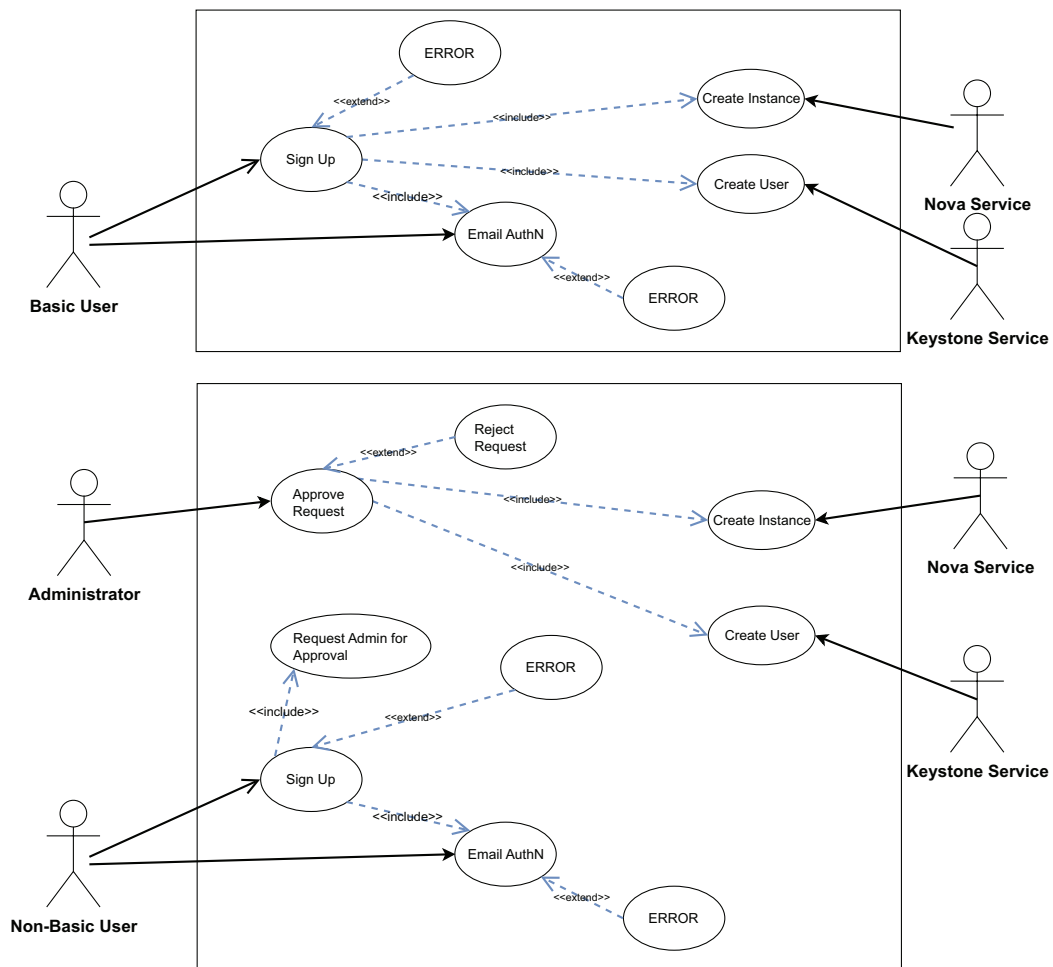


Figure 4.16: Use Case Diagram for User Sign-Up

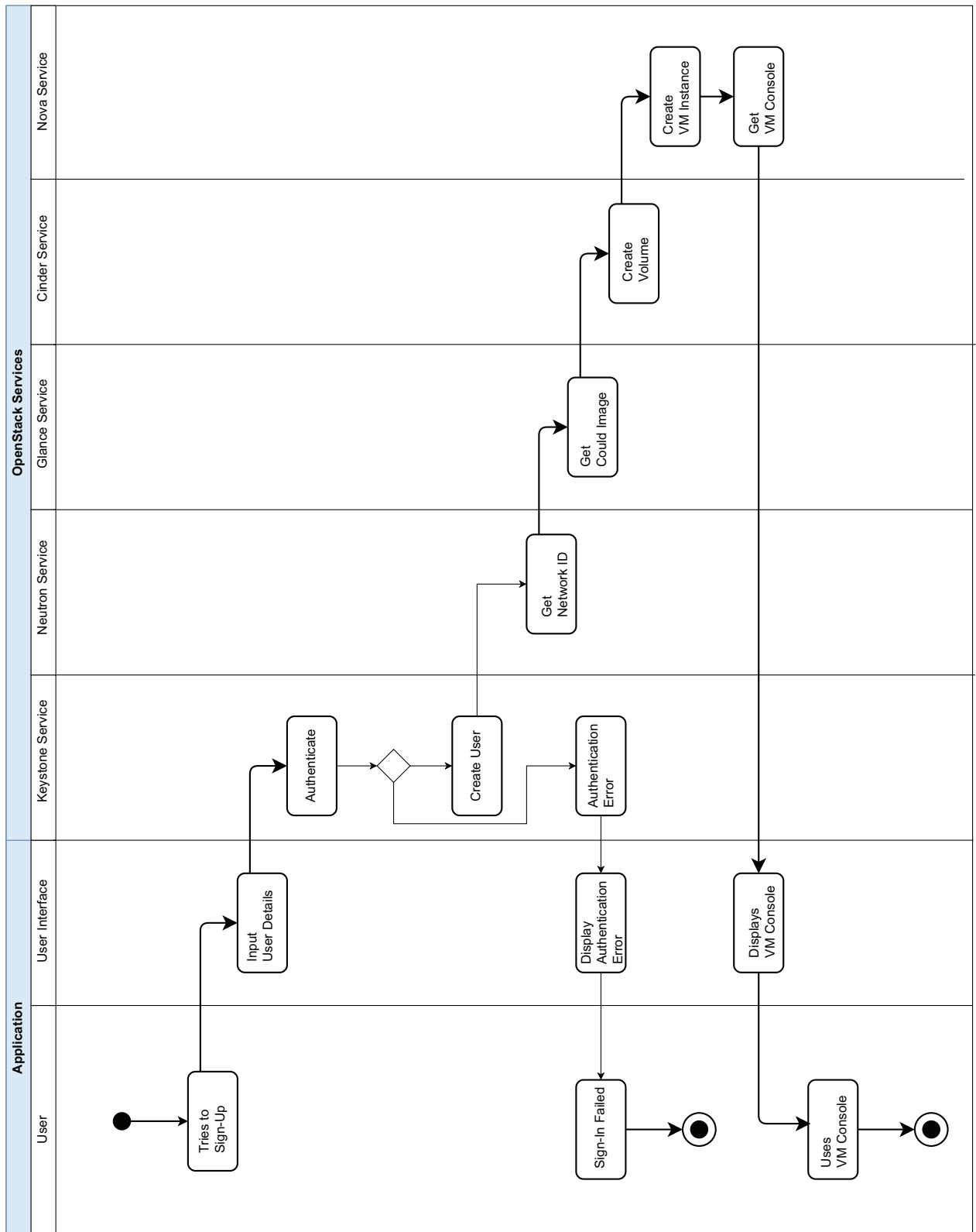


Figure 4.17: Activity Diagram for User Sign-Up

---

**Algorithm 2:** Algorithm for sign-up

---

**Input** : Type, ID, Name, Email, Mobile, Password

**Output:** url of VM

```
1 get profile_name using input (type) and return (OS, Network, Flavor_Name)
2 run Create_Instance with inputs (Name_default, OS, Flavor_Name, Network)
3 match 5 digit activation code from the user email
4 if Code matches then
5   | Continue to step 4
6 else
7   | display Error Message
8   | exit
9 end
10 if User Type is NOT student then
11   | Save data into Requests Table for admin approval
12   | exit
13 end
14 if Create Instance is successful then
15   | save Data Inputs (Type, ID, Name, Email, Mobile) into Users Table
16 else
17   | Display Error Message
18 end
```

---

## 4.5 Approval of user accounts

The system allows different kinds of users to create an account by using the *User Type* input. However this also makes the system vulnerable to impersonation. To prevent such exploitation the non-basic users (e.g., Faculty/Staff/administrators in a University), who require higher resource will require a one-step authentication from the IT administrator. This allows the administrator to ensure that the request is made by an authorized user.

Also for an existing user, the administrator can choose to delete, reset or block the account.

A screenshot of user approval form is shown in figure 4.18.

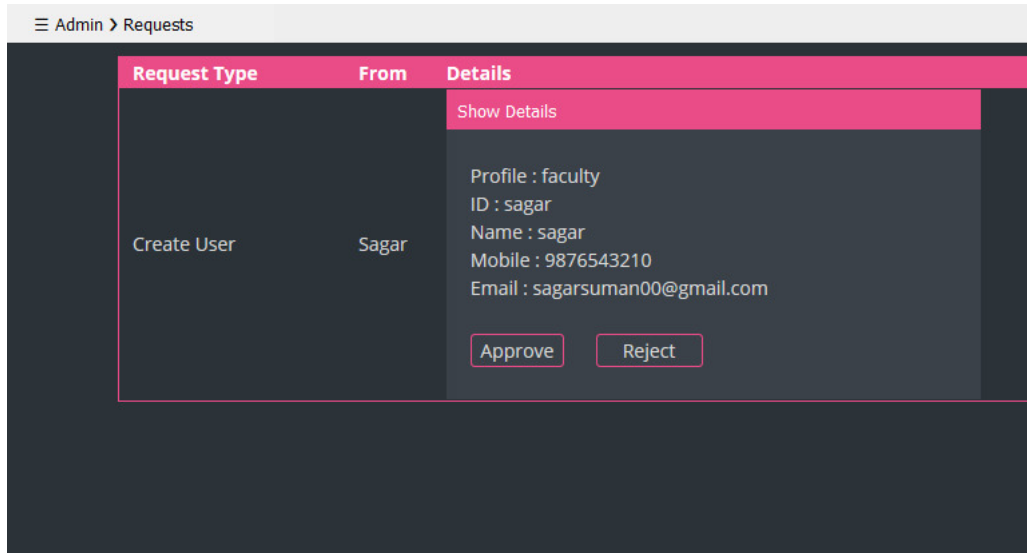


Figure 4.18: Approval of user account

The use case diagram, activity diagram and data flow diagram for the approval operation are given in Fig. 4.16, Fig. 4.19 and Fig. 4.20 respectively.

### 4.5.1 Algorithm for user account approval

---

**Algorithm 3:** User Account Approval

---

**Input** : Request Details

**Output:** console url/rejection message

```
1 fetch Request details from table
2 if request is approved then
3   create user
4   create Instance
5   send Success email to user
6 else
7   send rejection email to user
8   exit
9 end
10 delete request entry from table
```

---

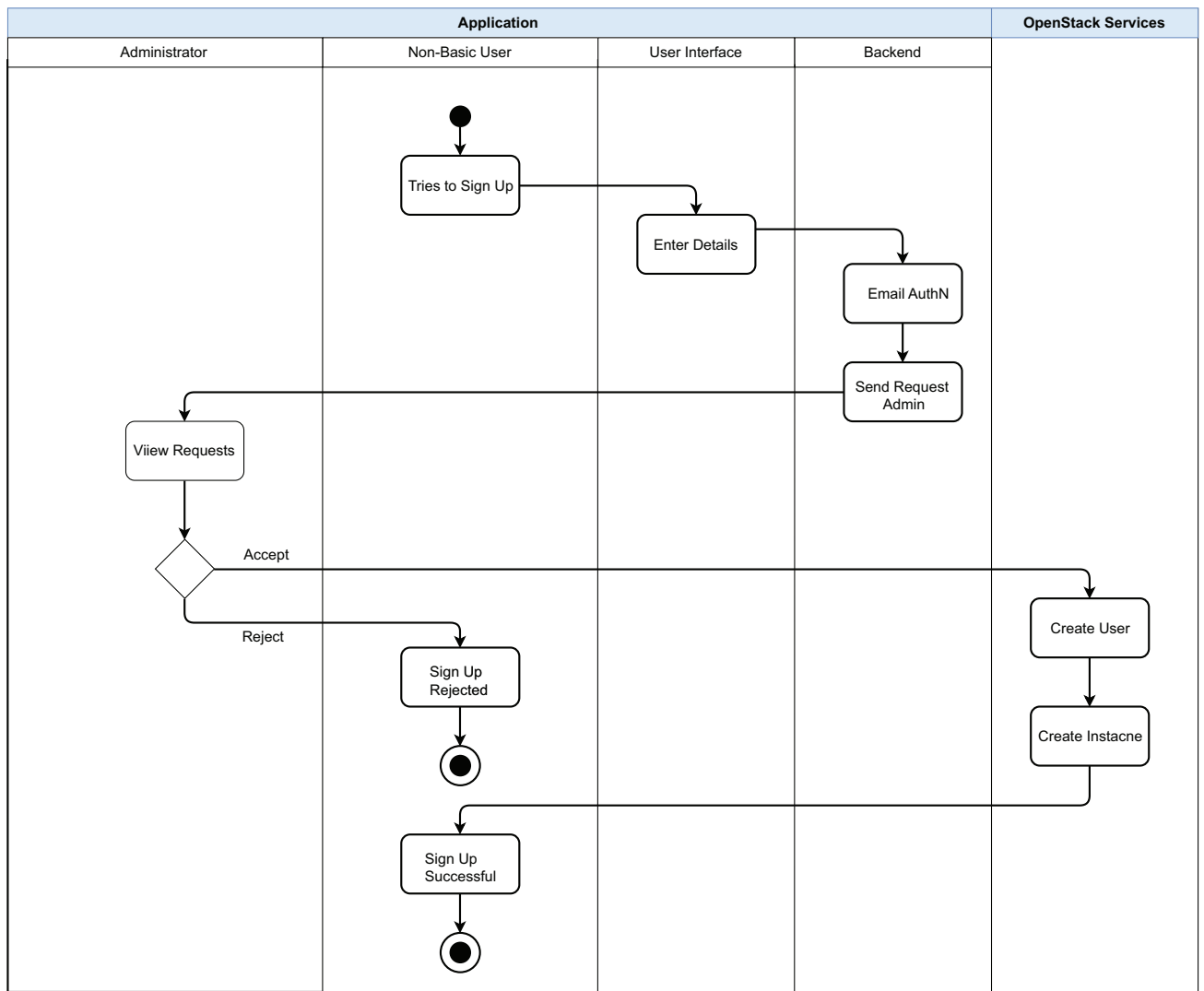


Figure 4.19: Activity Diagram for approval of user account

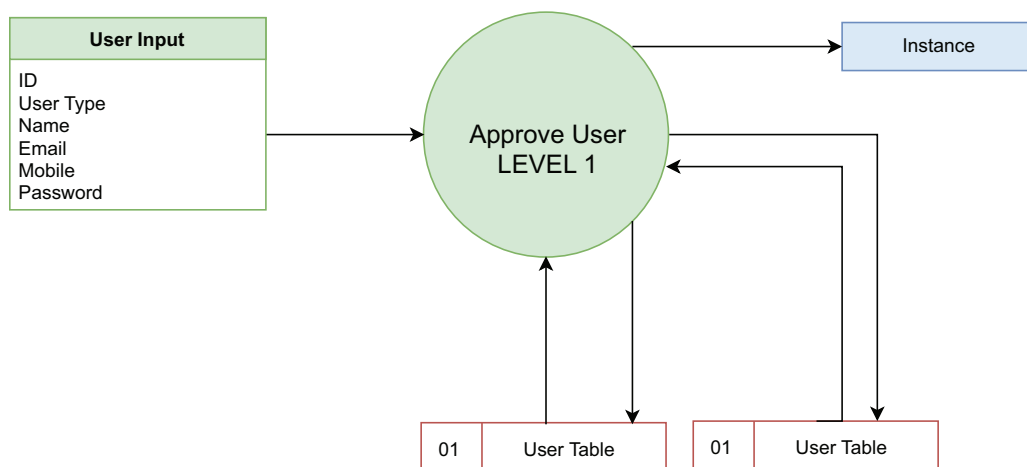


Figure 4.20: DFD for approval of user account

## 4.6 Email validation

The cloud administrators often need to send notification emails to the users. Therefore it is very important to ensure that the Email IDs are accessible and active. The user's email ID is verified using a five digit verification code while creating a user account. Fig. 4.21 shows the form used for this validation.

Greetings!

Hi sagar,

We've received your request for creating account linked with Registration ID **SAGAR**.  
An activation code has been sent on your Email ID **sagarsuman00@gmail.com**, kindly enter the 5 digit code to complete the Sign Up process.

0 0 0 0 0

VERIFY ACTIVATION CODE

Figure 4.21: Email validation using 5 digit code

The Level 1 DFD for the email validation is shown in Fig. 4.22.

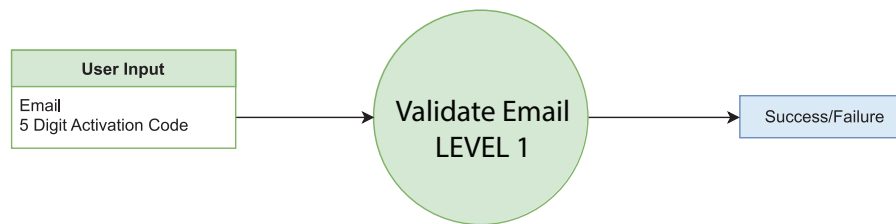


Figure 4.22: DFD of email validation

The use case and activity diagrams for the same are shown in Fig. 4.23 and Fig. 4.24 respectively.

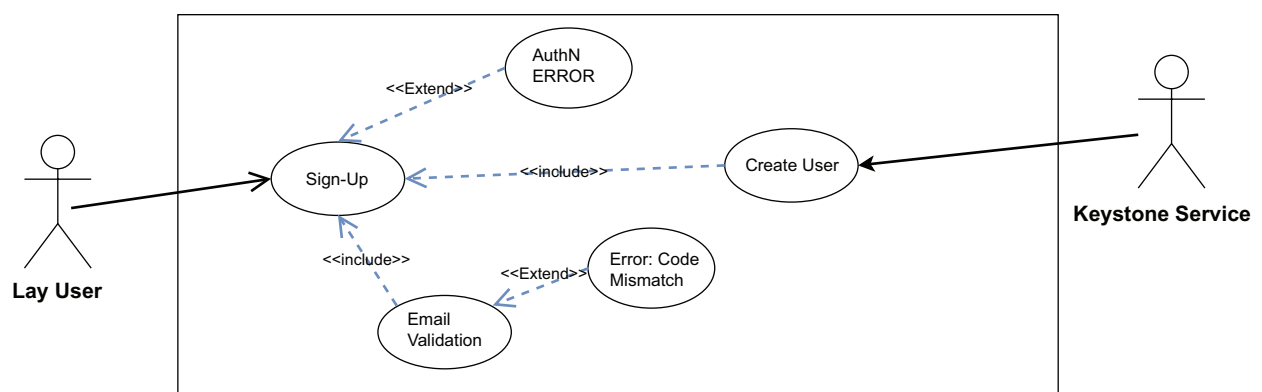


Figure 4.23: Use Case Diagram for email validation

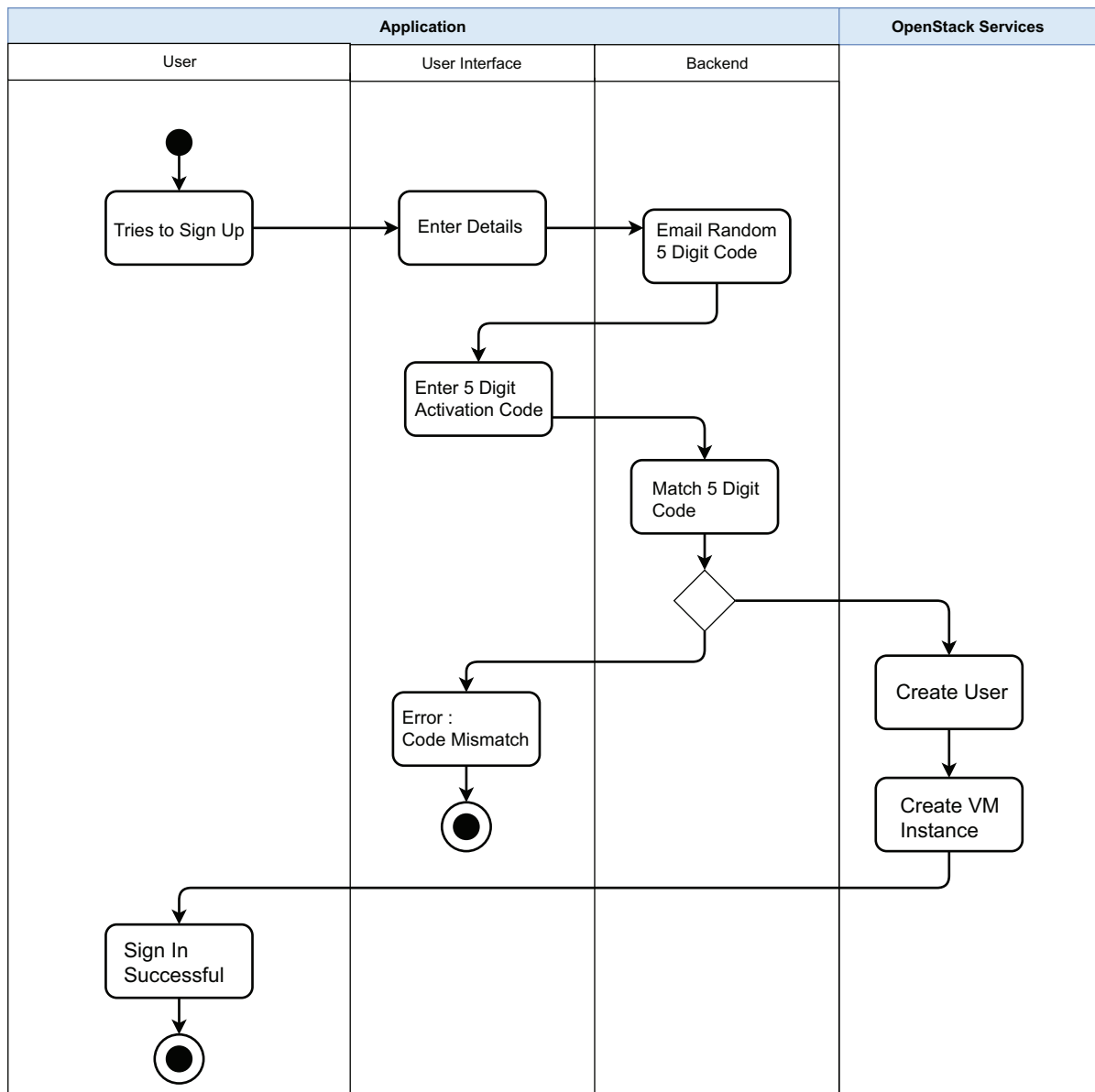


Figure 4.24: Activity Diagram for email validation



### 4.6.1 Algorithm for email validation

---

**Algorithm 4:** Email validation

---

```
1 generate 5 digit Random code
2 send 5 digit code to email ID
3 take 5 digit input from user
4 if user_code = generated_code then
5   | return true
6 else
7   | return false
8 end
```

---

## 4.7 Sign-In

This is used by already registered users to log into the cloud infrastructure.

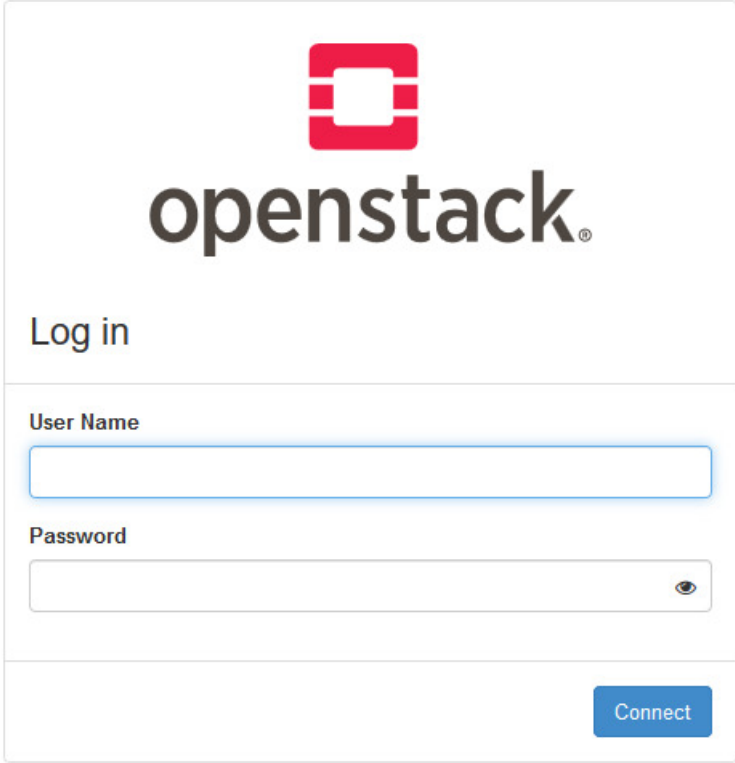
The image shows the OpenStack login interface. At the top is the OpenStack logo, which consists of a red square with a white 'O' inside, followed by the word 'openstack' in a dark grey, lowercase, sans-serif font. Below the logo is the text 'Log in' in a blue, sans-serif font. Underneath is a form with two input fields. The first field is labeled 'User Name' in a small, dark grey font and has a light blue border. The second field is labeled 'Password' in a small, dark grey font and has a light grey border with a small eye icon on the right side to toggle visibility. At the bottom right of the form is a blue button with the word 'Connect' in white, sans-serif font.

Figure 4.25: Sign-in form of Horizon

### 4.7.1 Sign-In using Horizon

In Horizon, the home screen after log-in is not so friendly. When a user logs into the system, the VM is not available directly.

To access the VM, the user has to go through a number of steps:

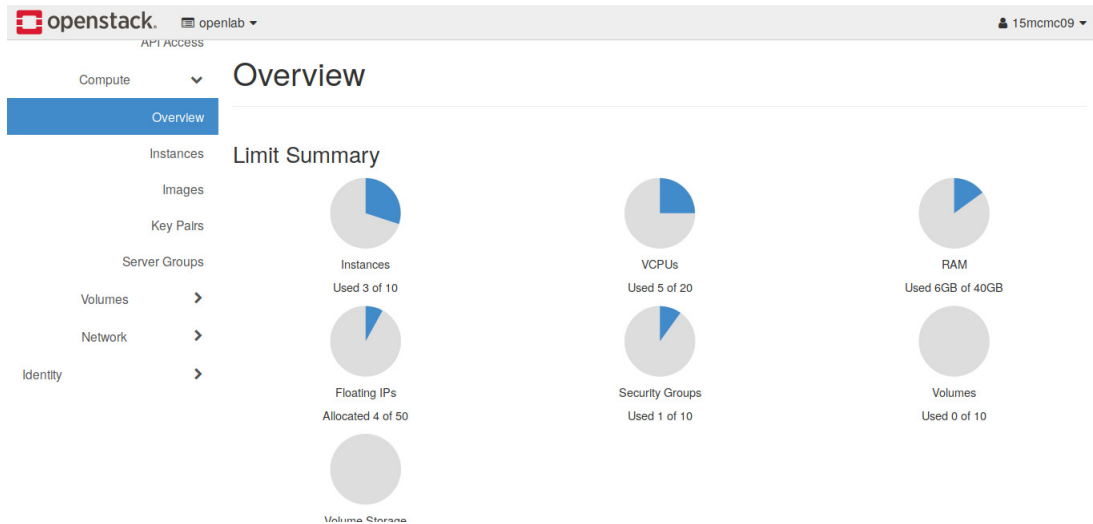


Figure 4.26: Home Screen after Sign-in in Horizon

1. As soon as user logs in, the “Resource Overview Summary” screen is displayed.
2. Now the user has to click on the **project** tab.
3. Then click on the **compute** tab.
4. Then he/she has to select **instances** from the menu.
5. After that the user has to find his/her VM from the **instance list**.
6. Finally from the drop-down menu of user’s VM, the **console** is accessed.

A screenshot of the horizon after log-in is shown in Fig. 4.26.

### 4.7.2 Sign-In using our application

To access the user account, the user has to enter *username*, *password* and *Sign-in type* in the Sign-In form as shown in Fig. 4.28. These credentials are then authenticated by the Keystone service of OpenStack. Once the user is authenticated, depending on the *Sign-In Type* – Custom or Standard – the corresponding VM with the required configuration is launched and then the user is directed to the VM console.

## 4.8 Standard Sign-In

If the user chooses the default ‘*Standard*’ Sign-In type in the Sign-In form, the default VM console is displayed on the user’s home screen. The configuration of this default VM is as per the profile defined by the administrator. The standard VMs are permanent, as the files and settings are retained even after the user signs out. The data flow diagram, activity diagram and use case diagram for the standard sign-in is shown in Fig. 4.29, Fig. 4.30 and Fig. 4.31.

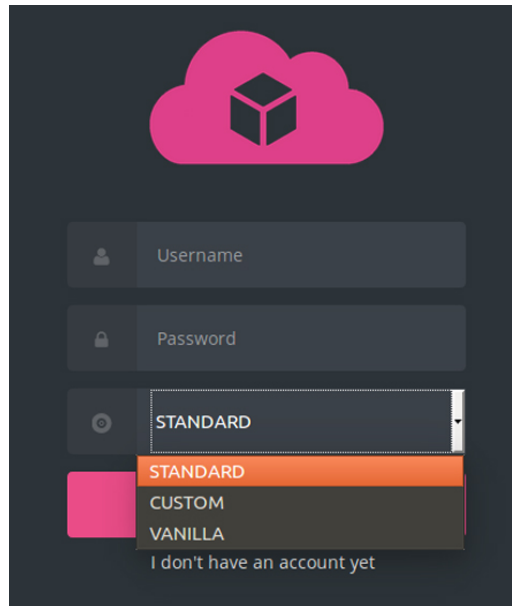


Figure 4.27: Sign-in form of our application

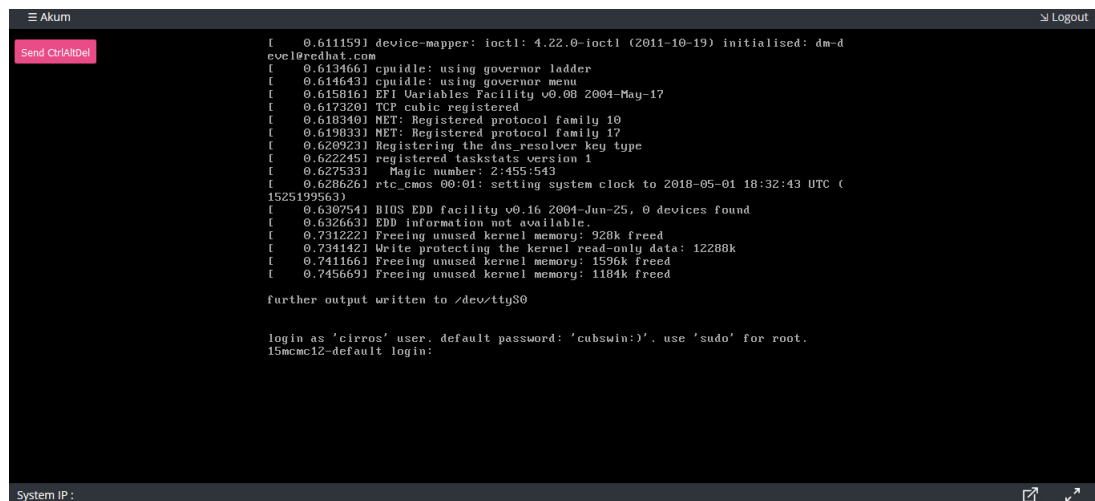


Figure 4.28: Home Screen after Sign-in of our application

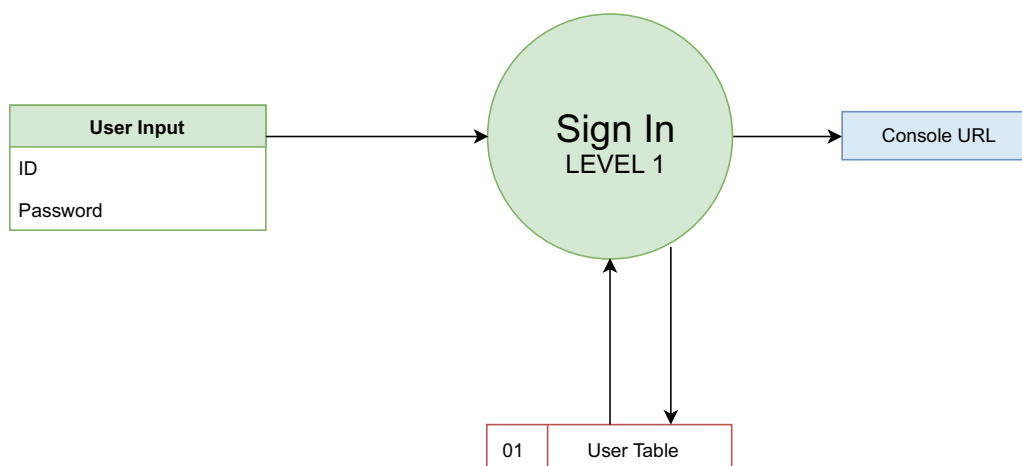


Figure 4.29: Data Flow Diagram for Sign-in

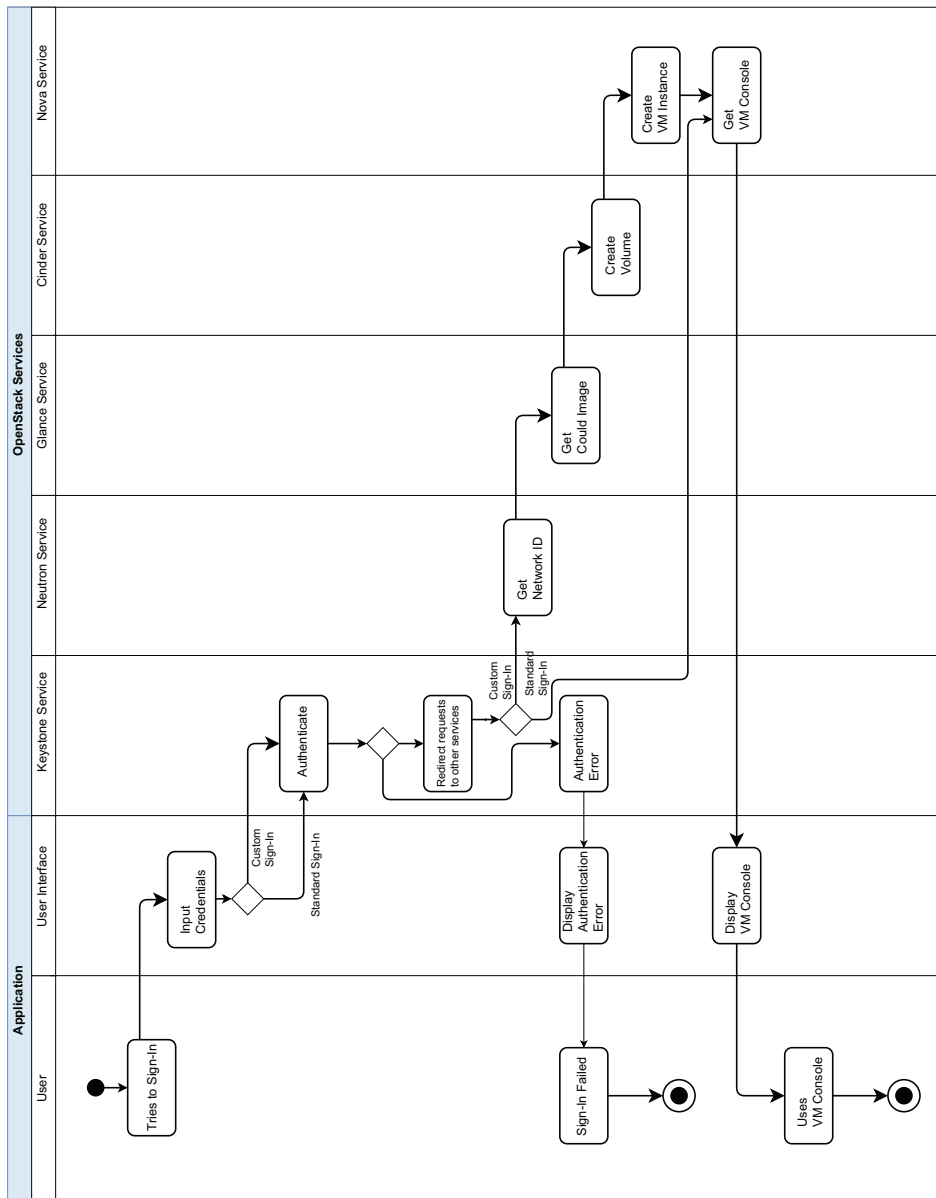


Figure 4.30: Activity Diagram for Sign-in

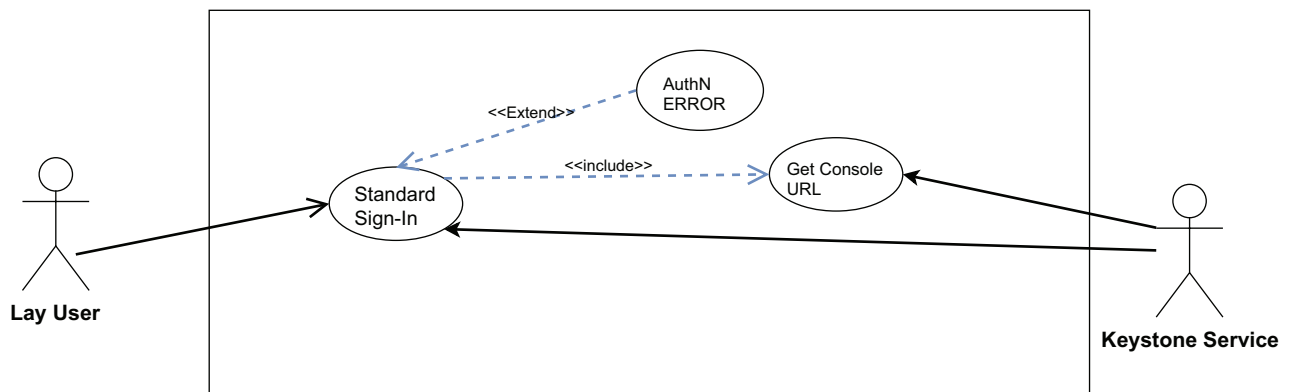


Figure 4.31: Use Case Diagram for standard sign-in

## 4.8.1 Algorithm for Standard Sign-In

---

**Algorithm 5:** Standard Sign-In

---

**Input** : id, password, type

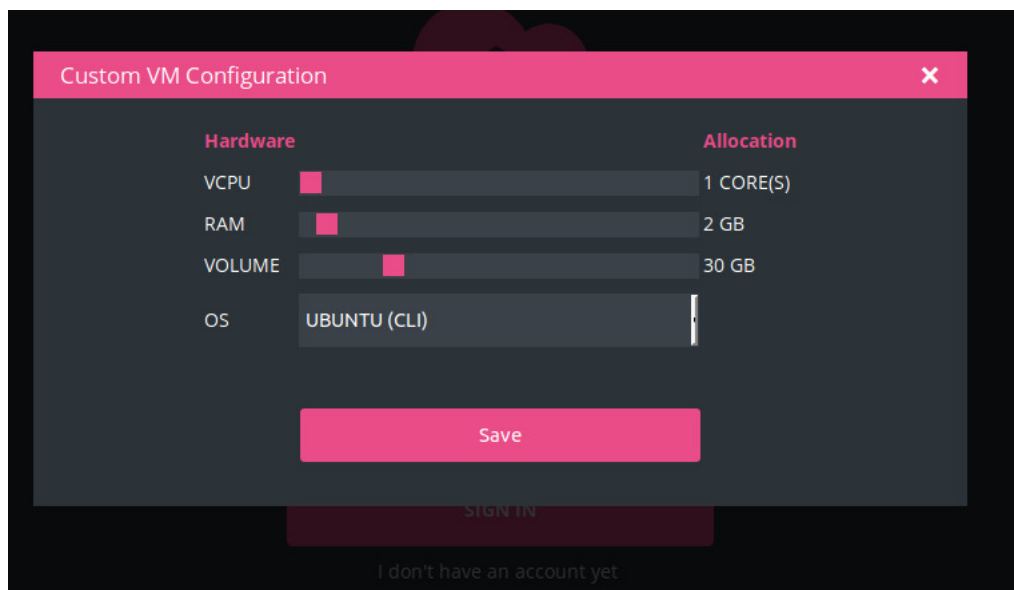
**Output:** VM console

```
1 get user-id, password, type
2 validate user-id, password
3 if invalid then
4   | display error message
5   | exit
6 end
7 if type = standard then
8   | return console
9 end
```

---

## 4.9 Custom Sign-In

If the user chooses the ‘*Custom*’ “Sign-In type” in the Sign-In form, a hardware configuration input form is displayed on the screen as shown in



The image shows a 'Custom VM Configuration' dialog box with a pink header and a dark grey body. It contains a table with two columns: 'Hardware' and 'Allocation'. The table has four rows: 'VCPU' with a slider set to 1 and '1 CORE(S)' in the allocation column; 'RAM' with a slider set to 2 and '2 GB' in the allocation column; 'VOLUME' with a slider set to 30 and '30 GB' in the allocation column; and 'OS' with a dropdown menu showing 'UBUNTU (CLI)'. Below the table is a pink 'Save' button. At the bottom of the dialog, there is a 'SIGN IN' button and a link that says 'I don't have an account yet'.

Hardware	Allocation
VCPU	1 CORE(S)
RAM	2 GB
VOLUME	30 GB
OS	UBUNTU (CLI)

Save

SIGN IN

[I don't have an account yet](#)

Figure 4.32: Use Case Diagram for custom sign-in

Fig. 4.32. The user can choose the hardware specification as per their requirements and then sign-in to get the VM console of the custom VM. These VMs are assigned Floating IPs by default. The custom VMs are temporary, as the files and settings are deleted along with the VM instance when the user signs out. The Level 1 DFD, use case diagrams and activity diagram for the custom login are shown in Figures 4.29, 4.33 and 4.30 respectively.

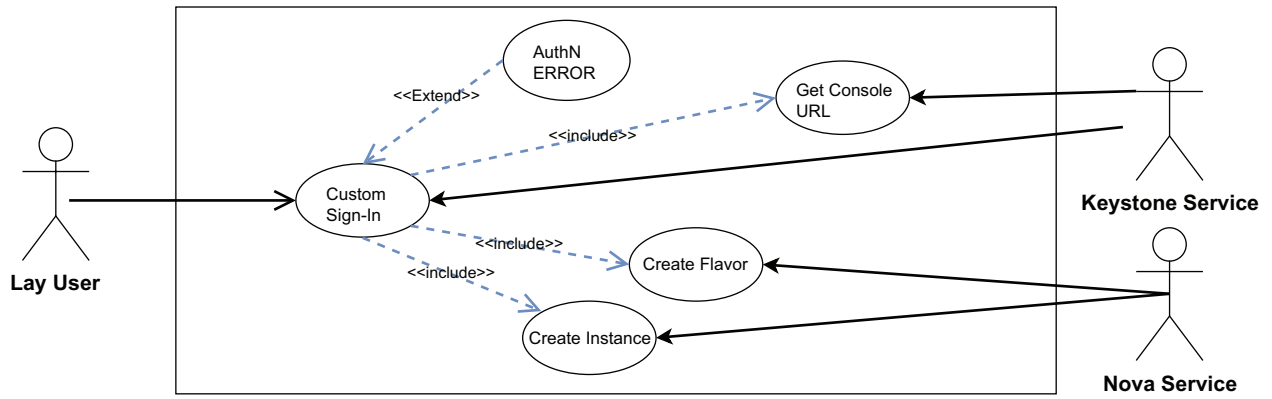


Figure 4.33: Use Case Diagram for custom sign-in

### 4.9.1 Algorithm for Custom Sign-In

---

#### Algorithm 6: Custom Sign-In

---

**Input** : id, password, type

**Output:** VM console

```

1 get user-id, password, type
2 validate user-id, password
3 if invalid then
4   | display error message
5   | exit
6 end
7 if type = custom then
8   | input OS, VCPU, RAM, HDD
9   | check_resources
10  | if available then
11    | create Flavor
12    | create VM
13    | return console
14  | else
15    | show error message
16    | exit
17  | end
18 end

```

---

## 4.10 Assigning Floating IP Addresses

By default, a VM is connected to OpenStack virtual private network and it has a private IP address. Using this address, the VM can be accessed within the virtual network only.

A Floating IP address is different from the IP addresses assigned to the VMs by OpenStack. Floating IP addresses can be instantly associated, dissociated or moved from one VM to another in the same datacenter. If we need to access a VM from outside the virtual network, we need to assign a floating IP address to it.

## 4.10.1 Assigning Floating IP Addresses through Horizon

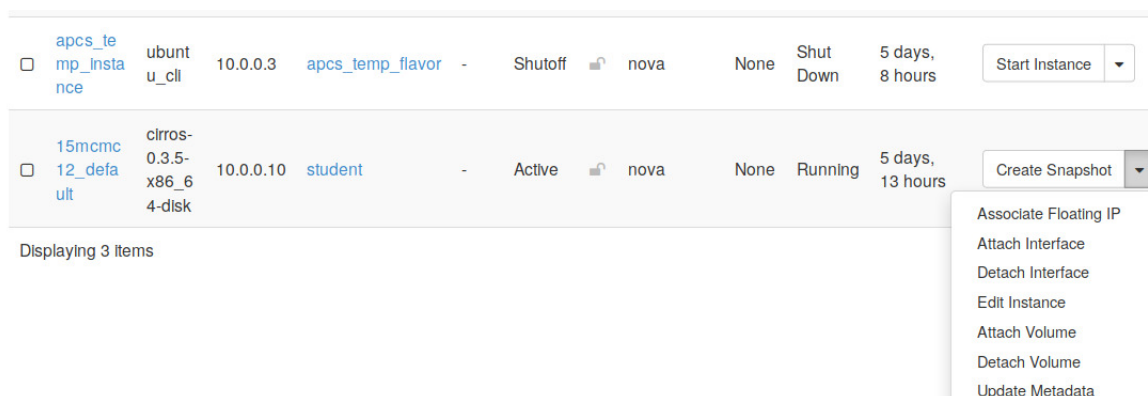


Figure 4.34: Choose “Associate Floating IP” from menu

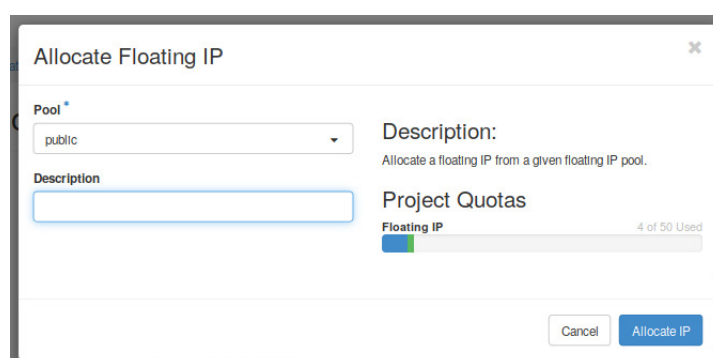


Figure 4.35: Allocate IP from public pool

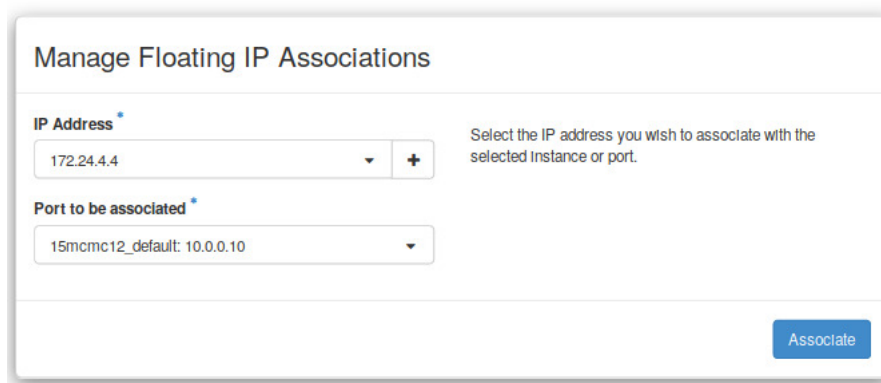


Figure 4.36: Associate to instance(VM)

Assigning a floating IP using Horizon UI is a 3 step process:

1. User has to select option of floating IP from a menu of VM (Fig. 4.34)
2. If there is no unallocated IP, then allocate one (Fig. 4.35)
3. Associate with the VM (Fig. 4.36)

✕
Account Details

ID	15MCMC09
Account Type	STUDENT
Name	SAGAR
Mobile	9876543210
Email	sagarsuman00@gmail.com
CPU Cores	2
Main Memory	2 GB
HDD	5 GB
Operating System	CIRROS-0.3.5-X86_64-DISK
IP Address	<span style="background-color: #e91e63; color: white; padding: 2px 10px; border-radius: 3px; cursor: pointer;">Assign IP</span>

Figure 4.37: Associate IP to instance(VM)

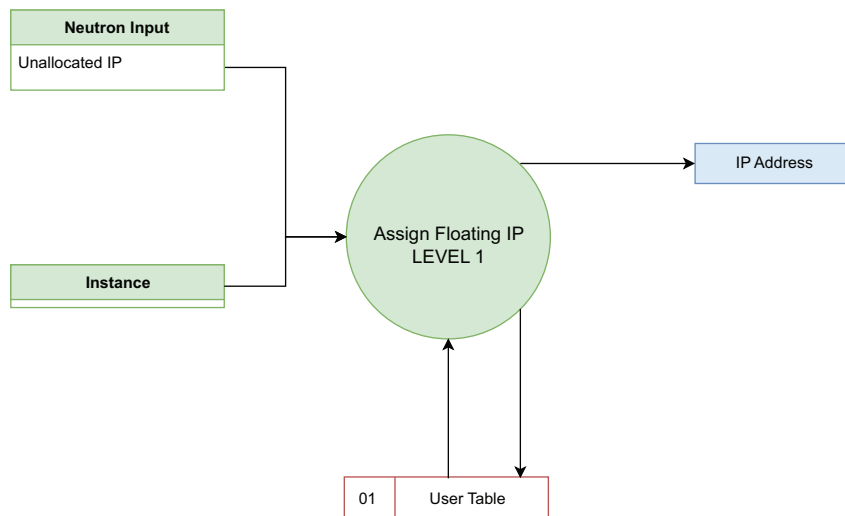


Figure 4.38: DFD of Floating IP assignment

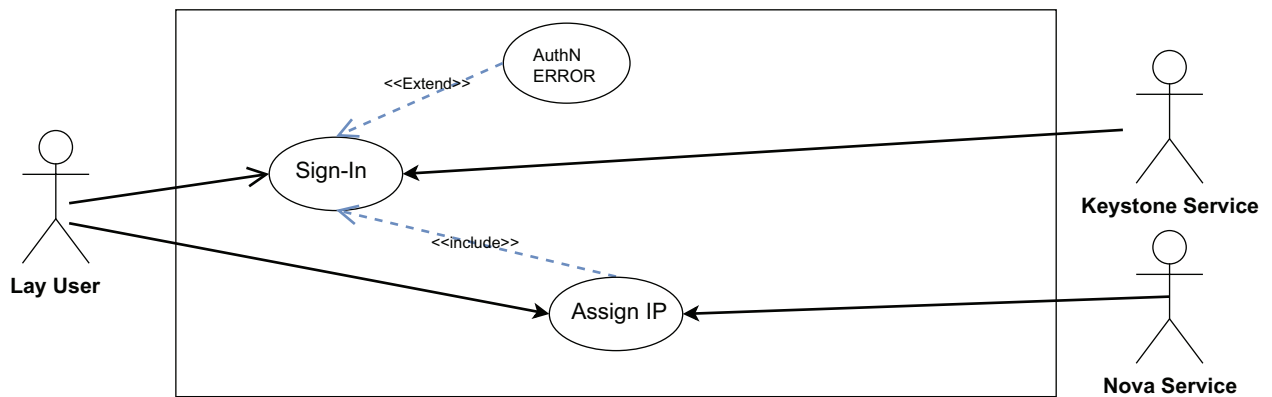


Figure 4.39: Use Case Diagram for Floating IP assignment

## 4.10.2 Assigning Floating IP Addresses in our Application

We have automated the allocation of floating IP addresses to the custom VM. The user will have to only click a button on their home screen as shown in Fig. 4.37.

The Level 1 DFD, use case diagrams and the activity diagram for the assignment of floating IP addresses are shown in Figures 4.38, 4.39 and 4.40 respectively.



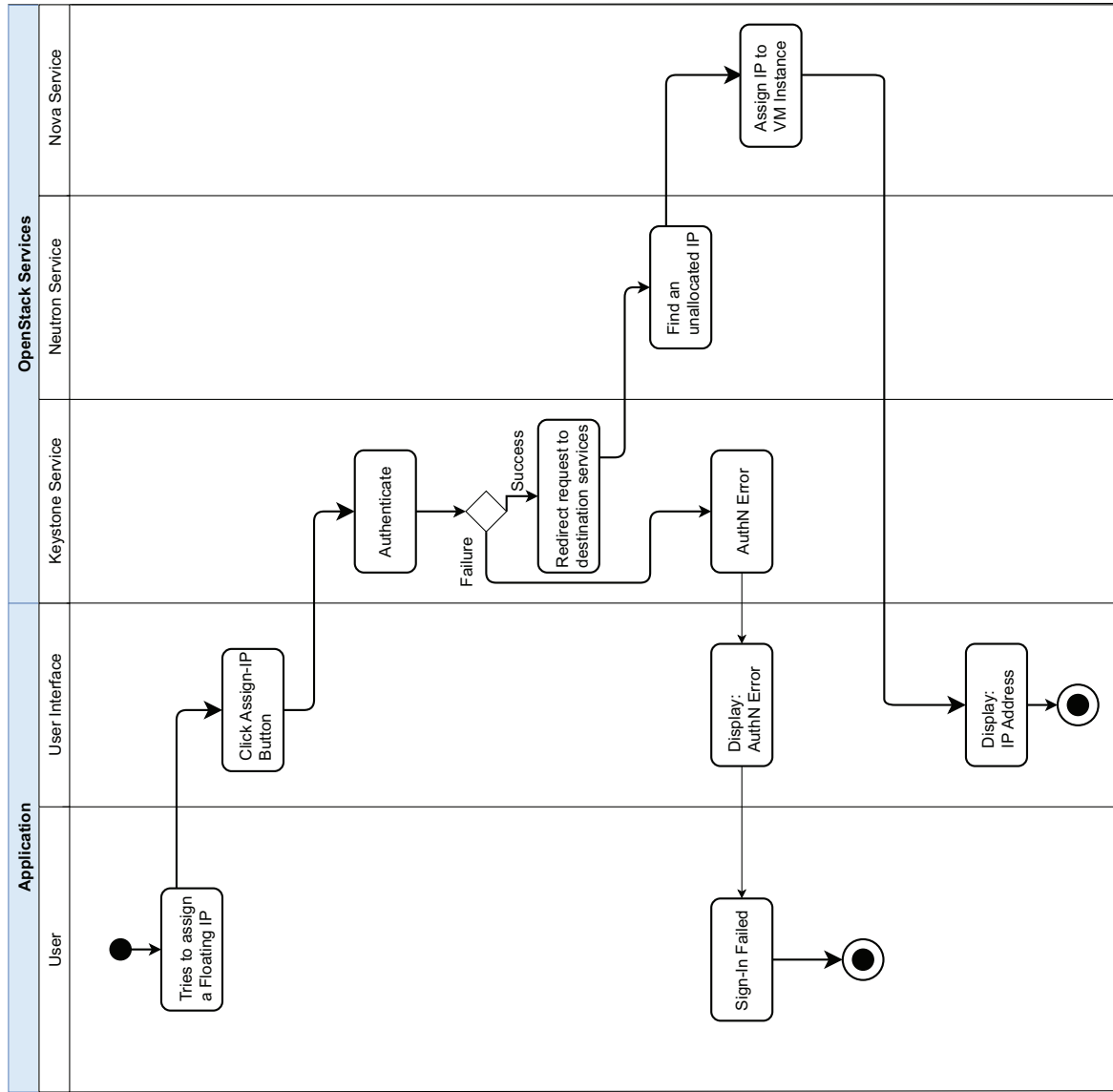


Figure 4.40: Activity Diagram for Floating IP assignment

### 4.10.3 Algorithm to Allocate Floating IP Addresses

---

#### Algorithm 7: Assigning floating IP

---

**Input** : instance\_name

**Output:** IP address

- 1 get instance name
  - 2 check availability of floating IP
  - 3 **if** *there is free floating IP* **then**
  - 4 | assign that IP to the instance
  - 5 **else**
  - 6 | allocate a Floating IP from the pool
  - 7 **end**
  - 8 assign IP to the instance
  - 9 return IP
-

## 4.11 Resource Monitoring

The resources can be monitored by the administrator using the admin panel dashboard. The resources include Number of Instances, VCPU, RAM and HDD. The dashboard displays the resources available and the resources being used in the form of pie charts. Also the the list of online users and the resources being used by each of them is displayed. The monitoring system also checks the set of available resources every time a new VM is created. If the resources available are not enough for creating a new VM as per the configuration requested, the system displays an error. A screenshot of resource monitor dashboard is shown in Fig. 4.41.

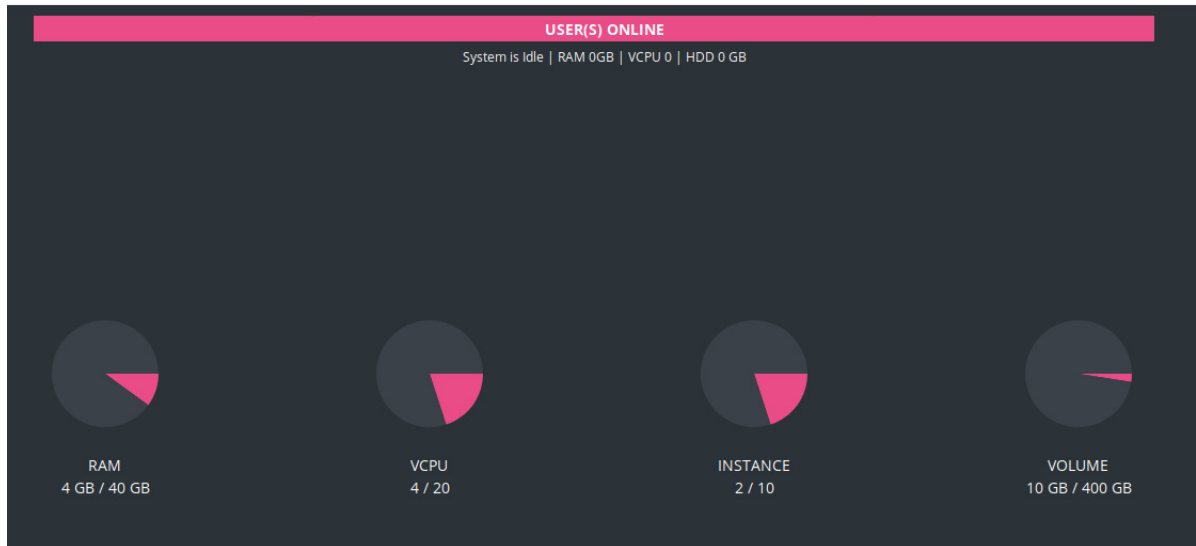


Figure 4.41: Resource Monitoring Dashboard

The Level 1 DFD and activity diagrams for resource monitoring are shown in Fig. 4.42 and Fig. 4.44 respectively.

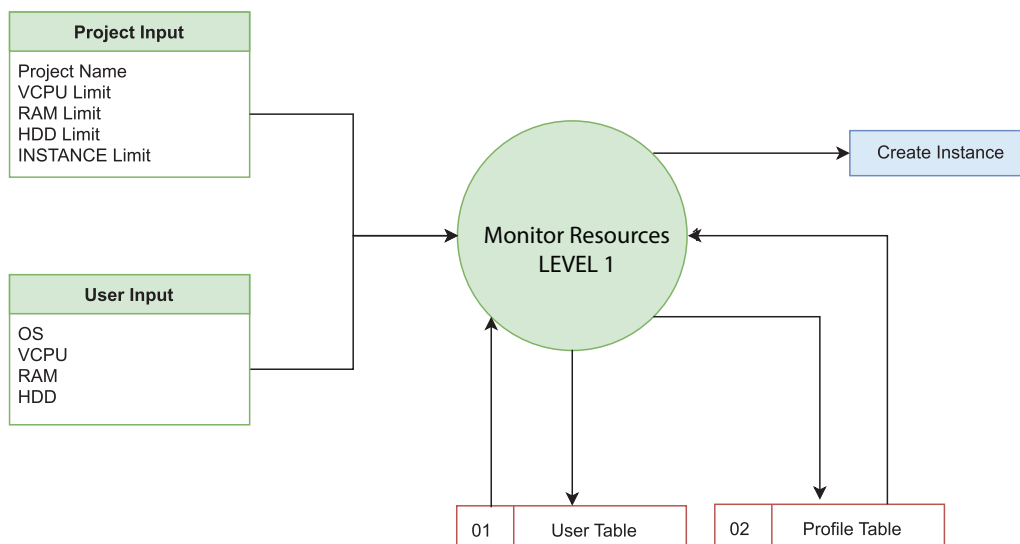


Figure 4.42: Data Flow Diagram for Resource Monitoring

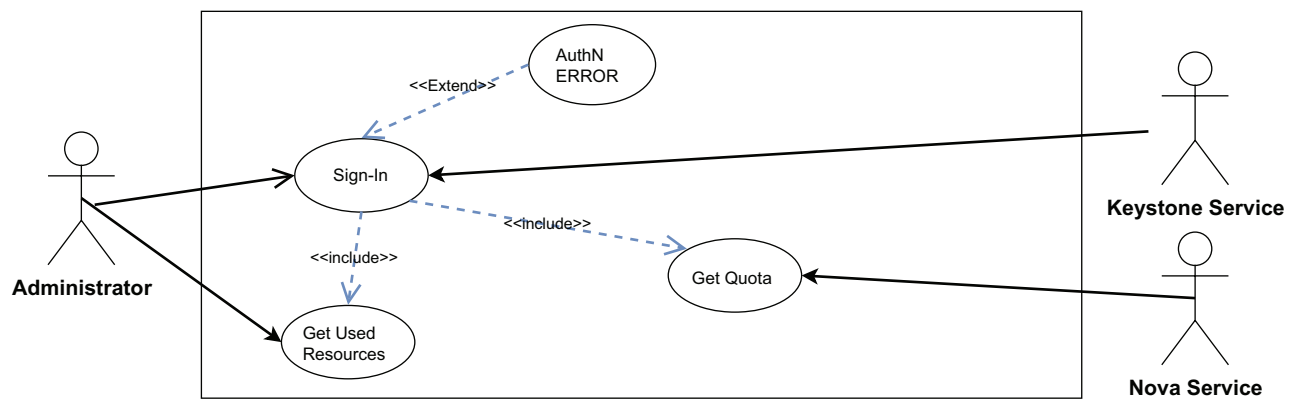


Figure 4.43: Use Case Diagram for Resource Monitoring

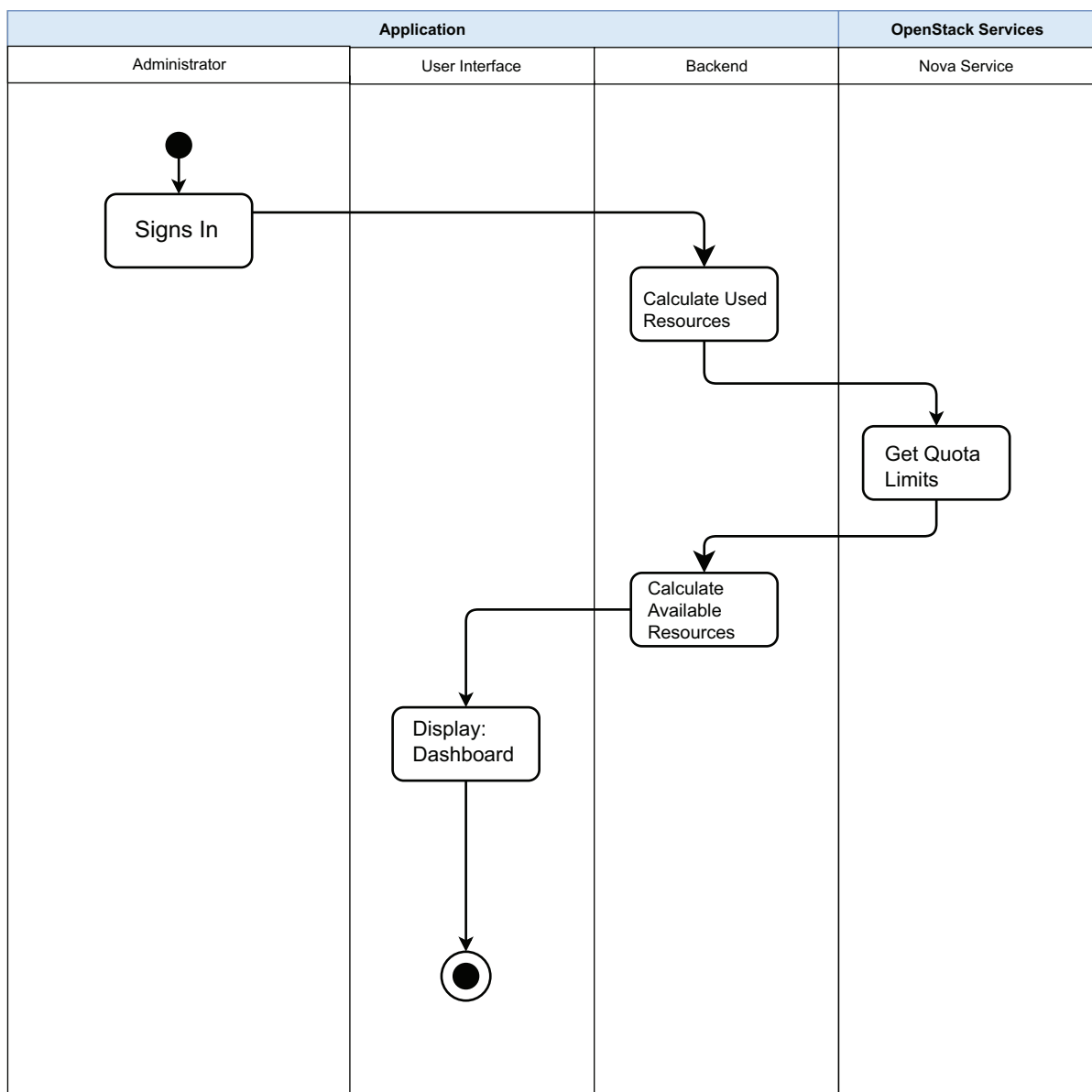


Figure 4.44: Activity Diagram for resource monitoring

### 4.11.1 Algorithm for Resource Checking

---

**Algorithm 8:** Resource Checking

---

**Input** : OS, RAM, CPU, HDD

**Output:**

```
1 Get User Input (ID, Password, OS, RAM, CPU, HDD)
2 Get Project_Name from application configuration file
3 Get Project Resource Quota using input(Project_Name)
4 Get SUM of resources allocated to the users
5 Available Resources= Resource Quota – Used Resources
6 if Requested Resources less than Available Resources then
7   | Display Error Message
8 else
9   | Create Instance
10 end
```

---

### 4.11.2 Algorithm for Resource Monitoring Dashboard

---

**Algorithm 9:** Resource Monitoring Dashboard

---

**Input** : project name

**Output:** dashboard

```
1 get project name
2 Get Project Quota_Limits from OpenStack
3 used_resources = sum(existing VMs' resources)
4 Available_Resources=Quota_Limits – Used_Resources
5 For all resources draw pie_chart using Available_Resources
```

---

# Chapter 5

## Conclusion

In conclusion, we have built an application on top of OpenStack that simplifies the job of the cloud administrator and a lay user. It hides many details of OpenStack from a lay user and directly displays the standard GUI of the chosen Operating System or the console on login.

### 5.1 Conclusion

The operations that have been added to a vanilla OpenStack are as follows:

1. Allows specification of different user profiles for an organization using a web form or through a configuration file.
2. Resource monitoring web form showing all the VMs currently running and their usage of resources.
3. Associating each user with the VM of their configuration and automatically running the instance on user login.
4. Allowing users to create custom VMs by specifying the resources needed without getting into the details of OpenStack.

Operations which have been simplified are as follows:

1. Creation of VM instances per user is simplified.
2. User registration and authorization improved upon.
- 3.

### 5.2 Future Work

1. **VM Migration:** The current version of application is incapable of restoring the VMs when the server recovers from a crash. When the server is restored, OpenStack needs to be unstacked and stacked again. This process most often requires the entire configuration settings to be reset. Therefore in order to retain the VMs even after a server crash, the application should provide VM Migration as a feature.

2. **File Transfer:** The VMs will be used by the users for running specific processor intensive tasks. The tasks can be in the form of an executable file. The current version of our application does not provide a convenient method for file transfer and retrieve. A simple GUI based file browse, upload and download feature will make the user interaction more seamless.
3. **Application Bundle:** The current version of the application requires manually downloading the files and configuring the app.conf file. Also the various dependencies like NGINX, PHP, PHP-MYSQL, PHP-FRM, OpenStack etc. need to be installed for using the application. The application can be packed into a single file as a bundle using Docker as a docker image.
4. **LDAP Integration:** OpenStack supports LDAP. It can be used for authenticating users by querying the LDAP server. For organizations that already have an LDAP server set up with the registered users, it is not necessary to do the registration again if the LDAP can be integrated with OpenStack. This also needs to be simplified, if possible.

# Appendix A

## Installation of OpenStack

OpenStack can be deployed as a single node setup or multiple nodes setup. In multinode setup, different services of OpenStack are installed on different node(machine). All the nodes need not to be at one place, they can be reside on different regions anywhere on the Earth. On the other hand, in single node setup, all the services are installed in one node. The node must be powerful enough in terms of CPU, Main Memory and Volume.

For our project purpose, we installed OpenStack as a single node setup.

DevStack [16] is a series of extensible scripts used to quickly bring up a complete OpenStack environment based on the latest versions of everything from git master.

### steps for setting up OpenStack with DevStack

- **install ubuntu [17] (16.04 LTS or above preferable)**

- **add a user named stack**

```
$ sudo useradd -s /bin/bash -d /opt/stack -m stack
```

- **give root privileges to stack**

DevStack will do several changes in the operating system, so the user “stack” must have root permissions. Make “stack” root as follows:

```
$ echo "stack ALL=(ALL) NOPASSWD: ALL" | sudo tee /etc/sudoers.d/stack
$ sudo su - stack
```

- **download devstack from github and cd in that folder**

```
$ git clone https://git.openstack.org/openstack-dev/devstack
$ cd devstack
```

- **disable nova networking and enable neutron networking**

```
$disable_service=n-net,c-api,c-sch,c-vol
$enable_service=q-svc,q-agt,q-dhcp,q-l3,q-meta,q-lbaasv2
```

- **write a configuration file**

DevStack installs everything using a configuration file known as *local.conf*. This file contains passwords, services to be installed, services to be removed etc. Contents of our *local.conf* are as follows:

```
[[local|localrc]]
GIT_BASE=${GIT_BASE:-https://git.openstack.org}
ADMIN_PASSWORD=password1
DATABASE_PASSWORD=$ADMIN_PASSWORD
RABBIT_PASSWORD=$ADMIN_PASSWORD
```

```
SERVICE_PASSWORD=$ADMIN_PASSWORD
HOST_IP=<IP Address of host machine>
enable_service s-proxy s-object s-container s-account
SWIFT_REPLICAS=1
SWIFT_HASH=66a3d6b56c1f479c8b4e70ab5c2000f5
```

- run the script *stack.sh*  
./stack.sh

Note: The installation usually takes 20-45 to complete, depending upon Internet speed. If every step has succeeded without any error then **OpenStack has been successfully installed on the system.** The URL for accessing the Horizon Web UI will be displayed on the terminal.



# Appendix B

## Creating Cloud Image using VirtualBox

A cloud server(VM) is created using a cloud image. Cloud image is a snapshot of a running operating system with some modification.

Several vendors provide cloud images for OpenStack. However, anyone is free to create his/her own cloud image. A cloud image can be created using any hypervisor. For our purpose, we have chosen VirtualBox to create cloud images.

### Steps for creating a Cloud Image

- install VirtualBox
- create a virtual machine in it.  
During setting up the network for virtual machine, select NAT, and in network adapter type Select **Paravirtualized network adapter (virtio-net)**.
- choose an operating system and install in virtual machine.  
steps for creating a virtual machine in VirtualBox can be learned from here-  
<https://www.lifewire.com/run-ubuntu-within-windows-virtualbox-2202098>
- customize this operating system according to your requirement.
- install cloud-init [18] package in guest operating system.
- set metadata source. run this command  
`$dpkg-reconfigure cloud-init`  
and select EC2 data source.
- You can further configure cloud-init according to your requirement by editing the file `/etc/cloud/cloud.cfg`.
- shutdown the virtual machine.
- find the directory in which VDI file of the virtual machine exists.
- command to get raw file.  
`VBoxManage clonehd <input-file>.vdi <output-file>.img --format raw`
- command to convert raw file into qcow2 file `qemu-img convert -f raw -O qcow2 <input>.img <output>.qcow2`

- if qcow2 file is too large then run the command to compress it `qemu-img convert -O qcow2 -c input.qcow2 output.qcow2`

Now the image file is ready to upload on OpenStack.

# Appendix C

## Installing Dependencies

There are a number of dependencies required for our web based application to be accessible on a local network. These dependencies include NGINX Web server, PHP, PHP-FPM and PHP-MySQL

### Installing NGINX

- Install NGINX Web server using:

```
$sudo apt-get install nginx
```

- Check NGINX is running :

```
$systemctl status nginx
```

- Edit the following file to change the port number:

File Location: `/etc/nginx/sites-enabled/default`

Change the port number to anything other than 80, since port 80 is being used by apache web server, hosting Horizon web service.

Change **listen 80;** to **listen [your port number];**

Change **index index.html index.htm;** to **index index.php index.html index.htm;**

Restart NGINX for changes to reflect:

```
$sudo systemctl restart nginx
```

- Access the nginx home directory at:

```
http://[IP Address]:[Port Number]
```

### Installing PHP

- Install PHP using:

```
$sudo apt-get install php7.0
```

- Install PHP-FPM using:  
`$sudo apt-get install php7.0-fpm`

- Install PHP-MySQL using:  
`$sudo apt-get install php-mysql`

- **Change php.ini file:**  
`$sudo vi /etc/php7/fpm/php.ini`

Find: `#cgi.fix_pathinfo=1`  
Replace with: `cgi.fix_pathinfo=0`

- Change www.conf file:  
`$sudo nano /etc/php7/fpm/pool.d/www.conf`

Find: `listen = 127.0.0.1:9000`  
Replace with: `listen = /var/run/php7-fpm.sock`

Restart PHP-FPM service `$sudo service php7.0-fpm restart`

Now the Web server is ready to be accessed on a local network.

# References

- [1] Omar Sefraoui, Mohammed Aissaoui, and Mohsine Eleuldj. Openstack: toward an open-source solution for cloud computing. *International Journal of Computer Applications*, 55(3), 2012.
- [2] John V Guttag. *Introduction to computation and programming using Python*. Mit Press, 2013.
- [3] Stig Sæther Bakken, Alexander Aulbach, Egon Schmid, Jim Winstead, Lars Torben Wilson, Rasmus Lerdorf, and Zeev Suraski. Php manual. *Zend Technologies, Ltd*, 1997.
- [4] AB MySQL. Mysql, 2001.
- [5] Will Reese. Nginx: the high-performance web server and reverse proxy. *Linux Journal*, 2008(173):2, 2008.
- [6] Alfonso V Romero. *VirtualBox 3.1: Beginner's Guide*. Packt Publishing Ltd, 2010.
- [7] <https://aws.amazon.com/what-is-cloud-computing>.
- [8] [https://en.wikipedia.org/wiki/Network\\_Information\\_Service](https://en.wikipedia.org/wiki/Network_Information_Service).
- [9] [https://en.wikipedia.org/wiki/Lightweight\\_Directory\\_Access\\_Protocol](https://en.wikipedia.org/wiki/Lightweight_Directory_Access_Protocol).
- [10] [https://en.wikipedia.org/wiki/Network\\_File\\_System](https://en.wikipedia.org/wiki/Network_File_System).
- [11] Ryan B Bloom and Brian Foreword By-Behlendorf. *Apache Server 2.0: The Complete Reference*. Osborne/McGraw-Hill, 2002.
- [12] Institute of Electrical and Electronics Engineers. *IEEE Guide to Software Requirements Specifications*. IEEE, 1984.
- [13] [https://en.wikipedia.org/wiki/Amazon\\_Web\\_Services](https://en.wikipedia.org/wiki/Amazon_Web_Services).
- [14] [https://en.wikipedia.org/wiki/Data\\_flow\\_diagram](https://en.wikipedia.org/wiki/Data_flow_diagram).
- [15] Peter Pin-Shan Chen. The entity-relationship model—toward a unified view of data. In *Readings in artificial intelligence and databases*, pages 98–111. Elsevier, 1988.
- [16] <https://docs.openstack.org/devstack/latest/>.
- [17] <https://www.ubuntu.com/download/desktop>.
- [18] <http://cloud-init.org/>.