# MULTIPLE AUTHENTICATION IN OPENSTACK

**Submitted in partial fulfillment of the requirements**

**of the degree of**

**Bachelor of Engineering**

**by**

**Kunal Dandekar**
**Apurva Deore**
**Yash Deorukhkar**
**Tanaya Mulik**

**Supervisor:**

**Asst. Prof. Nilesh Ghavate**

**UNIVERSITY OF MUMBAI**

# MULTIPLE AUTHENTICATION IN OPENSTACK

**Submitted in partial fulfillment of the requirements**

**of the degree of**

**Bachelor of Engineering**

**by**

**Kunal Dandekar**
**Apurva Deore**
**Yash Deorukhkar**
**Tanaya Mulik**

**Supervisor:**

**Asst. Prof. Nilesh Ghavate**

**Department of Information Technology**

**Don Bosco Institute of Technology**
**Vidyavihar Station Road, Mumbai - 400070**
**2016-2017**

**DON BOSCO INSTITUTE OF TECHNOLOGY**
**Vidyavihar Station Road, Mumbai - 400070**

**Department of Information Technology**

# Certificate

This is to certify that the project entitled **"Multiple Authentication in OpenStack"** is a bonafide work of

| | |
|---|---|
| **Kunal Dandekar** | **Roll no. 13** |
| **Apurva Deore** | **Roll no. 14** |
| **Yash Deorukhkar** | **Roll no. 15** |
| **Tanaya Mulik** | **Roll no. 44** |

submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **Undergraduate** in **Bachelor of Information Technology.**

**Date: 24/04/2017**

**(Asst. Prof. Nilesh Ghavate)**
**Supervisor**

**(Prof. Janhavi Baikerikar)**
**HOD, IT Department**

**(Dr. Prasanna Nambiar)**
**Principal**

**DON BOSCO INSTITUTE OF TECHNOLOGY**
**Vidyavihar Station Road, Mumbai - 400070**

**Department of Information Technology**

# Project Report Approval for B.E.

This project report entitled **“Multiple Authentication in OpenStack”** by **Kunal Dandekar, Apurva Deore, Yash Deorukhkar, Tanaya Mulik** is approved for the degree of **Bachelor of Engineering in Information Technology**

**(Examiner’s Name and Signature)**

**1.** ____________________

**2.** ____________________

**(Supervisor’s Name and Signature)**

**1.** ____________________

**( Chairman)**

**1.** ____________________

**Date:24/04/2017**

**Place: Mumbai**

**DON BOSCO INSTITUTE OF TECHNOLOGY**

**Vidyavihar Station Road, Mumbai - 400070**

**Department of Information Technology**

# Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea / data / fact / source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

**(——————————- )**
**(Kunal Dandekar 13)**

**(——————————- )**
**(Apurva Deore 14)**

**(——————————- )**
**(Yash Deorukhkar 15)**

**(——————————- )**
**(Tanaya Mulik 44)**

**Date:24/04/2017**

# Abstract

OpenStack is a cloud environment that controls large pools of compute, storage, and networking resources throughout a data center, all managed through a dashboard that gives administrators control while empowering their users to provision resources through a web interface. There are various components in OpenStack and these components provide various services. Keystone is an OpenStack service that provides API client authentication, service discovery, and distributed multi-tenant authorization by implementing OpenStack's Identity API. This project aims at adding a single sign on feature to access the resources provided by OpenStack cloud. This project explains the role of Keystone component in OpenStack that allows the user to authorize and authenticate himself to the cloud environment using his credentials. This project enables the end user(s) to login to a cloud environment using their existing Google credentials using single sign-on approach.

OAuth is an open standard for authorization, commonly used as a way for Internet users to login to third party websites using their Google, Facebook, Microsoft, Twitter, One Network, etc. accounts without exposing their password. This project will be using OAuth 2.0 protocol. The project efficiently explains the working of OAuth 2.0 protocol and its implementation in Keystone. Finally the steps for including google authentication in keystone and the technologies required are presented.

**Keywords:** OpenStack, Cloud, Keystone, OAuth, Google

# Contents

# List of Figures

# Chapter 1

# Introduction

## 1.1 Problem Statement

To add multiple authentication in OpenStack. (Google sign-in)

## 1.2 Scope of the Project

The project aims at allowing end users to login to a cloud environment using their existing Google credentials.
The user sends an authentication request to the Keystone server. The Keystone server forwards this request to the external Google Authentication server which will perform the task of fetching identities and authenticating them, after which a token is passed to the Keystone server for authorization. The authentication of the user is done by using the OAuth 2.0 protocol that uses the OpenID Connect plugin. The project thus, adds an external authentication feature and also separates authentication and authorization procedure.

### 1.2.1 What is SSO:

SSO (Single Sign On) occurs when a user logs in to one Client and is then signed in to other Clients automatically, regardless of the platform, technology, or domain the user is using.
Google's implementation of login for their products, such as Gmail, YouTube, Google Analytics, and so on, is an example of SSO. Any user that is logged in to one of Google's products are automatically logged in to their other products as well.

### 1.2.2 How SSO Works:

SSO works by means of a central service. In the case of Google, this central service is Google Accounts. When a user first logs in, Google Accounts creates a cookie, which persists with the user as they navigate to other Google-owned services. The process flow is as follows:

1. The user accesses the first Google product.
2. The user receives a Google Accounts-generated cookie.
3. The user navigates to another Google product.
4. The user is redirected again to Google Accounts.

Google Accounts sees that the user already has an authentication-related cookie, so it redirects the user to the requested product.
Single Sign On is thus obtained as an output of the project in an open source cloud computing environment of OpenStack.

## 1.3 Current Scenario

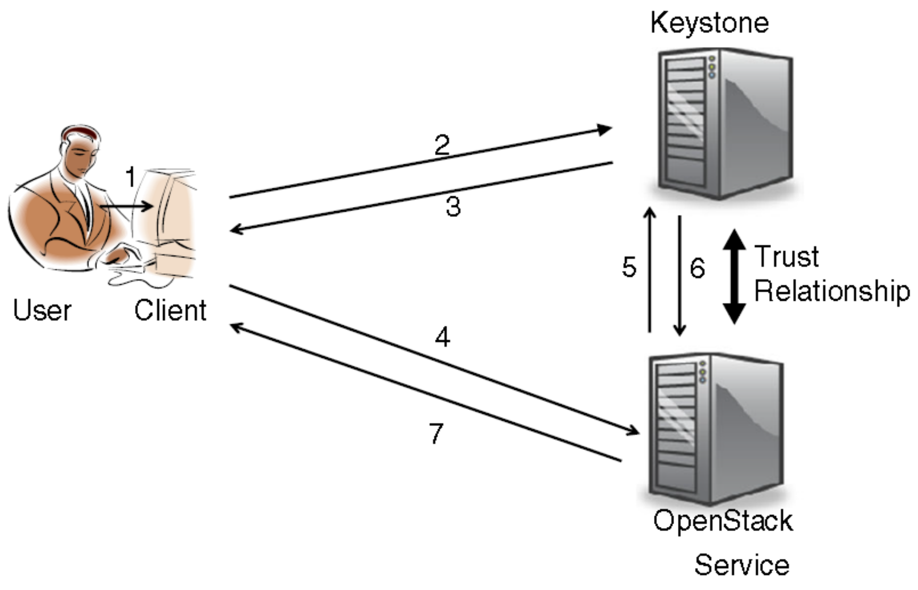


**Figure 1.1:** Current Mechanism of Keystone [1]

**(1)** The user enters his credentials to the OpenStack Horizon or the command line interface, along with the request he wishes to make.
Currently the OpenStack open source code only provides command line clients for each of its services, although a web based administrative client called Horizon is also available.
**(2)** The client sends an authentication request to the Keystone server using the user's credentials.
**(3)** If the credentials are correct, and the user specified the project he wishes to use, Keystone sends back a scoped token and a list of endpoints to the different services that are available to him/her. If the project was not specified, Keystone sends back an unscoped token. An unscoped token can only be used with Keystone to exchange it for a scoped token, by providing a project ID. The user needs a scoped token to access any of the OpenStack services.
**(4)** The client chooses the endpoint of its service and sends the scoped token along with the user's request to the chosen service.
**(5)** Assuming this is an opaque token and not a PKI based token, the service sends the scoped token to the Keystone server asking for it to be validated. (If it is a PKI based token the service can validate it itself by using the public key of the Keystone server.)
**(6)** The Keystone server checks if the token is valid. If it is, it sends back the token containing the user id, domain, project, and the roles the user has in the project. The user is then authenticated with the service.
**(7)** The service now makes an authorisation decision based on either the role and the project (and domain) that the user has (i.e. RBAC), or user id (i.e. ACL). If the user is authorised, then the service processes the request and sends the response to the client, otherwise the user's request is rejected.

**NOTE:** An opaque token contains unformatted data as a sequence of bytes. An opaque token cannot be parsed by the client. A PKI token is encrypted with the public key of Keystone Server and can be read only if the user has a valid private key. Therefore validations of PKI tokens does not require Keystone Server.

## 1.4 Need for the Proposed System

There is a need for cloud users to be able to login into the cloud system using their existing credentials. This system is needed to provide the concept of single sign on to cloud users using OpenStack. The concept of Singal Sign On (SSO) is explained below:

### 1.4.1 Single sign on:

It is a session and user authentication service that permits a user to use one set of login credentials (e.g, name and password) to access multiple applications.

**Example:**
The proposed system is needed in a situation where a company contains more than a thousand employees and all of them contain their own Google login credentials. In such a situation having a single user-name and password is much more feasible and easier than remembering different login-ids for different accounts. Therefore, there is a need for Single Sign On where in the user is not required to login everytime he needs to access his/her own cloud resources.

### 1.4.2 Advantages of Single Sign On:

**(1) Helps reduce desk costs**
User need not remember a long list of passwords and thus need not go through password reset options that are not cost-effective.

**(2) Improves customer satisfaction**
A user-friendly login process can be very important because the login screen is where the first interaction between users and the inner halls of your website will be taking place.

**(3) Boosts productivity**
In companies implementing strict password policies, simple login processes can take much longer than usual.

**(4) Improves compliance and security capabilities**
It encourages the use of strong passwords and other security applications, like a secure file transfer system.

**(5) Facilitates Business to Business collaboration**
Many products do not come from just one large enterprise. SSO facilitates B2B collaboration by allowing users to use only one password for multiple products coming from different enterprises.

# Chapter 2

# Review of Literature

## 2.1 Summary of the published papers

**1.** The architecture of OpenStack was studied [3]. It gave an overview of the diffent components of OpenStack.

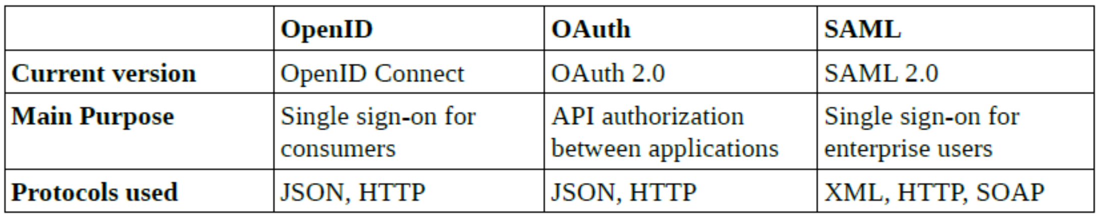
**2.** The paper was referred to learn how to add protocol independent federated identity management to the OpenStack services. The current mechanism of token generation of Keystone and its working is also explained. The mechanism explained in section 1.3 was also learnt from this paper.[1]
**Authors:** David W Chadwick, Kristy Siu, Craig Lee, Yann Fouillat, Damien Germonville

**3.** Concepts of Keystone and protocols used are explained in this book [2]. The book also contains mechanisms and a step-by-step guide for adding Google authentication to Keystone as well as token formats. It gives a clear understanding as to how Federated Identity can be achieved in OpenStack.
**Authors:** Steve Martinelli,Henry Nash and Brad Topol

**4.** This online document [10] gives the working of OAuth 2.0 protocol in detail. It specifies how a token is obtained and how a client can use protected resources without the leaking the credentials of the resource owner.
**Author:** Dick Hardt

## 2.2 Comparison between the tools / methods / algorithms

Following is the comparison between various alternatives, that can be adopted to authenticate a user in OpenStack, and their use cases.



| | **OpenID** | **OAuth** | **SAML** |
|---|---|---|---|
| **Current version** | OpenID Connect | OAuth 2.0 | SAML 2.0 |
| **Main Purpose** | Single sign-on for consumers | API authorization between applications | Single sign-on for enterprise users |
| **Protocols used** | JSON, HTTP | JSON, HTTP | XML, HTTP, SOAP |

**Figure 2.1:** Comparison between available protcols

We have chosen the federated identity alternative which includes authentication of a user with Google sign-in, which will use OpenID Connect running over the OAuth 2.0 protocol.

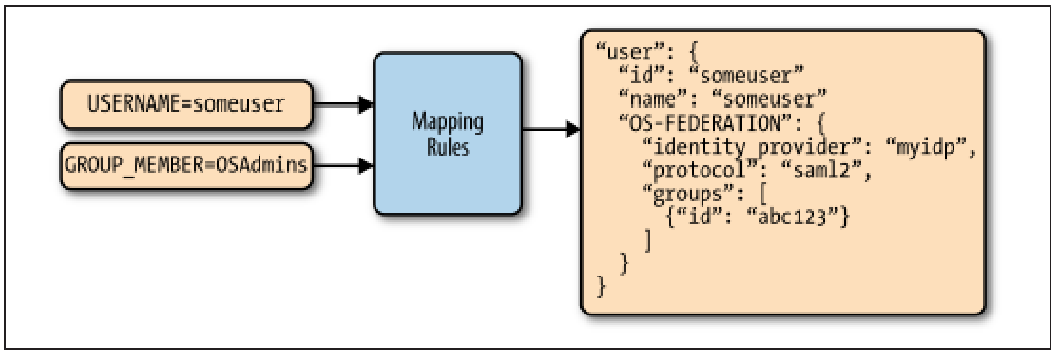
## 2.3 Steps for including external Google Authentication in Keystone

**OpenID Connect**
A newer standard approach to federated identity. Leverages OAuth 2.0, and ditches XML in favor of JSON. The resultant information about the user is called a Claim.

**Claims**
A standard method of representing information and attributes about a user. An identity provider may issue Claims, based on the standard they use, for service providers to consume.

**Mapping**
Mappings are the lynchpin to federated identity in Keystone. A mapping resource has now been created and served at the following API:
*/OS-FEDERATION/mappings*
This endpoint allows a cloud administrator to create, delete, update, retrieve, and list mappings. A mapping is specified when creating a protocol that an Google Authentication server supports. The actual contents of a mapping is a set of rules that will be evaluated when a Claim is seen by Keystone. These rules

will vary greatly between deployments, and depend on the attributes issued by the identity provider and how much access the cloud administrator wants to give to federated users.



```
"user": {
  "id": "someuser"
  "name": "someuser"
  "OS-FEDERATION": {
    "identity_provider": "myidp",
    "protocol": "saml2",
    "groups": [
      {"id": "abc123"}
    ]
  }
}
```

**Figure 2.2:** Mapping of HTTP headers to JSON format [2]

Claims as HTTP headers are sent to the mapping engine with the mapping rules for the identity provider and protocol, and the result is enough information to include in a token.

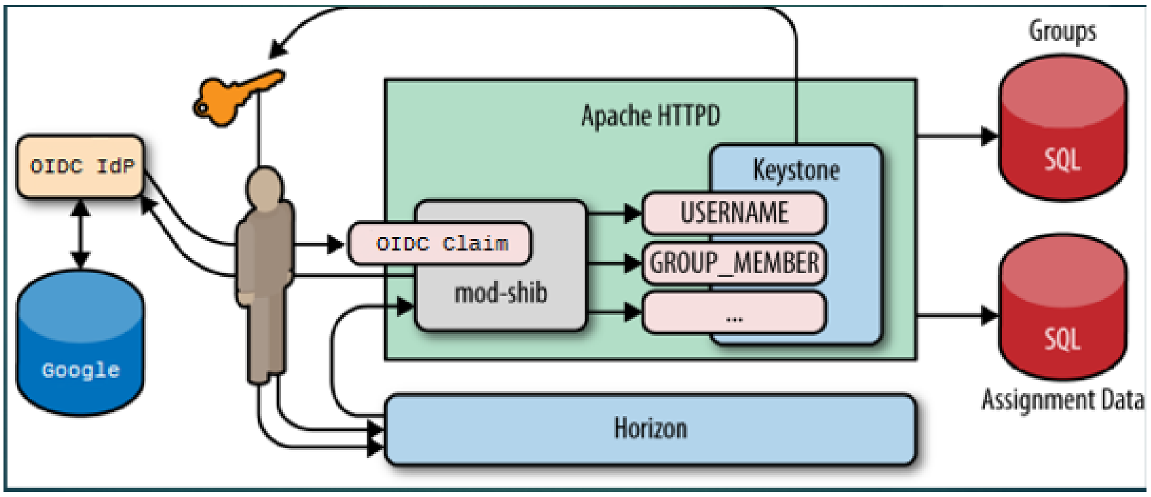


**Figure 2.3:** Schematic authentication flow across modules [2]

**(1)** A user hits the Horizon landing page. Upon loading the page, the user is given a list of methods to authenticate. This list will include a method to authenticate by username and password for service accounts, but also includes methods for authenticating by their protocol, or identity provider and protocol, depending on how Horizon is set up. A user accesses the protected URL above, which triggers the apache module (mod_auth_openidc in this case) to redirect the user to authenticate with the external identity provider (through a web page or otherwise).

**(2)** Once the user has authenticated with their identity provider, the identity provider returns user attributes (OpenID Connect claim) to the Apache plugin. The Apache module then transforms the user attributes into HTTP request headers and passes them along to Keystone.
**(3)** Keystone then determines which mapping to use, based on the identity provider and protocol, invokes the mapping engine, and, if there is enough information to construct a token (user information and group IDs), then an unscoped token is issued. (Not included in the diagram) The user can then use the unscoped token to look up the projects they are allowed to access and request a scoped token for that project, using their federated unscoped token.
**(4)** Where the process differs is what Keystone does with the token: it must post back the token ID to Horizon. This is done using JavaScript.

### 2.3.1 Configure Keystone to use OpenID Connect

If you intend to use an OpenID Connect identity provider, then mod_auth_openidc plugin is available.
Keystone is run on a Apache server. Httpd is the plugin type which means the partiular plugin handles HTTP requests which helps Keystone work with information exchanged on web. Following are the steps to configure Keystone to use OpenID Connect:
**1.** Install the Apache Httpd plugin *mod_auth_openidc*
**2.** Update the *etc/keystone/keystone.conf* file to support OpenID Connect as an authentication mechanism.
**3.** Add in new OpenID Connect entries to the */etc/apache2/sites- available/keystone.conf* file.

### 2.3.2 Configure Horizon for single sign-on

**1.** Change a few values in the *etc/openStack_dashboard/local_settings.py* file.
**2.** Restart all servers and services to finalize the changes

# Chapter 3

# Analysis and Design

## 3.1 Methodology / Procedure adopted

The methodology/model adopted was the Agile model.
Weekly meetings to discuss the project progress and updates were conducted.
Project progress were monitored by the project guide in the weekly meetings that are conducted. Also suggestions regarding the further steps to be carried out were given to the project members.

## 3.2 Analysis

The installation of the open source software required two desktop computers that would provide the required computing and storage capabilities to the system. Ubuntu sever 16.04 did not support Mitaka packages required for OpenStack installation. Therefore Ubuntu version 14.04 was selected.
Ubuntu server version of 14.04 was installed and the latest version of OpenStack Mitaka was chosen for installation.
Thus, project feasibility was carried out based on the availability of resources required and the dependencies of software packages on the OS version upon which it is installed.

## 3.3 Proposed System

The system is installed on two nodes compute and controller that component plugins and configuration of components respectively. From the available services, Identity, Image, SQL Database, Compute, Message queue, DHCP and L3 agents and Network time services were installed.The protocol used for authentication is OAuth 2.0.

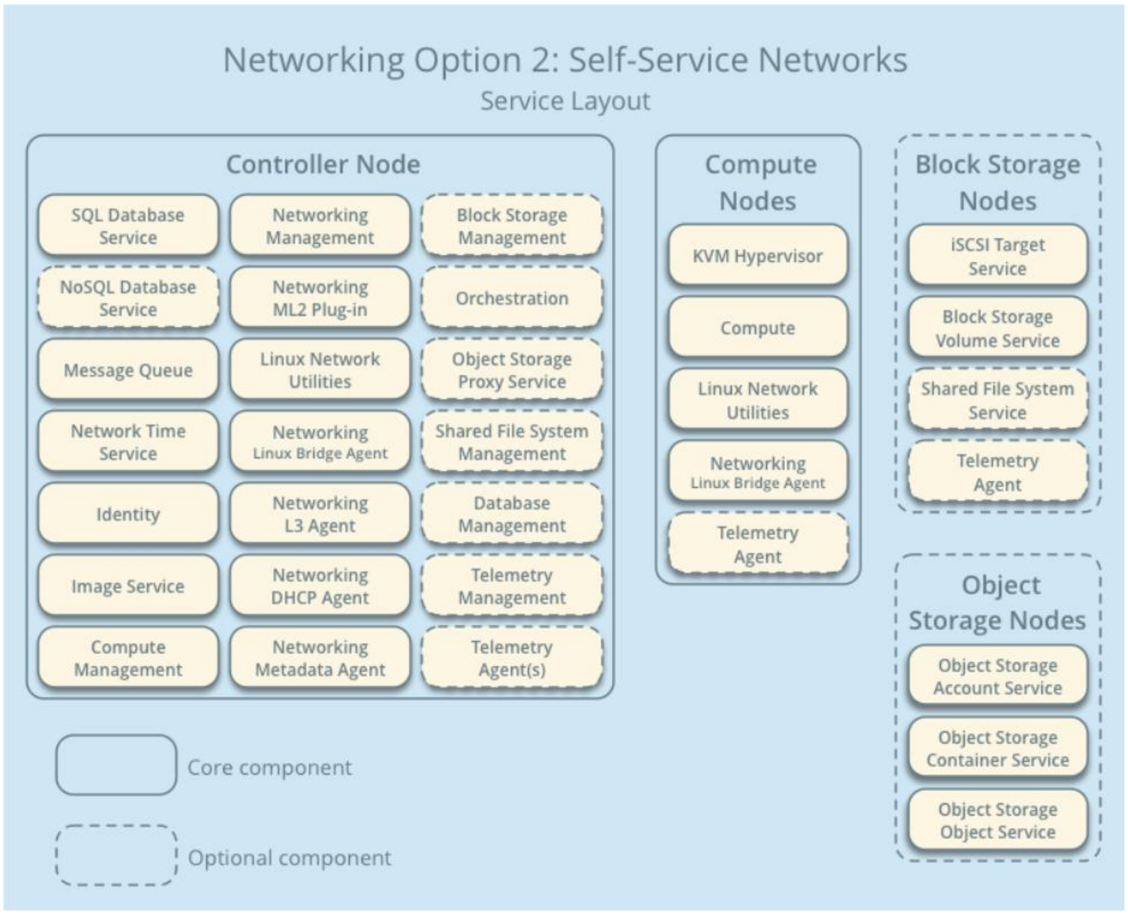


**Figure 3.1:** Network Architechture [3]

### 3.3.1 Google Authentication using OAuth Protocol

OAuth is an open standard for authorization, commonly used as a way for Internet users to login to third party websites using their Google, Facebook, Microsoft, Twitter, One Network, etc. accounts without exposing their password.[7]

### 3.3.2 Versions:

1. OAuth 1.0
2. OAuth 2.0

### 3.3.3 Working of OAuth:

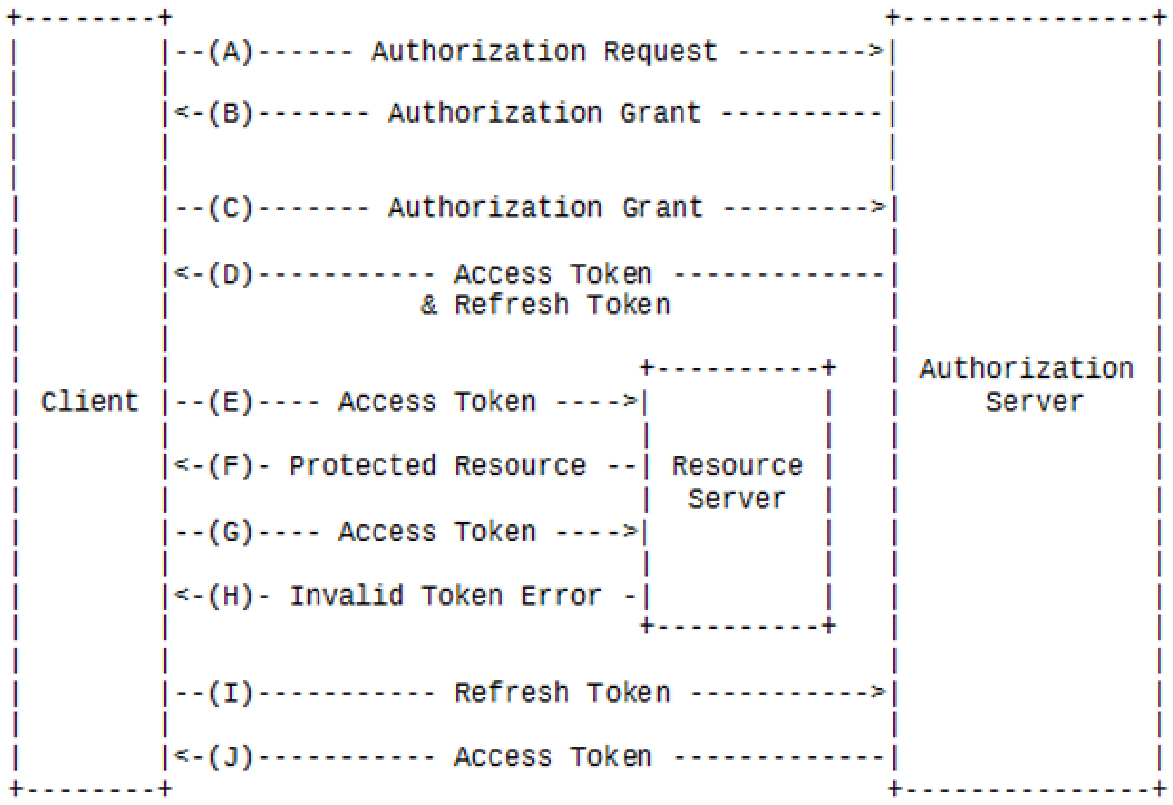


**Figure 3.2:** OAuth Authorization Flow [10]

**(A)** The client requests authorization from the resource owner via the Authorization server.
**(B)** The client receives an authorization grant, which is a credential representing the resource owner's authorization.
**(C)** The client requests an access token by authenticating with the authorization server and presenting an authorization grant.
**(D)** The authorization server authenticates the client and validates the authorization grant, and if valid, issues an access token and a refresh token.
**(E)** The client makes a protected resource request to the resource server by presenting the access token.
**(F)** The resource server validates the access token, and if valid, serves the request.
**(G)** Steps (E) and (F) repeat until the access token expires. If the client knows the access token expired, it skips to step (G); otherwise, it makes another protected resource request.
**(H)** Since the access token is invalid, the resource server returns an invalid to-

ken error.
**(I)** The client requests a new access token by authenticating with the authorization server and presenting the refresh token.
**(J)** The authorization server authenticates the client and validates the refresh token, and if valid, issues a new access token (and, optionally, a new refresh token).

### 3.3.4 Overview of the system's working:

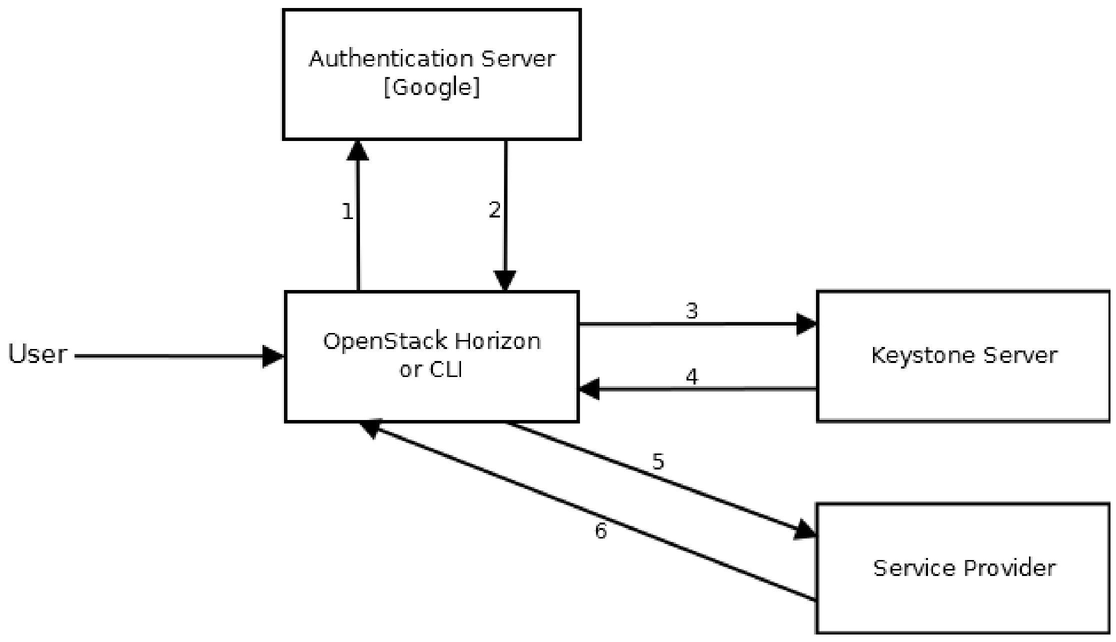


**Figure 3.3:** Proposed system's working

**(1)** The client program or Horizon sends an authentication request to the Google server that includes user credentials of that particular server.
**(2)** The authorization server provides an access token to the client program.
**NOTE:** The mechanism of access tokens and refresh tokens is explained above in the working of OAuth section.
**(3)** This access token that the client program receives, is then sent to the Keystone server for authorization.
**(4)** Keystone then sends a scoped or unscoped token (concept explained in current scenario page).
**(5)** It is necessary for the user to get a scoped token, which then is passed on to the service provider (OpenStack).
**(6)** The service provider then provides the necessary services requested to the client program.

### 3.3.5 Hardware / Software requirements

**2 Nodes**
1. Compute
2. Controller
These nodes are selected on basis of the network requirements of OpenStack.

**Minimum 8GB RAM**
This is required to guarantee fast processing and communication among the various components.

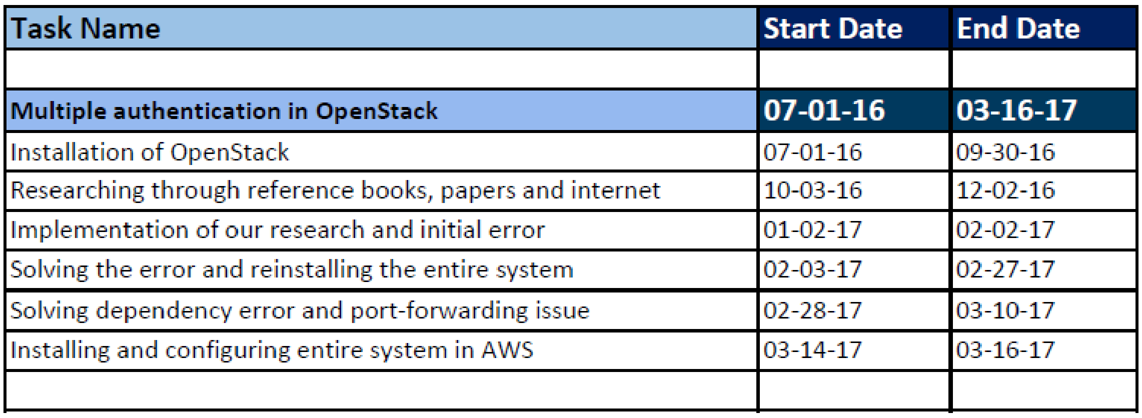
**Ubuntu Server 14.04 LTS**
Since some of the OpenStack Mitaka packages were not compatible with Ubuntu 16.04, 14.04 (Trusty Tahr) was chosen as the desired server OS.

# Chapter 4

# Implementation

## 4.1 Implementation Plan

### 4.1.1 Gantt Chart



| Task Name | Start Date | End Date |
|---|---|---|
| | | |
| **Multiple authentication in OpenStack** | **07-01-16** | **03-16-17** |
| Installation of OpenStack | 07-01-16 | 09-30-16 |
| Researching through reference books, papers and internet | 10-03-16 | 12-02-16 |
| Implementation of our research and initial error | 01-02-17 | 02-02-17 |
| Solving the error and reinstalling the entire system | 02-03-17 | 02-27-17 |
| Solving dependency error and port-forwarding issue | 02-28-17 | 03-10-17 |
| Installing and configuring entire system in AWS | 03-14-17 | 03-16-17 |
| | | |

**Figure 4.1:** Activities performed over the timeline specified

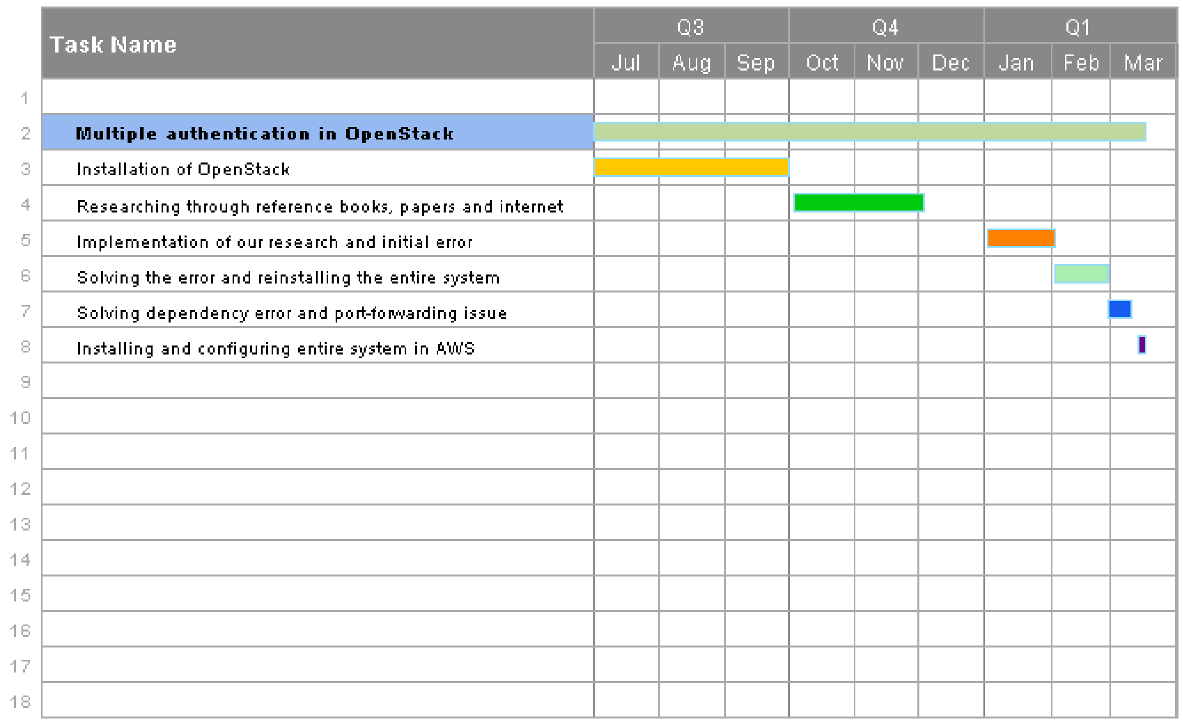


**Figure 4.2:** Gantt chart of the activities specified in Figure 4.1

## 4.2 Coding Standard

### 4.2.1 Python

Openstack uses the **pep8** coding guidelines for Python.[11]

#### 4.2.1.1 General

[H903] Use only UNIX style newlines (`\n`) , not Windows style (`\r\n`)

It is preferred to wrap long lines in parentheses and not a backslash for line continuation.

[H201] Do not write except:, use except Exception: at the very least. When catching an exception you should be as specific so you don't mistakenly catch unexpected exceptions.

[H101] Include your name with TODOs as in # TODO(yourname).
This makes it easier to find out who the author of the comment was.

[H105] Don’t use author tags. We use version control instead.

[H106] Don’t put vim configuration in source files (off by default).

[H904] Delay string interpolations at logging calls (off by default).

Do not shadow a built-in or reserved word. Shadowing built -in or reserved words makes the code harder to understand.

#### 4.2.1.2 Import

Do not import objects, only modules (*)

[H301] Do not import more than one module per line (*)

[H303] Do not use wildcard * import (*)

[H304] Do not make relative imports

[H306] Alphabetically order your imports by the full module path. Organize your imports according to the Import order template and Real-world Import Order Examples below.

(*) exceptions are:

imports from migrate package
imports from sqlalchemy package
function imports from i18n module

### 4.2.2 Django

Django follows the coding standards given below.[12]

#### 4.2.2.1 General

1. Views: The first parameter in a view function should be called a request.
2. Model: Field names should all be lowercase, using underscores instead of camelCase.

3. Template: Put one and only one space between the curly brackets and tag contents.

Use four and only four spaces for indentation.

Use InitialCaps for class names (or for factory functions that return classes).

#### 4.2.2.2 Import

Sort lines in each group alphabetically by the full module name.

Place all important module statements before "from module import objects" in each section.

Use absolute imports for other Django components and relative imports for local components.

## 4.3 Procedure

### 4.3.1 Installation of OpenStack

Installation on private cloud on the desktop systems was done. The components installed on respective controller and compute nodes is as follows [4]:

**Controller:**
1. Keystone Server
2. Horizon Server
3. Glance
4. Nova
5. Neutron

**Compute:**
1. Nova
2. Glance
3. Neutron

**NOTE:** The compute node contains only the necessary files required for the respective components and not the entire component package.

### 4.3.2 Setting up SSO with Google in Openstack

The steps to enable and configure Google sign-in after OpenStack installation are given below. [14]

#### 4.3.2.1 Changes in Keystone:

```
sudo apt-get install libjansson4 libhiredis0.10 libcurl3
sudo dpkg -i libapache2-mod-auth-openidc_1.8.3-1_amd64.deb
```

**Description:** The above commands download and unpack the OpenID Connect package which helps us use the OIDC plugin.

```
[auth]
methods = external,password,token,oidc
oidc = keystone.auth.plugins.mapped.Mapped
[federation]
remote_id_attribute = HTTP_OIDC_ISS
trusted_dashboard = http://dbit-openstack.com/horizon
```

**Description:** The above entries are to be made in the keystone configuration file.

Configuration is added in the Apache keystone file. This includes enabling OIDC, adding Google metadata, a redirect URI among other things.

```
openstack group create federated_users
openstack role add member –group federated_users –project demo
```

**Description:** These commands create a group called federated users in OpenStack and each new addition to this group is assigned the role of a member(minimum privileges).

A mapping.json file is created that will assign any user that logs in using their Google credentials to the federated users group.

*openstack identity provider create google –remote-id https://accounts.google.com*
*openstack mapping create google_mapping –rules mapping.json*
*openstack federation protocol create oidc –identity-provider google –mapping google_mapping*

**Description:** The above commands are used to create an identity provider (Google) for OpenStack and enable the usage of OIDC plugin for that particular identity provider.

Finally, the sso-callback template is set that helps to redirect the user back to Horizon.

#### 4.3.2.2 Changes in Horizon:

Configuration changes are made in the local setting python file to include multiple options for signing in to the cloud.

## 4.4 Errors encountered and solutions tried

OpenStack is installed on two computers according to the specification provided above. Since then various websites were referenced to know the content of the URLs in following files:
*1.OIDCRedirectURI – wsgi-keystone.conf*
*2.trusted_dashboard – keystone.conf*
*3.OPENSTACK_KEYSTONE_URL – local_settings.py*

It was found eventually that those URLs referenced the keystone server that is the server hosting the dashboard. There were a few more sites that indicated that a domain name for the web server (controller node) created was needed. We now have a domain name which is dbit-openstack.com that points to the keystone server.

There was still an error of *'Site cannot be reached'* that was faced on clicking on the opened connect option. There was a solution of forwarding port 5000 since it was a non-standard port, that needed opening by adding a rule in the router configuration.

After forwarding ports a ***401 Authorization error – "Request requires authentication from*** <ip address of the host machine (The computer from which browser requests are made)> was faced. (Since the web server/controller node is hosted inside virtualBox on a personal computer that uses a Netgear router.)

There is another error in the */var/log/keystone/keystone_wsgi_public.log* which says *"Missing entity ID from environment"* . On researching, it was found out that this error occurs when the server is not able to access the *remote_id_attribute = HTTP_OIDC_ISS* from the *[federation]* section of */etc/keystone/keystone.conf*. We tried inserting another section called [oidc] and adding the attribute value there but there was no change in the error.

Another WARNING from the log file indicates that under *[auth]* section in keystone.conf file the parameter *oidc = keystone.auth.plugins.mapped.Mapped is deprecated in Liberty in favor of its entrypoint for keystone.auth.oidc*. When changed to keystone.auth.oidc it gave an internal server error. On searching, it found out that it was a general warning that can be neglected.

The whole setup was then installed on Amazon Web Service virtual instance to check for the issue of opening of ports, but the same server error was encountered again. Adding a security group rule for the particular port was tried, which did not work.

The following solutions have been tried and tested:
1. Exported the *OS_URL=http;//controller:5000/v3*

2. Bought a domain name dbit-openstack.com and linked it to the OAuth client created on Google Cloud Platform

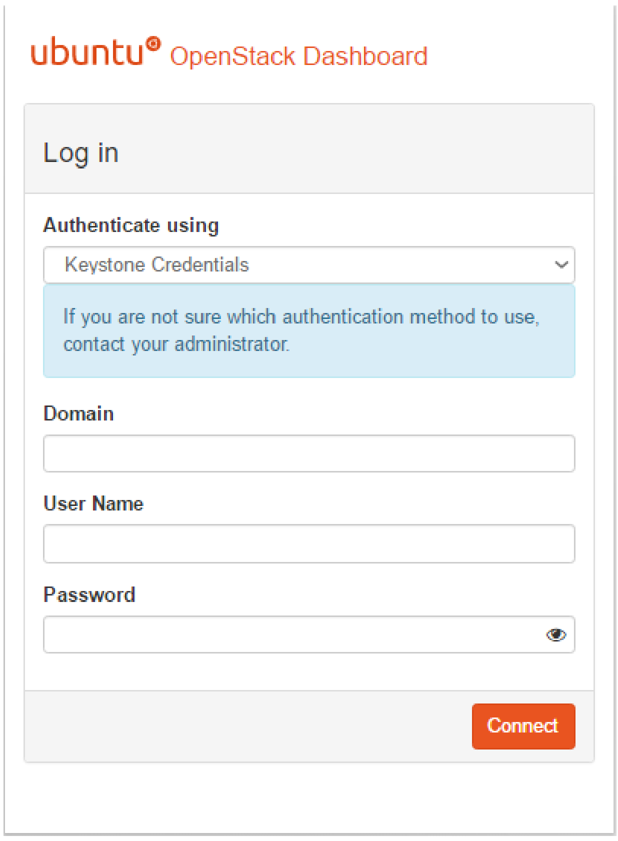
3.Changed the port to 35357 (because, a website stated there isn't much difference in the ports.)

4.Tried adding a [oidc] section in the keystone.conf file, yet the same 401 error persists.

5.Installed the whole OpenStack setup on an AWS virtual instance.

**NOTE:** We have created an OAuth client on Google cloud Platform and have put the Authorized Redirect URI in the given field and have also verified our domain name by following Google verification procedure.

## 4.5 Implementation Results


ubuntu® OpenStack Dashboard

Log in

Authenticate using

Keystone Credentials

If you are not sure which authentication method to use, contact your administrator.

Domain

User Name

Password

**Figure 4.3:** Default Authentication Option provided by OpenStack

**Description:** The above image shows the login screen where the default mechanism for OpenStack is provided.

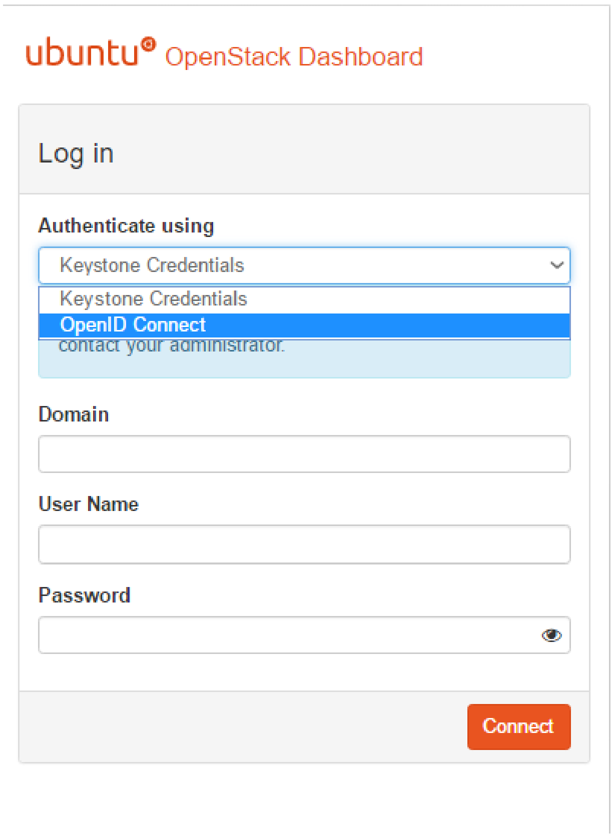
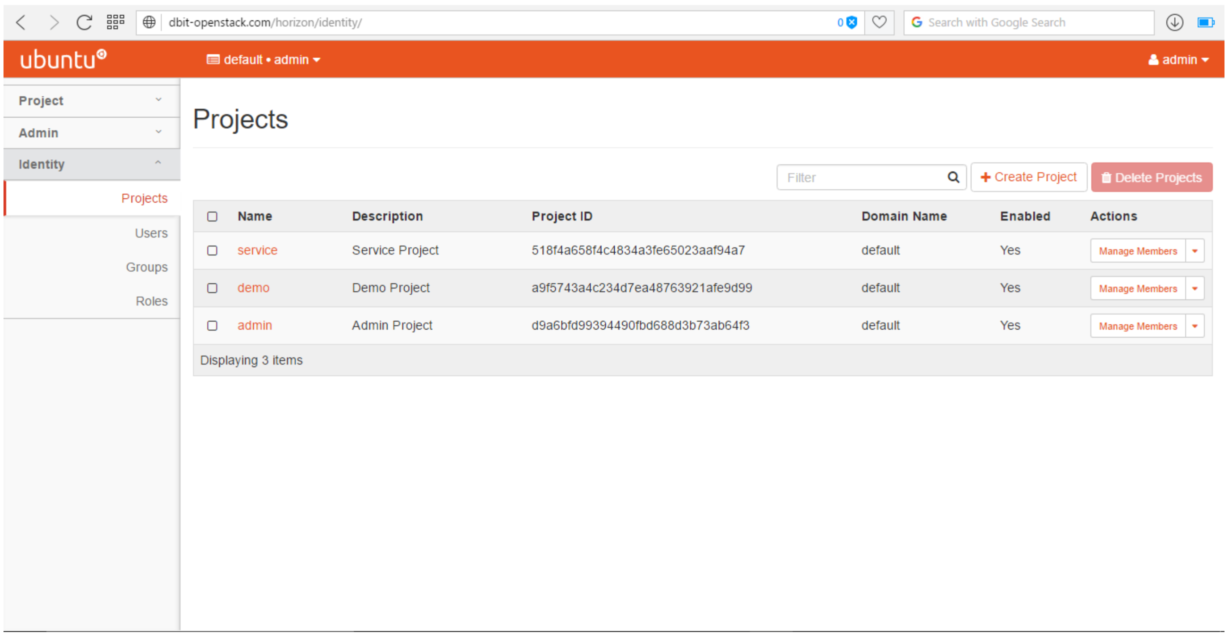


**Figure 4.4:** Authentication using OpenID Connect option

**Description:** The above picture is a snapshot of the cloud system with additional provisions for authentication using OpenID Connect.



Projects

| | Name | Description | Project ID | Domain Name | Enabled | Actions |
|---|---|---|---|---|---|---|
| ☐ | service | Service Project | 518f4a658f4c4834a3fe65023aaf94a7 | default | Yes | Manage Members |
| ☐ | demo | Demo Project | a9f5743a4c234d7ea48763921afe9d99 | default | Yes | Manage Members |
| ☐ | admin | Admin Project | d9a6bfd99394490fbd688d3b73ab64f3 | default | Yes | Manage Members |

Displaying 3 items

**Figure 4.5:** Dashboard when a user is logged in

**Description:** The picture above shows the admin dashboard of OpenStack.

# Chapter 5

# Results and Discussion

## 5.1 Summary

OpenStack software controls large pools of compute, storage, and networking resources throughout a datacenter, managed through a dashboard or via the OpenStack API. OpenStack works with popular enterprise and open source technologies making it ideal for heterogeneous infrastructure.[3]
There is a growing need to make login procedures easy for all applications. Also managament of user related database becomes easier and less tedious. Installation of a private cloud has become a future necessity due to increasing generation and processing of data on a day-to-day basis.
This requirement is fulfilled by the above mentioned open source software with easy access and data management. To make the authentication and authorization procedure easier, Single Sign On using Google was necessary.
Google is the most used platform for variety of services provided by it. Thus, adding the above mentioned feature using Google as an identity provider seemed the most convenient option.
The open source cloud operating system of OpenStack is installed and authentication mechanisms are altered.
The procedure used to alter the system was finalized.
The workflow for adding Google Authentication into the system with the help of OAuth 2.0 was also finalized.
The errors that were faced were to due use of non-standard ports required for communication with the Keystone server. In the process of solving the issues many new provisions provided by the software were encountered. Debugging and analysing log files and researching about the specifics were procedures followed.

# Chapter 6

# Conclusion

## 6.1 Summary

OpenStack is an open source software used to set up a private cloud in any organization or enterprise. Keystone is the primary component that we worked on. Our project is based on providing multiple ways for a user to authenticate themselves to use the cloud resources, and thus allowing Single Sign-On(SSO). This means that a user can sign in on OpenStack using their existing Google credentials. They will also be signed in on OpenStack if they are signed in on Google or any other Google service. This goes a long way in providing a satisfactory customer experience. Enabling Google sign-in in OpenStack will provide a much simpler and convenient authentication method without the need of additional login credentials.

**Figure 6.1:** OpenStack + Google

# Appendix - I

**1. OpenStack** - OpenStack is a cloud environment that controls large pools of compute, storage, and networking resources throughout a datacenter, all managed through a dashboard that gives administrators control while empowering their users to provision resources through a web interface.

**2. SSO** - Single Sign-On. It enables a user to log in with a single ID and password to gain access to a connected system or systems without using different usernames or passwords.

**3. OAuth** - OpenAuthorization protocol. It is used to authorize an external identity provider to authenticate users.

**4. OIDC** - OpenID Connect. It is an authentication layer above OAuth (authorization layer.

**5. IdP** - Identity provider.

**6. SAML** - Security Assertion Markup Language. It is an XML-based data format for exchange of authentication and authorization data.

**7. Keystone** - OpenStack component that deals with authentication and authorization of users.

**8. Horizon** - OpenStack component that gives a user interface to the cloud.

**9. Nova** - OpenStack component that deals with computing.

**10. Glance** - OpenStack component that deals with image file(iso) management.

**11. Neutron** - OpenStack component that deals with networking between all OpenStack components.

# References

[1] Chadwick, David W., et al. ”Adding federated identity management to OpenStack”. Journal of Grid Computing 12.1 (2014): 3-27.

[2] Steve Martinelli, Henry Nash, Brad Topol. Identity, Authentication, and Access Management in OpenStack. O’Reilly Media. 2015. [Book].

[3] Introduction to OpenStack. [Online].
Available:`https://www.openstack.org`

[4] OpenStack Mitaka installation guide. [Online].
Available:`https://docs.openstack.org/mitaka/install-guide-ubuntu/`

[5] OpenStack Foundation. Keystone Federation and Horizon. [Online].
Available:`http://docs.openstack.org/developer/keystone/`

[6] How is Oauth different from Oauth 1 [Online].
Available:`http://stackoverflow.com/questions/4113934/how-is-oauth-2-different-from-oauth-1`

[7] Oauth 2.0. [Online]
Available:`https://oauth.net/2/`

[8] OAuth. From Wikipedia. [Online].
Available:`https://en.wikipedia.org/wiki/OAuth`

[9] Lightweight Directory Access Protocol. From Wikipedia. [Online].
Available:`https://en.wikipedia.org/wiki/Lightweight_Directory_Access_Protocol`

[10] The OAuth 2.0 Authorization Framework Author: Dick Hardt
Available:`https://tools.ietf.org/html/rfc6749`

[11] Python Coding Standards. [Online].
Available:`https://docs.openstack.org/developer/hacking/`

[12] Django Coding Standards. [Online].
Available:https://docs.djangoproject.com/en/dev/internals/contributing/writing-code/coding-style/

[13] OpenID Connect. [Online].
Available:http://openid.net/connect/

[14] Adding OpenID Connect Authentication to OpenStack. [Online].
Available:https://docs.openstack.org/developer/keystone/mitaka/federation/websso.html

## Acknowledgements

This research was supported by Don Bosco Intitute Of Technology, Kurla. We are thankful to our institution who provided resources and expertise that greatly assisted the research. We thank our project guide Asst. Prof. Nilesh Ghavate for sharing his knowledge with us during the course of this research. We are also grateful to Asst. Prof. Tayyabali Sayyad who moderated this project and in that line improved it significantly. We would like to show our gratitude to Asst. Prof. Prasad Padalkar who constantly guided us throughout the semester. We are immensely grateful to them for their comments on any working related to our project. Any errors are our responsibility and should not tarnish the reputations of these esteemed professionals.

**(——————————- )**
**(Kunal Dandekar 13)**

**(——————————- )**
**(Apurva Deore 14)**

**(——————————- )**
**(Yash Deorukhkar 15)**

**(——————————- )**
**(Tanaya Mulik 44)**

**Date: 24/04/2017**