Cloud based IT Infra with Central Identity

Phase II – Project Report

Project Guide

T. Chandra Shekar
Dept. of CSE – RGUKT Nuzvid
chandra.indra@gmail.com

Project Team

T. Aneesh Kumar	N090247
P. Nageswarao	N091030
P. Anesh	N090977
P. Jyothi Ram	N090990
K. Naresh Chowdary	N090331
N. Venkata Sateesh	N090935
M. Sanyasi Rao	N090891

Dept. of Computer Science and Engg.
R.G.U.K.T. - Nuzvid
Krishna Dt. - Andrha Pradesh - 521202

Jan 2015 – Apr 2015

Abstract

The main objective of "Cloud based IT Infra with Central Identity" Phase II is provide the implementation to our objectives.

Implementing the Web based central identity, network based central identity, achieve the combination and deploy all these in the private cloud

Contents

List of Figures

Chapter 1

Introduction

1.1 Introduction

"Cloud Based IT Infra with Central Identity" is a complete solution, based on private cloud to enhance and effiecient utilization the IT Infrastructure of an emerging Universities and Organizations with Central Identity for all its users to access its services.

It is going to be developed in 3 phases

- *Private cloud*
- *Deploying Network Services*
- *Central Identity*

1.1.1 Private Cloud

Private Cloud establishment is targeted for hardware resource pooling, providing high computational and scalable virtual machines for deploying network based applications (smtp, proxy, ftp), web application and Network storage.

1.1.2 Deploying Network Services

Configuration of Uniform hardware experience over the complete university includes single sign on on every device, configuration of mail servers etc.

1.1.3 Central Identity

Essential part that combines normal network services(proxy, mail, etc.) and organizational web & native applications. In addtion to that this central identity is available to thrid party developers as API with dynamic based role user authentication protocols.

Chapter 2

Phase I Work

As part of Phase I, we have done literature survey and anlyzed feasability of the several components

2.1 Components

- Central Identity
  - Single Sign-On with REST API
  - Identity Management
  - Dynamic Role Based Access Control
- Network Based Central Identity
  - LDAP Servers
  - NFS Servers
- Cloud Computing
  - Cloud Characterstics
  - Service Models
  - Deployment Models
- Private Clouds
  - Introduction
  - Open Source Tools

Chapter 3

Phase II Work

As part of Phase II, we have tried to implement some of the above mention components

3.1 Components

- Web based Signle Sign On
  - OAuth Provider
  - University Users Profiles
  - REST API
  - Support of assigin roles to users with their permission set
  - Testing oauth client library in PHP using php-curl
- Network Components
  - LDAP Server
  - NFS Server
  - Haproxy
  - GlusterFS
  - XtreemFS
- Private Infrastructure Cloud
  - Openstack Architecture
  - Installation
  - Virtual Machines

Chapter 4

Web based Single Sign-On

4.1 OAuth Provider

Here in this section the overall workflow of Authentication in a Single Sign-On System is explained. In order to Authenticate users with the given credentials we must use a robust and stable protocol. And many Single Sign-On systems uses OAuth protocols for this purpose. Single Sign-On uses OAuth protocol for both Authentication & Authorization.

4.1.1 OAuth Protocol

OAuth is an authentication protocol that allows users to approve application to act on their behalf without sharing their password. The below figure explains the flow of Authentication in OAuth Protocol.

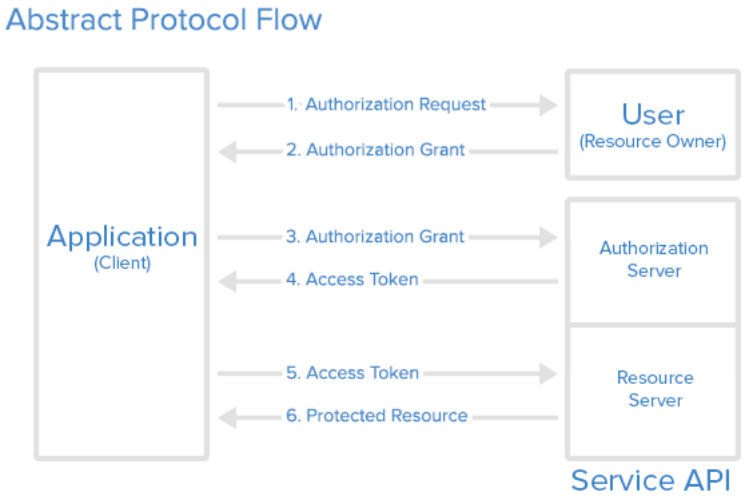

Figure 4.1: OAuth Protocol Work Flow Diagram

Steps invloved in the above flow diagram

- The application requests authorization to access service resources from the user
- If the user authorized the request, the application receives an authorization grant

- The application requests an access token from the authorization server (API) by presenting authentication of its own identity, and the authorization grant
- If the application identity is authenticated and the authorization grant is valid, the authorization server (API) issues an access token to the application. Authorization is complete.
- The application requests the resource from the resource server (API) and presents the access token for authentication
- If the access token is valid, the resource server (API) serves the resource to the application

4.2 University User Profiles

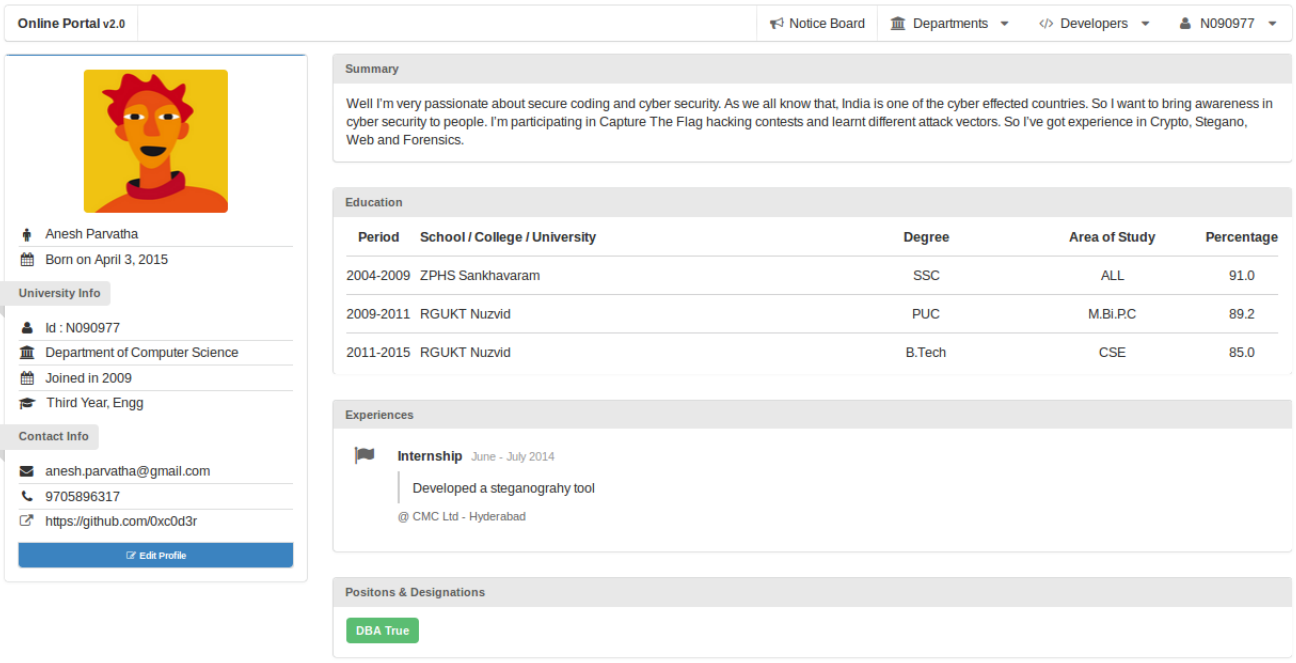

Figure 4.2: User Profile

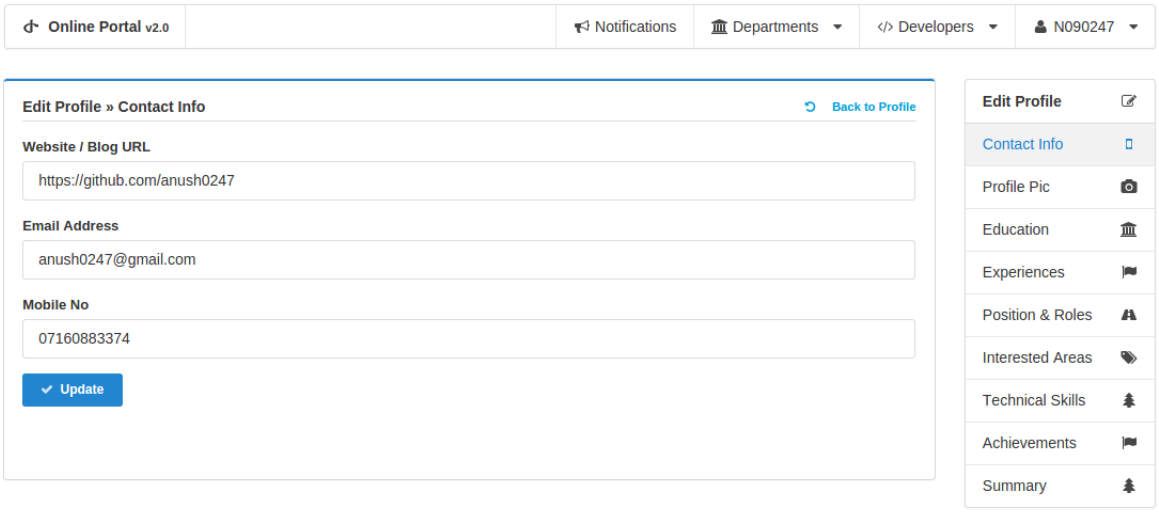

Figure 4.3: Edit User Profile

4.3 REST API

REST stand for **RE**presentational **S**tate **T**ransfer. It's a collection of simple URIs, and HTTP calls like GET,PUT,POST,DELETE to those URIs to get some Protected data from a Resource Server. Once User provides a set of valid credentials to OAuth Provider it will generate an access token to that particular user, with that token user can fetch protected resources from the resource server by making basic HTTP calls through REST API. REST API provides users a flexibility to perform Basic CRUD(Create,Read,Update,Delete) on resource server.

operation	HTTP verb	action
Create	`POST /posts`	`create`
Read	`GET /posts/1`	`show`
Update	`PUT /posts/1`	`update`
Delete	`DELETE /posts/1`	`destroy`

Figure 4.4: CRUD Operations

4.3.1 Technologies Used

- django >= 1.6.0
- SQLite3
- django-rest-framework
- oauth-tool-kit
- semantic ui 2.0

4.3.2 API Testing

/api/basic_info/?access_token=<token>


```
{
    "rid": "N090977",
    "first_name": "Anesh",
    "last_name": "Parvatha",
    "date_of_birth": "2015-04-03",
    "gender": "M"
}
```


/api/contact_info/?access_token=<token>


```
{
    "mobile": "9705896317",
    "url": "https://github.com/0xc0d3r",
    "email": "anesh.parvatha@gmail.com"
}
```


/api/education/?access_token=<token>


```
[
    {
        "school": "ZPHS Sankhavaram",
        "period": "2004-2009",
        "degree": "SSC",
        "stream": "ALL",
        "grade": 91.0
    },
    {
        "school": "RGUKT Nuzvid",
        "period": "2009-2011",
        "degree": "PUC",
        "stream": "M.Bi.P.C",
        "grade": 89.2
    },
    {
        "school": "RGUKT Nuzvid",
        "period": "2011-2015",
        "degree": "B.Tech",
        "stream": "CSE",
        "grade": 85.0
    }
]
```


/api/skills/?access_token=<token>


```
[
    {
        "skills": [
            {
                "id": 1,
                "title": "C"
            },
            {
                "id": 2,
                "title": "java"
```

```
            },
            {
                "id": 3,
                "title": "cloud"
            },
            {
                "id": 4,
                "title": "computing"
            },
            {
                "id": 5,
                "title": "python"
            },
            {
                "id": 6,
                "title": "mongldb"
            }
        ]
    }
]
```

/api/roles/?access_token=<token>

```
[
    {
        "role": {
            "id": 2,
            "title": "DBA",
            "is_verified": false,
            "permissions": [
                {
                    "id": 4,
                    "title": "DB Delete",
                    "is_verified": true
                },
                {
                    "id": 5,
                    "title": "DB Edit",
                    "is_verified": true
                },
            ]
        },
        "is_verified": true
    }
]
```

4.4 Roles & Permissions

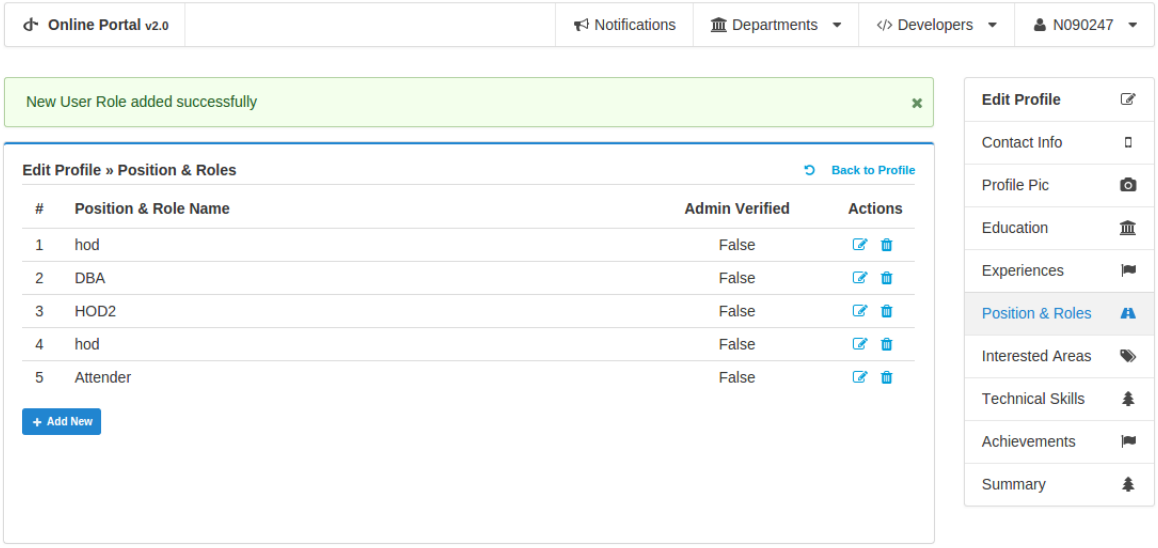

Figure 4.5: List of Roles of User

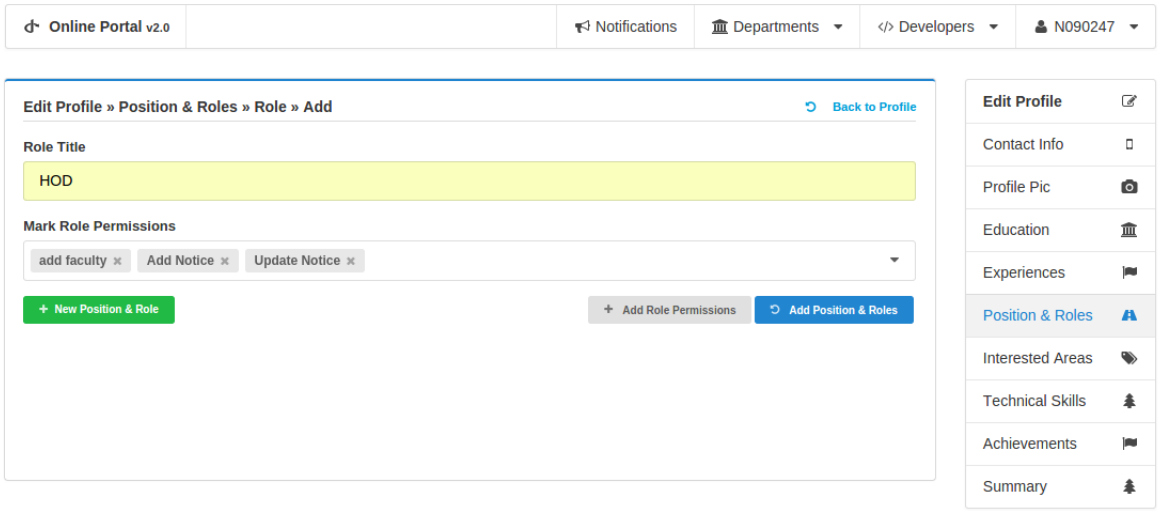

Figure 4.6: Adding Permissions Option

4.5 PHP Client Libraray

We developed a Client Library for PHP Applications. We used PHP-cURL to perform all the http calls and post requests to get protected data from API Server. And We developed it in a modular way with Object-Oriented approach. And all the function calls in the PHP library is self-explanatory.

4.5.1 Initializing the Client Library

```
<?php
include("Class.RIDOAuth.php");
$oauth=new OAuth("<ClientID>","<ClientSecret>");
?>
```

4.5.2 Get Authorization URL

```
$url=$oauth->getAuthorizeURL("<RedirectURI>");
```

4.5.3 Get Access Token

```
$token=$oauth->getAccessToken("<AuthorizationCode>","<RedirectURI>");
```

4.5.4 Initializing API with Access Token

```
$api=new API("<Access Token>");
```

4.5.5 Getting User Info from API

```
$user=$api->get("<API Endpoint>");
```

Chapter 5

Network Single Sign-On

5.1 Introduction

Single sign-on (SSO) is a session/user authentication process that permits a user to enter one name and password in order to access multiple applications. The process authenticates the user for all the applications they have been given rights to and eliminates further prompts when they switch applications during a particular session.

Components Used:

- LDAP Server
- phpLdapAdmin
- LDAP Client
- HAProxy
- GlusterFS
- XtreemFS

5.2 LDAP Server

LDAP, or Lightweight Directory Access Protocol, is a protocol for managing related information from a centralized location through the use of a file and directory hierarchy. It functions in a similar way to a relational database in certain ways, and can be used to organize and store any kind of information. LDAP is commonly used for centralized authentication.

5.2.1 Installation and Configuration

The OpenLDAP server is in Ubuntu's default repositories under the package "slapd". We have to install some additional utilities in order to use it in full pledged way.

- sudo apt-get **update**
- sudo apt-get install **slapd ldap-utils**

After the installation is complete, we actually need to reconfigure the LDAP package by the following

- sudo **dpkg-reconfigure slapd**

By following below steps we have to configure the LDAP

- Omit OpenLDAP server configuration? **No**
- DNS domain name? **reboot.org**
- Organization name? **reboot**
- Administrator password? **Password**
- Database backend to use? **HDB**
- Remove the database when slapd is purged? **No**
- Move old database? **Yes**
- Allow LDAPv2 protocol? **No**

5.3 phpLDAPadmin

Its a web-based LDAP client. It provides easy, anywhere-accessible, multi-language administration for LDAP server. By this configuration and monitor of LDAP Server will be done in an easy way.

Its hierarchical tree-viewer and advanced search functionality make it intuitive to browse and administer your LDAP directory. Since it is a web application, this LDAP browser works on many platforms, making your LDAP server easily manageable from any location.

5.3.1 Installation and Configuration

- sudo apt-get install **phpldapadmin**

After the installation is complete configuration will be done by making following changes in the config.php file of phpLDAPadmin.

- sudo nano **/etc/phpldapadmin/config.php**

```
$servers->setValue('server','host','10.4.34.47');
$servers->setValue('server','base',array('dc=reboot, dc=org'));
$servers->setValue('login','bind_id','cn=admin, dc=reboot, dc=org');
$config->custom->appearance['hide_template_warning'] = true;
```

Listing 5.1: PHP Config file

5.3.2 Web Interface of phpLDAPadmin:

Figure 5.1: phpLDAP

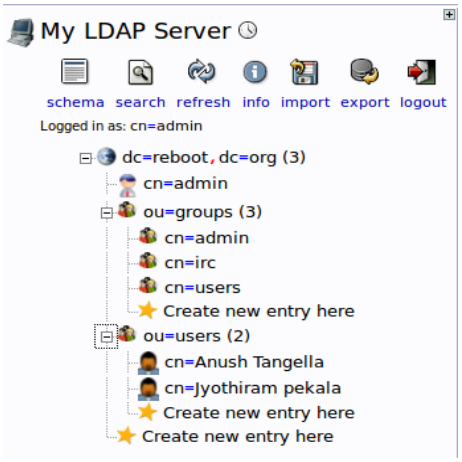

Figure 5.2: Complete category of LdapServer

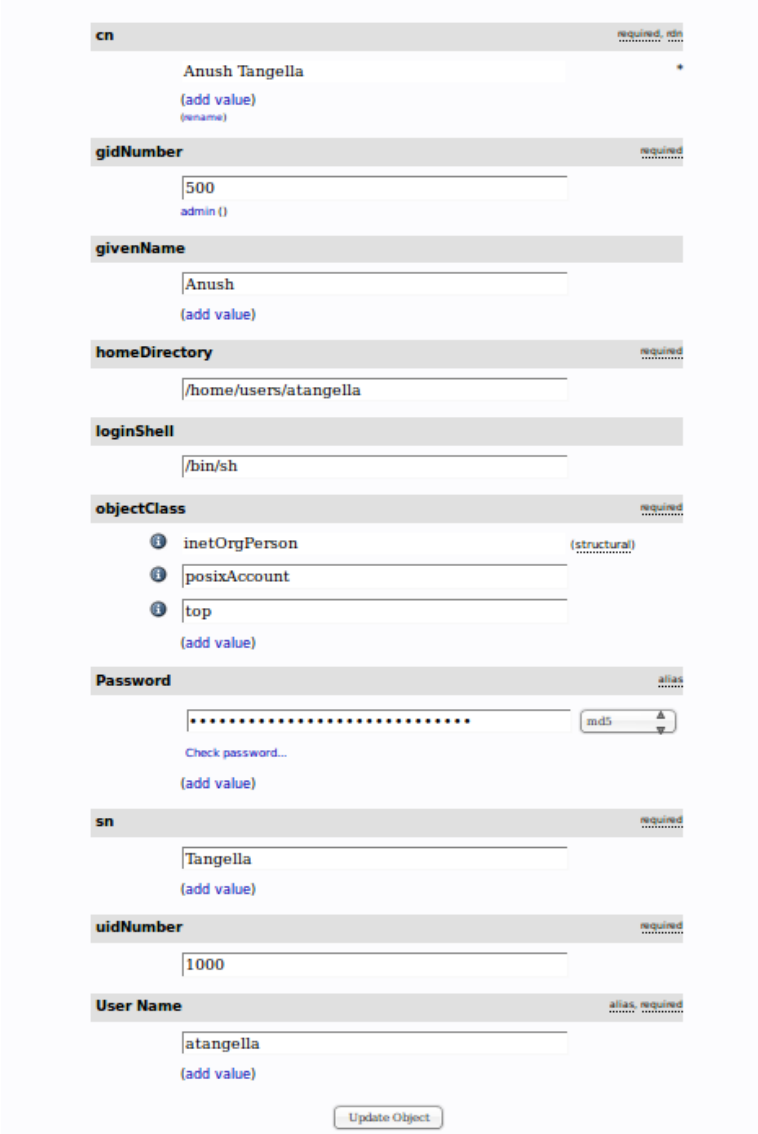

Figure 5.3: User Creation

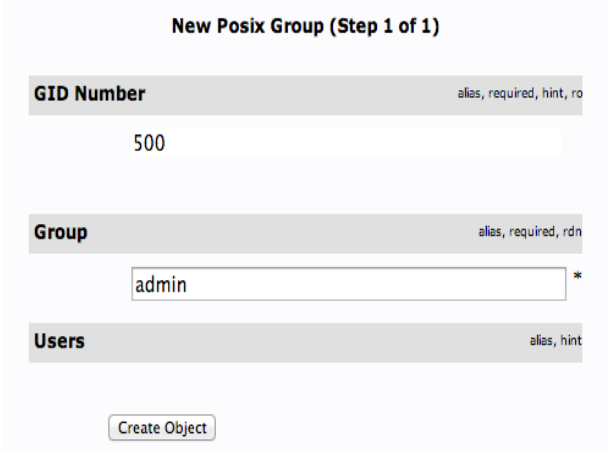

Figure 5.4: Group Creation

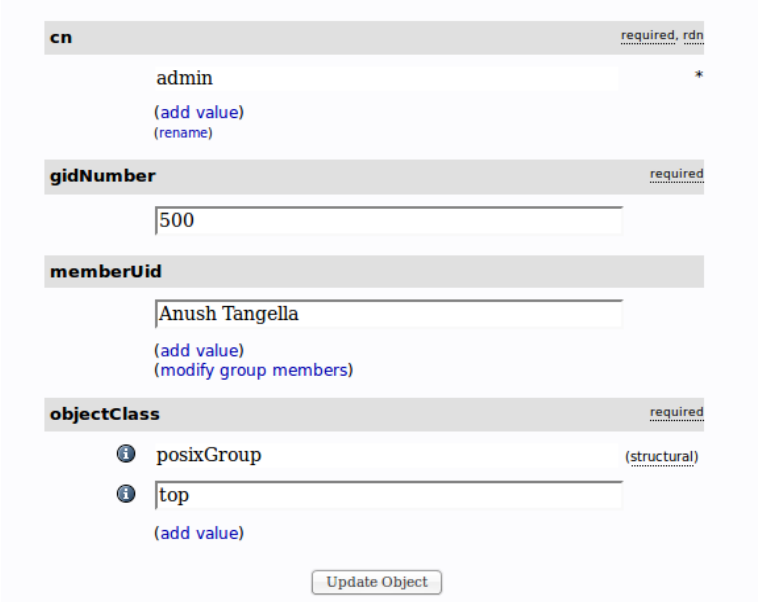

Figure 5.5: Adding User to Group

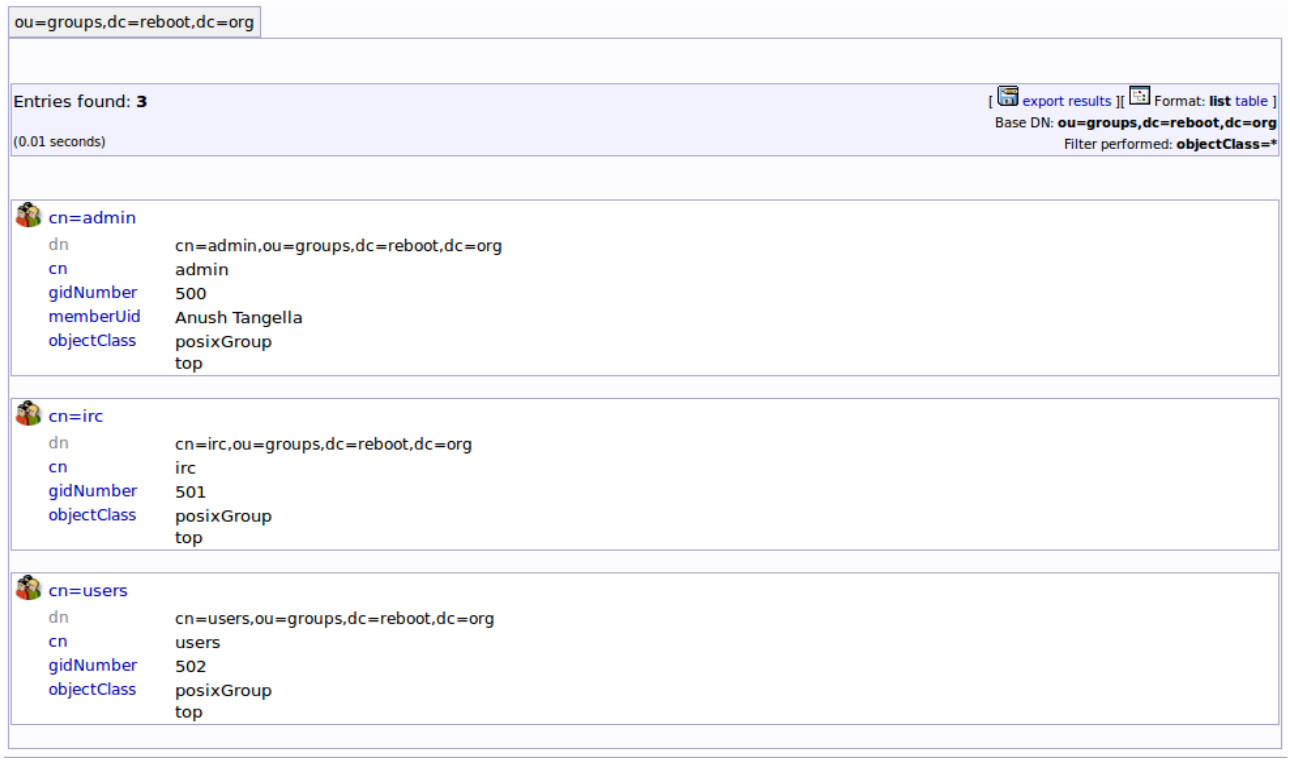

Figure 5.6: Groups information

5.4 LDAP Client:

LDAP, or Lightweight Directory Access Protocol, is one way of keeping authentication information in a single centralized location. We need another droplet to act as the client machine.

5.4.1 Installation and Configuration:

On the client machine, we need to install a few packages to make authentication function correctly with an LDAP server.

- sudo apt-get install **libpam-ldap nscd**

By following these below steps we need to configure the LDAP Client

- LDAP server Uniform Resource Identifier: **ldap://10.4.34.47/** from **"ldapi:///"**

Distinguished name of the search base: This should match our values in LDAP server's /etc/phpldapadmin/config.php file.

- We have to replace " 'server','base',array " within the file to **"dc=reboot,dc=org"**
- LDAP version to use: **3**
- Make local root Database admin: **Yes**
- Does the LDAP database require login? **No**
- LDAP account for root:
  - This should also match with our values in your **/etc/phpldapadmin/config.php**
  - Search for: **" ' login ' , ' bind_id ' "** within the file
  - Our example was **"cn=admin,dc=reboot,dc=org"**

LDAP root account password: **Our-LDAP-root-password**

If made a mistake and need to change a value, we can go through the menu again by issuing this command:

- sudo **dpkg-reconfigure ldap-auth-config**

To configure client we adjust a few files that they can look to our LDAP server for authentication information. First, we have to edit the /etc/nsswitch.conf file. This will allow us to specify that the LDAP credentials should be modified when users issue authentication change commands

- sudo nano **/etc/nsswitch.conf**

The three lines we are interested in are the "passwd", "group", and "shadow" definitions. Modify them to look like this:

```
passwd : files ldap
group : files ldap
shadow : files ldap
```

Listing 5.2: Config file

We have to add the values to our PAM configuration.

PAM, or Pluggable Authentication Modules, is a system that connects applications that can provide authentication to applications that require authentication. When we installed and configured our LDAP PAM module, most of the needed information was added to the configuration files and we need to edit below file.

- sudo nano **/etc/pam.d/common-session**
- sudo nano **/etc/pam.d/login**
- sudo nano **/etc/pam.d/lightdm**

We have to add the below piece of code to each of the above PAM configuration files

- session required **pam_mkhomedir.so skel=/etc/skel umask=0022x**

The above will create a home directory on the client machine when an LDAP user logs in who does not have a home directory. We have to restart a service for these changes to be implemented:

- sudo **/etc/init.d/nscd restart**

5.4.2 Log In as an LDAP User:

We have now configured our client machine enough to be able to log in as one of our LDAP users. This user does not have to exist on the client machine. In order to connect to LDAP Client, we have to ssh into that particular machine.

- ssh **atangella@10.4.34.45**

Chapter 6

Private Infrastructure Cloud

To support this central identity both the network and web network central identity we want to go for the private cloud deployment it includes creating the Private Infrastructure Cloud with openstack and creating Virtual Machines for instaling these services and assign them the IP address.

6.1 Openstack Architecture

Openstack is a cloud operating system that provides the 3 main services for the Infrastructure clouds namely Stoage, Compute, Networking and some other components are can be added later as addons

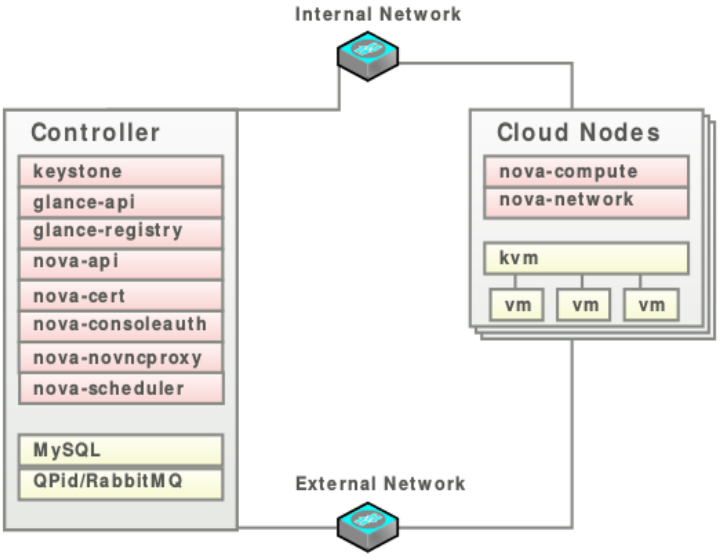

Figure 6.1: Openstack Architecture.

6.2 Installation

Installing openstack includes component wise installation namely NTP, MySQL, Rabbitmq-Server, Keystone, Nova, Cinder, Glance, Neutron

6.2.1 NTP

apt-get install ntp

6.2.2 MySQL

apt-get install mysql-server

6.2.3 Rabbitmq-server

apt-get install rabbitmq-server

6.2.4 Keystone

apt-get install keystone

6.2.5 Glance

apt-get install glance python-glanceclient

6.2.6 Nova

apt-get install nova-api nova-cert nova-conductor nova-consoleauth nova-novncproxy nova-scheduler python-novaclient

6.2.7 Neutron

apt-get install neutron-server neutron-plugin-ml2

6.3 Virtual Machines

This Virtual Machines are created from the resource pool after successfull installation openstack

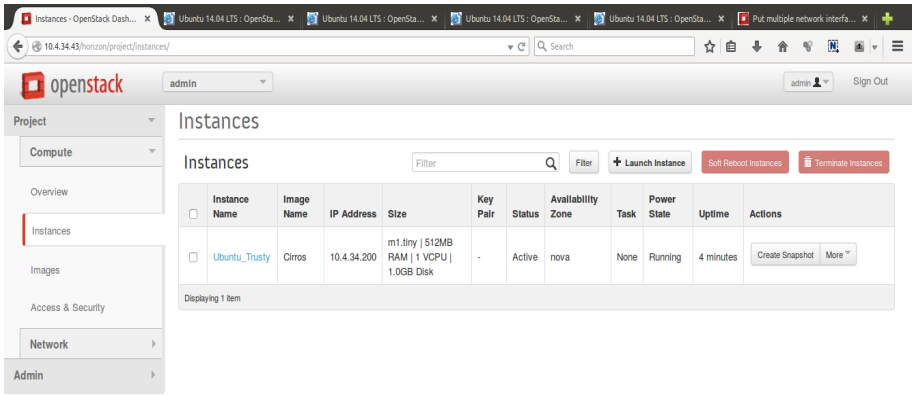

Figure 6.2: Openstack Virtual Machines.

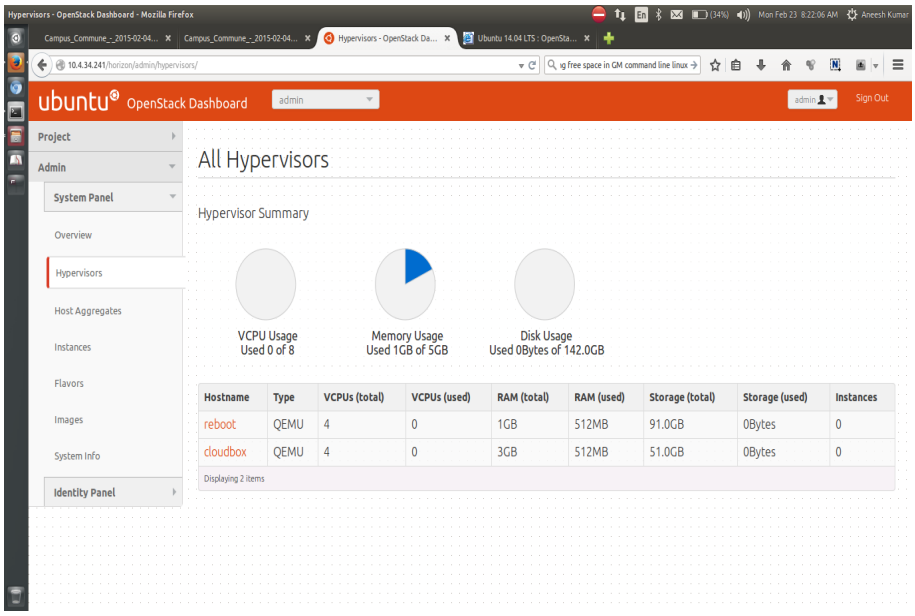

Figure 6.3: Openstack Resource Pool.

Chapter 7

Conclusion & Future Work

7.1 Conclusion

We tried GlusterFS for replication among systems, but its not working if any one of the system fails. Then we found that XtreemFS works well in distributed system and provides fault tolerant solution.

We developed network based sign on using LDAP and web based single sign on along with REST API using Oauth 2.0 and Django. We tried to create private cloud using openstack but lot of errors came because of proxy based internet and low configured PCs.

7.2 Future Work

We would like to combine Network single sign-on with Web based single sign on along with XtreemFS and HAProxy. Creating virtual machines and Private cloud is not possible with the available systems. But if we could provide systems with enough configuration, sure we can create better sophisticated solution

Chapter 8

References