Cloud based IT Infra with Central Identity

A project report
submitted in partial fulfillment of the
requirements for the degree of

Bachelor of Technology

in

Computer Science and Engineering

by

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

Under the Guidance of

Mr. T. Chandra Shekhar M.Tech (IIT KGP, AU)

Lecturer, Dept. of CSE – RGUKT Nuzvid

Department of Computer Science and Engineering
Rajiv Gandhi University of Knowledge Technologies - Nuzvid
Krishna Dt. - Andrha Pradesh - 521202

Sep 2014 – Apr 2015

Abstract

The main objective of “Cloud based IT Infra with Central Identity” is to create a private cloud and availing access of all its services using central identity with single signon through dynamic role based management along with REST API to third party for application developers and users.

We would like to work on Network single sign-on using LDAP, NFS and Web based single sign on using Django and Oauth2. We will develop distributed systems with fault tolerant file replication and load balancing environment using GlusterFS, XtreemFS and HAProxy.

We will develop REST API for third party application developers. We will combine all these components and deploy in private cloud developed using Openstack.

Contents

List of Figures

List of Tables

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

Motivation & Approach

2.1 Existing System

- The environment we observed is our university, it consists of 7000 students and more than 500+ faculty with 6 core Engg. departments apart from 2 years of PUC course.
- Each Department is having strength of 700 students they arrenged these students into various various classes of 60 to 70 each, total 10 to 12 number.
- Each student is provided with one Laptop with 2G RAM, 1.5GHz Clock speed and 200 GB Harddisk Storage.
- All students are using these sytems for more than 8 hours in a day.
- All these students has to provided with course content and course labs, they are maitaining dedicated labs with 50 - 60 machines.

2.2 Problems under consideration

We have observed these problems over our University

- Failed to maintain large user load web services and network applications.
- No Central Identity, Storage & High capacity hardware resource pool.
- Inadequate resource requirements for Research.
- Dedicated computer course labs like Matlab, VLSI, etc. and these labs are useful only at lab hourse most of time they are idle.
- Redundent data and failed to moniter the content over student laptops.

2.3 Proposed System

To avoid above mentioned observations we are proposing one new system with

- Cloud based hardware resources clustering.
- Central Identity for Network Applicaitons with REST API.
- Dynamic user role management.
- Providing Virtual Labs (MatLab, etc.,).
- High Configurational Virtual Machines for research.

2.4 Approach

We want to make use of entire departmental hardaware resources more on its students laptops capacity and create a common pool of resources hence easy to maintain and moniter.

We want to provide signle sign on implentation in each class of 60-70 laptops. such that user can use his own unique username to access university resources and later he can use the same password for network based or web based applications such as updates, mails, examinations, results etc.

User data can be data is retrevied from the Storage server like nfs while loging in to his laptop in any class room and his laptop's computational and storage capacity is used by the private cloud extentions that it make his laptop as slave node when ever laptop is available.

User can develop application and they can use Central Identity in their application with user control and access specifations for API calls.

Chapter 3

System Design

3.1 Users & IT Services

We are grouping all IT Services that are required for University into one and identifing the user who will going to use them. All Users are catagorized into 4 groups [1]

- Studens
- Developers
- Staff, faculty
- Researches

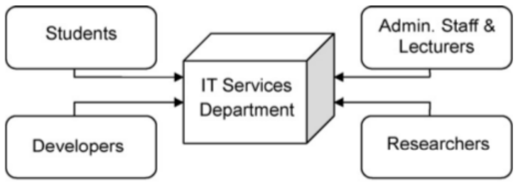


Figure 3.1: Simplified structure of the main users of IT services in a typical university.

All University IT Services are deployed in a private cloud, constructed over exsiting infrastructure, that can be broadly viewed as

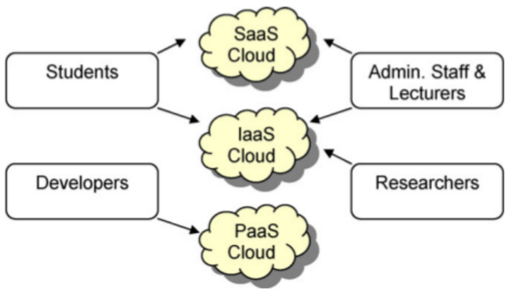


Figure 3.2: IT Services and Users in Cloud Computing [1]

3.2 Components Identified

In this project we are restricting ourselves to some componets of the above mentioned system, we are planing to develop them in upcoming semister. We have done literature survey about these componets

- Central Identiy
  - Single Sign on
  - Identity Management
  - Dynamic Role Based Access Control
  - Hybrid version with REST API to third party
- Network Components
  - AAA, LDAP, NFS
- Cloud Infrastructure
  - Cloud Computing, Private Clouds, Open Source tools

3.3 Cloud Infrastructure Architecture

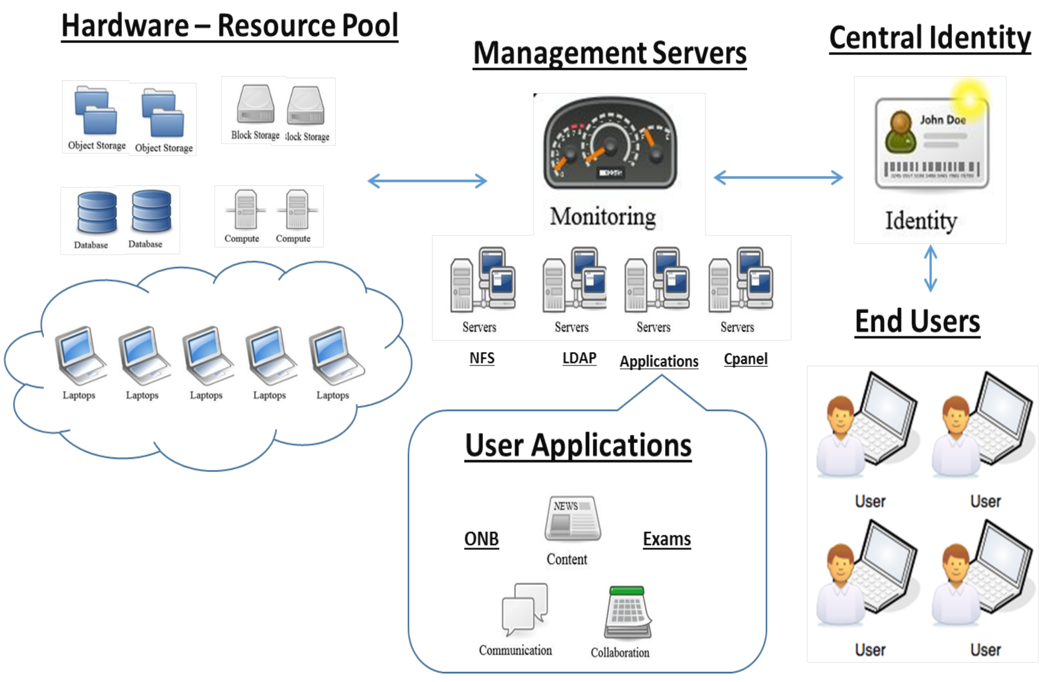


Figure 3.3: Cloud Infrastructure Architecture

Chapter 4

Central Identiy

4.1 Single Sign-On with REST API

4.1.1 Introduction

Single sign on is generally a process that allows the user to access multiple applications requiring authentication by passing his credentials only once. The user first authenticates to some trusted authentication authority and then is granted access to all the applications trusting that authority.

So one of the main advantages of SSO systems is the convenience for the user. Another major advantage is security. There is only one place of authentication, which receives users credentials.Also, the user authenticates only once, so there is minimum transfer of sensitive information over the network.

There is always a central authority, which handles user authentication. It may support various backend authentication mechanisms like Kerberos, LDAP, relational database, etc. The central authentication server may provide also a user interface needed to retrieve users credentials - usually a login form.

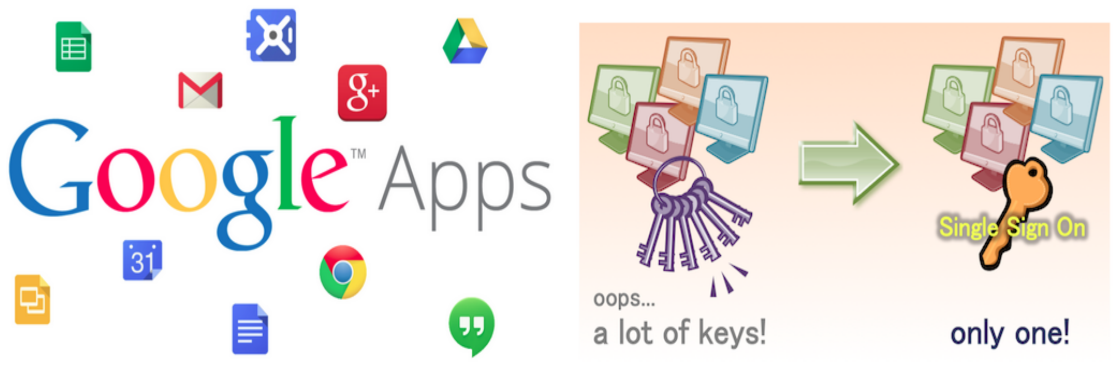


Figure 4.1: Google Single Sign-On System[19]

4.1.2 Why Single Sign-On?

The proliferation of web applications on the Internet has led to a sharp increase in the number of accounts (and corresponding credentials) that a web user has to create and remember. Inconveniences stemming from having to keep track of the accounts, and the tendency of web sites to suffer from security breaches, in which user credentials are exposed, has resulted in a recent push to adopt single sign-on (SSO) systems more widely. As the name implies, these systems help reduce the large amount of account credentials a web user has to keep track of, by replacing these credentials with a single identity with which she can authenticate herself at many different web sites.

In a University scenario there exists so many applications for different purposes. And each application provides it's own sign up form for users and follow their own password policies. It would be very difficult for a user to sign up every time when a new application comes in. And storing the user information at each applications database is a waste of memory and that is a redundant process.

By implementing Single Sign-On in the above scenarios will results in a stable & convinient web experience for the users. And this Single Sign-On has so many advantages.

4.1.2.1 Advantages

- Reduces the number of passwords a user has to remember and change before they expire.
- Reduces the risk of identity theft by limiting the number of password stored on or off campus.
- Users can login once and switch applications without having to login again.
- Allows for better audit controls and management of resource availability.
- Most applications/services are easier to integrate, and often can be plugged in for other services on campus.
- Saves time and effort of a Web Developer by providing a single button like "Login with Central Identity". No need of creating Sign Up & Login pages for every application.
- Reduces phising. If users are accustomed to seeing one and only one screen when entering their credentials it may be easier for them to identify phishing attacks.
- Transfer of sensitive data(like User credentials) across network is minimized.

4.1.3 Single Sign-On Work Flow

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.

4.1.3.1 OAuth Protocol

OAuth[14] 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.

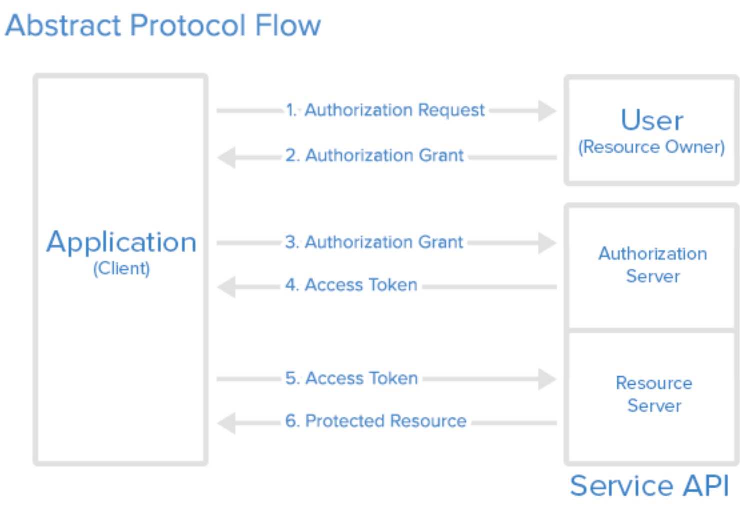


Figure 4.2: OAuth Protocol Work Flow Diagram

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 Identity Management

4.2.1 Introduction

The increasing amount of applications (web, desktop, etc.,) is one of the most important reasons why users have to remember different passwords to authenticate themselves. To simplify the use of the web applications users tend to choose the same password for every application or save the different passwords e.g., directly in the web browser. While this SSO solution is convenient for accessing the applications it also holds some security risks. For example a possible attack is phishing attack.

Secure Single Sign-On mechanism for web applications that were introduced in the past years can be classified in federated and user-centric identity management. Latter ones are more convenient for users as they allow them to log in with a globally unique username (e.g., their e-mail or a URL) and extended privacy by letting the user choose which personal information to send to the web applications. In referred literature they describes a solution to extended SAML-based federations with features of user-centric authentication.

4.2.2 Authentication & Authorization

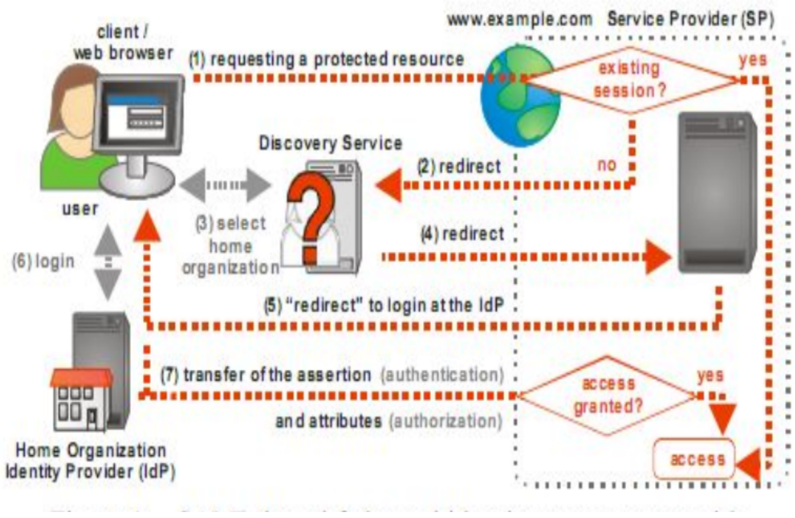
Due to the number and autonomy of the institutes a variety of authentication and authorization technologies are used for the different institutions. A large number of institutes use directory services (LDAP) or Kerberos, some applications use relational databases for authentication. Several legacy applications even use authentication by source IP address, which needs to be migrated in the near future. On the other hand some institutes are starting to use modern authentication techniques e.g., X.509 certificate based.

Traditionally identity management was done at a central point to unify authentication and authorization in heterogeneous IT infrastructures. Currently beside custom identity management there are two solutions to allow decentralized authentication and authorization in web applications: federated identity management normally SAML-based) and user-centric identity management (like OpenID).

4.2.2.1 Federated Identity Management

Single Sign-On was introduced on a large scale with the Kerberos protocol. For Web-based Single Sign-On OASIS issued the Security Assertion Markup Language (SAML) Standard. SAML divides the authentication and authorization in two parties (similar to Kerberos): the Service Provider (SP), being the party that holds the resources the user wants to access, and the Identity Provider (IdP), that holds the identities of the users for example at a local institute (home organization). SPs and IdPs are joined in a federation. The federation is also called an Authentication and Authorization Infrastructure (AAI).

Within the federation all SPs and IdPs trust each other using X.509 certificates. These certificates and addresses to resolve the handlers of the SPs and IdPs are stored in a metadata file that is distributed to all participants and builds the federation. Identity Providers issue a digitally signed assertion (or token) authenticating the user. SPs use the certificates of the IdPs published in the federation to verify the digital signature and accept the authentication. Authorization is done using attributes that the IdPs sends along with the assertion. Examples for this are Shibboleth, Liberty, and ADFS. Below figure illustrates the access to a resource protected by a Shibboleth Service Provider.



client / web browser
user
www.example.com Service Provider (SP)
(1) requesting a protected resource
existing session?
yes
no
Discovery Service
(2) redirect
(3) select home organization
(4) redirect
(6) login
(5) "redirect" to login at the IdP
(7) transfer of the assertion (authentication)
and attributes (authorization)
access granted?
yes
access
Home Organization Identity Provider (IdP)

Figure 4.3: SAML based federated identity management with Shibboleth 2.0. [3]

Users accessing the SP are redirected to a Discovery Service (DS or WAYF server). At the DS the user selects his home organization and is redirected to his IdP that redirects him back to the SP containing the requested resource after successful authentication. A reference to the assertion issued to the user by the IdP is held as a cookie in the web browser to enable Single Sign-On. As the user subsequently accesses SPs belonging to the same federation, the browser sends the cookie referencing the assertion to the IdP and the user is directly able to access the resource without logging in again.

Advantages

- Fine grained authorization and the wide-acceptance of SAML.
- The SAML standard guarantees interoperability between different implementations
- More Secure and prevalent

Dis-Advantages

- Complexity of the protocol and user while logging
- Single Logout
- Unable to manage private info

4.2.2.2 User-Centric Identity Management

The latest development in the evolution of identity management are user-centric approaches. They extends the users privacy as the user can decide which private information to send to the SP (Service Provider). The user presents his/her information as a card (called I-Card) to the SP. It is up to the user which card to show and what information should be on the card.

There is no need for a DS in this. Users can either select their I-Card to get access or log in using a globally unique identifier (e-mail or URL). The responsible Identity Provider (IdP) can be resolved by the Service Provider (SP) using the domain of the e-mail address or the URL given by the user.

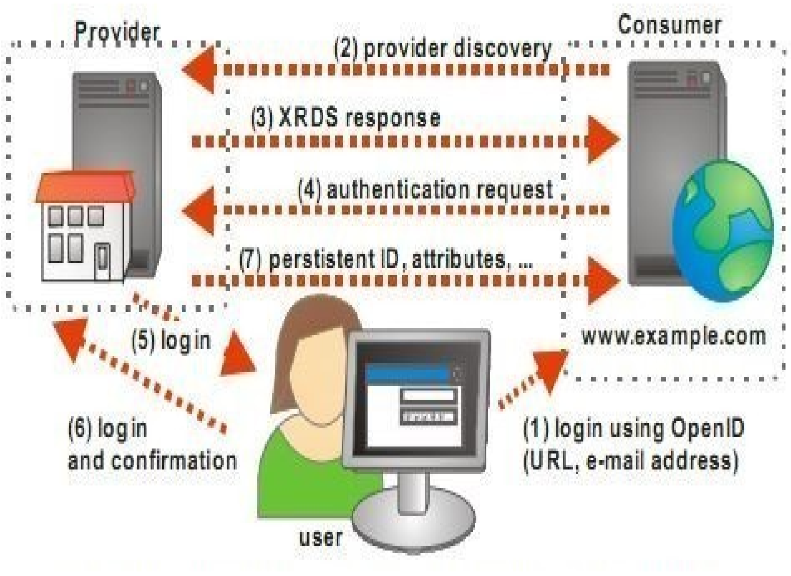


Figure 4.4: OpenId as an example for user-centric identity management. [3]

Above shows an example for user-centric identity management using OpenID. Some other typical implementations are CardSpace, sxip, OAuth, Higgins. Most important extension is the ability to exchange attributes for authorization as shown for SAML.

Advantages

- Higher Usability (use of e-mail or URL)
- Privacy (decide which information to send to the Service Provider)
- Simplification of the protocol & configuration

Dis-Advantages

- Security risks (like Phishing, replay attacks or web application logic flaws e.g., XSS, CSRF)

Besides the disadvantages because of the evolving security fixes e.g., in OpenID, user-centric solutions suffer from the fact that there is no single standard. Another fact that strengthens the role of OpenID is that major web companies like Google,Yahoo offer OpenID Providers for their accounts. This retention bares another disadvantage. The user has to bound to a specific Provider, if his Provider changes his terms or closes down his service, the services the user is registered for are lost.

4.2.2.3 Relating to Our University

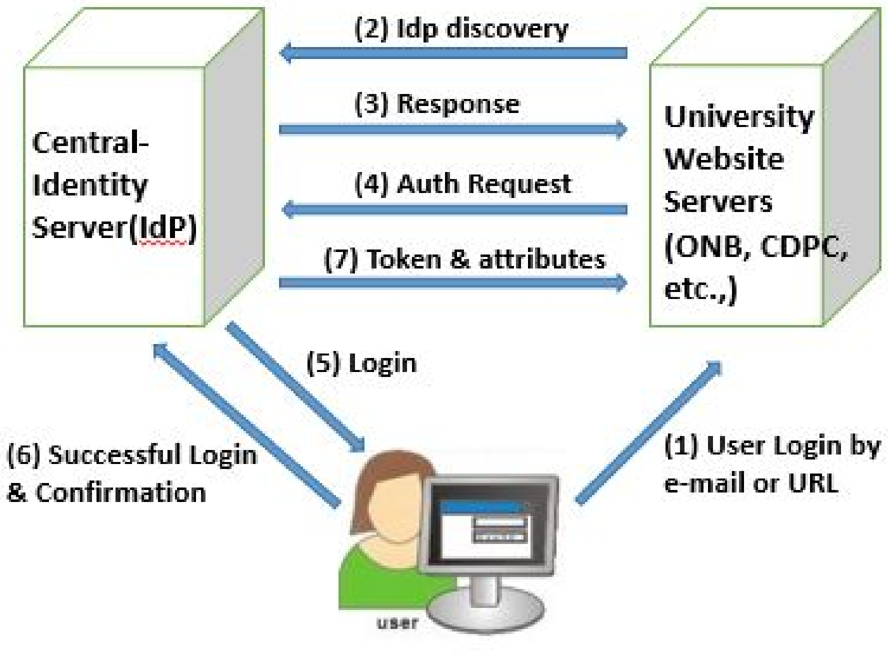

(2) Idp discovery
(3) Response
(4) Auth Request
(7) Token & attributes
Central-Identity Server(IdP)
University Website Servers (ONB, CDPC, etc.,)
(5) Login
(6) Successful Login & Confirmation
(1) User Login by e-mail or URL
user

Figure 4.5: Intra Campus

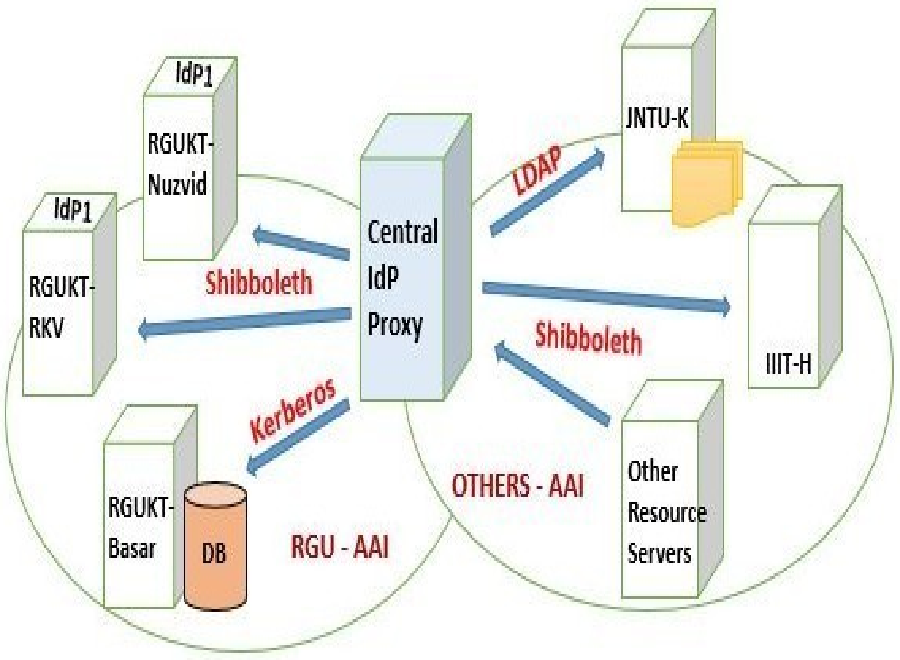


Figure 4.6: Inter Campus

4.3 Dynamic Role Based Access Control

4.3.1 Introduction

Role Based Access Control (RBAC), also known as Non discretionary Access Control, takes more of a real world approach to structuring access control. Access under RBAC is based on a user's job function within the organization to which the computer system belongs. Essentially, RBAC assigns permissions to particular roles in an organization. Users are then assigned to that particular role.[4]

With the rapid development of network and the coming of information age, access control is particularly important. RBAC is an access control which is popular. RBAC authorizes and controls the roles corresponding to the users to operate the object. It solves problems of least privilege, separation of duties and so on. RBAC has better security, better flexibility, and can be well applied to the workflow system.

In a role-based access control (RBAC) mechanism, these rights are defined based on the role that individuals are assigned to in an organization. It overcomes the problems in DAC (Discretionary Access Control) which is flexible but not secure and MAC (Mandatory Access Control) which is Secure but not flexible. We use roles because,

- Fewer relationships to manage so reduces complexity.
- Roles add a useful level of abstraction as organizations operate based on roles.
- A role is more stable than the collection of users and the collection of permissions.

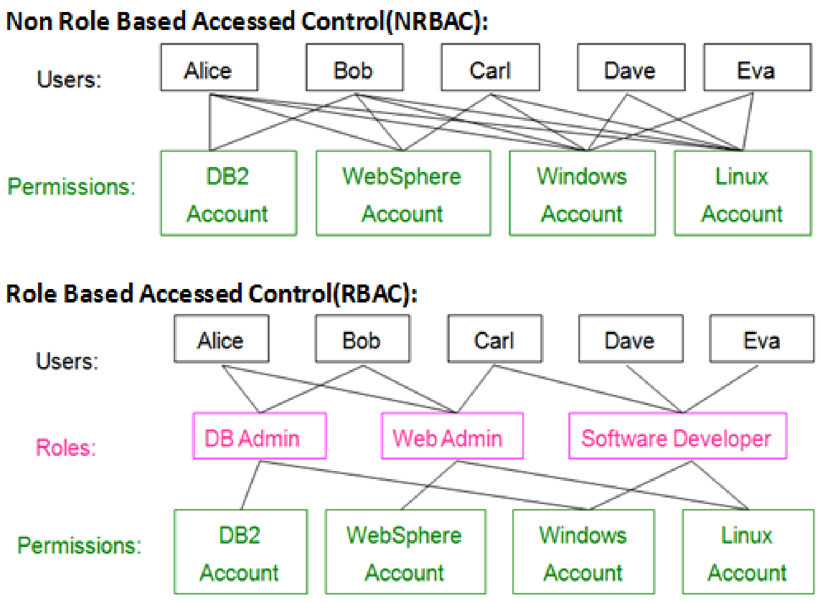


Figure 4.7: Difference between NRBAC and RBAC.

4.3.2 Basic Idea of RBAC

Access Control policy is embodied in various components of RBAC such as [4]

- Role-Permission relationships
- User-Role relationships
- Role-Role relationships

These components collectively determine whether a particular user will be allowed to access a particular piece of data in the system. RBAC components may be configured directly by the system owner or indirectly by appropriate roles as delegated by the system owner. The policy enforced in a particular system is the net result of the precise configuration of various RBAC components as directed by the system owner. The ability to modify policy to meet the changing needs of an organization is an important benefit of RBAC.

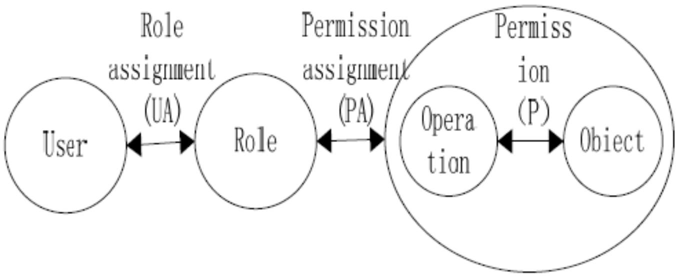


Figure 4.8: The Core Idea of RBAC.

RBAC model is defined in terms of three model components-Core RBAC, Hierarchical RBAC and Constraint RBAC. The core RBAC model includes five basic elements, i.e. U (users), R (roles), O (objects), OP (Operations) and P (permissions). The core idea of RBAC is that adds roles between users and permissions. It forms relationships among users, roles and permissions. Users get roles corresponding permissions by getting roles to operate on the objects.

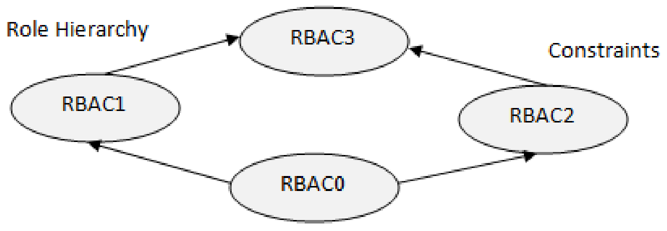


Figure 4.9: The Family of RBAC Model.[5]

4.3.3 Structure Diagram of RBAC Model

The Structure diagram of role based access control model consists of hierarchies and constraints. Role hierarchical relationship expresses the inheritance in roles permissions, such as the role A inherits role B, and then Bs permissions also belong to A. A role may inherit from multiple roles or be inherited by multiple roles. The use of role hierarchical relationship not only makes the system closer to reality, but also makes it easier to add and remove roles, easier to manage. The different types of inheritance can be,

- User inheritance
  - means every user that is a member of r1 is also a member of r2
- Permission inheritance
  - means every permission that is authorized for r2 is also authorized r1
- Activation inheritance
  - means that activating r1 will also activate r2

Constraint RBAC model in RBAC adds separation of duty relations. Separation of duty can be static or dynamic. There is no formal model specified for constraints. Administrator can improve his own constraints based on requirement. Some example for constraints can be,

- Mutual Exclusion
  - No user should have more than one role in a session
- Pre-condition
  - Must satisfy some condition to be member of some role
- Cardinality
  - How many permission are assigning for a role

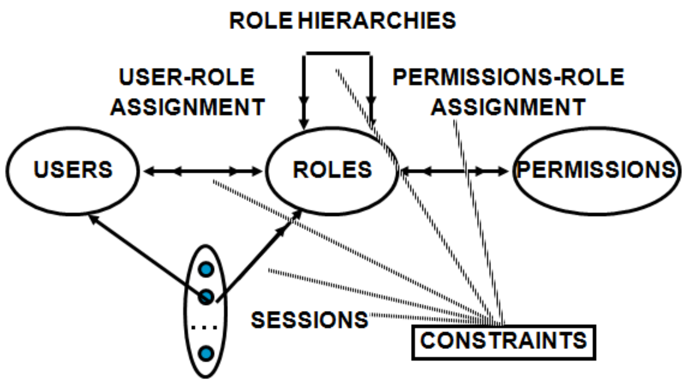


Figure 4.10: RBAC3 Model.

4.3.4 Dynamic Role Based Access Control Model

Although RBAC has many advantages, it also has some disadvantages. it ignores the need to make dynamic constraints executed on sequential order. This problem can be solved by dynamic constraint module, so the permissions required by the sequential order can be executed by order, ensuring when the workflow occurs can operate his dynamic permission, preventing abuse of the users own permissions, making the system more security.

Dynamic RBAC overcomes the shortages of the traditional RBAC by adding with dynamic constraints and dynamic permissions, so that it makes the system easier to manage and more close to the real world. The App creator no need to go for administration, Himself, he can create roles of his own requirement. If a role is already exist, he can just add. Otherwise he can request for new roles. We can deal application depends on time.

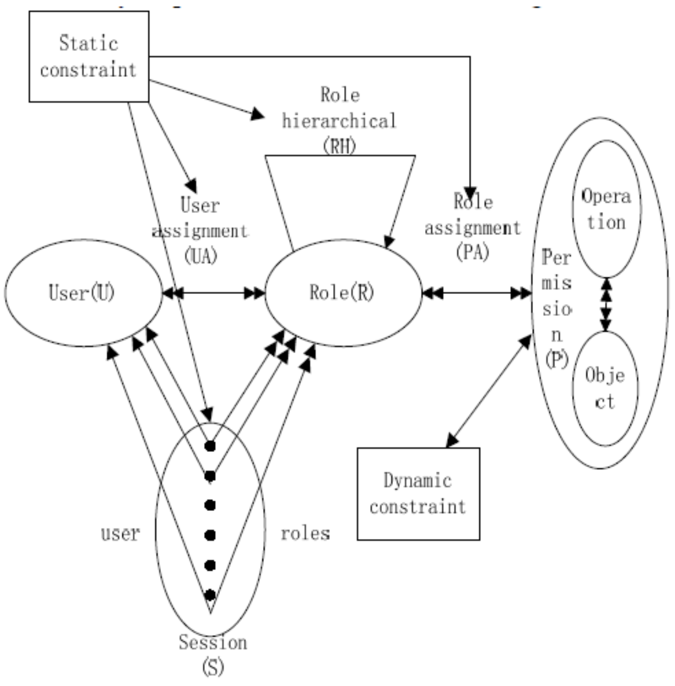


Figure 4.11: RBAC with Dynamic constraints.

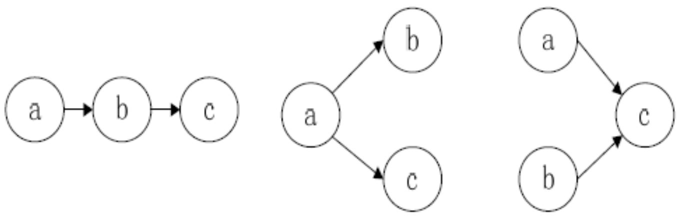


Figure 4.12: Different Types of Association.

4.4 Conclusion

Federated identity management is still state of the art for Web-based Single Sign-On in scientific communities. Integration of multiple federations is a difficult task especially if usability and privacy is considered and it is very important in distributed environment.

Single sign on helps us to assign only single key for each user for all available applications hence no redundant information about the user identification information.

Dynamic role based access control improves security and reduces complexity by users with assigned roles and permissions to access resources of the organization.

REST API used to connect third party developers with the orgnaizational identitiy services to their application developers and users.

Federated central Identity with single sign on through dynamic role based access control along with the REST API for third party application developers Implementation is our main aspect.

Chapter 5

Network Components

5.1 Network Components

5.1.1 Introduction

Network devices are components used to connect computers or other electronic devices together so that they can share files or resources like printers or fax machines. Devices used to setup a Local Area Network (LAN) are the most common types of network devices used by the public. A LAN requires a hub, router, cabling or radio technology, network cards, and if online access is desired, a high-speed modem.

To construct a Campus Area Network(CAN), building network existed in that network is an important and complex task. So in order to make that complex task easier and very robust in nature we have to use some of these components such as LDAP, AAA and NFS. In constructing a network with complex architecture and more people present is very difficult, unless efficient configuration of the network components present in that network will be there. In this section we will see why these components are so important and very efficient in designing such a complex task.

5.1.2 Basic Networking Components

5.1.2.1 Router

A router is a device that forwards data packets along networks. A router is connected to at least two networks, commonly two LANs or WANs or a LAN and its ISP's network. Routers are located at gateways, the places where two or more networks connect. Router is a networking device, commonly specialized hardware, that forwards data packets between computer networks. This creates an overlay internetwork, as a router is connected to two or more data lines from different networks.

5.1.2.2 Switch

A network switch is a computer networking device that connects devices together on a computer network, by using a form of packet switching to forward data to the destination device. A network switch is considered more advanced than a (repeater) hub because a switch will only forward a message to one or multiple devices that need to receive it, rather than broadcasting the same message out of each of its ports.

5.2 AAA Server

AAA is the acronym for **A**uthentication,**A**uthorization, and **A**ccounting. Authentication controls access by requiring valid user credentials, which are typically a username and password. Authentication asks the user who they are. Authorization controls access per user after users authenticate, asks the user what privileges they have, and controls the services and commands available to each authenticated user. Accounting tracks traffic that passes through the security appliance, enabling you to have a record of user activity. If you enable authentication for that traffic, you can account for traffic per user.

Without authentication of traffic, there's no account for traffic per IP address. Accounting information includes when sessions start and stop, username, the number of bytes that pass through the security appliance for the session, the service used, and the duration of each session.

Types of AAA Servers: RADIUS Server, TACACS+ Server, Kerberos Server, LDAP Server Support and Local Database Support. Depending on the size of the network and available resources, AAA can be implemented on a device locally or can be managed from a central server running RADIUS or TACACS+ (Terminal Access Controller Access Control System Plus) protocols. The most functionally server type is the TACACS+ Server and it is chosen here for that purpose.

The AAA server first checks to see if the user has been authenticated. If a valid authentication entry exists for the user, the session is allowed and no further intervention is required by the authentication proxy. If no entry exists, the authentication proxy responds to the connection request by prompting the user for a username and password. If the authentication fails, the AAA server reports the failure to the user and prompts the user for a configurable number of retries.

TACAC+ is generally considered superior because of the following reasons:

- TACACS+ encrypts the entire TACACS+ packet, while RADIUS only encrypts the shared secret password portion.
- TACACS+ separates authentication and authorization, making possible distributed security services.

	TACACS +	**RADIUS**
Functionality	Separates AAA	Combines Authentication, Authorization
Transport protocol	TCP	UDP
CHAP	Bidirectional	Unidirectional
Protocol Support	Multi- Protocol	No ARA No NetBEUI support
Confidentiality	Entire Packet	Password- Encrypted Encrypted
Accounting	Limited	Extensive

Table 5.1: Comparision of Features Between TACACS+ AND RADIUS

5.3 LDAP Server

Lightweight **D**irectory **A**ccess **P**rotocol is a protocol for accessing a directory. A directory contains objects related to users, groups, computers, printers and so on etc. Lightweight Access Protocol (LDAP) is a directory service access protocol. It's having layered structure, access control mechanism, customized data storage format etc. Company structure information (although frankly you can extend it and store anything in there).

Directory is an important part of Internet technology to support such needs. Directory exists in a multitude of applications ranging from operating systems, asset management systems, security systems, etc. All entries in the directory are organized by the hierarchical model, which is tree structure called Directory Information Tree(DIT). Entry held in DIT is named directory service entry witch is identified by unique Distinguished Name(DN).

LDAP gives you query methods to add, update and remove objects within a directory (and a bunch more, but those are the central ones). Distinguished Name(DN) having three major operations

- Add
- Modify
- Delete

LDAP v3 specifies three authentication types

- No Authentication
- Basic Authentication
- Simple Authentication and Security Layer (SASL)

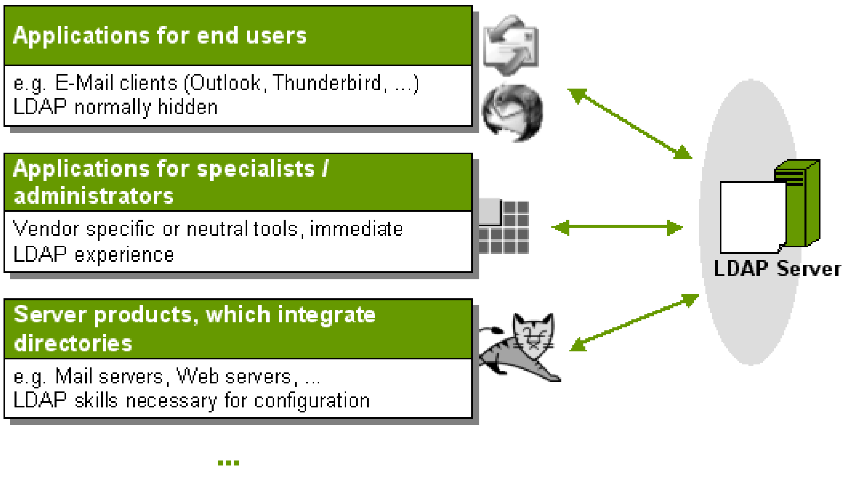

Applications for end users
e.g. E-Mail clients (Outlook, Thunderbird, ...)
LDAP normally hidden
Applications for specialists / administrators
Vendor specific or neutral tools, immediate LDAP experience
Server products, which integrate directories
e.g. Mail servers, Web servers, ...
LDAP skills necessary for configuration
...
LDAP Server

Figure 5.1: LDAP Hierarchy

5.4 NFS Server

NFS stands for **N**etwork **F**ile **S**ystem, a file system developed by Sun Microsystems, Inc. It is a client/server system that allows users to access files across a network and treat them as if they resided in a local file directory. For example, if you were using a computer linked to a second computer via NFS, you could access files on the second computer as if they resided in a directory on the first computer. This is accomplished through the processes of exporting (the process by which an NFS server provides remote clients with access to its files) and mounting (the process by which file systems are made available to the operating system and the user).

The NFS protocol is designed to be independent of the computer, operating system, network architecture, and transport protocol. This means that systems using the NFS service may be manufactured by different vendors, use different operating systems, and be connected to networks with different architectures. These differences are transparent to the NFS application, and thus, the user.
Using NFS, the user or a system administrator can mount all or a portion of a file system (which is a portion of the hierarchical tree in any file directory and subdirectory, including the one you find on your PC or Mac). The portion of your file system that is mounted (designated as accessible) can be accessed with whatever privileges go with your access to each file (read-only or read-write). The operation of NFS is defined in the form of three main components as it is having a crucial role in providing its functionalities.

- Remote Procedure Call
- External Data Representation
- NFS Procedures and Operations

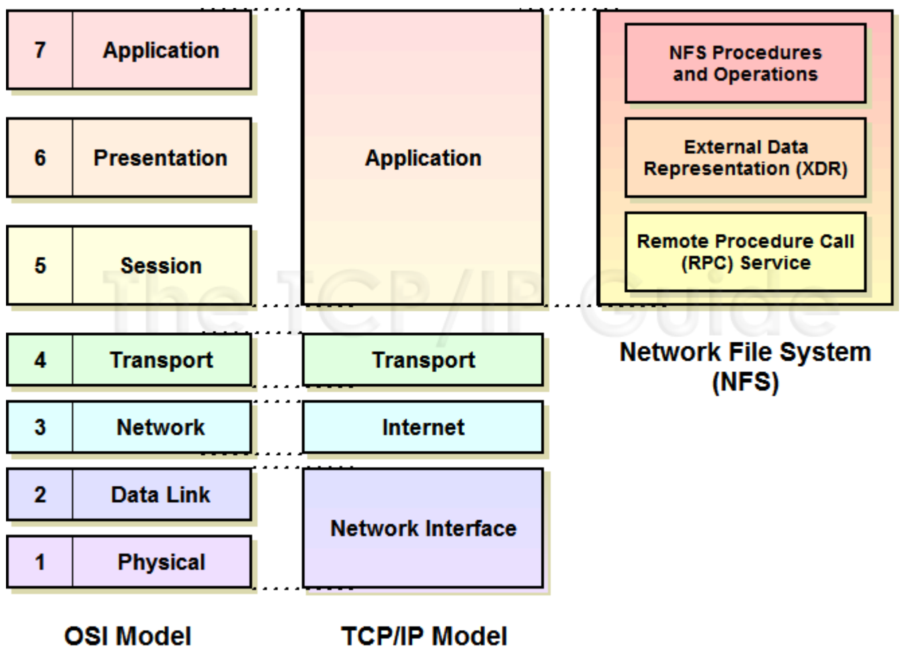


Figure 5.2: Components in NFS Server

Chapter 6

Cloud Infrastructure

6.1 Cloud Computing

6.1.1 Introduction

Cloud computing refers to both the applications delivered as services over the Internet and the hardware and system software in the data centers that provide those services.

Definition – *"Cloud computing is a model for enabling convenient, on- demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction"* [2]

This definition includes cloud architectures, security, and deployment strategies.

6.1.2 Cloud Characterstics

Charactersitcs of Cloud Computing can be broudly defined as

On-demand self-service: A consumer with an instantaneous need at a particular timeslot can avail computing resources (such as CPU time, network storage, software use, and so forth) in an automatic (i.e. convenient, self-serve) fashion without resorting to human interactions with providers of these resources.

Broad network access: These computing resources are delivered over the network (e.g. Internet) and used by various client applications with heterogeneous platforms (such as mobile phones, laptops, and PDAs) situated at a consumer's site.

Resource pooling: When we combine available resource and we can experiance accessing one super computer kind of thing is the goal of this resource pooling

Rapid elasticity: For consumers, computing resources become immediate rather than persistent: there are no up-front commitment and contract as they can use them to scale up whenever they want, and release them once they finish to scale down. Moreover, resources provisioning appears to be infinite to them, the consumption can rapidly rise in order to meet peak requirement at any time.

Measured Service: Although computing resources are pooled and shared by multiple consumers (i.e. multi-tenancy), the cloud infrastructure is able to use appropriate mechanisms to measure the usage of these resources for each individual consumer through its metering capabilities.

Virtualization: Virtualization is key to use the one resource or group of resource to use up to multiple instances this can be done by directly or partially

6.1.3 Service Models

If we providing any thing as a service comes, that will comes into Cloud Computing. Various Service Delivery Models listed bellow.[3]

- Software as a Service (SaaS) – is where in many users can make use of the software hosted by the service provider and pay only for time its being used.
  - Salesforce, Zoho Office and Google Apps
- Platform as a Service (PaaS) – provides a high-level integrated environment to design, build, test, deploy and update online custom applications.
  - Googles App Engine , Microsoft Azure and SalesForce
- Infrastructure as a Service (IaaS) – refers to the services provided to the users to use processing power, storage, network and other computing resources, to run any software including operating systems and applications.
  - AWS, Eucalyptus, Open Stack, GoGrid and Flexiscale
- Data storage as a Service (DaaS) – provides data store as a service for the storage requirements of the organizations and users on the basis of pay per use model in public cloud.
  - Amazon S3, DropBox, SkyDrive, etc
- Anything as a Service (XaaS) – is broder way of say the services in cloud computing which includes anything that is offered as a service.

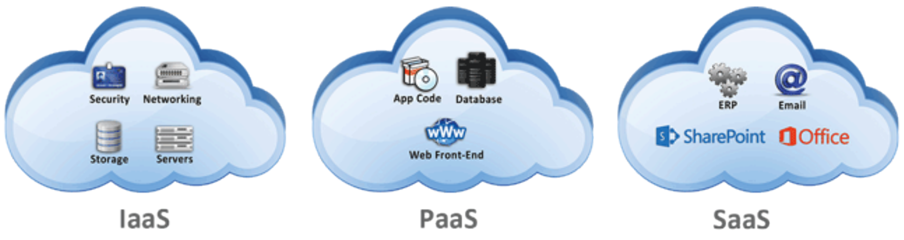


Figure 6.1: Cloud Computing - Servcice Models

6.1.4 Deployment Models

More recently, four cloud deployment models have been defined in the Cloud community

- *Public cloud* – This is the dominant form of current Cloud computing deployment model. The public cloud is used by the general public cloud consumers and the cloud service provider has the full ownership of the public cloud with its own policy, value, and profit, costing, and charging model.
  - Amazon EC2, S3, Google AppEngine, and Force.com
- *Private cloud* – The cloud infrastructure is operated solely within a single organization, and managed by the organization or a third party regardless whether it is located premise or off premise.
  - Openstack, Cloudstack, OpenNebula, Nimbus etc.
- *Hybrid cloud* – The cloud infrastructure is a combination of two or more clouds (private, community, or public) that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability.
  - Openstack APIs, AWS APIs
- *Community cloud* – Several organizations jointly construct and share the same cloud infrastructure as well as policies, requirements, values, and concerns. The cloud community forms into a degree of economic scalability and democratic equilibrium.

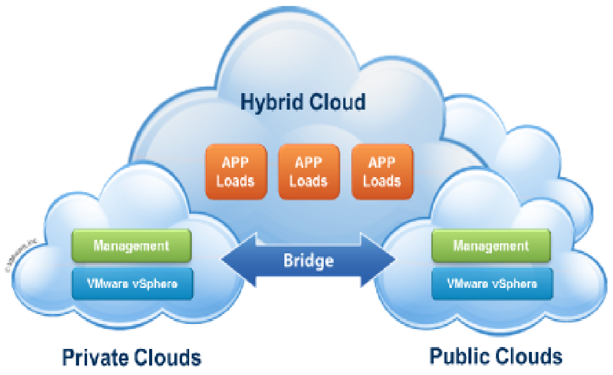


Figure 6.2: Cloud Computing - Deployment Models

6.2 Private Clouds

6.2.1 Introduction

Definition – "It is one of the cloud deployment model where the resources of small or medium organization are united and cattered to users of the that organization or outsourced through internet"

The motivation to setup a private cloud within an organization has several aspects.

- To maximize and optimize the utilization of existing in-house resources.
- Security concerns including data privacy and trust also make Private Cloud an option for many firms.
- Organizations always require full control over mission-critical activities that reside behind their firewalls.
- Academics often build private cloud for research and teaching purposes.

6.2.2 Open Source Tools

We can construct private cloud using some open source tools like Openstack, Cloudstack, OpenNebula.

We can use this private cloud to deploy various services like Departmental Websites, Notice Boards, Events portal, High Computational Virtual Machines for Virtual Labs, High Performance Computing, Big data analytics.

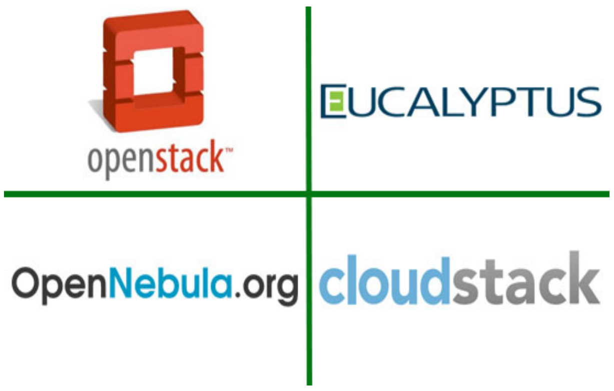


Figure 6.3: Private Cloud - Open source tools

6.3 Conclusion

Hardware can be effinciently utilized by tronsforming them into cloud based infra. To maintain the institutional security we are going for privte clouds. We want to go with open source tools to maintain economically optimized.

Chapter 7

Implentation and Specifications

7.1 Implentation

For maintining above kind of envronment we have to setup one private cloud which will gives the highly scalable virtual machines to server the user need services on fly.

We want to develop minal system with the above guidelines in a lab environment of 10 Master nodes and 5 client machines with uniform operating system and nfs data share. Later extended upto departmental level.

7.2 Specifications

- 15 Laptops with below configuration (10 Master Nodes + 5 Slave Nodes)
  - 4GB RAM, 500 GB Harddisk, Intel i3 processor 1.5GHz Clock
- Class C Network
  - Static IP for Master Nodes
  - DHCP / Static IP for Slave Nodes
- Uninterrupted power suply for Master Nodes.

7.3 Technologies and Tools

- Opensource private cluod tools such as Openstack, Cloudstack, etc.
- Ubuntu 14.04 Server LTS.
- LDAP Active Directories, Kerbrose, Squid for Network proxy.
- OAuth 2.0 with extended dynamic roles managment.
- Django for Implementation of OAuth 2.0 as REST API.
- Git for Version control.
- Python shpenix & Latex for Documentation.

Chapter 8

Web based Single Sign-On

8.1 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.[3]

8.1.1 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. [5]

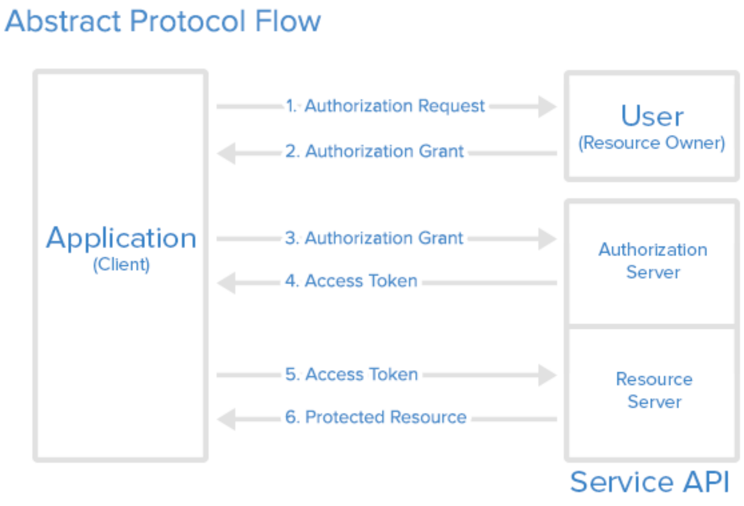


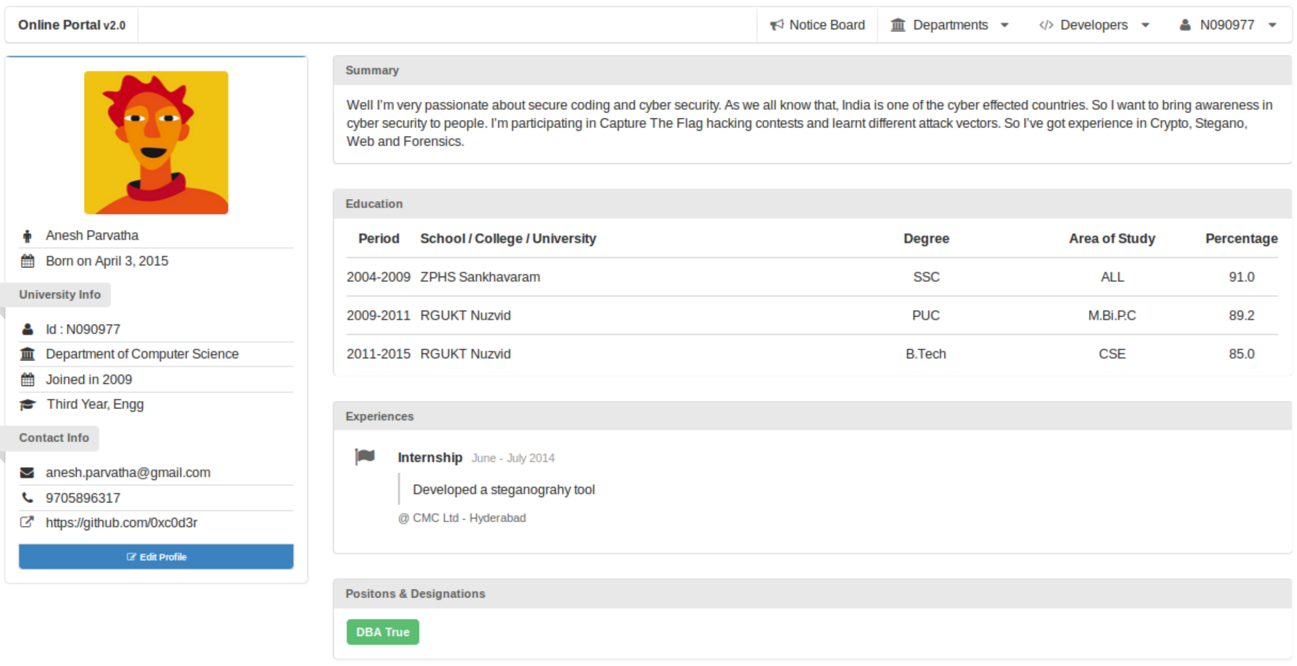
Figure 8.1: OAuth Protocol Work Flow Diagram

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

8.2 University User Profiles



Anesh Parvatha
Born on April 3, 2015

University Info

Id : N090977
Department of Computer Science
Joined in 2009
Third Year, Engg

Contact Info

anesh.parvatha@gmail.com
9705896317
https://github.com/0xc0d3r

Summary

Well I'm very passionate about secure coding and cyber security. As we all know that, India is one of the cyber effected countries. So I want to bring awareness in cyber security to people. I'm participating in Capture The Flag hacking contests and learnt different attack vectors. So I've got experience in Crypto, Stegano, Web and Forensics.

Education

Period	School / College / University	Degree	Area of Study	Percentage
2004-2009	ZPHS Sankhavaram	SSC	ALL	91.0
2009-2011	RGUKT Nuzvid	PUC	M.Bi.P.C	89.2
2011-2015	RGUKT Nuzvid	B.Tech	CSE	85.0

Experiences

Internship June - July 2014

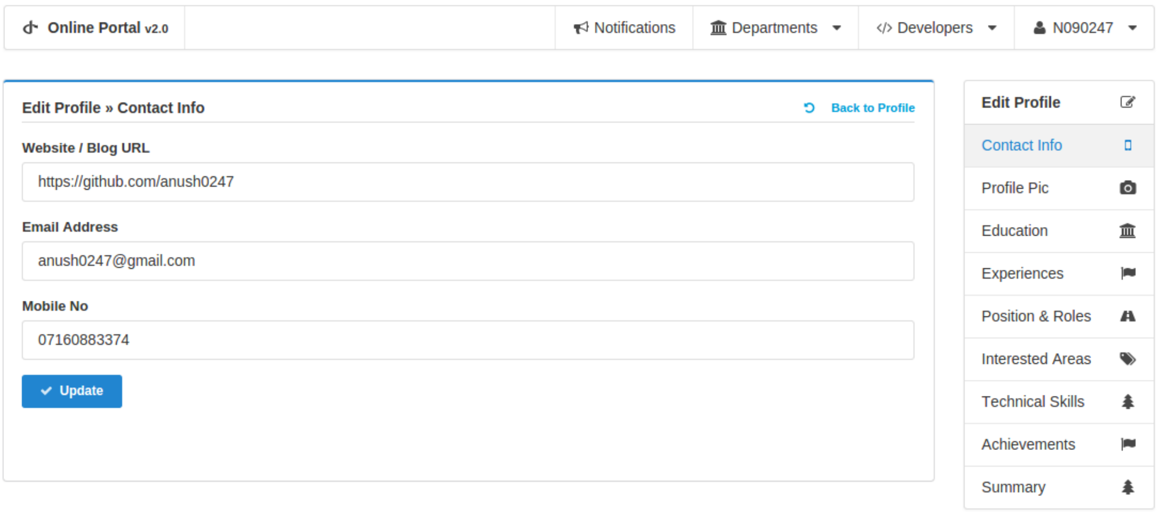
Developed a steganograhy tool

@ CMC Ltd - Hyderabad

Positons & Designations

DBA True

Figure 8.2: User Profile



Edit Profile » Contact Info

Website / Blog URL: https://github.com/anush0247

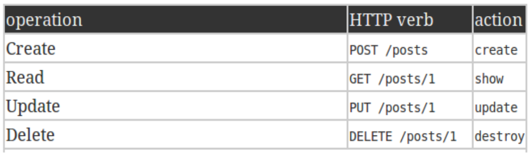
Email Address: anush0247@gmail.com

Mobile No: 07160883374

Figure 8.3: Edit User Profile

8.3 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.^{4}



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

Figure 8.4: CRUD Operations

8.3.1 Technologies Used

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

8.3.2 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
    }
]
```

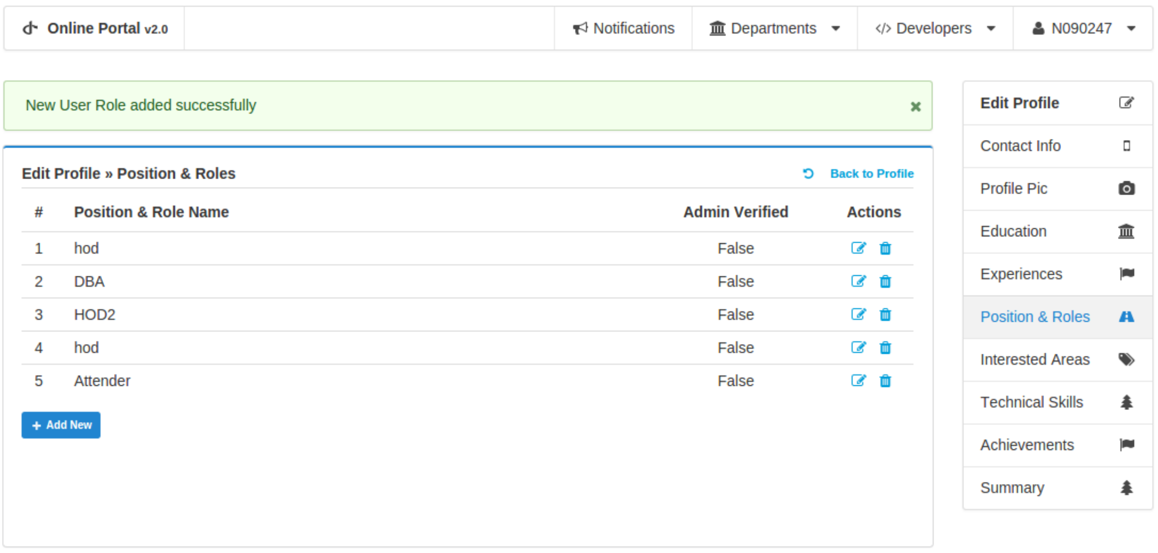
/api/skills/?access_token=<token>

```
[
    {
        "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
    }
]
```

8.4 Roles & Permissions



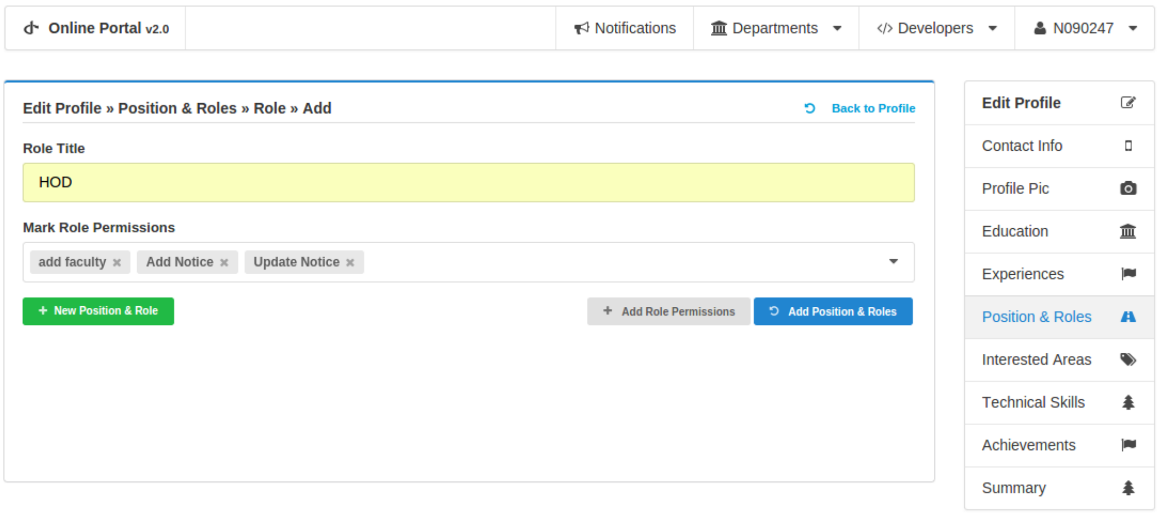
New User Role added successfully

Edit Profile » Position & Roles

Back to Profile

#	Position & Role Name	Admin Verified	Actions
1	hod	False	
2	DBA	False	
3	HOD2	False	
4	hod	False	
5	Attender	False	

Figure 8.5: List of Roles of User



Edit Profile » Position & Roles » Role » Add

Back to Profile

Role Title

HOD

Mark Role Permissions

add faculty × Add Notice × Update Notice ×

Figure 8.6: Adding Permissions Option

8.5 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.

8.5.1 Initializing the Client Library

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

8.5.2 Get Authorization URL

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

8.5.3 Get Access Token

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

8.5.4 Initializing API with Access Token

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

8.5.5 Getting User Info from API

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

Chapter 9

Network Single Sign-On

9.1 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

9.2 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.

9.2.1 Installation and Configuration [9]

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**

9.3 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.

9.3.1 Installation and Configuration

- sudo apt-get install **phpldapadmin**

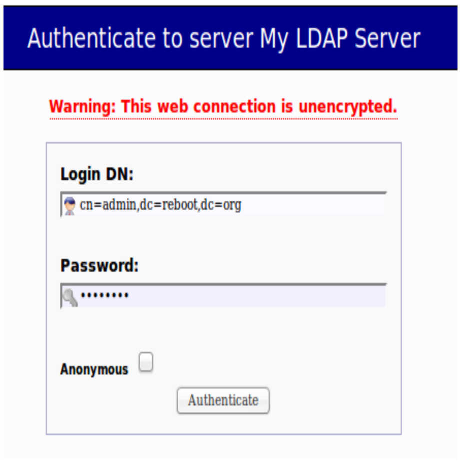
After the installation is complete configuration will be done by making following changes in the config.php file of phpLDAPadmin.

- 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 9.1: PHP Config file

9.3.2 Web Interface of phpLDAPadmin:


Authenticate to server My LDAP Server

Warning: This web connection is unencrypted.

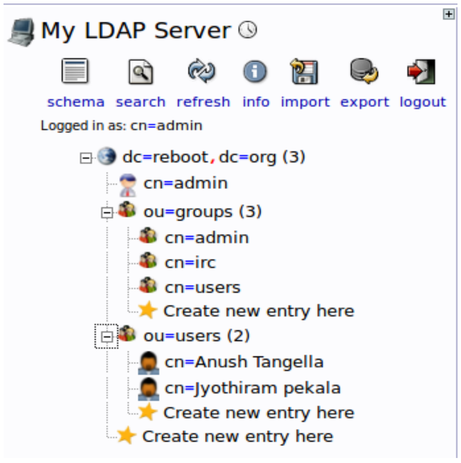
Login DN:
cn=admin,dc=reboot,dc=org

Password:
••••••••

Anonymous

Authenticate

Figure 9.1: phpLDAP

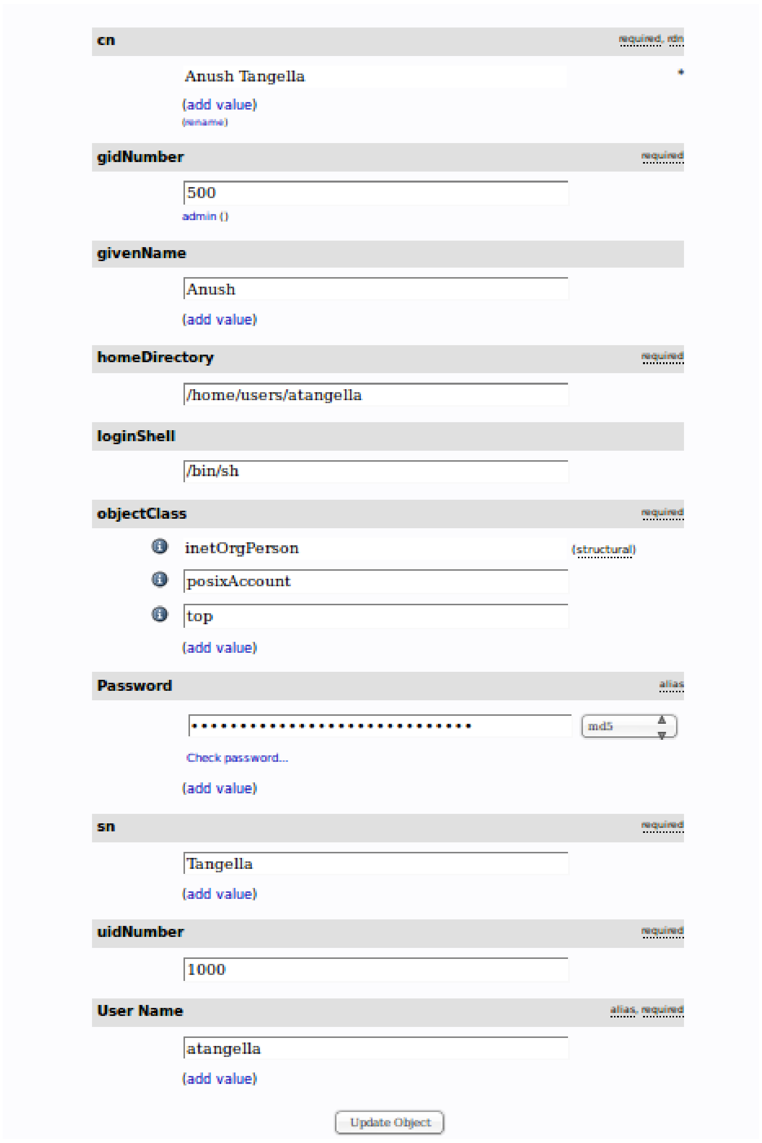

My LDAP Server

schema search refresh info import export logout

Logged in as: cn=admin

dc=reboot, dc=org (3)
cn=admin
ou=groups (3)
cn=admin
cn=irc
cn=users
Create new entry here
ou=users (2)
cn=Anush Tangella
cn=Jyothiram pekala
Create new entry here
Create new entry here

Figure 9.2: Complete category of LdapServer



cn required, rdn

Anush Tangella *

(add value)
(rename)

gidNumber required

500
admin ()

givenName

Anush
(add value)

homeDirectory required

/home/users/atangella

loginShell

/bin/sh

objectClass required

inetOrgPerson (structural)
posixAccount
top
(add value)

Password alias

•••••••••••••••••••••••••••• md5
Check password...
(add value)

sn required

Tangella
(add value)

uidNumber required

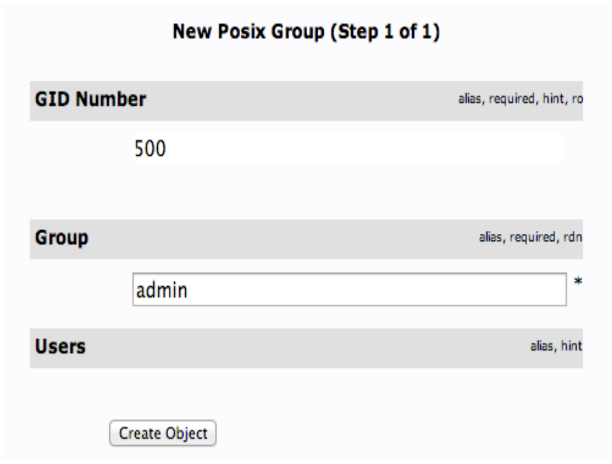
1000

User Name alias, required

atangella
(add value)

Update Object

Figure 9.3: User Creation



New Posix Group (Step 1 of 1)

GID Number alias, required, hint, ro

500

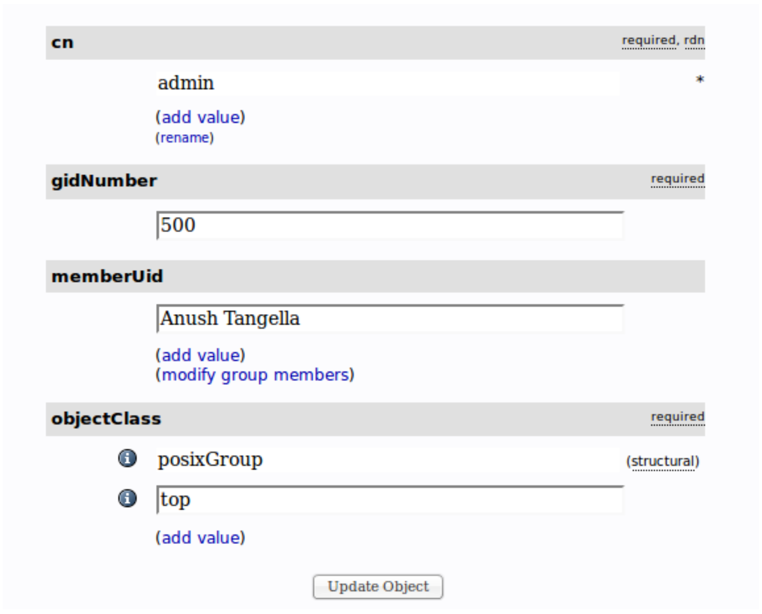
Group alias, required, rdn

admin *

Users alias, hint

Create Object

Figure 9.4: Group Creation



cn — required, rdn

admin *

(add value)
(rename)

gidNumber — required

500

memberUid

Anush Tangella

(add value)
(modify group members)

objectClass — required

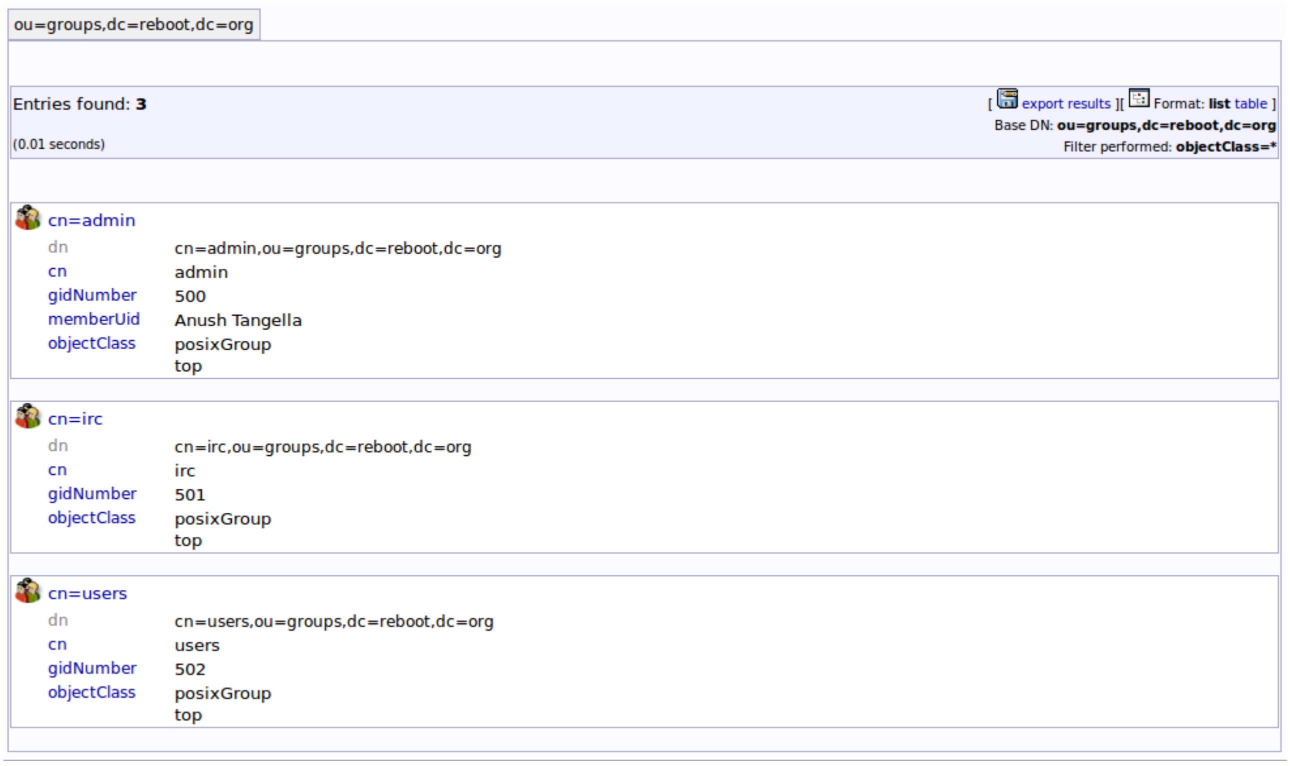
posixGroup (structural)

top

(add value)

Update Object

Figure 9.5: Adding User to Group



ou=groups,dc=reboot,dc=org

Entries found: **3**

(0.01 seconds)

[export results][Format: **list** table]
Base DN: **ou=groups,dc=reboot,dc=org**
Filter performed: **objectClass=***

cn=admin

dn	cn=admin,ou=groups,dc=reboot,dc=org
cn	admin
gidNumber	500
memberUid	Anush Tangella
objectClass	posixGroup
	top

cn=irc

dn	cn=irc,ou=groups,dc=reboot,dc=org
cn	irc
gidNumber	501
objectClass	posixGroup
	top

cn=users

dn	cn=users,ou=groups,dc=reboot,dc=org
cn	users
gidNumber	502
objectClass	posixGroup
	top

Figure 9.6: Groups information

9.4 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.

9.4.1 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 9.2: 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**

9.4.2 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**

9.5 HAProxy

HAProxy(High Availability Proxy) is an open source load balancer which can load balance any TCP service. It is particularly suited for HTTP load balancing as it supports session persistence and layer 7 processing. HAProxy can be configured as a front-end to load balance two VPS through private network connectivity.

9.5.1 Installing HAProxy [8]

```
# to install haproxy
sudo apt-get install haproxy
# to get started by init script
edit /etc/default/haproxy, set ENABLED option to 1
# using haproxy
sudo /etc/init.d/haproxy start—stop—reload—restart—status
```

9.5.2 Configuring HAProxy

edit gedit /etc/haproxy/haproxy.cfg

```
frontend sunny
bind 10.4.34.250:8080
default_backend sunny-backend
backend sunny-backend
balance roundrobin
mode tcp
server sunny 10.4.34.250:80 check
server ram 10.4.34.242:80 check
server knc 10.4.34.245:80 check
```

Requests come to frontend PC will be send to any one of the backend PC's based on the algorithm. Here algorithm can be roundrobin, leastconn.

9.6 GlusterFS

GlusterFS is a unified, poly-protocol, scale-out file system serving many peta bytes of data. While many databases and other software allows you to spread data out in the context of a single application, other systems can operate on the file system level to ensure that data is copied to another location whenever it is written to disk. A clustered storage solution like GlusterFS provides this exact functionality.

We are writing by considering three systems. we are considering the two of our machines as cluster members and the third as a client. We will be configuring the computers we labeled as gluster0 and gluster1 as the cluster components. We will use gluster2 as the client. [8]

9.6.1 Configure DNS solution

Go to **sudo nano /etc/hosts** and add below lines
#first_ip gluster0.droplet.com gluster0
#second_ip gluster1.droplet.com gluster1
#third_ip gluster2.droplet.com gluster2

9.6.2 Install server components

# On our cluster member machines (gluster0 and gluster1), we can install the GlusterFS server package
sudo apt-get install glusterfs-server
#On one of the hosts, we need to peer with the second host.
sudo gluster peer probe gluster1.droplet.com

9.6.3 Create a storage volume

#Creating and enabling replication property for volume1
sudo gluster volume create volume1 replica 2 transport tcp gluster0.droplet.com:/gluster-storage gluster1.droplet.com:/gluster-storage force
#And we can activate the storage by command
sudo gluster volume start volume1

9.6.4 Install and configure client components

#On our client machine, we can install the GlusterFS client package
sudo apt-get install glusterfs-client
#for mounting our remote storage volume on our client computer
sudo mkdir /storage-pool
sudo mount -t glusterfs gluster0.droplet.com:/volume1 /storage-pool

9.6.5 Restrict access to the volume

```
#for restriction of storage volume for clients
sudo gluster volume set volume1 auth.allow *
#for removing restrictions
sudo gluster volume set volume1 auth.allow gluster_client1_ip,gluster_client2_ip
```

At this point, you should have a redundant storage system that will allow us to write to two separate servers simultaneously. This can be useful for a great number of applications and can ensure that our data is available even when one server goes down. But GlusterFS is failing in distributed environment at some situations. For that we moved an efficient one XtreemFS, which works very well in distributed environment and tackles all errors.

9.7 XtreemFS

XtreemFS is a fault-tolerant distributed file system for all storage needs. It is simple to setup as it does not use any kernel modules. It's easy to integrate with clients for Linux and Windows. XtreemFS is also a parallel, object-based file system. You can stripe your files across many storage servers for high-performance parallel access. The stripe width can be configured per file.

It can stripe files within your cluster and can replicate your data across clusters. This allows you to have high-performance access within your cluster and to share your data in your virtual organization.

XtreemFS' replication is fault-tolerant. A broken hard drive or unhealthy storage server does not result in data loss or even degraded service quality.

This matters if you have 10 or 1000s of machines, as your jobs finish without interruptions and you can delay repairs to whenever you have time.

9.7.1 Installation, Creating & Mounting Filesystem

- Added the XtreemFS repo to our system and then installed the packages xtreemfs-server and xtreemfs-client.
- Starting the Directory Service: $sudo /etc/init.d/xtreemfs-dir start
- Starting the Metadata Server: $sudo /etc/init.d/xtreemfs-mrc start
- Starting OSD: $sudo /etc/init.d/xtreemfs-osd start
- Loading the FUSE kernel module: $sudo modprobe fuse
- We can check the registry by opening the DIR status page in our web browser at http://localhost:30638
- Creating a new volume with default settings : $sudo mkfs.xtreemfs localhots/myVolume
- Creating mounting point : $sudo mkdir /xtreemfs
- Mounting XtreemFS: $sudo mount.xtreemfs localhost/myVolume /xtreemfs
- We can unmount using : $sudo umount.xtreemfs /xtreemfs

9.7.2 Distributed Step

We assume a setup with two machines:

- One system running the directory service (DIR), metadata server (MRC), storage server (OSD).
- Another system running the only storage server (OSD).

First, installed the binary packages using repositories made available by the XtreemFS team. Once the repository file is registered, we can install these packages:

```
sudo apt-get install xtreemfs-client xtreemfs-server xtreemfs-tools
```

Now suppose one server to host the DIR and MRC service is called osd1, and my OSDs will be running on osd1 and osd2. On osd2, we have to edit the following config files under /etc/xos/xtreemfs and **set dir_service.host = kdc01**. For the mrc and osd config files, we have to set up **hostname = kdc01**

- mrcconfig.properties
- osdconfig.properties

On WKS01 where a second OSD will be running, edited the osd config file, point **dir_service.host = kdc01** and **hostname = wks01**

- osdconfig.properties

Then we have to start the services. On KDC01, the following services are started:

```
sudo /etc/init.d/xtreemfs-dir start
sudo /etc/init.d/xtreemfs-mrc start
sudo /etc/init.d/xtreemfs-osd start
```

On WKS01, OSD service

```
sudo /etc/init.d/xtreemfs-osd start
```

Then we created the XtreemFS filesystem using following commands:

```
sudo mkfs.xtreemfs kdc01/myVol # on KDC01
sudo mount.xtreemfs kdc01/myVol /data # on KDC01 and WKS01
```

By default, 1 replica will be stored on the entire XtreemFS volume. XtreemFS client is smart enough to contact the best OSD for the file resource. In case its desirable to maintain more than 1 replica of the same file, it can be done with the xtfsutil tool. For example, to configure XtreemFS to replicate a file to all available OSDs:

Chapter 10

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.

10.1 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

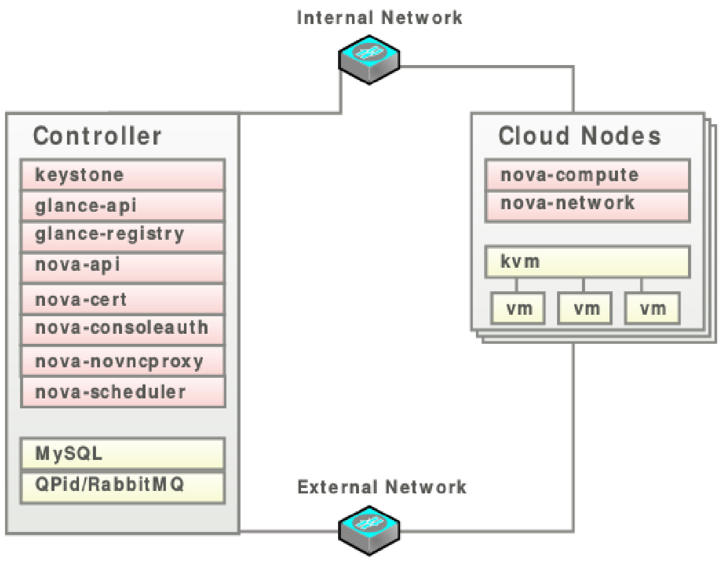
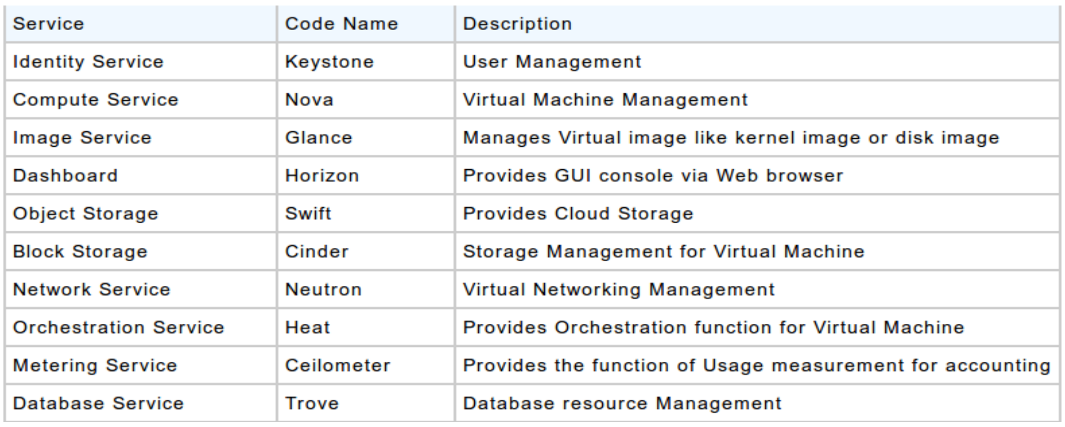


Figure 10.1: Openstack Architecture.

10.2 Installation

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



Service	Code Name	Description
Identity Service	Keystone	User Management
Compute Service	Nova	Virtual Machine Management
Image Service	Glance	Manages Virtual image like kernel image or disk image
Dashboard	Horizon	Provides GUI console via Web browser
Object Storage	Swift	Provides Cloud Storage
Block Storage	Cinder	Storage Management for Virtual Machine
Network Service	Neutron	Virtual Networking Management
Orchestration Service	Heat	Provides Orchestration function for Virtual Machine
Metering Service	Ceilometer	Provides the function of Usage measurement for accounting
Database Service	Trove	Database resource Management

Figure 10.2: Openstack Service Components. [12]

10.2.1 NTP

# apt-get install ntp

10.2.2 MySQL

# apt-get install mysql-server

10.2.3 Rabbitmq-server

# apt-get install rabbitmq-server

10.2.4 Keystone

# apt-get install keystone

10.2.5 Glance

# apt-get install glance python-glanceclient

10.2.6 Nova

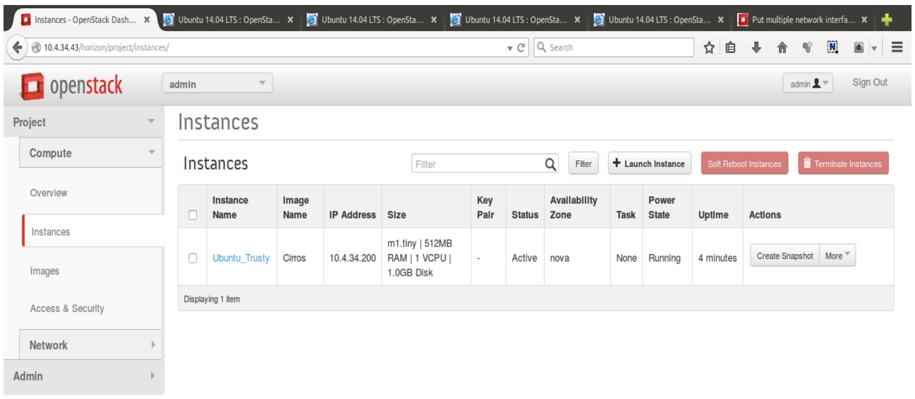
# apt-get install nova-api nova-cert nova-conductor nova-consoleauth nova-novncproxy nova-scheduler python-novaclient

10.2.7 Neutron

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

10.3 Virtual Machines

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



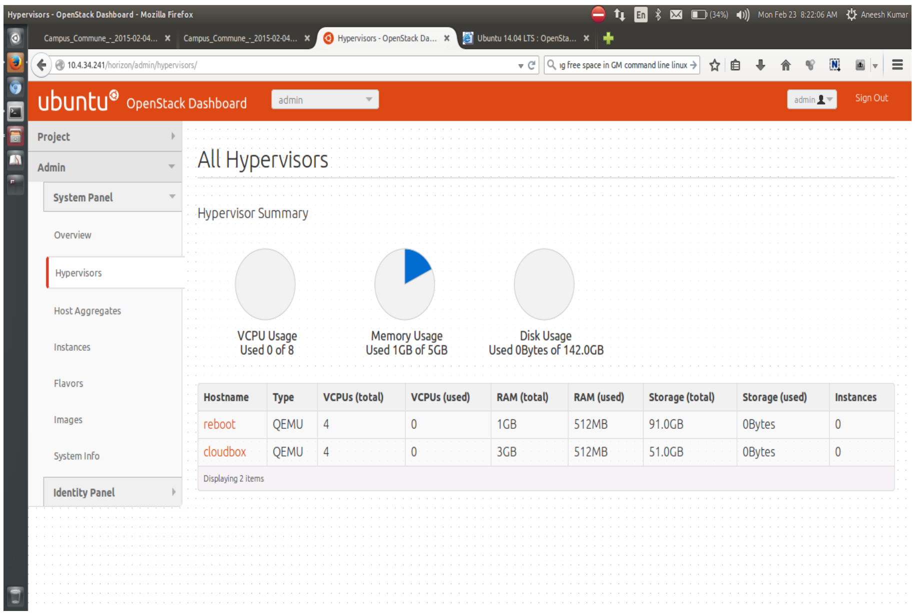
Instances

Instances

	Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Uptime	Actions
☐	Ubuntu_Trusty	Cirros	10.4.34.200	m1.tiny | 512MB RAM | 1 VCPU | 1.0GB Disk	-	Active	nova	None	Running	4 minutes	Create Snapshot More

Displaying 1 item

Figure 10.3: Openstack Virtual Machines.



Hypervisors - OpenStack Dashboard - Mozilla Firefox
Campus_Commune_-_2015-02-04...
Hypervisors - OpenStack Da...
Ubuntu 14.04 LTS : OpenSta...
10.4.34.241/horizon/admin/hypervisors/
Mon Feb 23 8:22:06 AM
Aneesh Kumar
ubuntu OpenStack Dashboard
admin
Sign Out
Project
Admin
System Panel
Overview
Hypervisors
Host Aggregates
Instances
Flavors
Images
System Info
Identity Panel

All Hypervisors

Hypervisor Summary

VCPU Usage
Used 0 of 8

Memory Usage
Used 1GB of 5GB

Disk Usage
Used 0Bytes of 142.0GB

Hostname	Type	VCPUs (total)	VCPUs (used)	RAM (total)	RAM (used)	Storage (total)	Storage (used)	Instances
reboot	QEMU	4	0	1GB	512MB	91.0GB	0Bytes	0
cloudbox	QEMU	4	0	3GB	512MB	51.0GB	0Bytes	0

Displaying 2 items

Figure 10.4: Openstack Resource Pool.

Chapter 11

Conclusion & Future Work

11.1 Conclusion

We worked on GlusterFS for replication of files 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 worked on network based sign on using LDAP,NFS and web based single sign on along with REST API using Oauth2 and Django. We tried to combine all these componets to deploy in private cloud. we worked on creating private cloud using openstack

11.2 Future Work

We would like to combine Network single sign-on with Web based single sign on along with XtreemFS and HAProxy for fault tolerant distributed environment. Creation of private cloud, virtual machines and deploying all components in private cloud avails us to use resources efficiently and all this work can be done on workstations.

Chapter 12

References

12.1 References

1. Nabil Sultan, “Cloud Computing for Education, International journal information management, 30, pp 109-116, 2010.

2. Tharam Dillon, Chen Wu and Elizabeth Chang, “Cloud Computing: Issues and Challenges”, 24thIEEE International Conference on Advanced Information Networking and Applications, 2010

3. Sebastian Rieger, “User-centric Identity Management in Heterogeneous Federations”, Fourth international conference on Internet and Web Applications and Services, 2009.

4. Jun Zheng, Qikun Zhang ,Shangwen Zheng and Yuan Tan, “Dynamic Role Based Access Control”, JOURNAL OF SOFTWARE, VOL. 6, NO. 6, JUNE 2011.

5. R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E.Youman, ”Role-Based Access Control Models,” [C] IEEE Computer, vol. 29, pp. 38-47, 1996.

6. Ivan Novakov, “Web Single Sign On Systems”, CESNET technical report number 21/2006

7. C.S.Yang, C.Y.Liu, J.H.Chen, C.Y.Sung, “Design and Implementations of Secure Web-based LDAP Management System”.

8. Ning Li, Qing Wang, Zhongliang Deng, “Authentication Framework of IIEDNS Based on LDAP and Kerberos”, Proceedings of IC-BNMT2010.

9. Salah A.Jaro Alabady, “Design implementation of Network Security Model Using Static VLAN and AAA Server

10. Shengli Liu, Wenbing Wang, Yuefei Zhu, “A New-Style Domain Integrated Management of Windows and UNIX”, The Ninth International Conference on Web-Age Information Management.

11. Ubnutu OS, http://www.ubuntu.com/

12. Openstack, https://openstack.org/

13. Linux Bible, http://tuxnetworks.blogspot.com

14. OAuth 2.0, http://oauth.net/

15. Git, https://github.com

16. Bootstrap, http://getbootstrap.com

17. Stack Overflow http://stackoverflow.com/

18. Google, http://www.google.com/

19. Django Docs – https://docs.djangoproject.com/en/1.7/

20. Django – https://djangoproject.com/

21. Django OAuth Tool Kit – https://github.com/evonove/django-oauth-toolkit

22. Django REST Framework – http://www.django-rest-framework.org/

23. OAuth 2.0 – http://oauth.net/2/

24. Semantic UI – http://beta.semantic-ui.com/

25. GlusterFS – https://www.digitalocean.com/HowToCreateaRedundantStoragePoolUsingGlus
DigitalOcean.htm

26. HAProxy – www.digitalocean.com/HowToUseHAProxytoSetUpHTTPLoadBalancingonanUbuntuV
DigitalOcean.htm

27. LDAP – https://www.digitalocean.com/community/tutorials/how-to-install-and-confi

28. NFS Server – http://www.server-world.info/en/note?os=Ubuntu_14.04&p=nfs

29. NFS Client – http://www.server-world.info/en/note?os=Ubuntu_14.04&p=nfs&
f=2

30. Openstack – http://www.server-world.info/en/note?os=Ubuntu_14.04&p=openstack_
icehouse