Cloud based IT Infra with Central Identity

{ Project reboot }

Phase I – Project Report

Project Guide

T. Chandra Shaker
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

Sep 2014 – Dec 2014

Abstract

The main objective of “Cloud based IT Infra with Central Identity” is to utilize exisiting hardware, turn them into private clouds and access all of its services using Central Identity, which can be available to third party developers as API with dynamic role management and service endpoints.

New private cloud based IT Infra is aimed to develop using some opensource tools like OpenStack, NFS, LDAP, Ubuntu and etc

Expecting to surve with high computational virtual machines to the research, academic, learning purpose, virtual labs rather than dedicated lab hardware.

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 cosists 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.
- Redendent 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

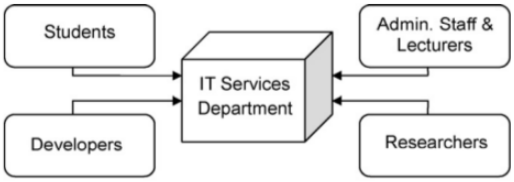

Figure 3.1: Simplified structure of the main users of IT services in a typical university.

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

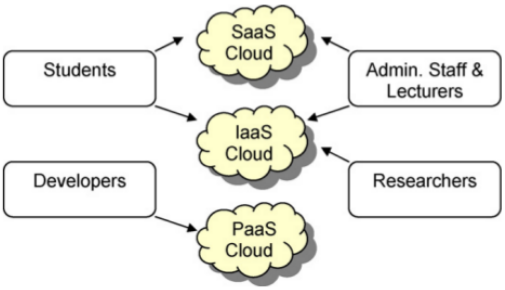

Figure 3.2: Simplified structure of the main users of IT services in a typical university now using the services of cloud computing

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

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

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 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.

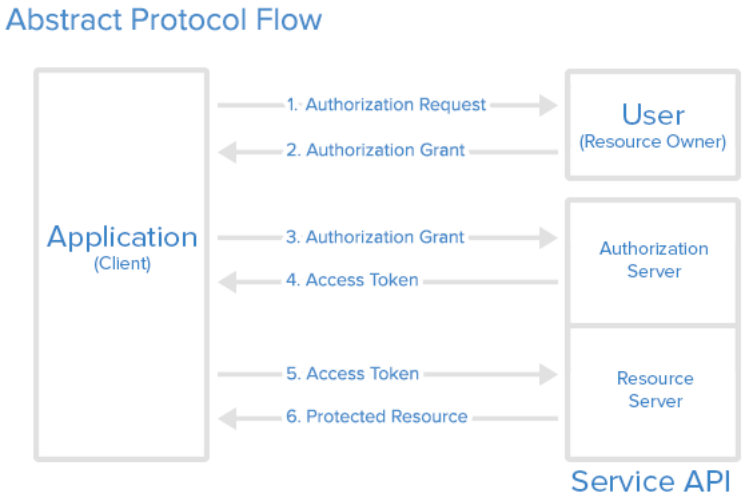

Figure 4.2: OAuth Protocol Work Flow Diagram

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

4.3 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.

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.

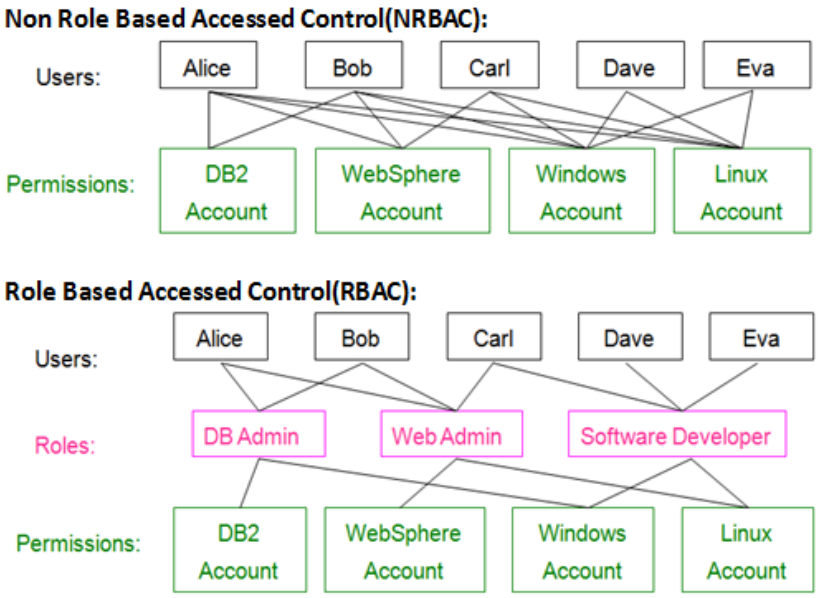

Figure 4.3: Difference between NRBAC and RBAC.

4.3.2 Basic Idea of RBAC

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

- 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.

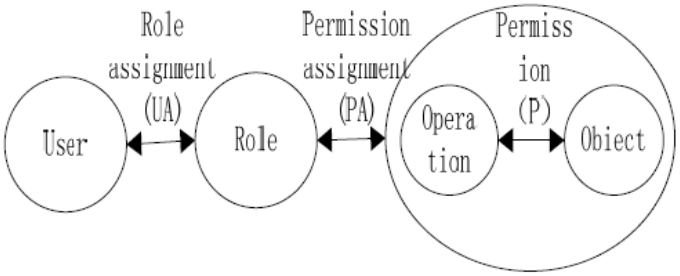

Figure 4.4: The Core Idea of RBAC.

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.

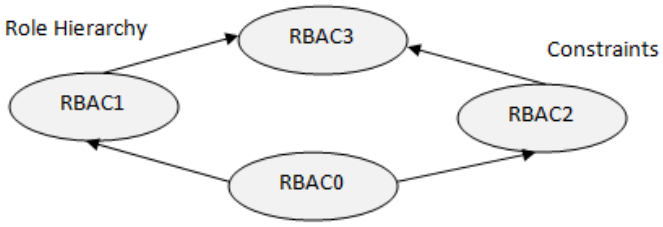

Figure 4.5: The Family of RBAC Model.

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

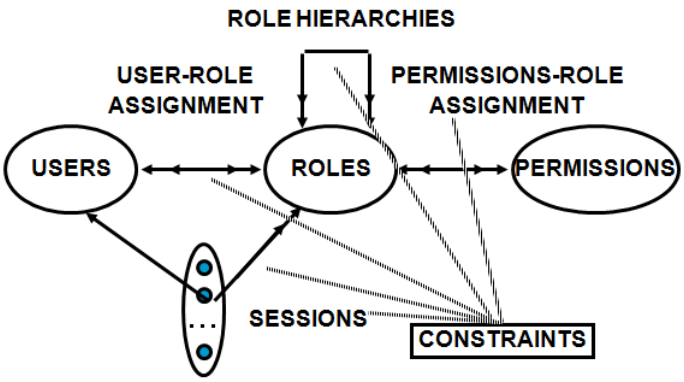

Figure 4.6: RBAC3 Model.

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.

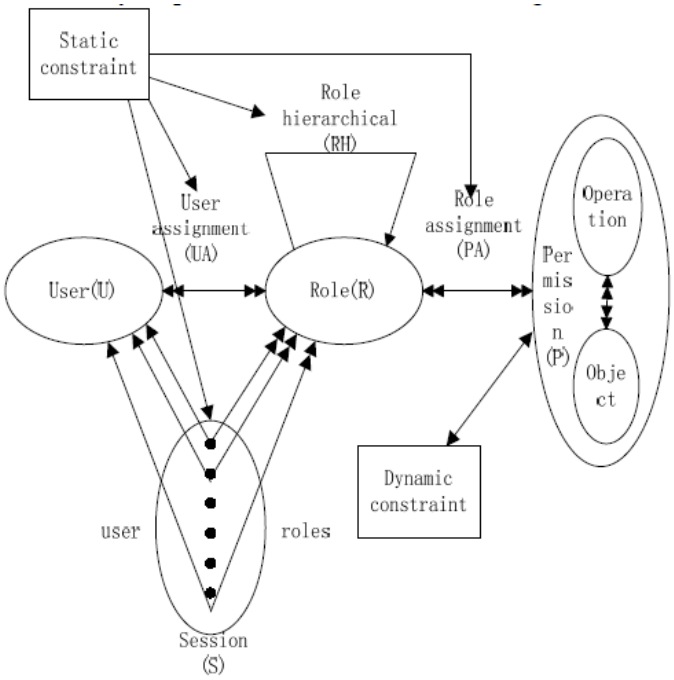

Figure 4.7: RBAC with Dynamic constraints.

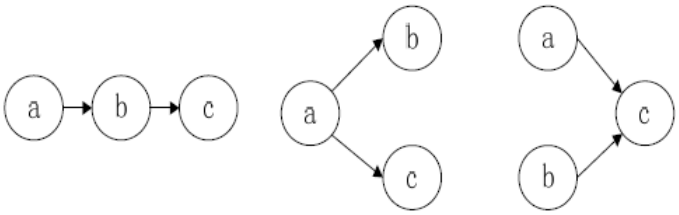

Figure 4.8: Different Types of Association.

4.4 Hybrid version with REST API to third party

4.5 Conclusion

Chapter 5

Network Components

5.1 Introduction

5.2 AAA

5.3 LDAP

5.4 NFS

5.5 Conclusion

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.

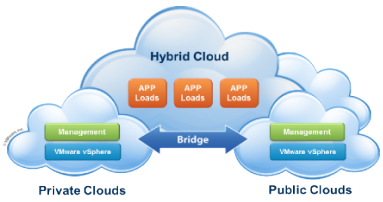

Figure 6.2: Cloud Computing - Deployment Models

6.2 Private Clouds

6.2.1 Introduction

6.2.2 Open Source Tools

6.3 Conclusion

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 Desired Technologies

- 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.
- Nodejs for Implementation of OAuth 2.0 as REST API.
- Git for Version control.
- Ascii doc / python shpenix & Latex for Documentation.

7.4 Expcted Results

We are expecting the below results after successfull implemenation of "Cloud Based IT Infra with Central Identity" in lab enviroment with above mentioned specifications

- Hardware resoucre clusters from Master Nodes
  - 40 GB RAM
  - 5TB of Hard disk
  - 15 GHz Clock Speed
- Hardware resoucre clusters from Slave Node (available only at active period)
  - 20 GB RAM
  - 2TB of Hard disk
  - 7.5 GHz Clock Speed
- Well Documented Implementation of Central Identity for
  - Single Sign on of Client Machines.
  - Network applications proxy, mails.
  - University Web Services and Departmental websites.
  - Third party OAuth 2.0 Implemenation as API.
- Dynamic user role management for both Web and Native applications as API.
- Central Cloud Storage Pool.
- High Computational Virtual Machines.
- Virtual Labs insted of Dedicated labs in Remote Desktop or in SSH protocol.
- Content or Data Monitering in a Organization.
- Get full recovery and achieve more than 99.99% services up time.
- Extendable to several departments and Universities.

Chapter 8

References

8.1 Web References

- Ubnutu OS, http://www.ubuntu.com/
- Openstack, https://openstack.org/
- Linux Bible, http://tuxnetworks.blogspot.com
- OAuth 2.0, http://oauth.net/
- Node.js https://nodejs.org
- Git, https://github.com
- Ascii doc, http://asciidoc.org
- Bootstrap, http://getbootstrap.com
- Stack Overflow http://stackoverflow.com/