

# 1.Introduction

## 1.1. Overview

This project aims to create sub versioning system in private cloud using IaaS (Infrastructure as a Service). Main purpose of the system is to provide the facility to the users to access all the versions of his/her file which he/she has synced in the master repository using intranet or internet.

## 1.2. Problem Summary

Subversion is a tool that enables version control for any kind of files. In normal scenario if user saves the file with the same name that he/she may save before in same directory, it will overwrite by new data and old file will lost. Now if we want old file, we couldn't get it. But by using subversion technology every time user saves the file with same name, system will make new version of file in master repository and we can get any previously created data in any instance of time. This all are done using IaaS (Infrastructure as a Service) architecture. There is a one center repository system that keeps all versions of files and their Meta data of all clients connected in intranet connection. By using intranet connection, there is no need of internet connectivity. Just all clients must be connected in single network topology. So all client machines may access all center stored data and if any one makes a change, new version is created and it will committed to master branch. This all are done using private cloud platform based on IaaS (Infrastructure as a Service) architecture. If we want to share files among 2 intranet network, only at that time internet connectivity is needed and it is done by online web application.

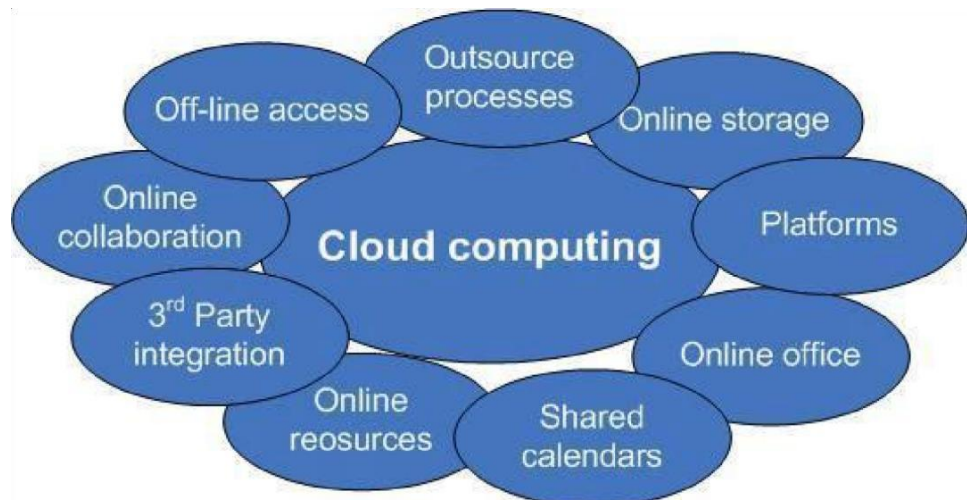
## 1.3. Objectives

This system will help users to store and update their data very effectively. By using subversion technology every time user saves the file with same name, the system will make new version of the file in master repository and we can get any previously created data at any instance of time. User machines can access all centrally stored data and if any one makes a change, new version is created and it will be committed to master branch.

## 1.4. Literature Review

### 1.4.1. Cloud Computing

Cloud computing is not a technology it is concept which overlays the principals of the distributed computing, grid computing, cluster computing. The Definition of the cloud computing was given by NSIT as per “Cloud computing is a model for enabling ubiquitous, 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 providerinteraction.”



Benefits of Cloud Computing

The above figure shows some benefits of cloud computing. As mentioned in the diagram we can have many platforms for cloud. We can share, collaborate, Store, Integrate our data in cloud.

### Essential Characteristics

- **On-demand self-service**

A user can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider.

- **Broad network access**

Capabilities are available over the network and accessed through standard mechanisms that promotes use by heterogeneous thin or thick client platforms.

- **Resource pooling**

The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to user demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter). Examples of resources include storage, processing, memory, and network bandwidth.

- **Rapid elasticity**

Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the Consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.

- **Measured service**

Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service).

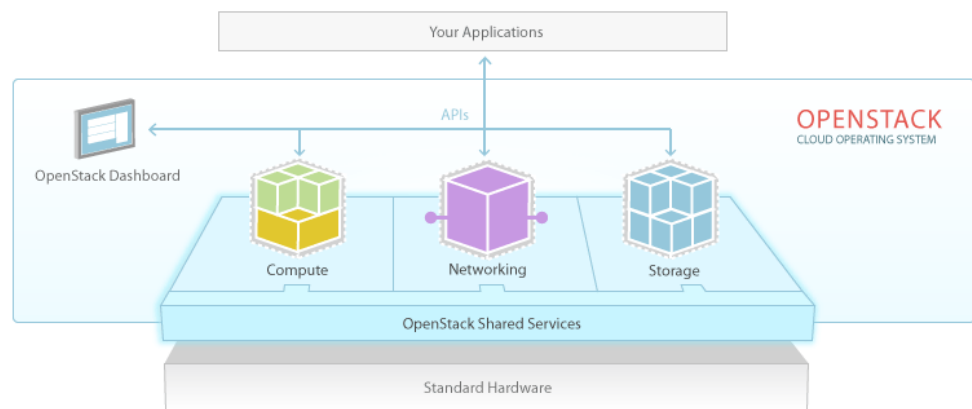
#### 1.4.2. Introduction to OpenStack

OpenStack is a free and open-source cloud computing software platform. Users primarily deploy it as an infrastructure as a service (IaaS) solution. The technology consists of a series of interrelated projects that control pools of processing, storage, and networking resources throughout a data center—which users manage through a web-based dashboard, command-line tools, or a RESTful API. OpenStack.org released it under the terms of the Apache License.

OpenStack lets users deploy virtual machines and other instances which handle different tasks for managing a cloud environment on the fly. It

makes horizontal scaling easy, which means that tasks which benefit from running concurrently can easily serve more or less users on the fly by just spinning up more instances. For example, a mobile application which needs to communicate with a remote server might be able to divide the work of communicating with each user across many different instances, all communicating with one another but scaling quickly and easily as the application gains more users.

OpenStack is open source software, which means that anyone make any changes or modifications they need, and freely share these changes back out to the community at large. So it has the benefit of thousands of developers all over the world working in tandem to develop the strongest, most robust, and most secure product that they can.



### 1.4.3. Introduction to J2EE

Using the Java 2 Platform, Standard Edition (J2SE) as a basis, Java 2 Platform, Enterprise Edition (J2EE) builds on top of this to provide the types of services that are necessary to build large scale, distributed, component based, multi-tier applications. Essentially, J2EE is a collection of APIs that can be used to build such systems, although this is only half of the picture. J2EE is also a standard for building and deploying enterprise applications, held together by the specifications of the APIs that it defines and the services that J2EE provides. In other words, this means that the "write once, run anywhere" promises of Java apply for enterprise applications too:

- Enterprise applications can be run on different platforms supporting the Java 2 platform.
- Enterprise applications are portable between application servers supporting the J2EE specification.

#### 1.4.4. Introduction to MySQL

MySQL is a freely available open source Relational Database Management System (RDBMS) that uses Structured Query Language (SQL).

SQL is the most popular language for adding, accessing and managing content in a database. It is most noted for its quick processing, proven reliability, ease and flexibility of use. MySQL is an essential part of almost every open source PHP application.

Facts about MySQL:

- MySQL is a database system used on the web
- MySQL is a database system that runs on a server
- MySQL is ideal for both small and large applications
- MySQL is very fast, reliable, and easy to use
- MySQL uses standard SQL
- MySQL compiles on a number of platforms
- MySQL is free to download and use
- MySQL is developed, distributed, and supported by Oracle Corporation

#### 1.4.5. Apache Commons

The Apache Commons is a project of the Apache Software Foundation, formerly under the Jakarta Project. The purpose of the Commons is to provide reusable, open source Java software. The Commons is composed of three parts: proper, sandbox, and dormant.

#### 1.4.6. AWS (Amazon Web Services)

Amazon Web Services (AWS) is a collection of remote computing services, also called web services, that make up a cloud computing platform offered by Amazon.com. These services are based out of 11 geographical regions across the world. The most central and well-known

of these services are Amazon Elastic Compute Cloud and Amazon S3. These products are marketed as a service to provide large computing capacity more quickly and cheaper than a client company building an actual physical server farm.

## 1.5. Project Planning and Management

Project Management is an important part of project development. It deals with all the main areas for project development like Feasibility, Requirement analysis, Project Schedule, Project Plan etc. We have used the Project management approach to deal with all these areas. It is achieved by proper selection of Software Life Cycle Model.

### 1.5.1. Project Scheduling

Project Scheduling is perhaps one of the most important works in developing any project. Before the project can begin estimate regarding work to be done, what resources will be required and how much time will elapse from start to the finish of a project. Planning helped us to prepare a framework that enabled to make us a reasonable estimate of all such things.

#### **Project Development Approach**

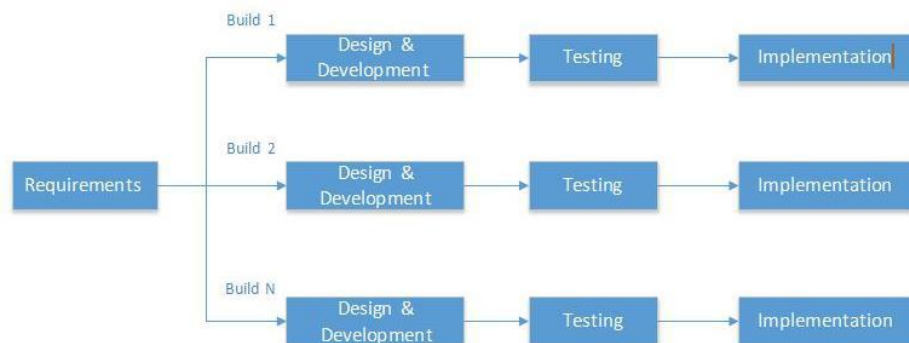
We are using conventional approach for system development. Reasons for choosing conventional approach are:

- Simple to understand and adopt.
- Consistent and aesthetically attractive notations.
- in the context of our project we decided that object oriented approach would create very complex diagrams and thus system would not be understood clearly. So we selected the conventional approach.

There are many types of Software Process Model like:

- Spiral Model
- Linear Sequential Model
- RAD (Rapid Application Development) Model
- Incremental Model
- Waterfall Model

### Incremental model



The Incremental model combines elements of the linear sequential model with the iterative philosophy of the prototyping. This model has been explicitly designed to accommodate a product that evolves over time.

When an incremental model is used, the first increment is often a core product. The core product is used by the customer or undergoes a detailed review. As a result of use and/or evaluation a plan is developed for the next increment. The plan addresses the modification to the core product to better meet the needs of the customer and delivery of additional features and functionality. Software is constructed in a step-by-step manner. While a software product is being developed, each step adds to what has already been completed.

### Advantages of Incremental Model

- System is developed and delivered in increments after establishing an overall architecture. Requirements and specifications for each increment may be developed.
- Users may experiment with delivered increments while others are being developed.
- Intended to combine some of the advantages of prototyping but with more manageable process and better system structure.

- Incremental development is especially useful when staffing is unavailable
- for a complete implementation by the business deadline. Early increments can be implemented with fewer people

### Project Timeline

Task No	Task	Date	Status
1	Research	2 July 2014 -15 July 2014	Completed
1.1	Project definition analysis	5 Aug 2014 - 15 Aug 2014	Completed
1.2	Searching for resources such as documentation and Tools	20 Aug 2014- 30 Aug 2014	Completed
2	Requirement and analysis information gathering	1 Sep 2014- 20 Sep 2014	Completed
2.1	Search Related Application	1 Sep 2014- 10 Sep 2014	Completed
2.2	Discuss About What Is Make	11 Sep 2014 - 15 Sep 2014	Completed
2.3	Finalize Modules	16 Sep 2014 - 20 Sep 2014	Completed
3.1	Private cloud setup using Open Stack	8 Oct 2014 - 9 Oct 2014	Networking Failed
3.2	Module for finding IP Address of devices connected in network. Module for getting modification notifications	6 Dec 2014 - 10 Dec 2014	Completed
3.3	Private cloud setup using OpenStack	1 Jan 2015-20 Jan 2015	Instance Successful and Networking Failed
4	Server API	2Feb 2015-15 Feb 2015	Completed
4.1	Java modules	20 Feb 2015-20 March 2015	Completed



4.2	Merging and Upgrading modules, Native Installer	6 April 2015-12 April 2015	Completed
4.3	Server Portal GUI and testing application	13 April 2015-15 April 2015	Completed

### Brief History of Work done

- **Task 1:** During the survey phase we found various Subversion systems relevant to the idea we developed. The difference between the systems and our project was that traditional Subversion systems provided the facility of public cloud whereas our main aim is to provide public as well as private cloud. In private cloud Infrastructure as a Service will be used. By various sources such as internet, magazines and eBooks we collected various information regarding for the setting up private cloud in the college.
- **Task 2:** After finalizing the Subversion System as project domain, we started to gather the information regarding how small scale private cloud can be established. We got many ways by which we can set up a private cloud such as FOSS Cloud, OpenStack, and Microsoft Finally we decided to set up using OpenStack Cloud.
- **Task 3:** Then we started by setting up the OpenStack in our system for creating the private cloud and connecting two private clouds through AWS services.
- **Task 4:** Then we started creating application in java for connection establishment purpose by installing NetBeans.
- **Task 5:** We created an application which can fetch all the IP's of the systems connected in a network and display it to the admin.
- **Task 6:** Then we deployed an application which can show the connections in GUI of the admin.
- **Task 7:** Then we started making an application which shows all the modifications done in any of the folder of systems such as delete, add, update etc.
- **Task 8:** Java Application GUI.
- **Task 9:** Server Portal GUI.

### 1.5.2. Milestones and Deliverables

- **Milestones** can add significant value to project scheduling. When combined with a scheduling methodology such as Program Evaluation and Review Technique (PERT) or the Critical Path Method (CPM), milestones allow project management to much more accurately determine whether or not the project is on schedule. By constraining the dates associated with milestones, the critical path can be determined for major schedule intervals in addition to the entire project. Slack/float can also be calculated on each schedule interval. This segmentation of the project schedule into intervals allows earlier indication of schedule problems and a better view into the activities whose completion is critical.
- Milestones are frequently used to monitor the progress, but there are limitations to their effectiveness. They usually show progress only on the critical path, and ignore non-critical activities. It is common for resources to be moved from non-critical activities to critical activities to ensure that milestones are met. This gives the impression that the project is on schedule when actually some activities are being ignored. Milestones are like dashboard reviews of a project. Number of activities which were planned at the beginning of the project with their individual timelines are reviewed for their status. It also gives an opportunity to check the health of the project.
- **Deliverable** is a term used in project management to describe a tangible or intangible object produced as a result of the project that is intended to be delivered to a customer (either internal or external). A deliverable could be a report, a document, a server upgrade or any other building block of an overall project. A deliverable may be composed of multiple smaller deliverables. It may be either an outcome to be achieved (as in "The Corporation says that becoming profitable this year is a deliverable.") or an output to be provided (as in "The deliverable for the completed project consists of a special-purpose electronic device and its controlling software.").
- A deliverable differs from a project milestone in that a milestone is a measurement of progress toward an output whereas the deliverable is the result of the process. For a typical project, a milestone might be the

completion of a product design while the deliverable might be the technical diagram of the product. A deliverable also differs from a project document in that project document is typically part of a project deliverable, or a project deliverable may contain number of documents and physical thing.

### 1.5.3. Risk management

A risk is any anticipated unfavorable event or circumstances that can occur while a project is underway.

#### **Risk Identification**

Risk Identification is the first stage of risk management. It is connected with discovering possible risk to the project. Following are the risk identified to the project.

##### **a) Schedule Risk**

Requirements are not clear about the project and also it is a new area of concern to deal with, it is slightly difficult to maintain the strict schedule for the project because we cannot guarantee the user as rather than using the system he can be aggressive to test the system.

##### **b) Process Risk**

As this project is a part of the system there can be some problem regarding the fact that user can use the system aggressively.

##### **c) Technical Risk**

There are many technical problems we faced during coding and execution like memory issue, hardware capability issues.

##### **d) Other Risks**

Other risks can be changes in the requirement, which may require major design rework.

### **Risk Analysis**

Risk Monitoring involves regularly assessing each of the identified risk to decide whether or not the risk is becoming more or less probable and whether the effects of the risk have been changed. Risk Monitoring was a continuous process throughout the development phase.

### **Risk Planning**

For scheduling risk we had decided the strategy that as everything is not properly defined, any requirement which comes to us will be kept by us and we will check its feasibility. There is no alternative of the process risk; we need to study current structure and standard of the current system deeply. Changing requirement risk, strategy has been decided that traceability information should be maintained every time the requirement added.

<b>Strategy</b>	<b>Maintainability</b>
<b>For Scheduled Risk</b>	Average
<b>For Technological Risk</b>	Average
<b>For Process Risk</b>	Average
<b>For Change in Requirements</b>	High

#### **1.5.4. Estimation**

Software measurement is concerned with deriving a numeric value for some attributes of a software product or a software process. By comparing these values to each other and to standards which apply across on organization it is possible to draw conclusion about the quality of software process

### **Cost and Effort Estimation**

It can be done using any of these four techniques. Along with the ways to estimate justifications are given to state why the strategy was not selected or why was it selected.

- Delay estimation until late in the project.
- Base estimation on similar projects that have already been completed.
- Use relatively simple decomposition techniques to generate project cost and effort estimates.
- Use one or more models for software cost and effort estimation.

**Description:**

- First is very simple and provides accurate results but, it was not possible to delay cost of software until end as it could increase risk of project.
- Second is not preferable as it is not possible to find resemblance and estimated cost of project using some other previous projects.
- Obvious choice remaining is third and fourth out of which third is given more priority and is worked on.
- Decomposing project in small parts and estimating is very crucial.

## 1.6. Prior Art

### 1. **Research Publication:** SECURE CLOUD STORAGE AND SYNCHRONIZATION SYSTEMS AND METHODS (US2014344572 (A1))

- **Patent Search (Summary):** The present inventors have developed an integrated solution that addresses all of these issues and more, all while strictly maintaining the privacy of the user's data. These solutions are alternatively referred to herein individually and collectively as the system, method, or application. In various example embodiment the system may allow a user to mark any file on any of his/her devices for backup and synchronization, and mark any directory on any of his/her devices for backup and synchronization. This may provide backup/synchronization of all files in the marked directory, and recursive marking of all sub-directories (and the files contained therein) of the marked directory. Files subsequently added to the marked directory may likewise be backed up and synchronized in these embodiment.
- **Limitation:** This prior art does not applied to sub-version technology.

### 2. **Research Publication:** LOCAL SERVER FOR SYNCED ONLINE CONTENT MANAGEMENT SYSTEM (US2014289360 (A1))

- **Patent Search (Summary):** Additional features and advantages of the disclosure will be set forth in the description which follows, and in part will be obvious from the description, or can be learned by practice of the herein disclosed principles. The features and advantages of the disclosure can be realized and obtained by means of the instruments and combinations particularly pointed out in the appended claims. These and other features of the disclosure will become more fully apparent from the following description and appended claims, or can be learned by the practice of the principles set forth herein. Disclosed are systems, methods, and non-transitory computer-readable storage media for a local server operable with a synchronized online content management system. Various embodiments of the present disclosure enable data to be stored and synchronized at both a local computing device and an online content management system. In some embodiments, a client computing device can implement a client application corresponding to an online content management system. The client application can enable selected data (e.g., content items, files, folders, etc.) stored locally on the client computing device to be stored and synchronized at the

online content management system. The synchronization process between the locally stored data (i.e., local data) and the data (i.e., online data) stored at the online content management system can occur at one or more specified times (e.g., at a schedule time, in response to a synchronization command, when data is modified, etc.). In one example, the client computing device can be running a web browser. In this example, a user of the web browser can use the web browser to surf the Internet. During or after surfing, the user can decide to use the web browser to access one or more of his content items (e.g., files) stored at the online content management system. Since the browser is already open and being used by the user, it can be more convenient to the user to use the browser to access the online data than it would be to switch to a content navigation (e.g., file navigation) application to access local data (which is synchronized to the online data). However, the user experience associated with accessing online data using the web browser can vary depending on the speed and/or reliability of the network connection (e.g., Internet speed and/or availability).

- **Limitation:** This prior art doesn't provide private cloud features like reliability, scalability, virtualization, load balancing etc.

### 3. Research Publication: SYSTEM AND METHOD FOR A CLOUD COMPUTING ABSTRACTION LAYER WITH SECURITY ZONE (US 20120185913 A1 )

- **Patent Search (Summary):** In embodiments of the present invention improved capabilities are described for a virtualization environment adapted for development and deployment of at least one software workload, the virtualization environment having a meta model framework that allows the association of a policy to the software workload upon development of the workload that is applied upon deployment of the software workload. This allows a developer to define a security zone and to apply at least one type of security policy with respect to the security zone including the type of security zone policy in the meta model framework such that the type of security zone policy can be associated with the software workload upon development of the software workload, and if the type of security zone policy is associated with the software workload, automatically applying the security policy to the software workload when the software workload is deployed within the security zone.
- **Limitation:** This prior art doesn't have information about synchronization process and sub-version technology.

## 2. Design: Analysis, Design Methodology and Implementation Strategy

### 2.1. Hardware Requirements

- Linux Server/PC for performing OpenStack operations.
- Minimum 2 PC's for subversion and sync operations.
- Router with Wi-Fi certified.
- Ethernet cables for connecting clients to the repository.

### 2.2. Software Requirements

- GIT system for cloning the setup files of Dev stack online library to the local libraries.
- Open Stack for establish cloud on PC's.
- Python software to run cloud on a device or PC
- JDK
- NetBeans 8.0
- Dreamweaver
- Web Browser
- Apache Tomcat server for establishing connection to master repository.
- My SQL server.

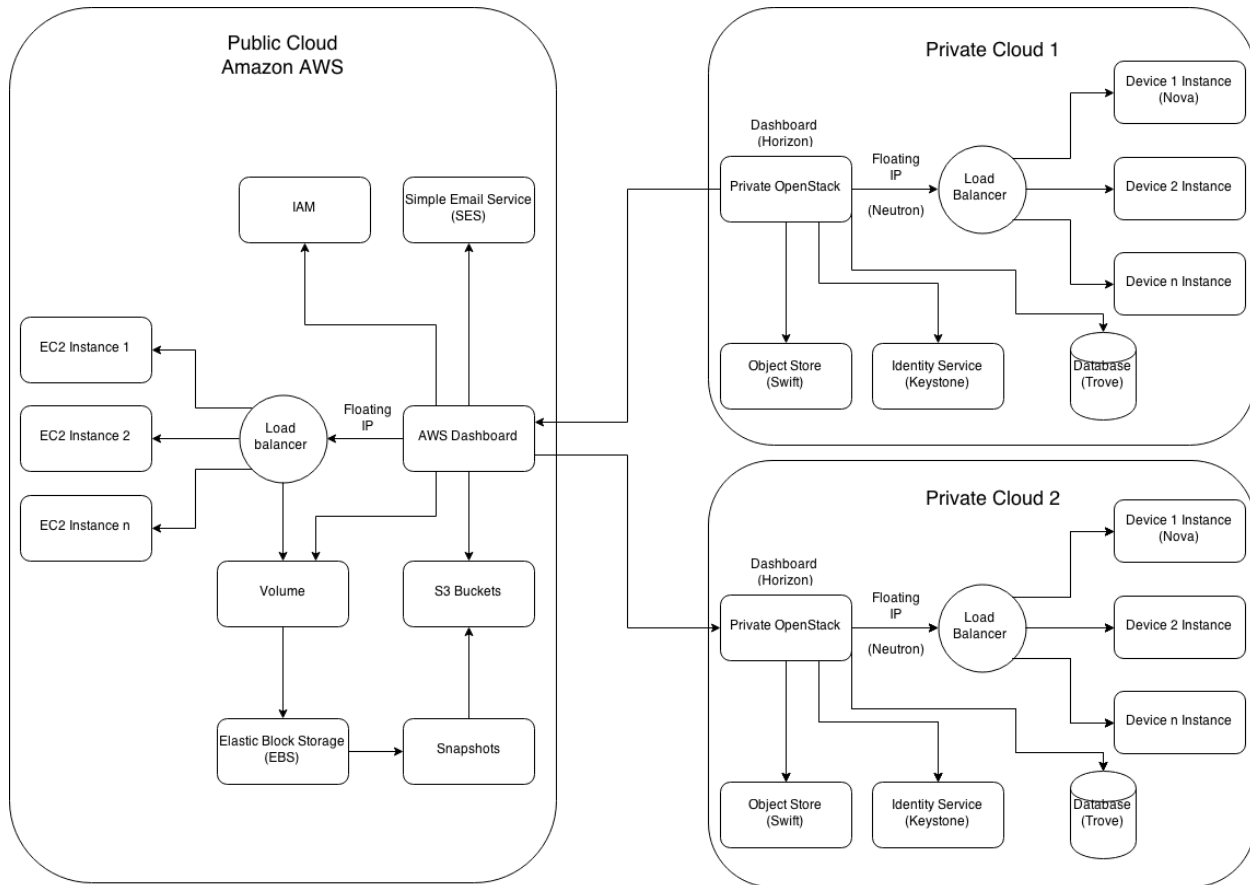


## 2.3. Non-functional requirements

- Ubuntu 12.04 or 14.04 required, not latest 15.04 because this version supports only OpenStack Kilo that is currently under beta mode, not production.
- JDK version 1.7 or higher required
- Some Java libraries required to perform subversion efficiently
  - Java NIO
  - JAVA JNI
  - JAVA JNA
  - 3<sup>rd</sup> party libraries that supports Windows32 DLL access
- Linux based VM Image for OpenStack instance

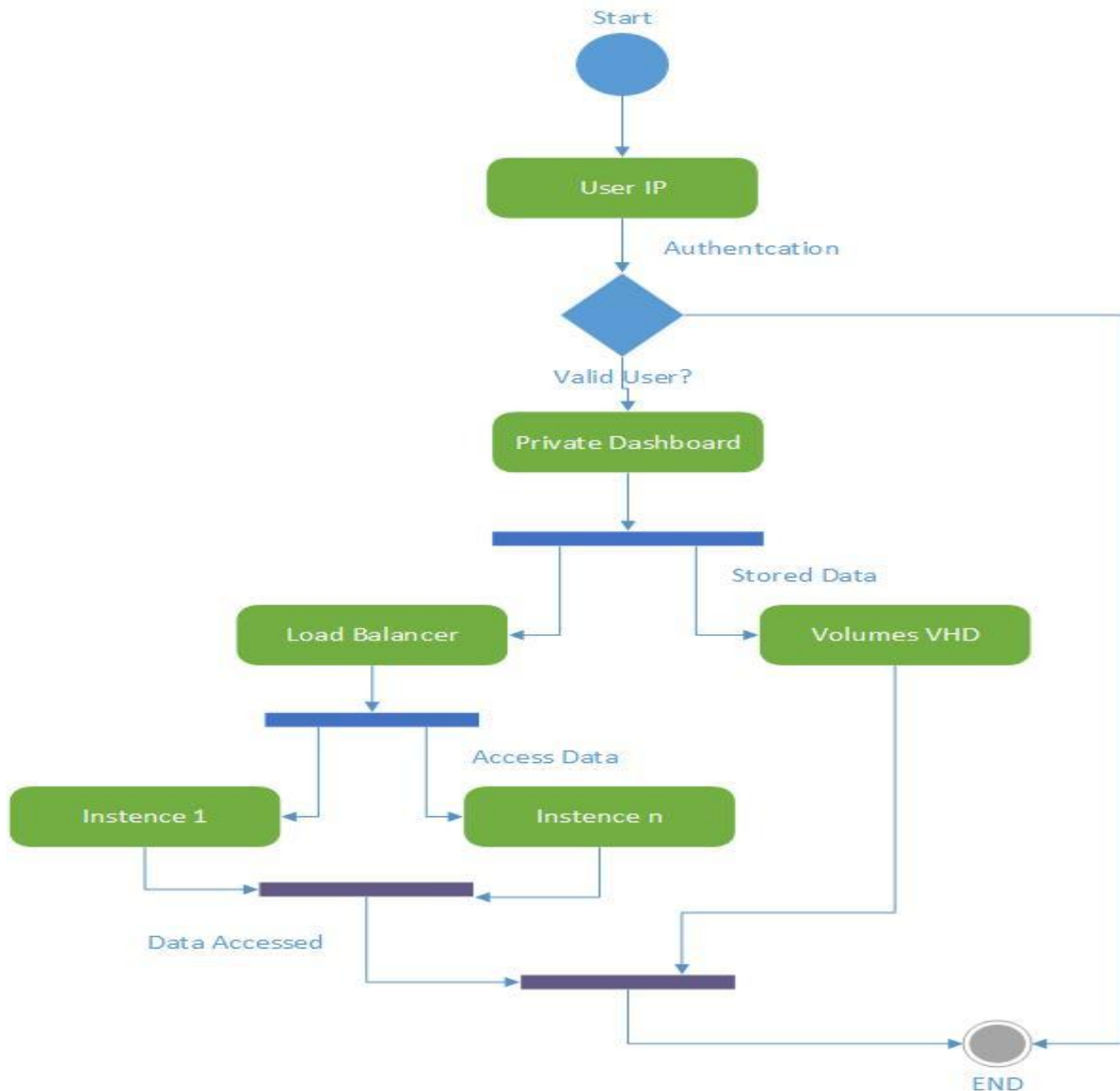
## 2.4. System Modeling and Design

### 2.4.1. System Architecture



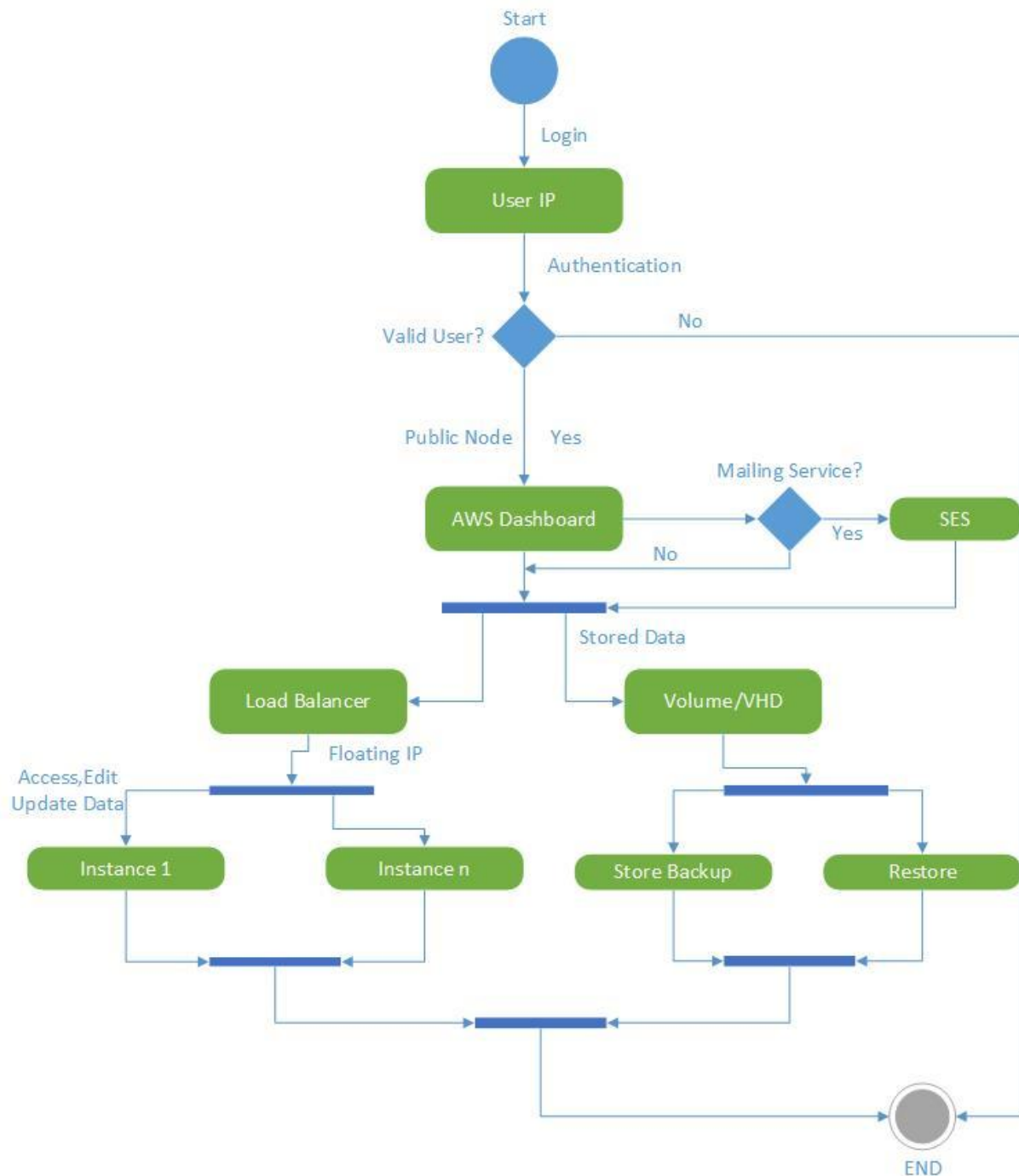
The above figure shows Subversion between two Private clouds. In private cloud, Load balancer Distribute working workloads across multiple computing resources, such as computers, a computer cluster, network links, central processing units or disk drives. Swift is used to store objects and keystone is to identify the services. If we have two different private cloud located in different places we would use Amazon Web Services (AWS). Here Load balancer do the same process as it done in a private cloud. AWS would help us sharing data in two different private clouds.

### 2.4.2. Activity Diagram for Private Cloud



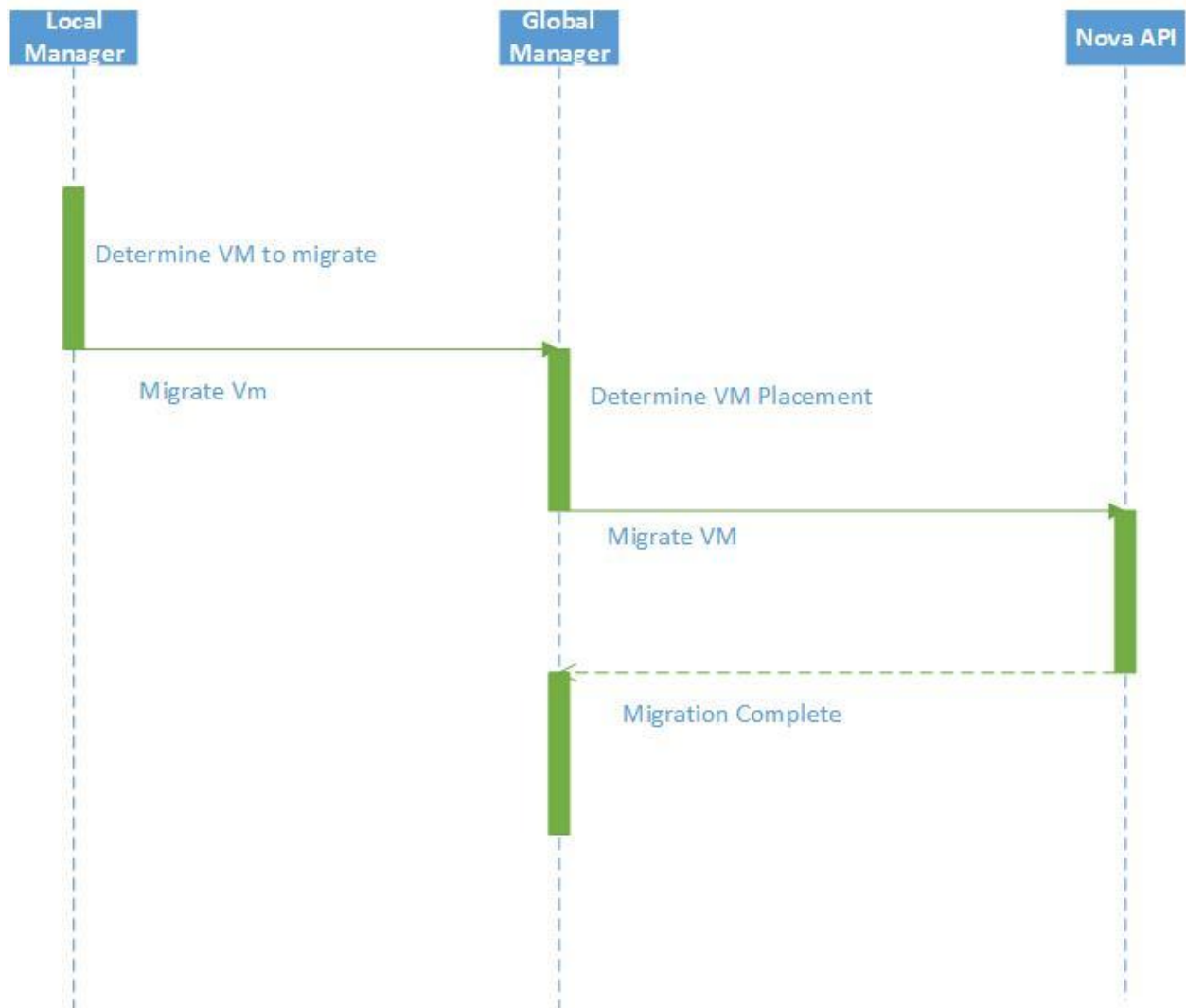
Here, first of all we will start private cloud. It would authenticate user IP. After authentication process user would see its own private dashboard. VHD, virtual hard disk is used to store data and objects. Load balancer would create instances which can access the data. We can create unlimited instances in the system. It completely depends on the device where you create your private cloud.

## 2.4.3. Activity Diagram for Public Cloud



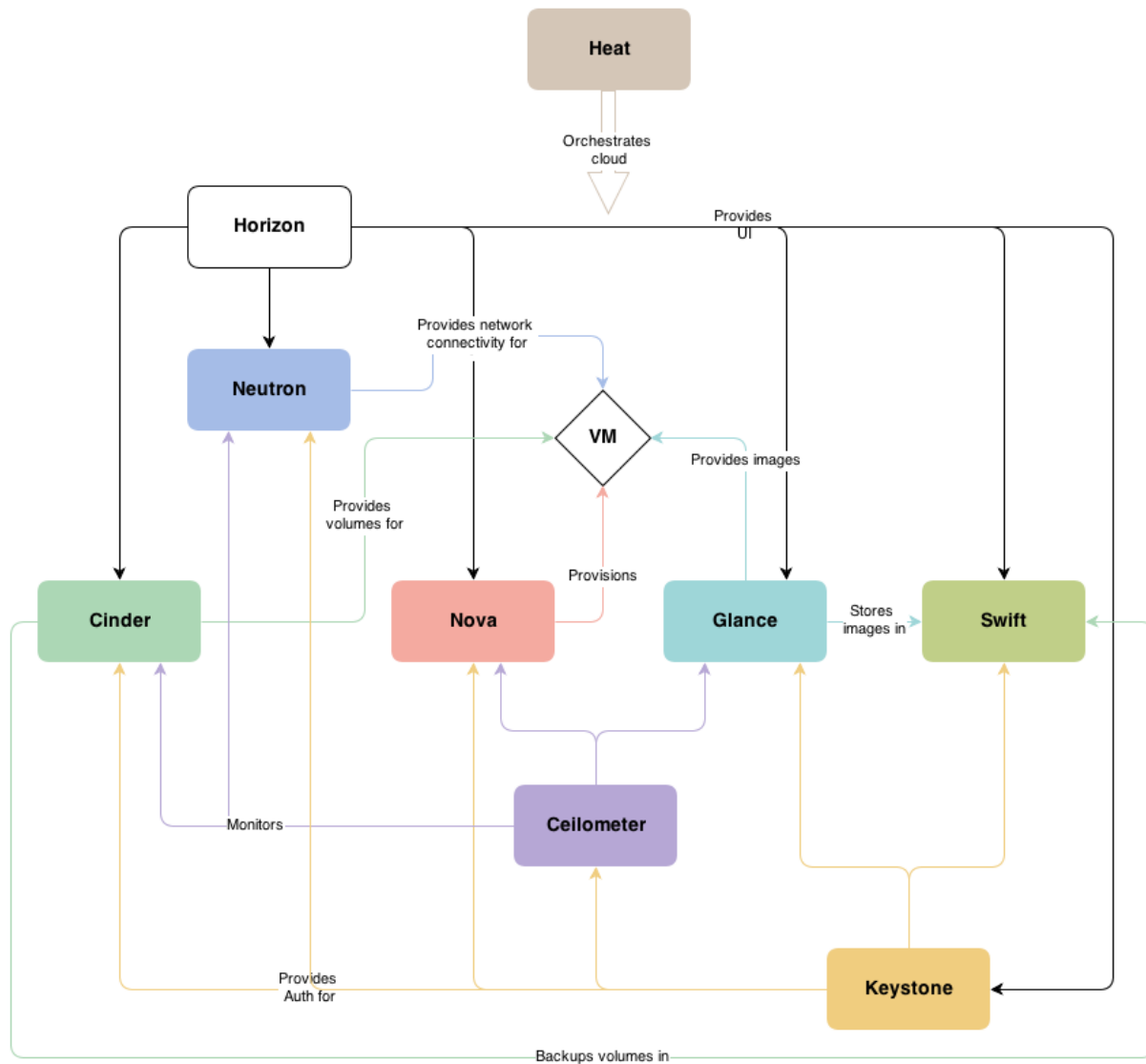
Public cloud is used to transfer, sync and share data from one private cloud from another private cloud. We are using Amazon web services (AWS) for transferring, syncing and sharing data. We would first login to the system, so it will take us to amazon dashboard. Here load balancer create instances to access, edit and update data. Virtual hard disk (VHD) store, backup and restore data.

#### 2.4.4. OpenStack Sequence Diagram



In OpenStack sequence diagram First Local manager determine virtual machine (VM) to migrate it. After that it will send it to global manager. Again this Global manager would determine VM to migrate. This Migrated VM would go to Nova API. Nova API is used to for the migration of the data in OpenStack private cloud.

### 2.4.5. OpenStack Architecture



**Horizon:** Horizon is the canonical implementation of OpenStack’s Dashboard, which provides a web based user interface to OpenStack services including Nova, Swift, Keystone, etc.

**Swift:** Swift is a highly available, distributed, eventually consistent object/blob store. Organizations can use Swift to store lots of data efficiently, safely, and cheaply.

**Keystone:** Keystone is an OpenStack project that provides Identity, Token, Catalog and Policy services for use specifically by projects in the OpenStack family. It implements OpenStack’s Identity API.

**Neutron:** OpenStack Neutron is a cloud networking controller and a networking-as-a-service project within the OpenStack cloud computing initiative.

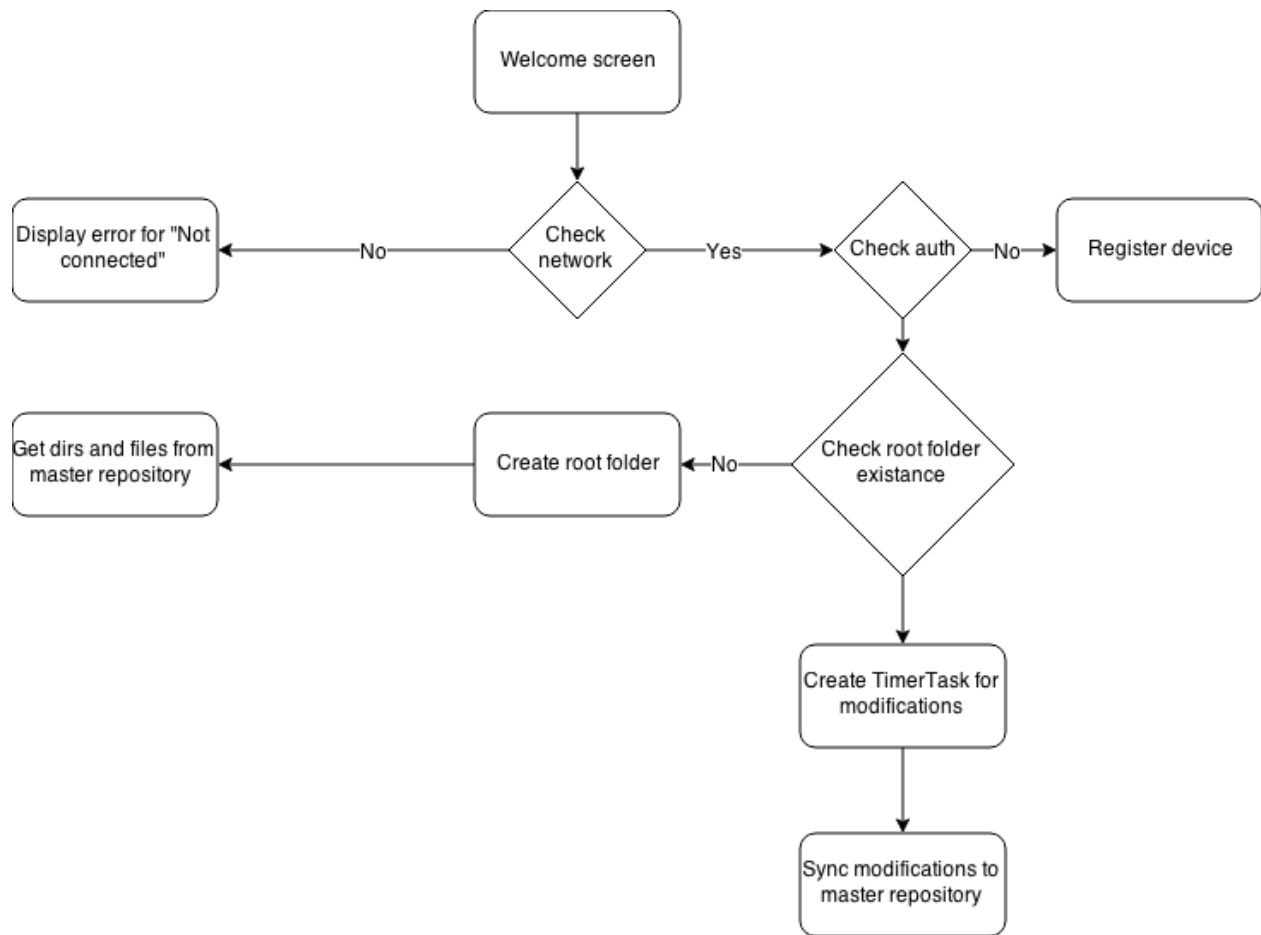
**Ceilometer:** The Ceilometer project aims to deliver a unique point of contact for billing systems to acquire all of the measurements they need to establish customer billing, across all current OpenStack core components with work underway to support future OpenStack components.

**Nova:** Nova is the project name for OpenStack Compute, a cloud computing fabric controller, the main part of an IaaS system. Individuals and organizations can use Nova to host and manage their own cloud computing systems.

**Cinder:** Cinder is a Block Storage service for OpenStack. It's designed to allow the use of either a reference implementation (LVM) to present storage resources to end users that can be consumed by the OpenStack Compute Project (Nova).

**Glance:** The Glance project provides a service where users can upload and discover data assets that are meant to be used with other services. This currently includes images and metadata definitions. Glance image services include discovering, registering, and retrieving virtual machine images. Glance has a RESTful API that allows querying of VM image metadata as well as retrieval of the actual image. VM images made available through Glance can be stored in a variety of locations from simple file systems to object-storage systems like the OpenStack Swift project.

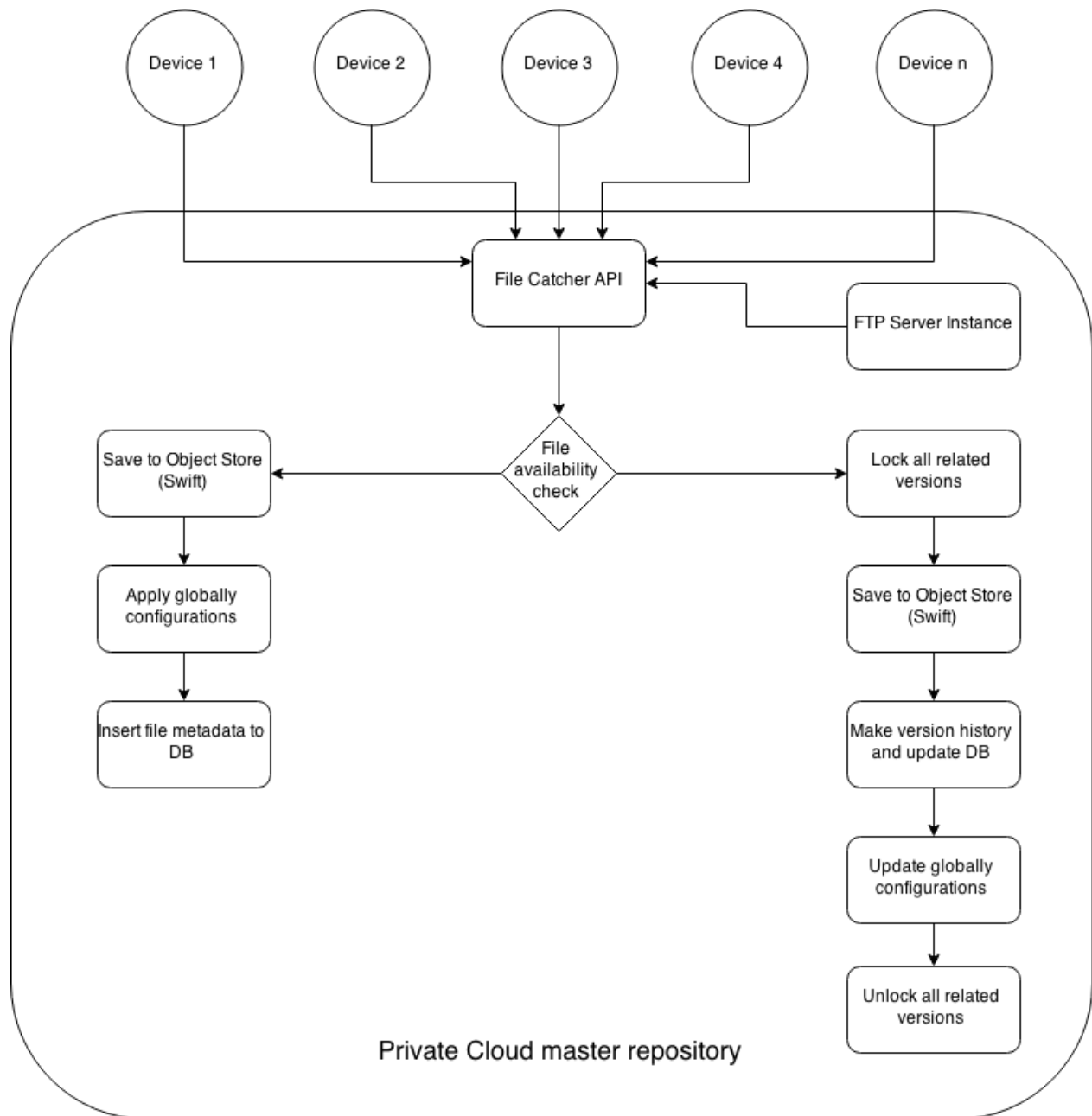
### 2.4.6. Client app working architecture



When we open a BuddsCloud app in our pc or laptop first welcome screen would appear on the screen. It would check the network. If the network is working properly it would check if the device is authenticated before or not. If not it will register a network or else it would go the root folder. If the network is not available it will show a message “Device Not Connected”. After authentication process it checks the root folder existence. If root exists it creates a timer task for modification and sync modification to master repository and if root folder does not exists it will create a new root folder to get directories and files from master repository.

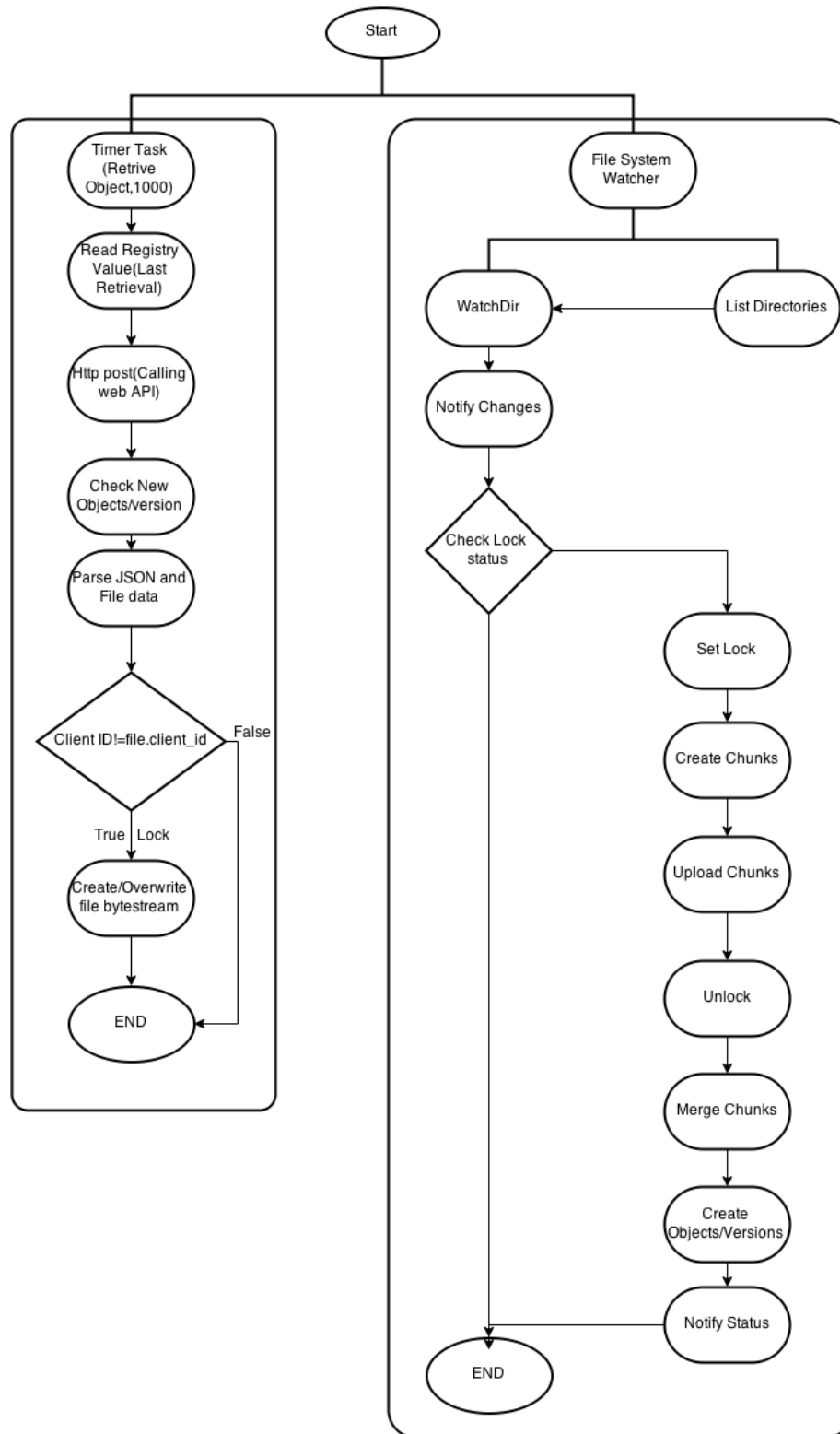


### 2.4.7. Subversion in master repository



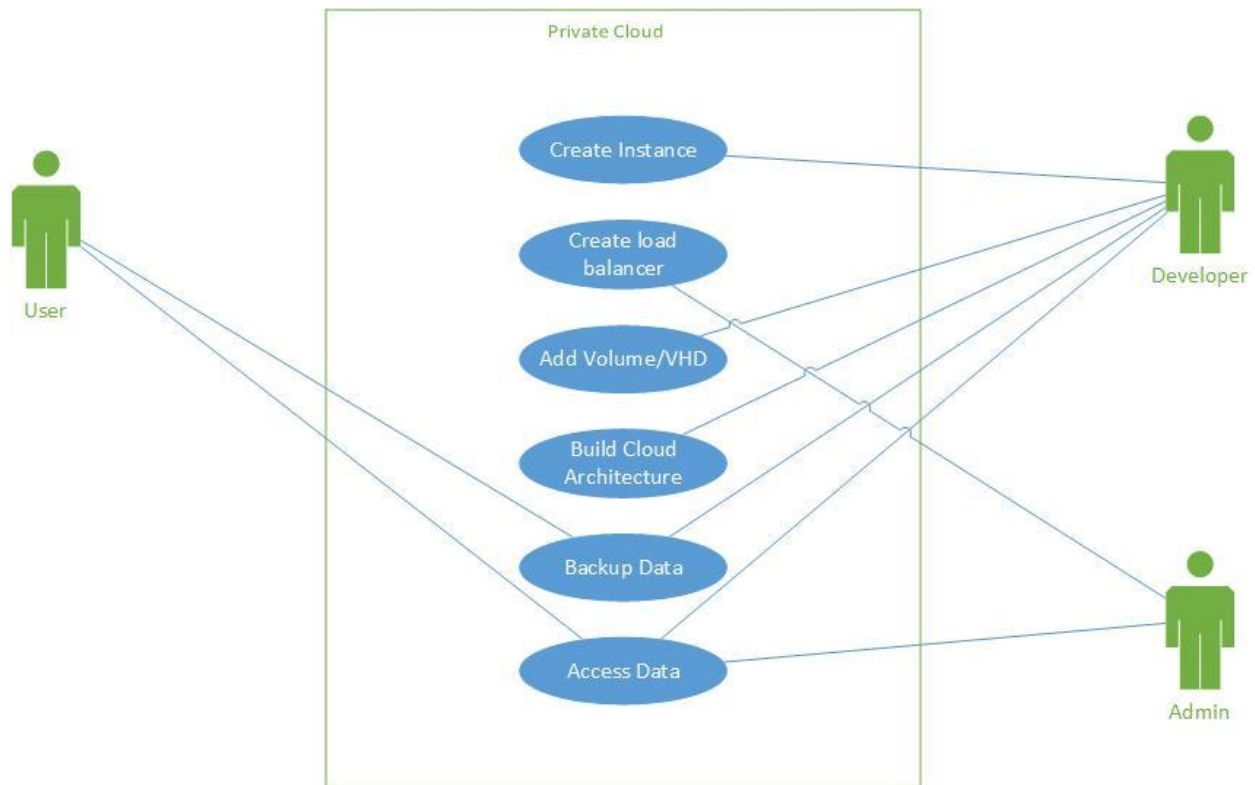
All devices are connected with private cloud. Here, file catcher API fetch the files from all connected device. Then it will check the availability of the particular file, means it will check whether file is already there or not. If file already exists it will lock all the versions of the file which are stored in private cloud and save it to the object store. It will make a version history and update the database. It will also update the global configurations and then it will unlock all related versions. Now If the file is not exists it will create a new object store to store the file. It will apply global configuration for that new file.

## 2.4.8. Synchronizing system



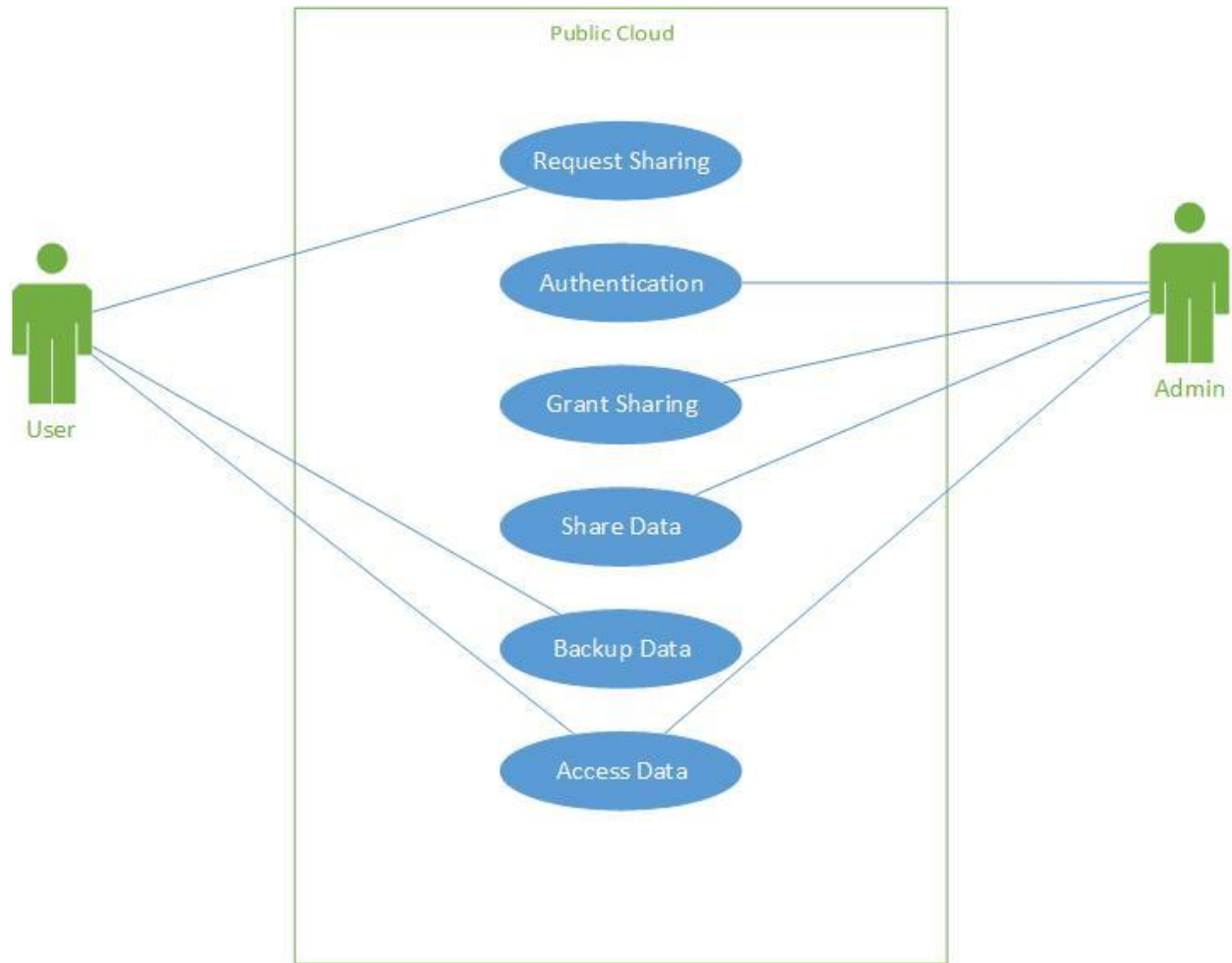
The above figure shows the syncing process. In sync system after every 1 second system will check whether any new object is created or not. It will also check if there is any versions created of an existing object. After that it will read the registry value. After reading registry value system will call HTTP post method to call web API. It will check the new version or new object if there is any. Then it will parse the JSON data and file data to client. But here it will check the client ID. Simultaneously File system watcher would work. Here It will check whether directory is listed in the database or not. If there is any changes in the directory it will notify changes to watch directory. Here Condition check lock status will check after the changes has made does the file locked or not. If the file is not locked it will set the lock. Then it will create chunks of that file. After that these chunks would uploaded to the server. After all the chunks upload to the server, It will unlock the file and merge all the chunks. That's how to simultaneous system Time watcher and file system watcher work in synchronization system.

### 2.4.9. Use case for private cloud



The above figure shows the use case diagram for private cloud. There are three actors who could access the private cloud. One is user and other two are developer and admin. User can access the data and backup the data in private cloud. There is no other authority user can have in private cloud. A developer who built a private cloud can have authority to access everything. Developer can create instance, load balancer, add virtual hard disk, build cloud architecture, access and backup the data. The admin of the private cloud can only access data and can create load balancer.

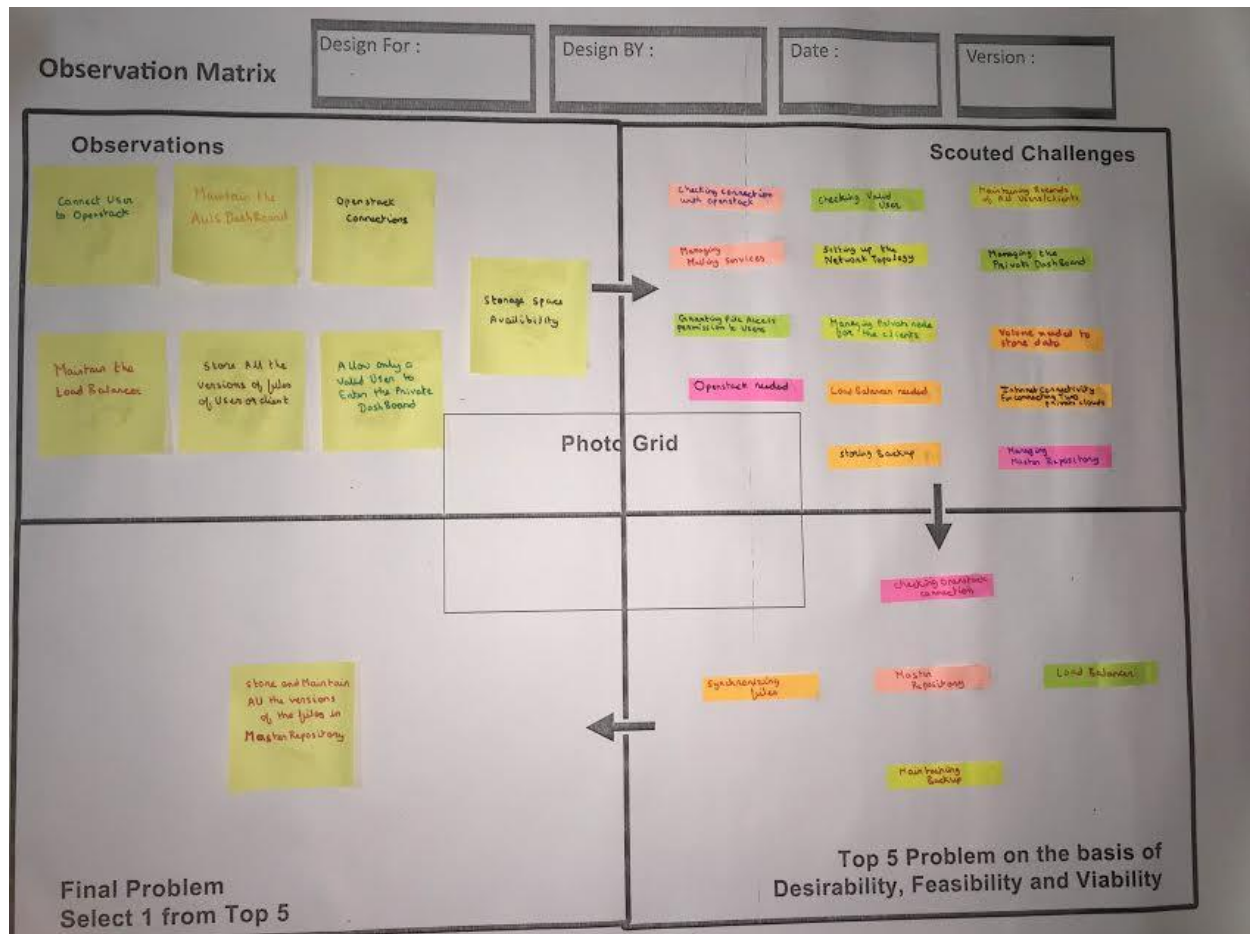
## 2.4.10. Use case for public cloud



The above diagram shows the use case diagram for public cloud. There are two actors in public cloud. One is user and another is admin. User can request admin for sharing data to another devices or users. User can also access and backup data. Admin has a major role in public cloud. Admin authenticate the user when user login to access, share or backup the data. Admin can allow user to share data.

## 2.5. Canvas Exercise

### 2.5.1. Observation Matrix



#### OBSERVATIONS:

- The observations of this project includes the following steps:
  1. Connection: Connect the user to the OpenStack.
  2. AWS Management: To maintain the AWS dashboard for the different versions of files and data.
  3. Establishment: Establishing the OpenStack connections.
  4. Load Balancer: To maintain the load balancer for data traffic.
  5. Storage: To store all the versions of files of different users with respect to timestamp.
  6. Validation: Check validity and allow only valid users to enter the private Dashboard.

7. Availability: Check the availability of storage space to store the files of users.

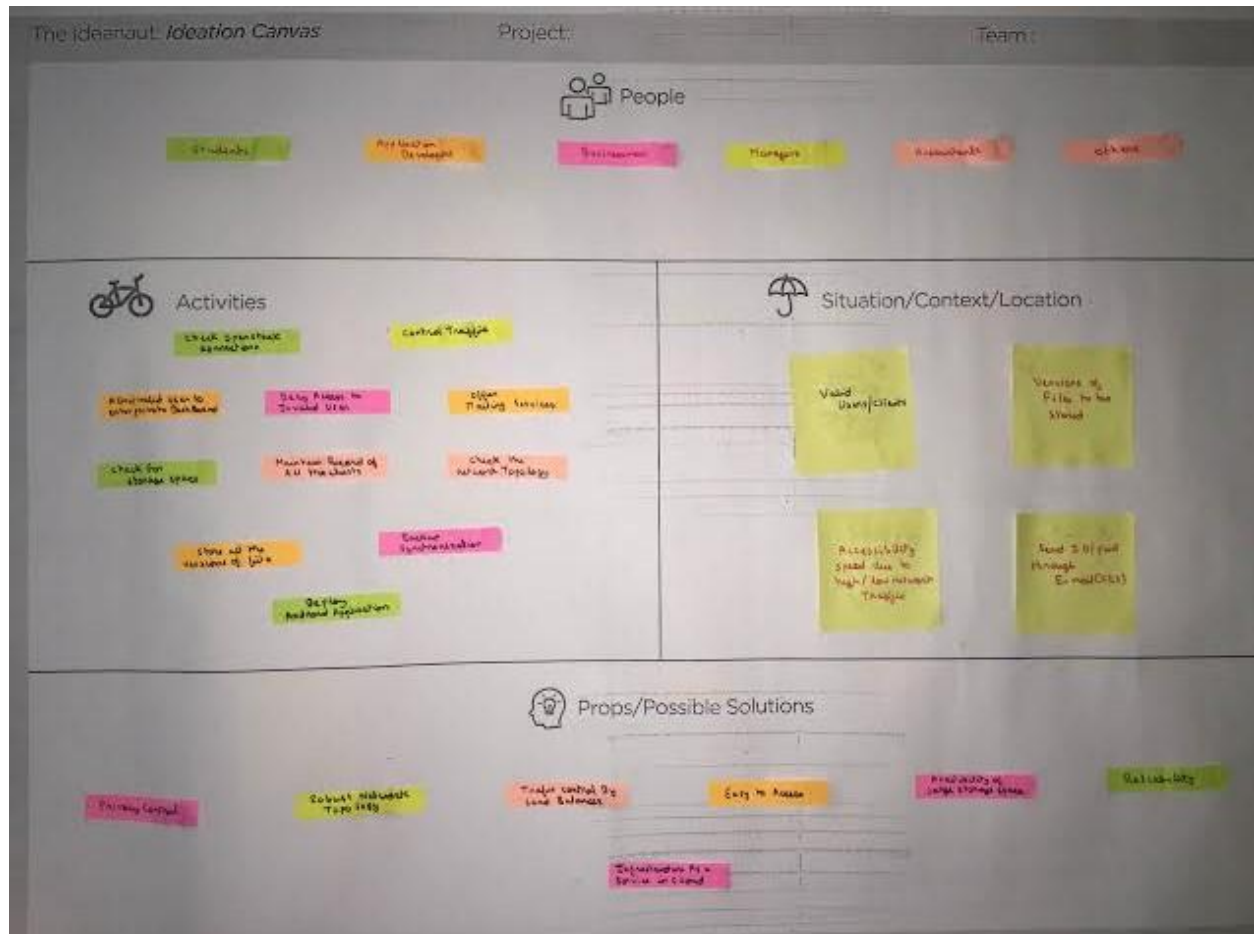
### **SCOUTED CHALLENGES:**

- The challenges during the implementation are as follows:
  1. Checking the connection with the OpenStack.
  2. Validating or Authenticating the users to use specific files.
  3. Maintaining records of all users and clients.
  4. Providing mailing services to the clients.
  5. Setting up the network topology.
  6. Maintaining the private dashboard for frequent users.
  7. Granting file access permission to the users.
  8. Managing the private node for the clients.
  9. Providing volume for storing data or files.
  10. Installation of OpenStack needed.
  11. Providing Load balancer for maintaining the data traffic.
  12. Storing and maintaining the backup of files or data.
  13. Checking the internet connectivity for the communication between two or more private clouds.
  14. Managing the master repository where all versions of files are stored.

### **TOP FIVE PROBLEMS:**

- Top 5 Problems faced during the setup of this project are:
  1. Checking the OpenStack connection.
  2. Synchronization of files and data of users.
  3. Maintaining the Load balancer.
  4. Maintaining the master repository for storage of all files.
  5. Storing and maintaining the backup for the versions of files.

## 2.5.2. Ideation Canvas



### PEOPLE TO WHOM IT WILL BE USEFUL:

- Students
- Application developers
- Businessman
- Manager
- Accountants
- Others



**ACTIVITIES:****• The activities involved in this project are as follows:**

1. Control traffic
2. Check the OpenStack connections.
3. Deny the access to invalid users.
4. Allow valid users to access the desired file entering the private dashboard.
5. Maintain the records of all clients.
6. Check the network topology.
7. Offer the mailing services to the clients.
8. Check for storage space for storing files.
9. Storing all versions of files of users.
10. Backup storage.
11. Synchronization of files.
12. Deploy application.

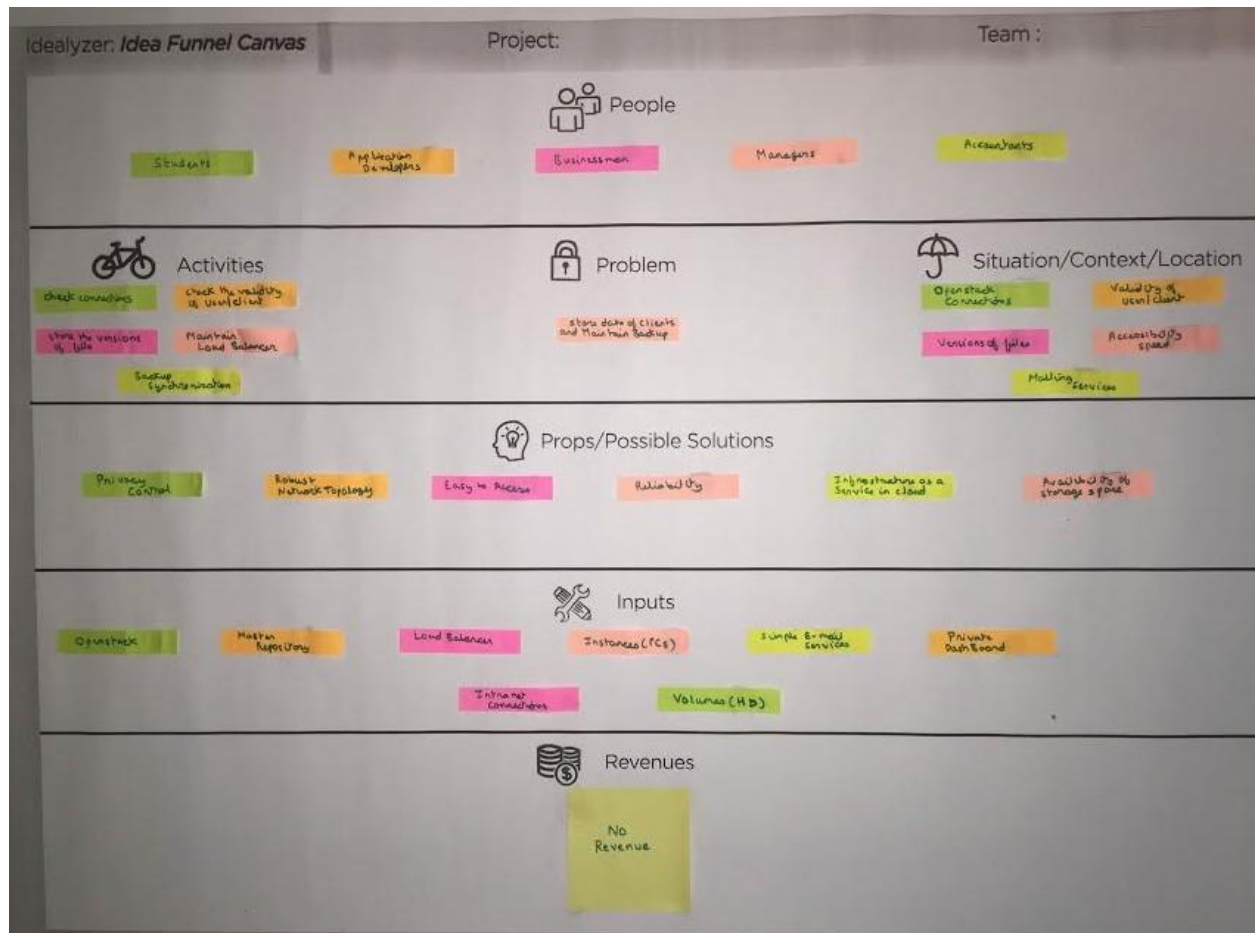
**SITUATIONS:****• We faced following situations:**

1. Valid Users or clients.
2. Versions of files to be stored.
3. Accessibility speed dues to high or low network traffic.
4. SES service for sending ID or PWD through E-mail.

**POSSIBLE SOLUTIONS:****• Following are the possible solutions to problems:**

1. Privacy Control.
2. Robust network topology.
3. Traffic control by Load balancer.
4. Easy to access mode.
5. Availability of large storage space.
6. Reliability.
7. Infrastructure as a Cloud service.

### 2.5.3. Idea Funnel Canvas



#### PEOPLE TO WHOM IT WILL BE USEFULL:

- Students
- Application developers
- Businessman
- Manager
- Accountants
- Others

## **ACTIVITIES:**

- **The activities involved in this project are as follows:**

- Control traffic
- Check the OpenStack connections.
- Deny the access to invalid users.
- Allow valid users to access the desired file entering the private dashboard.
- Maintain the records of all clients.

## **INPUTS:**

**Following are the inputs to be implemented in the system:**

1. OpenStack
2. Master repository
3. Load balancer
4. Instances (PCs)
5. Simple E-mail Services
6. Private dashboard
7. Internet connections
8. Volumes (Hard Drives)

## **MAIN PROBLEMS:**

Store, maintain and backup all versions of files of users.

## **REVENUE STRUCTURE:**

- **Till now we have not decided any revenue structure for our system.**

## 3. Implementation

### 3.1. Data Dictionary

#### Public Cloud – AWS

##### Account

Attributes	Data types	Description
acc_id	BIGINT	Auto-incremental field
Username	VARCHAR (30)	
Email	VARCHAR (200)	
client_key	BIGINT	Random numeric key for authenticity establishing
client_secret	VARCHAR (25)	25-digit alphanumeric code for authentication
total_storage_capacity	DOUBLE	In GB
storage_used	DOUBLE	In GB

##### Objects

Attributes	Data types	Description
file_id	BIGINT	Unique file id
Name	VARCHAR (300)	
root_path	Text	
created_time	BIGINT	
modified_time	BIGINT	
total_version_count	DOUBLE	
Permissions	CLOB	Who can access this file
public_url	TEXT	For sharing purpose
acc_id	BIGINT	From which it is belong

##### Public sync

Attributes	Data types	Description
_id1	BIGINT	User1_id
_id2	BIGINT	User2_id
last_sync	BIGINT	Timestamp of last sync

## Server at Private Cloud

### Clients

Attributes	Data types	Description
client_id	BIGINT	Registered client id in network
client_system_name	VARCHAR (300)	Host name of system
ip_address	VARCHAR (15)	
mac_address	BIGINT	
transmission_media	BIGINT	LAN of Wi-Fi
Is_authorized	BOOLEAN	For authentication purpose

### Versions

Attributes	Data types	Description
version_id	BIGINT	Global version id
file_id	BIGINT	File id which this subversion relates
version_number	INT	
physical_path	TEXT	Actual object path over cinder
created_time	BIGINT	File creation time-stamp

### Objects

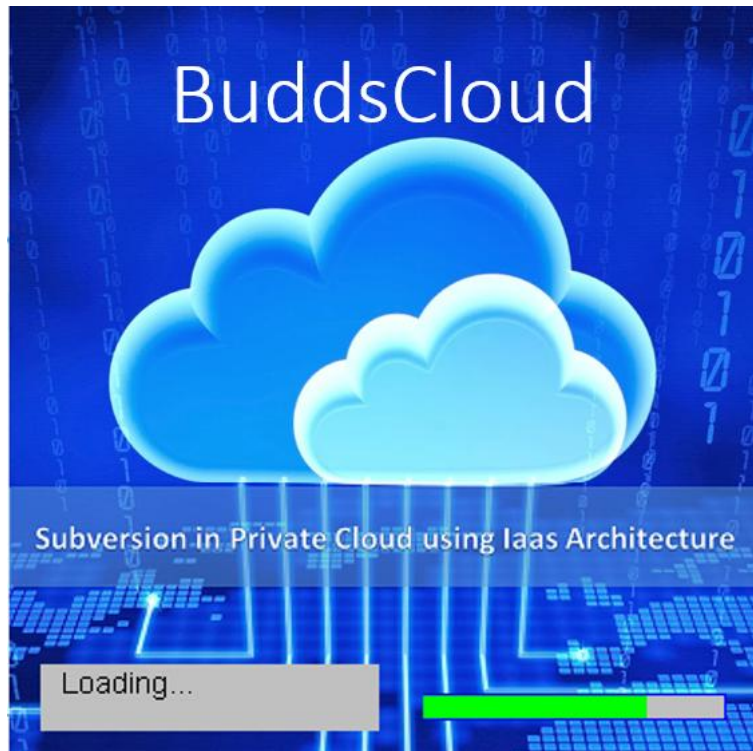
Attributes	Data types	Description
file_id	BIGINT	Global file id
file_name	VARCHAR (255)	
latest_version_number	INT	
physical_path	TEXT	Actual object path over cinder
Last_modified_time	BIGINT	File modification time-stamp

## Client at Private Cloud

### Object

Attributes	Data types	Description
file_id	BIGINT	Local file id
file_name	VARCHAR (255)	
root_path	INT	Path relative to BuddsCloud root folder
created_time	TEXT	Firstly created timestamp
modified_time	BIGINT	Latest modification time-stamp

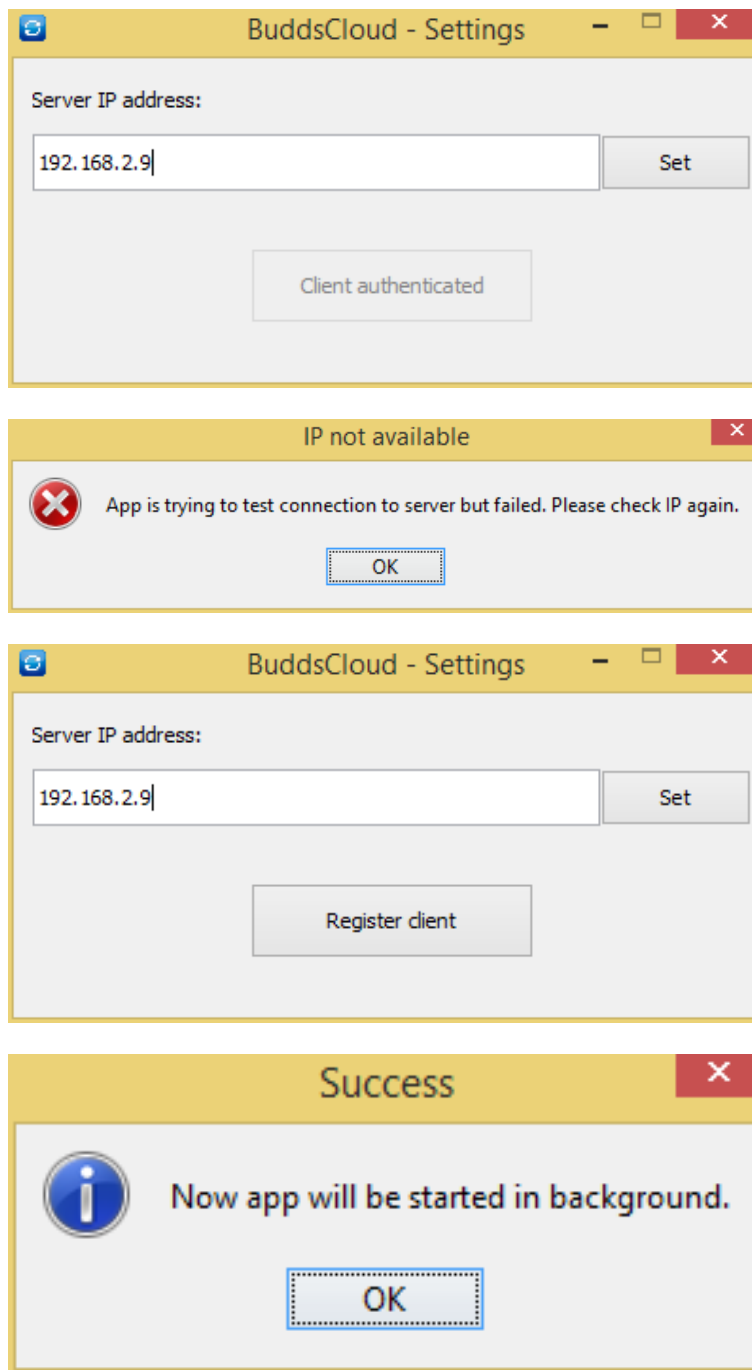
### 3.2. Screenshots



This is the welcome screen of the project. After completion of the loading a folder would open in your PC/Laptop name BuddsCloud.



This shows the logo of our projects application in system tray of windows/Linux



The above snapshots shows the authentication process of our system. If we entered a correct IP address of the system it would connect to the server otherwise the system will show an error message to the user.



```

dhruv@dhruv: ~/devstack
dhruv@dhruv:~$ cd devstack/
dhruv@dhruv:~/devstack$ ./unstack.sh
Site keystone disabled.
To activate the new configuration, you need to run:
  service apache2 reload
* Stopping web server apache2
*
* Starting web server apache2
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.0.1. Set the 'ServerName' directive globally to suppress this message
*
* Stopping web server apache2
*
tgt stop/waiting
tgtadm: failed to send request hdr to tgt daemon, Transport endpoint is not connected
tgt seems to be in a bad state, restarting...
stop: Unknown instance:
tgt start/running, process 5623
tgt stop/waiting
dhruv@dhruv:~/devstack$

```

```

dhruv@dhruv: ~/devstack
* Stopping web server apache2
*
tgt stop/waiting
tgtadm: failed to send request hdr to tgt daemon, Transport endpoint is not connected
tgt seems to be in a bad state, restarting...
stop: Unknown instance:
tgt start/running, process 12812
tgt stop/waiting
/opt/stack/data/stack-volumes-lvmdriver-1-backing-file: No such file or directory
dhruv@dhruv:~/devstack$ ./stack.sh
APT::Acquire::Retries "20";

#####
ENTER A PASSWORD TO USE FOR THE DATABASE.
#####
This value will be written to your localrc file so you don't have to enter it
again. Use only alphanumeric characters.
If you leave this blank, a random default value will be used.
Enter a password now:
aspirine
Using mysql database backend

#####
ENTER A PASSWORD TO USE FOR RABBIT.
#####
This value will be written to your localrc file so you don't have to enter it
again. Use only alphanumeric characters.
If you leave this blank, a random default value will be used.
Enter a password now:
aspirine

#####
ENTER A SERVICE TOKEN TO USE FOR THE SERVICE ADMIN TOKEN.
#####
This value will be written to your localrc file so you don't have to enter it
again. Use only alphanumeric characters.
If you leave this blank, a random default value will be used.
Enter a password now:
aspirine

#####
ENTER A SERVICE PASSWORD TO USE FOR THE SERVICE AUTHENTICATION.
#####
This value will be written to your localrc file so you don't have to enter it
again. Use only alphanumeric characters.
If you leave this blank, a random default value will be used.
Enter a password now:

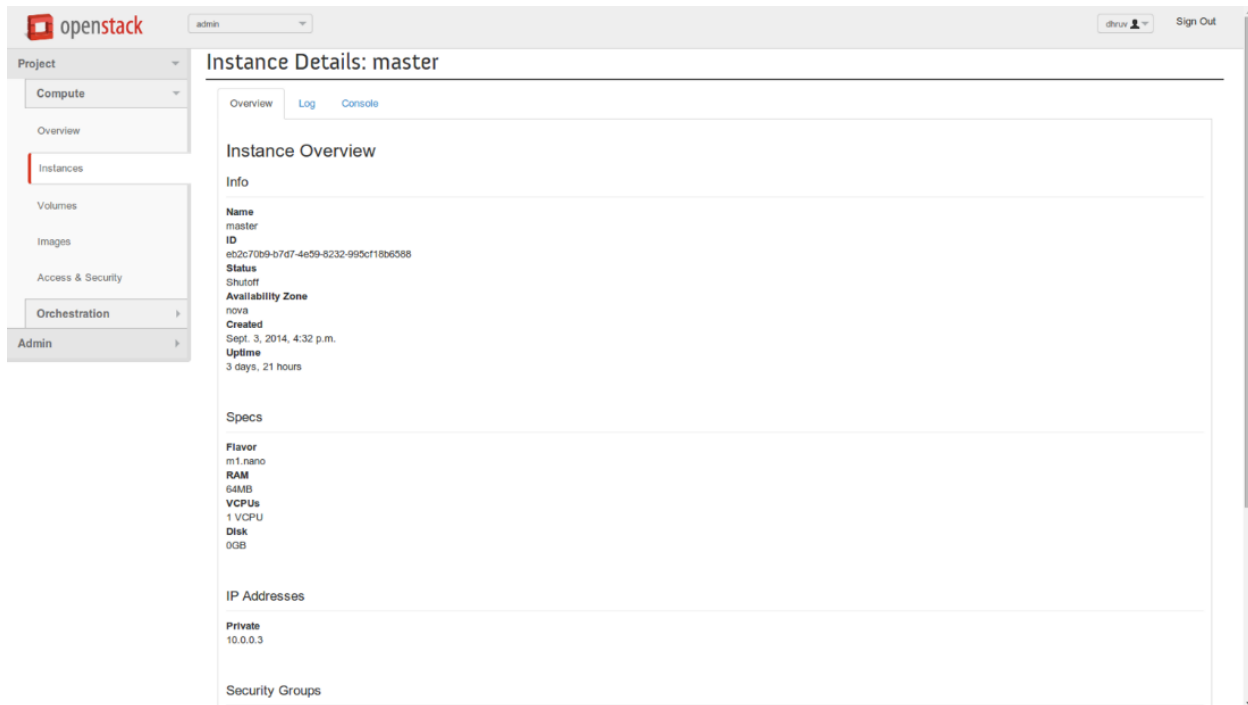
```

```

dhruv@dhruv: ~/devstack
python-anyjson
python-routes
python-xattr
python-sqlalchemy
python-webob
python-kombu
pylint
python-eventlet
python-nose
python-sphinx
python-mox
python-kombu
python-coverage
python-cherrypy3
python-migrate
libxslt1-dev
++ is ubuntu
++ [[ -z deb ]]
++ '[' deb = deb ']'
++ grep -q dkms
++ echo bridge-utils pylint python-setuptools screen unzip wget psmisc gcc git lsof openssh-server openssl python-virtualenv python-unittest2 iputils-ping wget curl tcpdump euca
ZooIs tar python-dev python2.7 bc libyaml-dev libffi-dev libssl-dev libxml2-dev python-eventlet python-routes python-greenlet python-sqlalchemy python-wsgiref python-pastedeploy
python-xattr python-iso8601 python-lxml python-pastescript python-pastedeploy python-paste sqlalchemy python-pysqlite2 python-sqlalchemy python-mysqldb python-webob python-greenle
t python-routes libldap2-dev libsass2-dev libkrb5-dev python-dateutil mspack-python fping dnsmasq-base dnsmasq-utils conntrack kpartx parted iputils-arping python-mysqldb pytho
n-xattr python-lxml gawk iptables ebtables sqlite3 sudo pm-utils libjs-jquery-tablesorter vlan curl genisoimage socat python-mox python-paste python-migrate python-greenlet pyth
on-libxml2 python-routes python-numpy python-pastedeploy python-eventlet python-cheetah python-tempita python-sqlalchemy python-suds python-lockfile python-m2crypto python-boto
python-kombu python-feedparser python-iso8601 lvm2 open-iscsi genisoimage sysfsutils sg3-utils python-numpy tgt lvm2 qemu-utils libpq-dev open-iscsi python-beautifulsoup python-
dateutil python-paste python-pastedeploy python-anyjson python-routes python-xattr python-sqlalchemy python-webob python-kombu pylint python-eventlet python-nose python-sphinx p
ython-mox python-kombu python-coverage python-cherrypy3 python-migrate libxslt1-dev
++ install package bridge-utils pylint python-setuptools screen unzip wget psmisc gcc git lsof openssh-server openssl python-virtualenv python-unittest2 iputils-ping wget curl t
cpdump eucaZooIs tar python-dev python2.7 bc libyaml-dev libffi-dev libssl-dev libxml2-dev python-eventlet python-routes python-greenlet python-sqlalchemy python-wsgiref python-
pastedeploy python-xattr python-iso8601 python-lxml python-pastescript python-pastedeploy python-paste sqlalchemy python-pysqlite2 python-sqlalchemy python-mysqldb python-webob pyt
hon-greenlet python-routes libldap2-dev libsass2-dev libkrb5-dev python-dateutil mspack-python fping dnsmasq-base dnsmasq-utils conntrack kpartx parted iputils-arping python-my
sqldb python-xattr python-lxml gawk iptables ebtables sqlite3 sudo pm-utils libjs-jquery-tablesorter vlan curl genisoimage socat python-mox python-paste python-migrate python-gr
eenlet python-libxml2 python-routes python-numpy python-pastedeploy python-eventlet python-cheetah python-tempita python-sqlalchemy python-suds python-lockfile python-m2crypto p
ython-boto python-kombu python-feedparser python-iso8601 lvm2 open-iscsi genisoimage sysfsutils sg3-utils python-numpy tgt lvm2 qemu-utils libpq-dev open-iscsi python-beautifuls
oup python-dateutil python-paste python-pastedeploy python-anyjson python-routes python-xattr python-sqlalchemy python-webob python-kombu pylint python-eventlet python-nose pyth
on-sphinx python-mox python-kombu python-coverage python-cherrypy3 python-migrate libxslt1-dev
++ update_package_repo
++ [[ '' = \N\U\l\e ]]
++ is ubuntu
++ [[ -z deb ]]
++ '[' deb = deb ']'
+++ set +o
+++ grep xtrace
++ local 'xtrace=set -o xtrace'
++ set +o xtrace
0% [Connecting to archive.ubuntu.com] [Connecting to archive.canonical.com] [Co

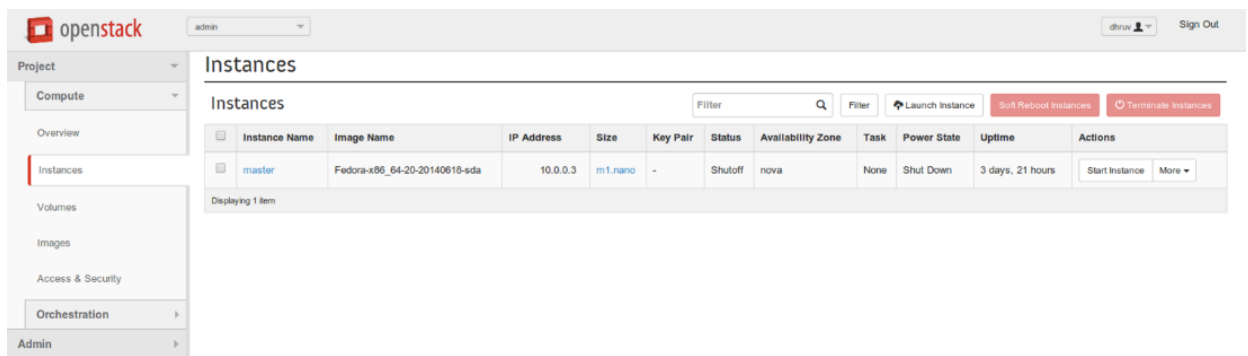
```

## Git and OpenStack Installation process



The screenshot shows the OpenStack dashboard interface for the 'master' instance. The left sidebar contains navigation links: Project, Compute, Overview, Instances (selected), Volumes, Images, Access & Security, Orchestration, and Admin. The main content area is titled 'Instance Details: master' and includes tabs for Overview, Log, and Console. The 'Overview' tab is active, displaying the following information:

- Info**
  - Name: master
  - ID: eb2c70b9-b7d7-4e59-8232-995cf18b6588
  - Status: Shutoff
  - Availability Zone: nova
  - Created: Sept. 3, 2014, 4:32 p.m.
  - Uptime: 3 days, 21 hours
- Specs**
  - Flavor: m1.nano
  - RAM: 64MB
  - VCPUs: 1 VCPU
  - Disk: 0GB
- IP Addresses**
  - Private: 10.0.0.3
- Security Groups**

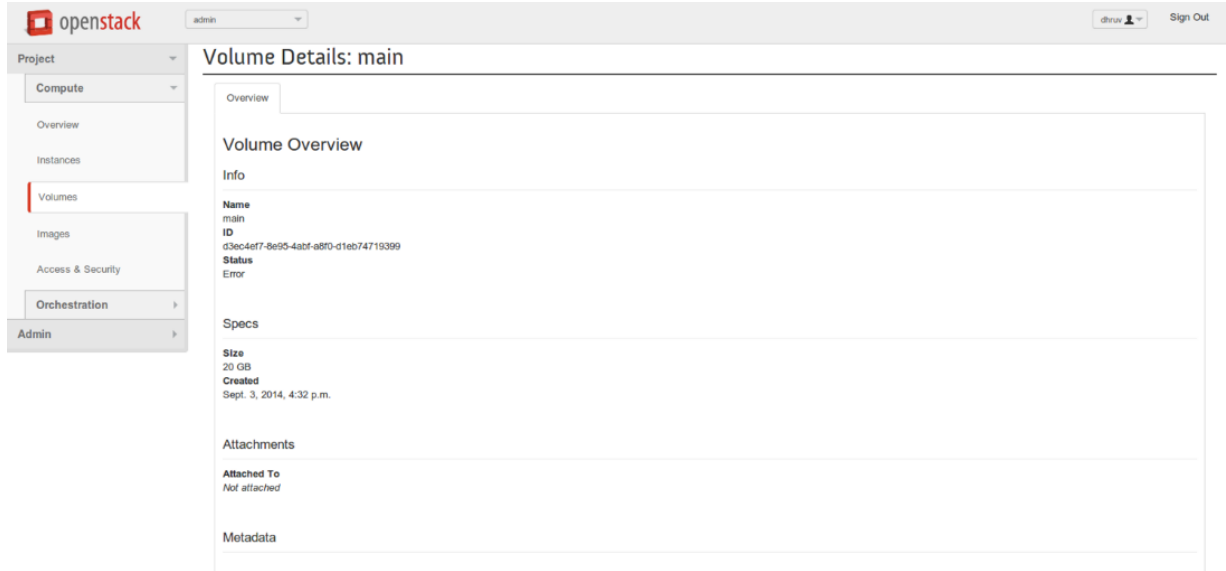
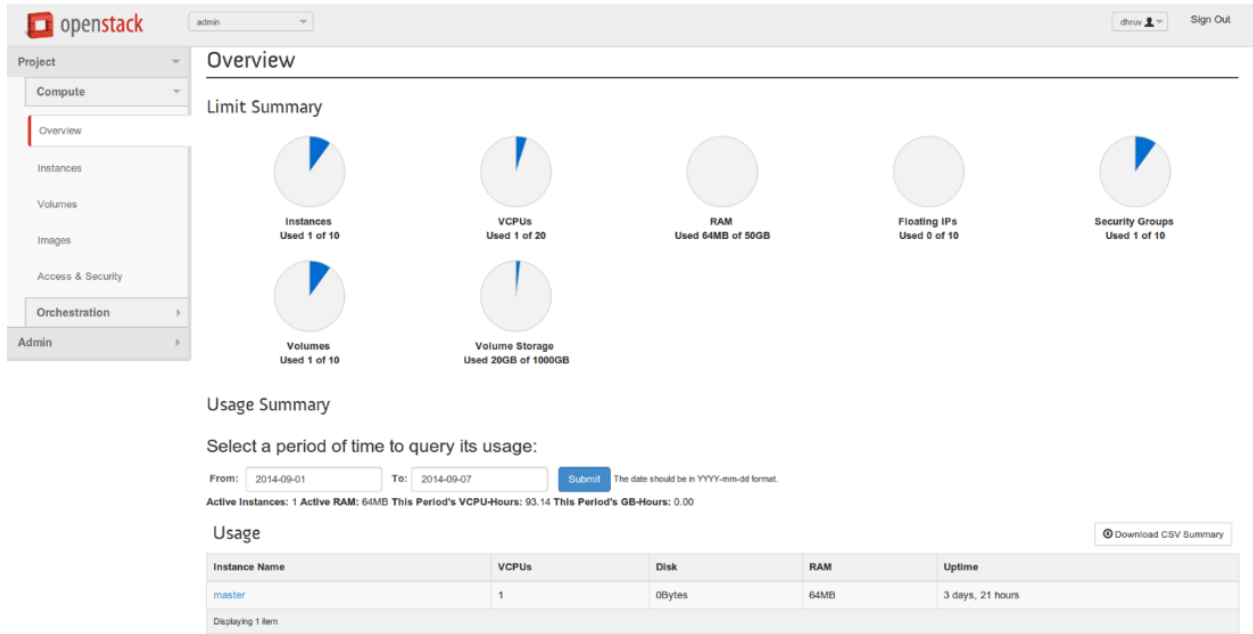


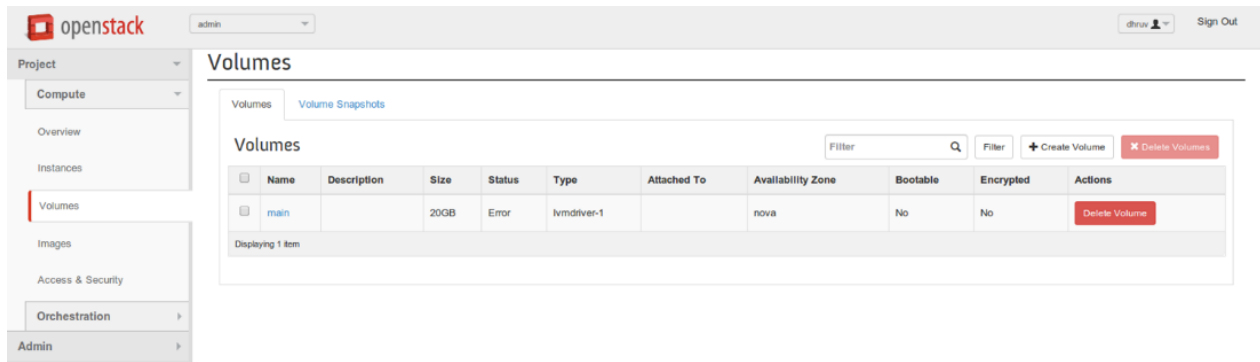
The screenshot shows the OpenStack dashboard interface for the 'Instances' page. The left sidebar contains navigation links: Project, Compute, Overview, Instances (selected), Volumes, Images, Access & Security, Orchestration, and Admin. The main content area is titled 'Instances' and includes a search bar, a filter dropdown, and buttons for 'Launch Instance', 'Soft Reboot Instances', and 'Terminate Instances'. Below the buttons is a table listing the instances:

Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Uptime	Actions
master	Fedora-x86_64-20-20140618-sda	10.0.0.3	m1.nano	-	Shutoff	nova	None	Shut Down	3 days, 21 hours	Start Instance More

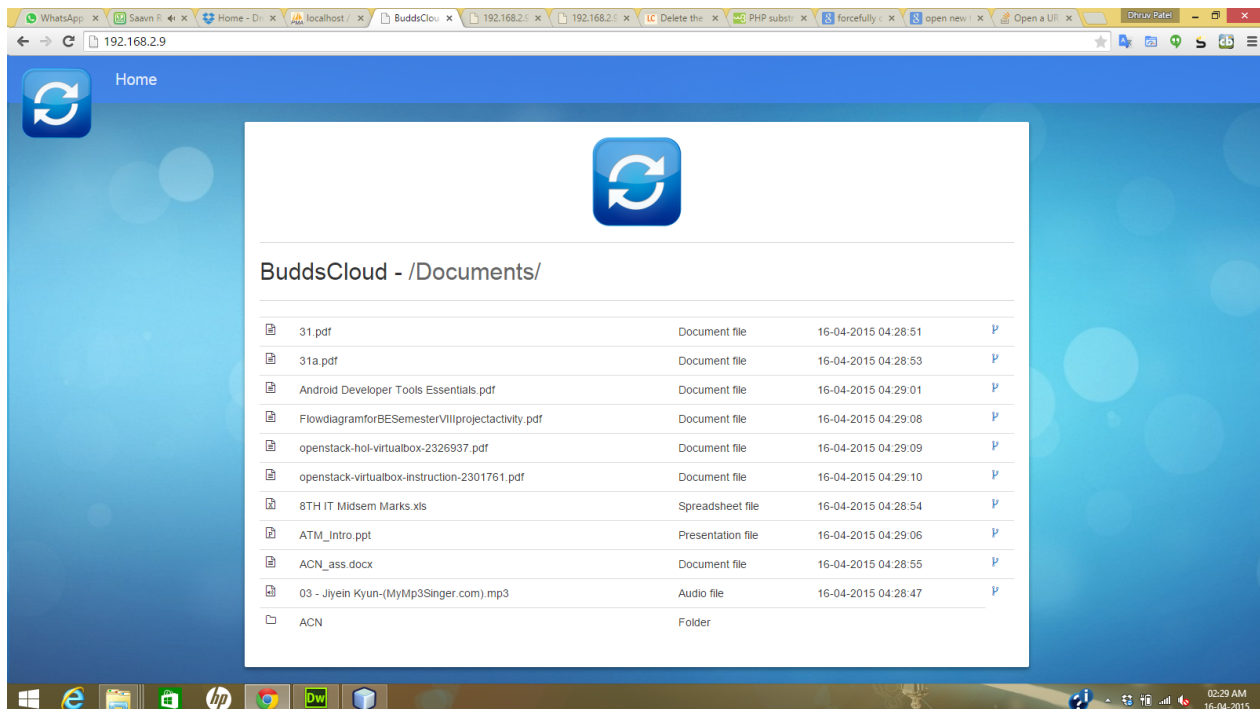
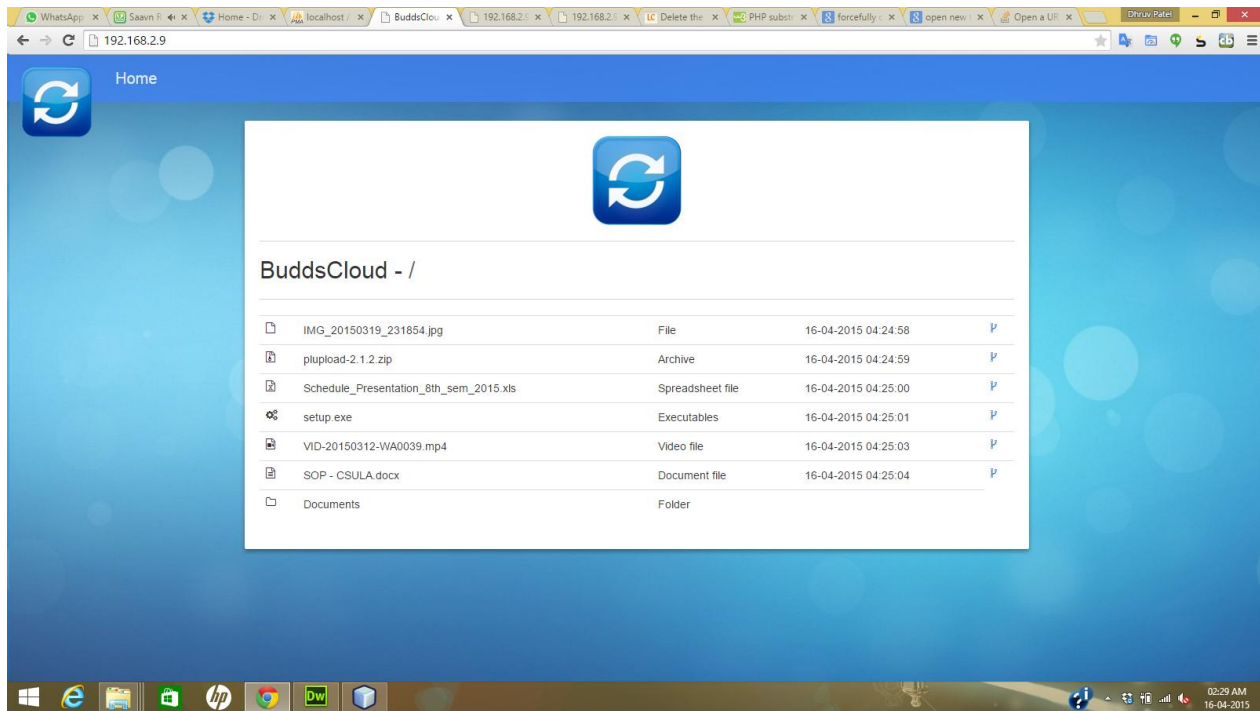
Displaying 1 item

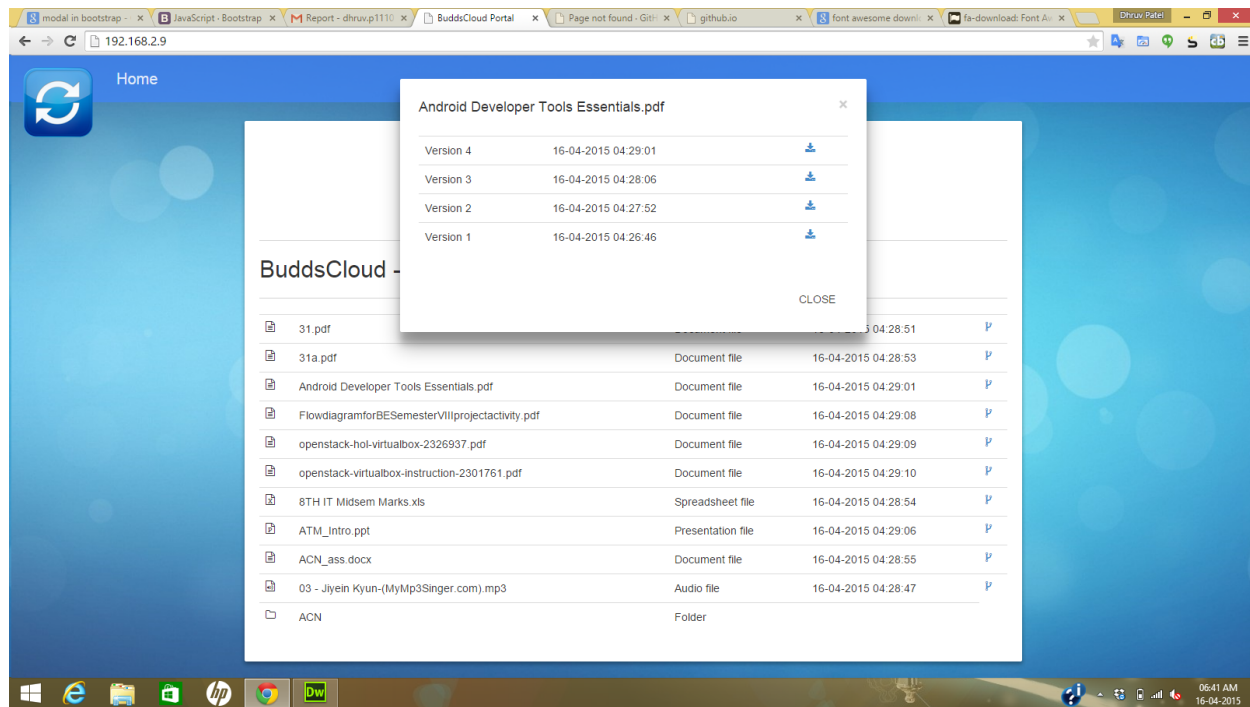
localhost/project/instances/





The above snapshots are the successful testing on an OpenStack. Once you install OpenStack cloud in your system the welcome page would see in the browser. After entering username and password user could able to create instances and keys. Here we have created only one Instance. To create more instances we need more RAM and space. To use basic functionality of OpenStack we need at least 2 GB of RAM.





After successful connection of our application we could use subversion feature of this app. Whenever we make change in our existing file or create a new file we would get a notification. The above snapshots shows the Web portal of our application. Here the new update file would see in the screen and if we want to know how many other version of that file has created we need to click on the version to see other older version of the file.

### 3.3. System Testing

Development of a complex client/server three tier application requires that a methodology be developed for more effective application or software testing and quality assurance. Testing is made to find errors in the application. The strategy adopted for the testing in our application.

#### 3.3.1. Test cases

- *“Testing cannot show the absence of defect. It can only show that software errors are present.”*
- Testing is the process of executing a program with the explicit intention of finding errors that is, making the program fail. Testing is very crucial and most expensive phase of the software development. Before delivering the system, the process of rigorous testing is done to check that software works as it is expected and meets its specifications. For that two testing strategies are there Code Testing & Specification Testing. We have used both of them at different levels of code development.

#### 3.3.2. Testing (White Box Testing)

- The code-testing strategy examines the logic of the program. To follow this testing method, test cases should be developed that result in executing every instruction in the program or module; that is, every path through the program is tested. A path is a specific combination of conditions that is handled by the program.
- This testing is used at initial stage of the development, as code volume is very less at this stage. It checks only the aspects are implemented correctly or not. But this strategy does not indicate the code meets its specifications nor does it determine whether all aspects are even implemented. So with this, another strategy is also used.



### 3.3.3. Specification Testing (Black Box Testing)

- In this strategy, the specifications stating what the program should do and how it should perform under various conditions are examined. Test cases are developed for each condition or combination of conditions
- The analyst does not look into the program to study the code and is not concerned about whether every instruction or path through the program is tested. This is more efficient method, since it focuses on the way software is expected to be used.

### 3.3.4. Testing methods

Different types of testing method are used

- Unit Testing
  - In it analyst tests the program making up a system. The software units in a system are the modules and routines that are assembled and integrated to perform a specific function.
  - It focuses on modules, independently of one another, to locate errors. This enables the tester to detect errors in coding and logic that are contained within the module alone.
- Bottom-Up Unit Testing
  - It can be performed from the bottom up, starting with the smallest and lowest-level modules and proceeding one at a time. For each module in bottom-up testing, a short program executes the module and provides the needed data, so that the module is asked to perform the way it will when embedded within the larger system.
- Top-Down Unit Testing
  - As the name implies, begins with the upper-level modules. However, since the detailed activities usually performed in lower-level routines are not provided, stubs are written. A stub is a module that can be called by the upper-level module and that, when reached properly, will return a message to the calling module, indicating a proper interaction occurred.
- System Testing

- System testing does not test the software per se but rather the integration of each module in the system. It also tests to find discrepancies between the system and its original objective, current specifications and system documentation.
- The primary concern is the compatibility of individual modules. Analysts are trying to find areas where modules have been designed with different specifications.
- Test cases are designed to test the system and according to the submitted test cases test data are determined and then his system is tested according to different test objectives.

## 4. Conclusion

### 4.1. Limitations

- The limitation of this system is every time an organization wants to establish this system in their premises there has to be a person who can have a knowledge about OpenStack and how to establish a connection.
- Non-Technical person cannot install this system by their own.
- For transfer, modification and storage of different versions of files in between two different premises, Internet connection becomes mandatory.

### 4.2. Future Enhancement

- We are planning to use this system as commercial product.
- In future we are also planning to create a Simple GUI package. Using this GUI package a Non-technical or a person who doesn't have enough knowledge about cloud can easily implement and establish a subversion in private cloud.
- We would create an android application for our project. Where person can send, update, delete and share data to other authenticated users using this system.

### 4.3. Conclusion

By the use of our system, organization can implement sub-versioning feature without internet connection with:

- High reliability
- Scalable Performance
- Faster Synchronization

## References

1. <http://aws.amazon.com/>
2. <http://www.linux.com/>
3. <http://linuxconfig.org/>
4. <http://www.openstack.org/>
5. <http://stackoverflow.com/>
6. <http://www.tutorialspoint.com/>
7. <http://docs.openstack.org/developer/devstack/>
8. <http://www.ubuntu.com/cloud>
9. [http://en.wikipedia.org/wiki/Git\\_\(software\)](http://en.wikipedia.org/wiki/Git_(software))
10. [http://en.wikipedia.org/wiki/Cloud\\_computing](http://en.wikipedia.org/wiki/Cloud_computing)