

# A RESTful Approach to the Management of a Cloud Infrastructure

A web portal for Cloud Infrastructure Management and Monitoring

Siddharth Sujir Mohan

School of Computing  
Informatics and Decision

Systems Engineering

[siddharthsujir.mohan@asu.edu](mailto:siddharthsujir.mohan@asu.edu)

Suraj Rao

School of Computing  
Informatics and Decision

Systems Engineering

[suraj.s.rao@asu.edu](mailto:suraj.s.rao@asu.edu)

Vageesh Bhasin

School of Computing  
Informatics and Decision

Systems Engineering

[vageesh.bhasin@asu.edu](mailto:vageesh.bhasin@asu.edu)

**Abstract** — The setup and management of a cloud based infrastructure is a challenging process. It requires a degree of expertise to understand and deploy a cloud infrastructure and its services. The tasks involved include creating or deleting a service instance, enabling or disabling a service, monitoring of resources available and used by services, and generating service usage reports. This project aims at providing a user friendly web based portal to handle these tasks. The current scope of the project is providing this portal for a DevStack cloud platform.

**Keywords**— *OpenStack, DevStack, Cloud Platform, Cloud Services, Web application, OpenStack API, RESTful API, Node.js, Express.js*

## I. INTRODUCTION

- a. The problems to be addressed in this project:
  - a. Cloud Infrastructure Management: Deploying and withdrawing of Cloud-Service instances, Enabling and Disabling of services
  - b. Cloud Infrastructure Monitoring: Resource-usage statistics and reporting of cloud services

- b. Why these problems are important?

Deployment of a cloud service is a very time-consuming problem that expects a high level of understanding of the cloud environment. It involves setting up configuration files and executing setup scripts on the host machine. Monitoring of available resources is another area that requires addressing due to the fact that the service instances have a limited resource pool.

- c. Applied technologies and solutions to address these problems:
  - a. Cloud Platform: DevStack/OpenStack
  - b. Web Application: Node.js as Server, Express.js as Web Application framework, JavaScript for user interaction, RESTful API for interfacing
- d. Expected outcomes of this projects.

A web application that provides tools for management and monitoring of OpenStack/DevStack cloud platform

- e. Project management plan (timeline, and group members, etc.)

This project needs to be completed in one month. The main tasks include setup of a cloud environment, developing the web application as well as testing and documentation of the complete system. This will be handled by Siddharth Sujir Mohan, Suraj Rao and Vageesh Bhasin.

## II. SYSTEM MODELS

### A. System Model

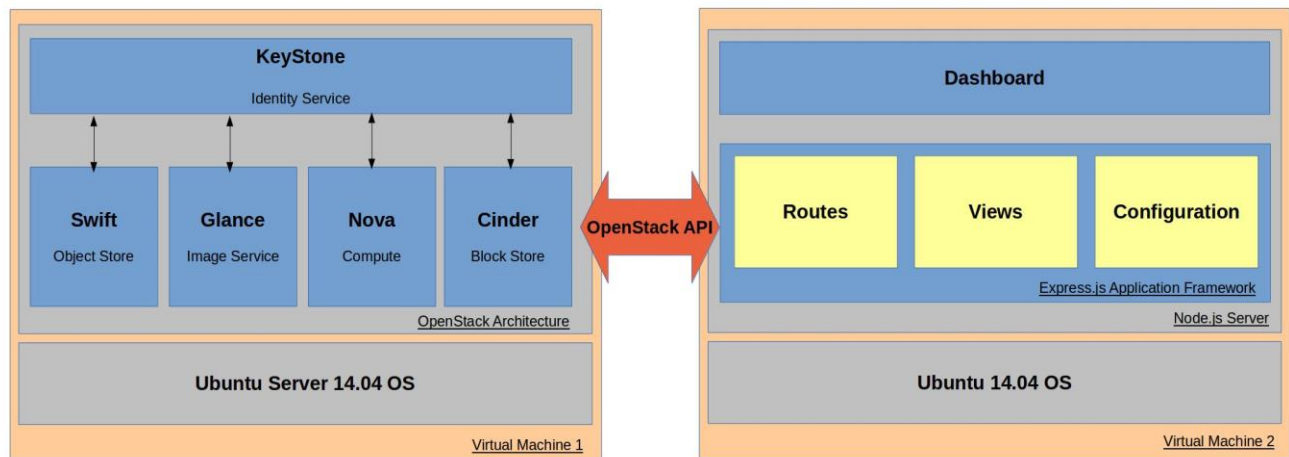
The system can be divided into two major components:

#### a. Cloud Environment:

This is an DevStack based cloud infrastructure running on a host OS(Ubuntu Server) in a virtual machine. The cloud services provided by this system will include Computing (Nova), Identity Service(Keystone), Image Service(Glance), Object Storage(Swift) and Block Storage(Cinder). The host machine (DOM0) will be communicating with the application using the DevStack/OpenStack API.

#### b. Web Application:

The web application will be built using Express.js Application framework and would be running on a Node.js server. The front-end of the application will be developed using HTML for markup, JavaScript for User interaction and CSS for styling. The whole application environment would be hosted on another virtual machine.



*Project System Model*

### B. Software

- Applications/Platform: Oracle VM VirtualBox, DevStack/OpenStack Cloud Platform, Ubuntu Server OS 14.04, Node.js
- Framework: Express.js
- API: DevStack/OpenStack API
- IDEs: Text Editor (Sublime Text)

## III. PROJECT DESCRIPTION

### A. Project Overview

The project aims at allowing users to manage and monitor various cloud services provided by the cloud platform. The project integrates the ability to manage and monitor cloud services with a user-friendly web portal allowing the user to perform the management and monitoring tasks on any DevStack/OpenStack cloud platform. A user will be able to login to a cloud infrastructure built on DevStack/OpenStack platform via the web. Once user authentication is successful, the user can then deploy and withdraw cloud services like Computing (Nova), Identity Service(Keystone), Image Service(Glance), Object Storage(Swift) and Block Storage(Cinder). The user will also be able to monitor the resource usage of the services as well as generate reports of resource utilization.

### *B. Task 1: Setting up of the Cloud Infrastructure*

In this task, a Virtual Machine will be configured to run Ubuntu Server 14.04 OS as the Host OS. Once the bare environment is set up, a 'localrc' file holding the configurations and variables will be created. After this, the DevStack platform will be set up using the earlier created config file. Lastly, once the cloud is up and running, all the enabled services will be tested for activation.

### *C. Task 2: Web-based Application*

In this task, a web-based application will be developed using Express.js application framework, which will be hosted on a Node.js server. The front-end of the application will be written using HTML, CSS and JavaScript. The application will be configured to connect to a local database for storing user activity and statistic reports. The application will allow user to create, delete and manage service instances. The user will also be able to see resource utilization of the running cloud services as well as generating usage reports.

### *D. Task 3: Integration of Web Application with Cloud Platform*

In this task, a network would be setup between the virtual machine hosting the web application and the virtual machine hosting the cloud platform. Also, web server methods will be written to allow communication between the web application and the cloud platform using the provided OpenStack/DevStack API. Lastly, the data provided by the web application via the API will be tested against the data provided by the cloud platform.

### *E. Project Task Allocation*

Task	Sub-Task	Team Member	Work Percentage
Cloud	Environment Setup	Siddharth/Suraj	7.5
	DevStack Configuration	Suraj/Vageesh	10
	DevStack Deployment	Suraj/Vageesh	10
	Services Testing	Vageesh/Siddharth	7.5
Web Application	Environment Setup	Suraj/Vageesh	5
	Server-Side Development	Vageesh/Siddharth	10
	Client-Side Development	Siddharth/Suraj	10
	Smoke Testing	Suraj/Vageesh	5
	Application Deployment	Vageesh/Siddharth	5
API Integration	Network Setup	Suraj/Siddharth	5
	Server-Side Methods Development	Vageesh/Siddharth	10
	Integration Testing	Siddharth/Suraj	5
Documentation	Reports	All	2.5
	Power-point Presentation	All	2.5
	Demo	All	5

### *F. Deliverables*

It includes source code, supporting documents, usage manual, along with:

- Running Cloud Platform
- Web Application

### *G. Mid Term Goals*

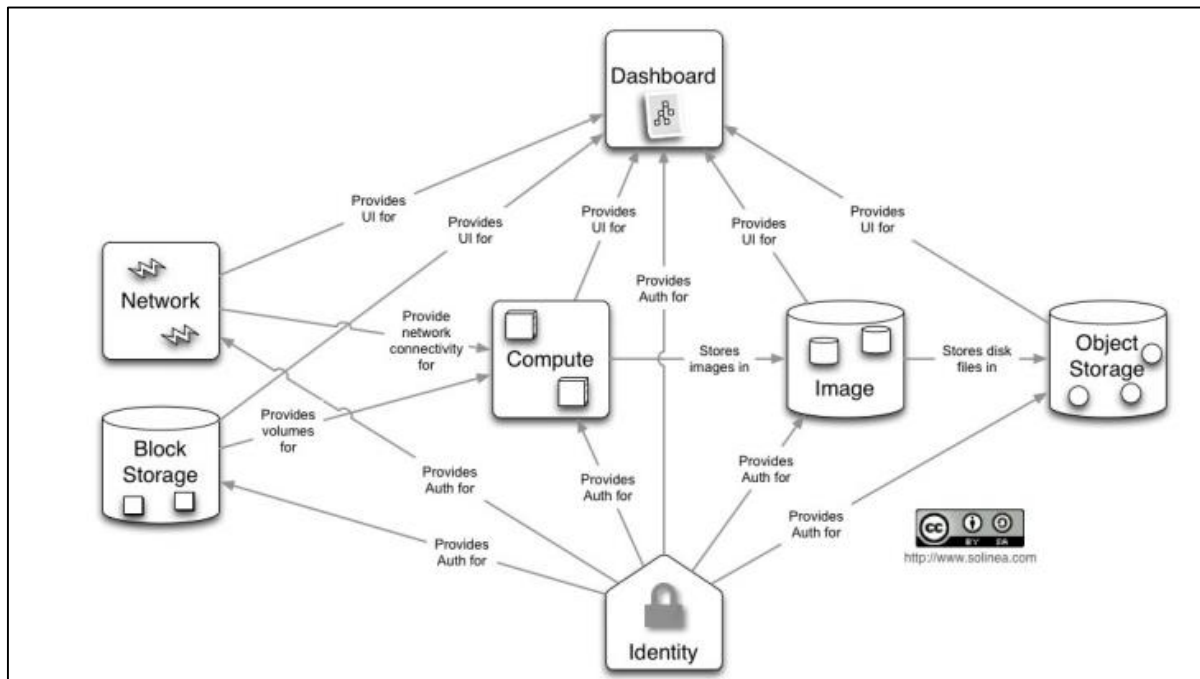
- Setting up and configuration of the Dev Stack environment on the virtual machine.
- Setting up the application environment on the local machine.

- Dev Stack Deployment and configuration on the virtual machine.

### *OpenStack/DevStack*

OpenStack is a cloud operating system that manages large pools of compute, storage and networking resource throughout a data center. OpenStack is a cloud computing project released under the terms of Apache License that provides Infrastructure as a Service (IaaS) to users. [10] DevStack is OpenStack for developers, but is not meant to be a production-ready OpenStack installer.

OpenStack has a modular architecture and has several shared services that span the three pillars of compute, storage and networking. These services- including identity, image management, and web interface- integrate the OpenStack environment components with each other as well as the external systems to provide unified for users.[10]



*Figure 1: High-level view of OpenStack Architecture[10]*

### *Compute (Nova):*

OpenStack provides the compute services through Nova, by provisioning and managing large network of virtual machines. It is designed to manage and automate large pool of computer resources and can work with wide range of virtualization technologies, bare metals and high performance computing (HPC) configurations. KVM and XenServers are the available choices for hypervisor technologies. [10] Nova is built on a shared-nothing, messaging-based architecture. All of the major nova components can be run on multiple servers. This means that most component to component communication must go via message queue. [20] Nova uses RabbitMQ for messaging, which is open source message broker software that implements the Advanced Message Queuing Protocol (AMQP). [20]

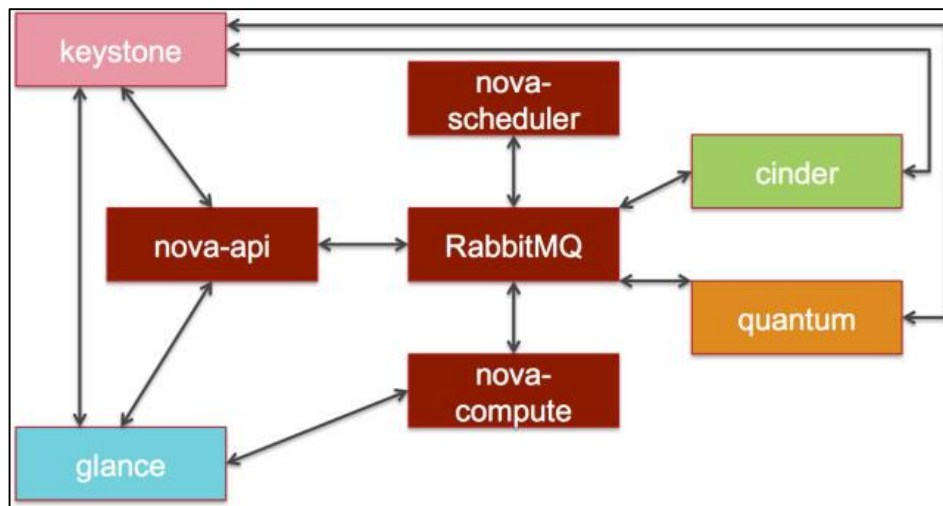


Figure 2 : OpenStack Nova Architecture[20]

### Object Storage (Swift):

The storage service is provided by OpenStack Swift. Swift is a highly available, distributed, eventually consistent object/blob store. [18] It provides APIs to write objects and files to multiple disks on a server spread across the datacenter. In case of hard drive failure, Swift replicates its contents from active nodes to new locations in the cluster. [10] It allows you to store or retrieve files (but not mount directories like a fileserver). Swift enables users to store, retrieve, and delete objects (with their associated metadata) in containers via a RESTful HTTP API. Swift can be accessed with HTTP requests directly to the API or by using one of the many Swift client libraries such as Java, Python, Ruby, or JavaScript. This makes it ideal as a primary storage system for data that needs to be stored and accessed via web based clients, devices and applications. [19]

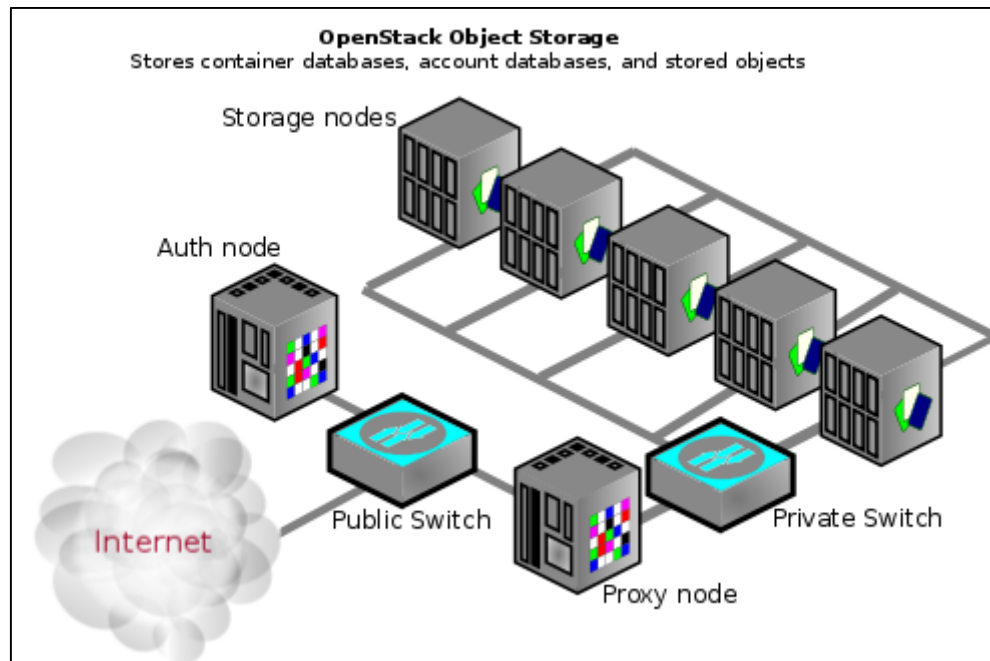
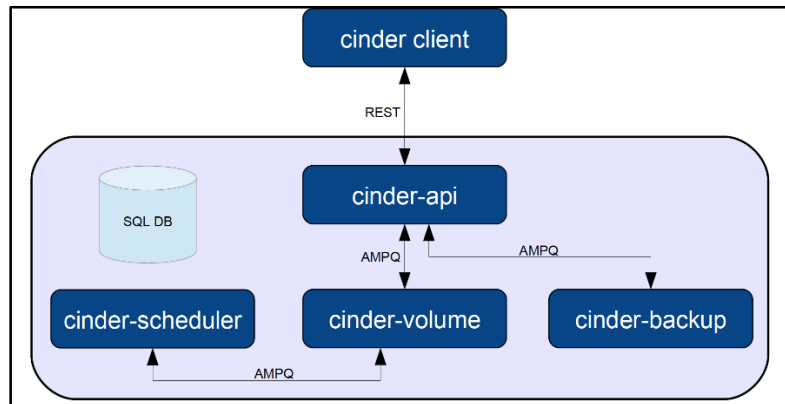


Figure 3: OpenStack Swift - Object Storage Architecture [19]

*Block Storage (Cinder):*

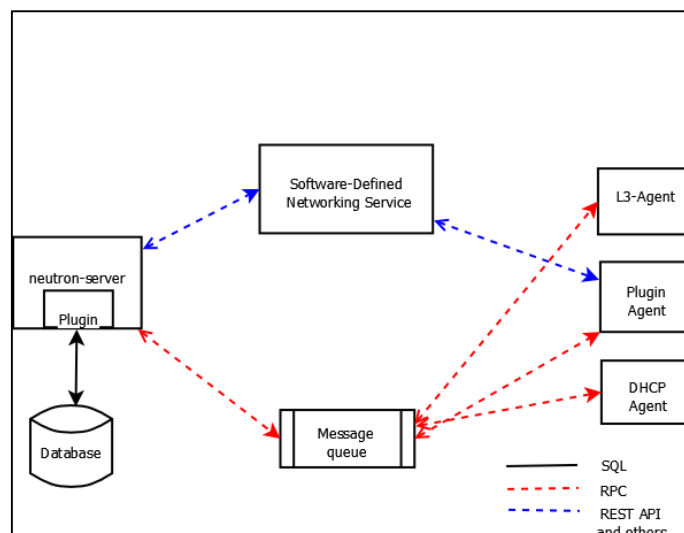
OpenStack Block Storage (Cinder) provides persistent block level storage services for use with OpenStack Compute instances. It manages attaching, detaching and creation of the block storage devices to servers. It is fully integrated with OpenStack Compute and Dashboards so that users can manage their own needs. Snapshot management provides powerful functionality to back up data stored on block storage volumes. Snapshots can be used to restore lost data in case of failure and also create a new block storage volume. [10] It virtualizes pools of block storage devices and provides end users with a self-service API to request and consume those resources without requiring any knowledge of where their storage is actually deployed or on what type of device. [21]



*Figure 4 : OpenStack Cinder Architecture [21]*

*Networking (Neutron):*

OpenStack networking (Neutron) is a pluggable, scalable and API-driven system for managing networks and IP addresses. Like the other components of OpenStack it lets the administrators and users to manage the existing data center assets. It provides networking modes for different application group or user group. It uses standard models like flat network or VLANs for separation of servers and traffic. Administrators can make use of Software Defined Networking (SDN) technology to allow high levels of multi-tenancy and massive scale. [10]



*Figure 5: Flow diagram of OpenStack Networking Components [22]*

### Identity (Keystone):

The users get to access the OpenStack resource with the help of identity service provided by Keystone. The users are mapped to a common authentication system across the cloud operating system and can integrate with the existing backend resources such as LDAP. It supports multiple form of authentication such as username and password credentials, token-based authentication, and Amazon web service log in credentials. [10]

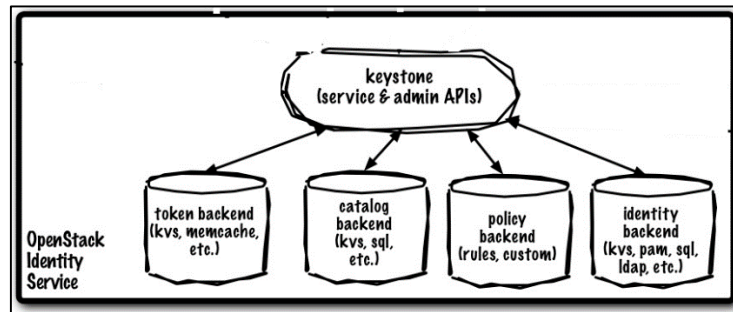


Figure 6 : Identity Architecture [25]

### Image Service (Glance):

OpenStack provides provision for discovery, registration and delivery services for disk and server images through Image Service (Glance). The image service can store disk and server images on variety of back end storage including OpenStack object storage. The Image service API provides a standard REST service for querying information about disk images and lets clients stream the images to new servers. [10]

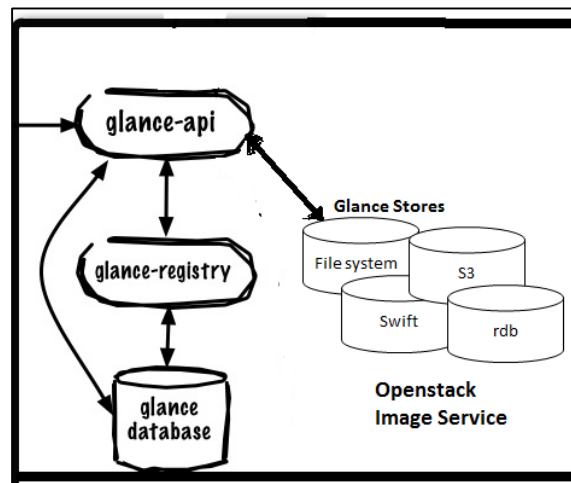


Figure 7: OpenStack Glance Architecture[24]

### OpenStack Setup and Configuration:

- We have installed DevStack on Ubuntu 14.04 installed in a virtual machine. The Virtual machine instance is created using oracle virtual box. The virtual machine is given an extendable disk space of 20GB along with 2GB RAM.
- DevStack is downloaded from Git repository:  
<https://github.com/openstack-dev/devstack.git>  
Note: We have used the Havana branch which is the current stable version.
- The *localrc* file is then created to provide the configuration details and passwords for the DevStack setup.



- Sample *localrc* file can be created as follows:  
*DATABASE\_PASSWORD=password*  
*RABBIT\_PASSWORD=password*  
*SERVICE\_TOKEN=password*  
*SERVICE\_PASSWORD=password*  
*ADMIN\_PASSWORD=password*  
*FLAT\_INTERFACE=eth0*  
*LOGFILE=/home/stack/stack.sh.log*

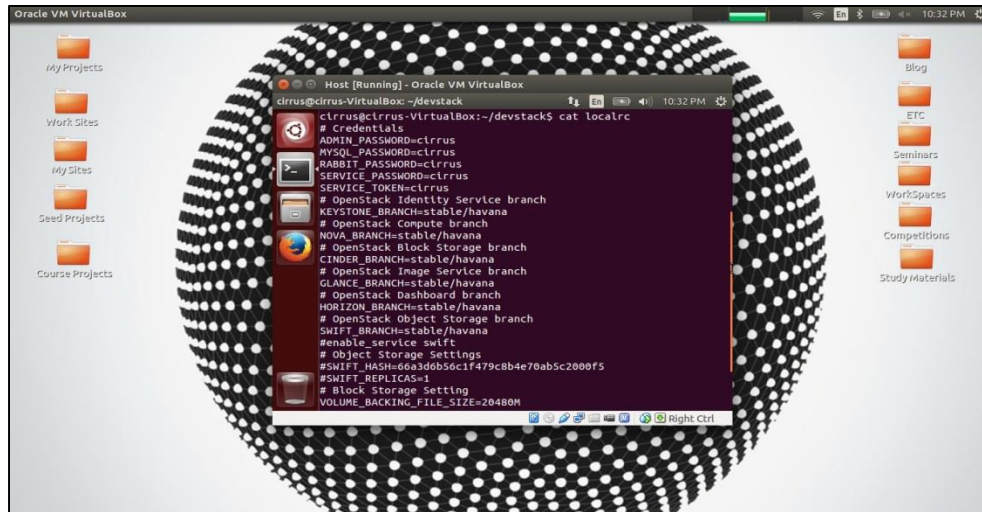


Figure 8 : Snapshot of *localrc* file for DevStack configuration

- DevStack provides an installation script file: *stack.sh*. This script installs DevStack and all of its services like Computing (Nova), Identity Service (Keystone), Image Service (Glance), Object Storage (Swift) and Block Storage (Cinder) and also all of the dependencies. The script uses the configuration details provided in *localrc* file.
- Network Setup:  
 The DevStack running in the virtual host needs to be accessible from the application server running in the host machine. The Oracle VirtualBox is configured with a host-only adapter and an IP address is assigned to it.

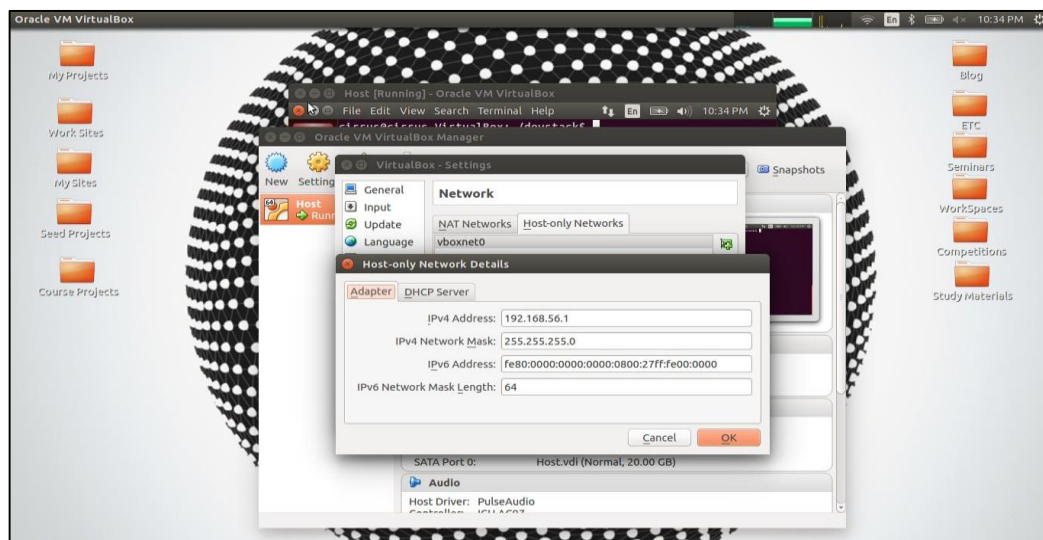


Figure 9 : Setting up network connection to host machine



- This adapter allows the virtual machine to communicate with the host and also any other guest virtual machines running in the host.
- The host-only network adapter is then added to the virtual machine along with the existing NAT adapter. The network interface is then configured in the ubuntu virtual machine for pointing to this adapter. The application server will then use this setup to communicate with the devstack configured in the vm.

#### *Node.js:*

Node.js is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications.

Node.js uses an asynchronous, non-blocking, event-driven I/O to be able to perform as a lightweight and efficient solution for data-intensive real-time applications that run across distributed devices. [11] A simple introduction to asynchronous programming on Node.js can be found at: [12].

Node.js has a built-in support for package management using NPM, or Node Package Manager. It is a set of publicly available, reusable components, available online through easy installation via an online repository, with version and dependency management. An introduction to using npm for downloading, updating, uninstalling and creating node packages can be found at: [13].

#### *Express.js:*

Express is a minimal and flexible node.js web application framework, providing a robust set of features for building single and multi-page, and hybrid web applications.[14] Express can be broken down into three layers of abstraction, the bottom layer – Node HTTP Server, serves as the starting point for making a webserver. The middle layer – Middleware, allows a developer to add functions to deal with the requests. A server can have a stack of middleware associated with it. When a request comes in, it is passed off to the first middleware function, along with a wrapped response object and a next callback. [15] The last layer is the routing layer, which allows mapping of a server request to a function handler.

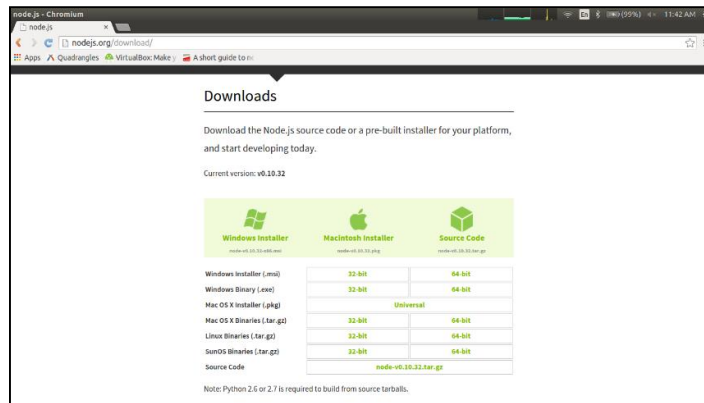
#### *RESTful Services:*

Representational State Transfer (REST) is an architectural style that specifies constraints, such as the uniform interface. In the REST architectural style, data and functionality are considered resources and are accessed using Uniform Resource Identifiers (URIs), typically links on the Web. The REST architectural style constrains an architecture to a client/server architecture and is designed to use a stateless communication protocol, typically HTTP. In the REST architecture style, clients and servers exchange representations of resources by using a standardized interface and protocol. [16] A concrete implementation of a REST Web service follows four basic design principles:[17]

- Use HTTP methods explicitly - The basic REST design principle establishes a one-to-one mapping between create, read, update, and delete (CRUD) operations and HTTP methods.[17]
- Be stateless - A REST Web service application (or client) includes within the HTTP headers and body of a request all of the parameters, context, and data needed by the server-side component to generate a response.[17]
- Expose directory structure-like URIs - The structure of a URI should be straightforward, predictable, and easily understood.[17]
- Transfer XML, JavaScript Object Notation (JSON), or both

#### *Application Environment Setup:*

- Download *node.js* for your OS from <http://nodejs.org/download/>  
*Note: Python 2.6 or more is required to build from source tarballs.*



- Verify successful installation by running:  
`$ node --version`

```
vageesh@vageesh-Inspiron-7520: ~  
vageesh@vageesh-Inspiron-7520:~$ node --version  
v0.10.28  
vageesh@vageesh-Inspiron-7520:~$
```

- Verify `npm` installation by running:  
`$ npm --version`

```
vageesh@vageesh-Inspiron-7520: ~  
vageesh@vageesh-Inspiron-7520:~$ npm --version  
1.4.16  
vageesh@vageesh-Inspiron-7520:~$
```

*Note: npm ships with node.js installation. If, in case, npm was not installed, manually install it by running:  
`$ curl -L https://npmjs.org/install.sh | sh`*

- Install `express` using `npm` by running:  
`$ npm install -g express`

```
vageesh@vageesh-Inspiron-7520: ~  
vageesh@vageesh-Inspiron-7520:~$ npm install -g express
```

- Install `express-generator` using `npm` by running:  
`$ npm install -g express-generator`

```
vageesh@vageesh-Inspiron-7520: ~  
vageesh@vageesh-Inspiron-7520:~$ npm install -g express-generator
```

- Create your application folder and scaffold the application using generator by running:  
`$ express my_app`  
`$ cd my_app && npm install`

```
vageesh@vageesh-Inspiron-7520: ~  
vageesh@vageesh-Inspiron-7520:~$ express my_app
```

```

vageesh@vageesh-Inspiron-7520: ~/Desktop/my_app
vageesh@vageesh-Inspiron-7520:~/Desktop$ express my_app

create : my_app
create : my_app/package.json
create : my_app/app.js
create : my_app/public
create : my_app/public/stylesheets
create : my_app/public/stylesheets/style.css
create : my_app/routes
create : my_app/routes/index.js
create : my_app/routes/users.js
create : my_app/views
create : my_app/views/index.jade
create : my_app/views/layout.jade
create : my_app/views/error.jade
create : my_app/bin
create : my_app/bin/www
create : my_app/public/javascripts
create : my_app/public/images

install dependencies:
$ cd my_app && npm install

run the app:
$ DEBUG=my_app ./bin/www

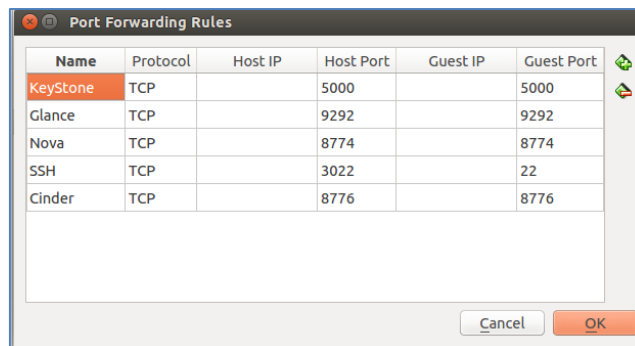
vageesh@vageesh-Inspiron-7520:~/Desktop$ cd my_app
vageesh@vageesh-Inspiron-7520:~/Desktop/my_app$ npm install

```

- Once the application is setup, use your IDE or Text Editor to develop your application.

### Port Forwarding:

Each of the OpenStack services operate on a different port number. In order to manage the OpenStack services from the host machine, we need to map the guest machine port numbers used by these services to the host machine. Port forwarding is a technique that allows translating the destination address to a new destination.[26] This is made possible by using NAT as one of the network adapters for the guest machine. For each service, we map the default port of the service to the corresponding port on the host machine.



Name	Protocol	Host IP	Host Port	Guest IP	Guest Port
KeyStone	TCP		5000		5000
Glance	TCP		9292		9292
Nova	TCP		8774		8774
SSH	TCP		3022		22
Cinder	TCP		8776		8776

Figure 10 : Port Forwarding

### Configuring the services to use Keystone:

Before making API calls to the services, we have to make sure that *Auth-Token* middleware is part of their authentication pipeline. This allows us to make direct API calls to services, and OpenStack will take care of authenticating the token for us in the background.

- Configuring Nova:

The '*nova-paste-config.ini*' is modified to include keystone in its authentication pipeline. Below is an example: [27]

```

[filter:authtoken]
paste.filter_factory = keystoneclient.middleware.auth_token:filter_factory
auth_port = 35357
auth_host = 127.0.0.1
auth_token = 012345SECRET99TOKEN012345
admin_user = admin
admin_password = keystone123

```

- Configuring Glance:  
The 'glance-api-paste.ini' and 'glance-registry-paste.ini' are modified to include keystone in its authentication pipeline. Below is an example:[28]

```
[filter:authtoken]
paste.filter_factory = keystonemiddleware.auth_token:filter_factory
auth_host = 127.0.0.1
auth_port = 35357
auth_protocol = http
admin_user = glance_admin
admin_tenant_name = service_admins
admin_password = password1234

[pipeline:glance-registry-keystone]
pipeline = authtoken context registryapp
```

Service Functions:

- Authentication:  
All API calls to OpenStack services are authenticated using a token based system, which is generated when a user provides valid credentials. The application takes the user's credentials and makes an API call to *Identity* to generate a token. This token is then stored in the application's session management. All future calls are called with this token extracted from the session. Currently, the application is supporting *Identity* v3 API.

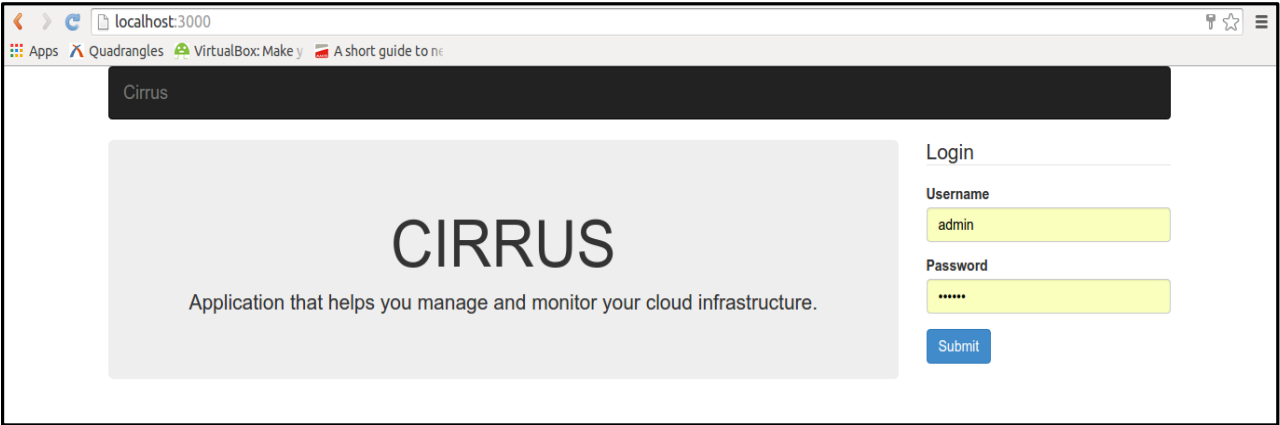


Figure 11 : Application Login

- User Management:  
The application provides an interface to map user actions for *viewing*, *creating*, *updating* and *deleting* of users in the system to API calls.

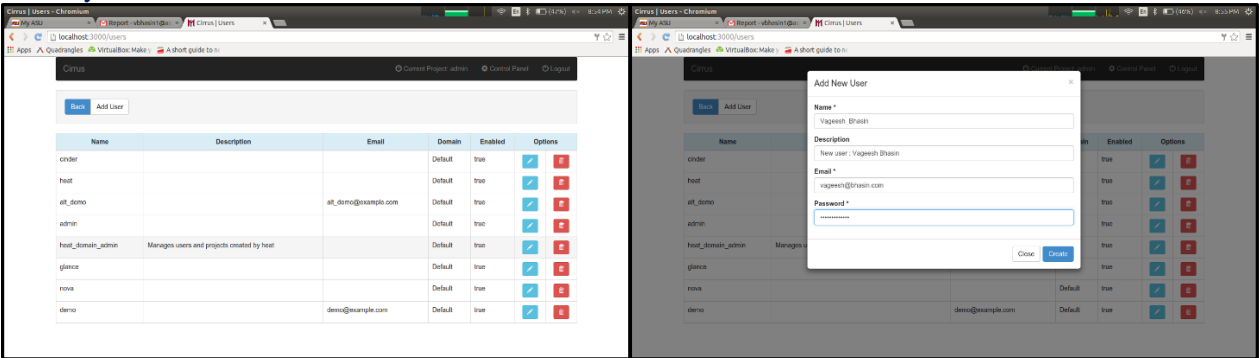


Figure 12: User List and Create new User

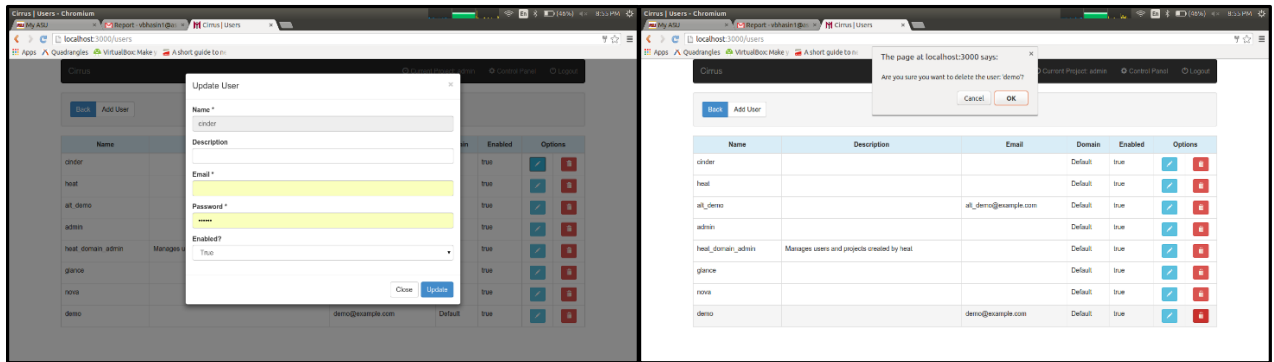


Figure 13 : User Update and User Deletion

- **Domain Management:**  
Domains represent collections of users, projects and groups.[29] Each domain defines a namespace where certain API-visible name attributes exist, which affects whether those names must be globally unique or unique within that domain.[29] The application provides interface to map user actions for *viewing*, *creating* and *deleting* domains to API calls.

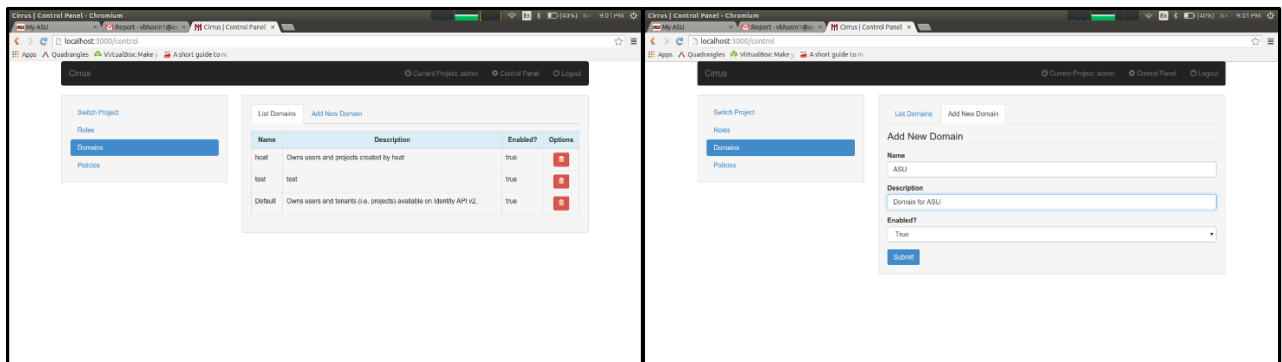


Figure 14: Domain list and New Domain

- **Project Management:**  
Projects (also called tenants) are isolated resource containers forming the principal organizational structure within the Compute service. They consist of a separate VLAN, volumes, instances, images, keys, and users.[30] A user logs into the project by specifying the *project\_id* in the login request. The application provides a simpler interface to switch project, and handles the background API calls to generate and store a new token for this project. The application also provides a user to *view*, *create* and *delete* projects.

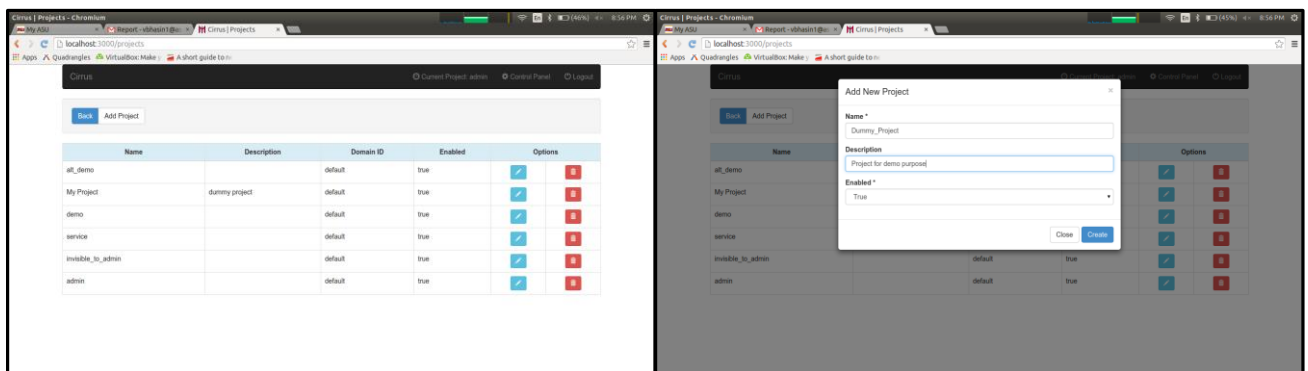


Figure 15: Project List and New Project

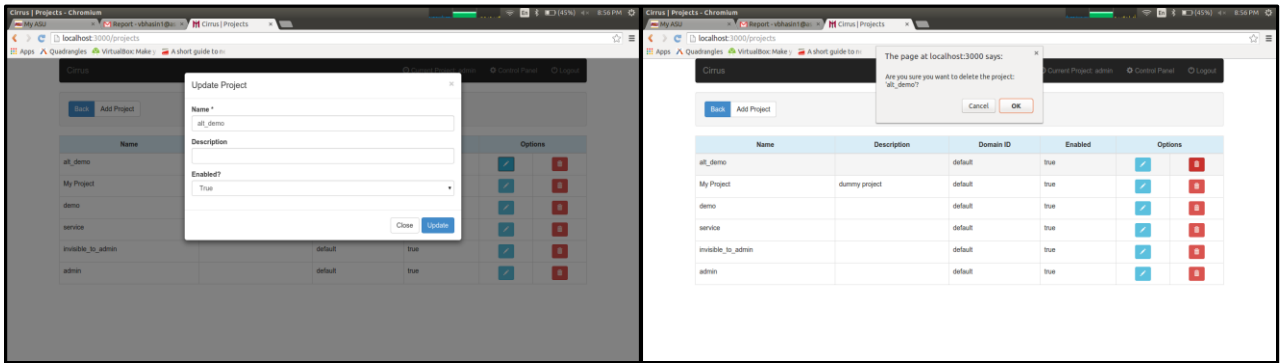


Figure 16 : Update Project and Delete Project

- **Role Management:**  
Roles define which actions users can perform. Roles are assigned to user-project pairs. The application provides an interface to *view* and *create* user roles

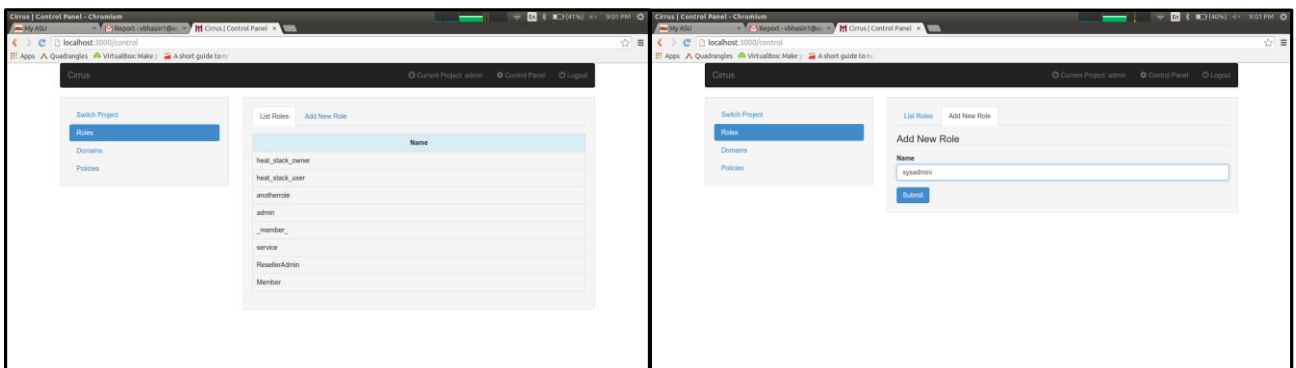


Figure 17 : Role List and New Role

- **Compute Services:**  
OpenStack Compute is a tool to orchestrate a cloud, including running instances, managing networks, and controlling access to the cloud through users and projects. The underlying open source project's name is Nova, and it provides the software that can control an Infrastructure as a Service (IaaS) cloud computing platform.[31] The application provides an interface to map the user actions of *viewing*, *creating*, *updating* (*rebooting*, *resizing* and *properties modification*) and *deleting* of server instances.

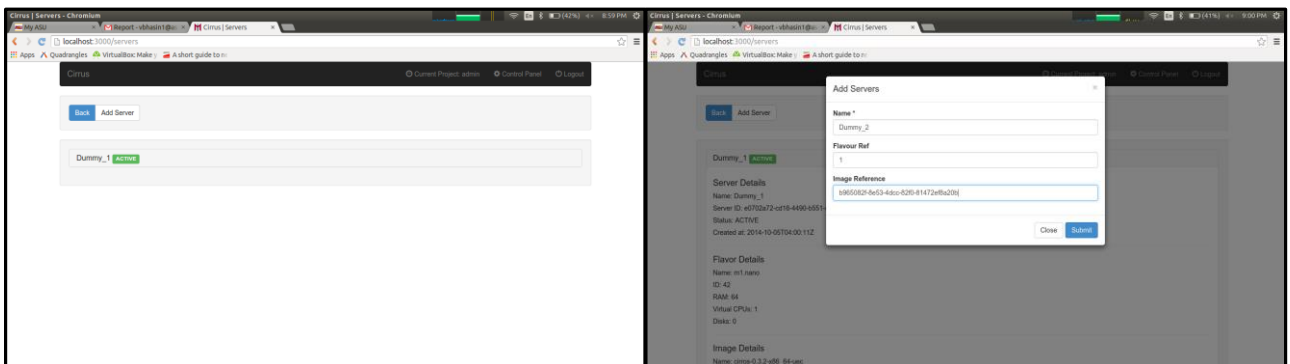


Figure 18 : Server List and New Server



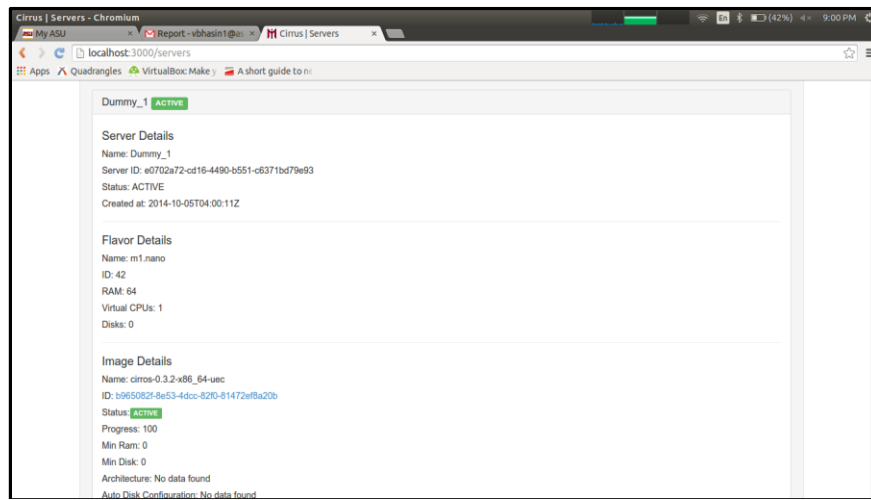


Figure 19 : Server Details

- Imaging Services:**  
 The OpenStack Image Service enables users to discover, register, and retrieve virtual machine images. Also known as the glance project, the Image Service offers a REST API that enables you to query virtual machine image metadata and retrieve an actual image.[32] The application provides an interface to map the user actions of *viewing*, *updating(metadata properties)*, *uploading(image files)* and *deleting* the images.

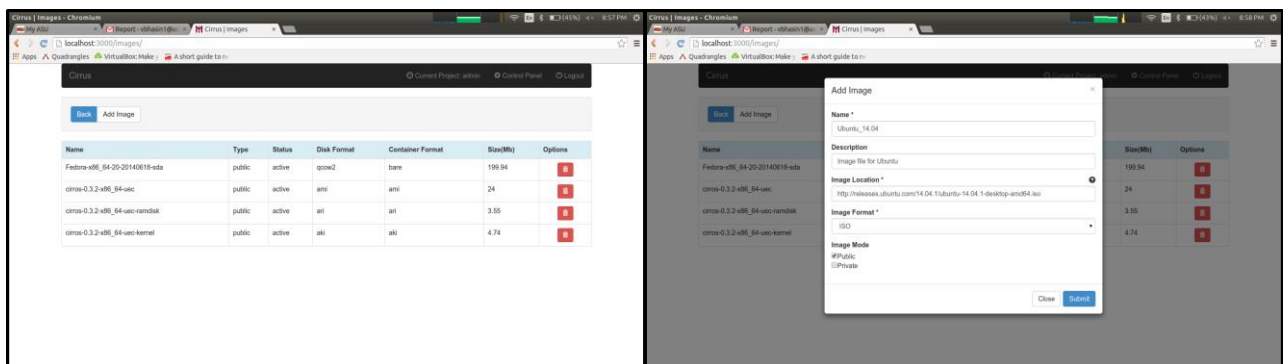


Figure 20 : Image List and New Image

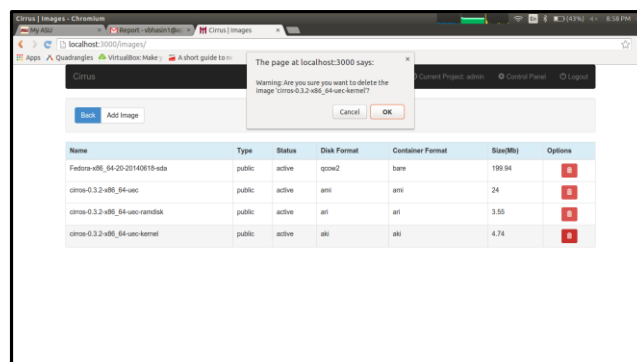


Figure 21 : Delete Image

- Block Storage Services:  
The Block Storage Service enables management of volumes, volume snapshots, and volume types.[33] It consists of *cinder-api* (API through which the application will contact the service), *cinder-volume* (responds to read-from and write-to calls to the block storage), *cinder-scheduler* (Picks the node to create volume) and a *messaging queue*. The application provides an interface to map user actions of *viewing*, *creating* and *deleting* the volumes.

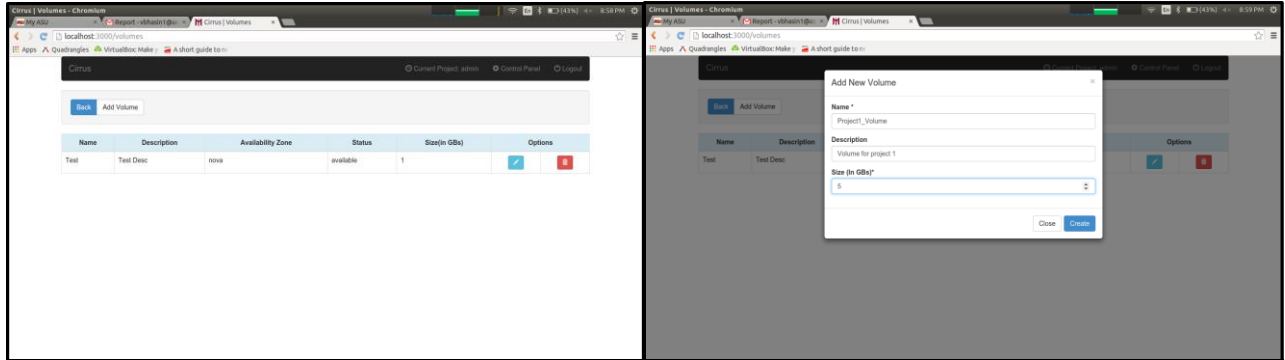


Figure 22 : Volume List and New Volume

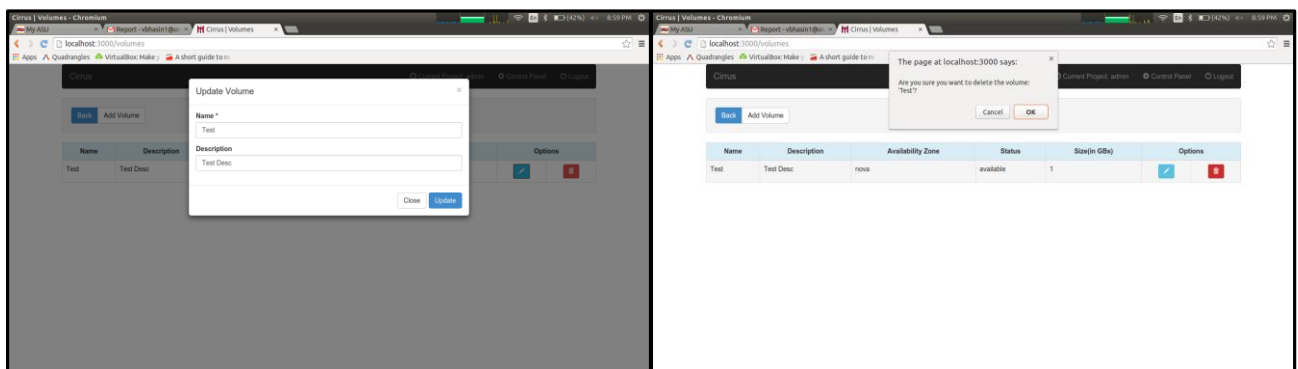


Figure 23 : Update Volume and Volume Delete

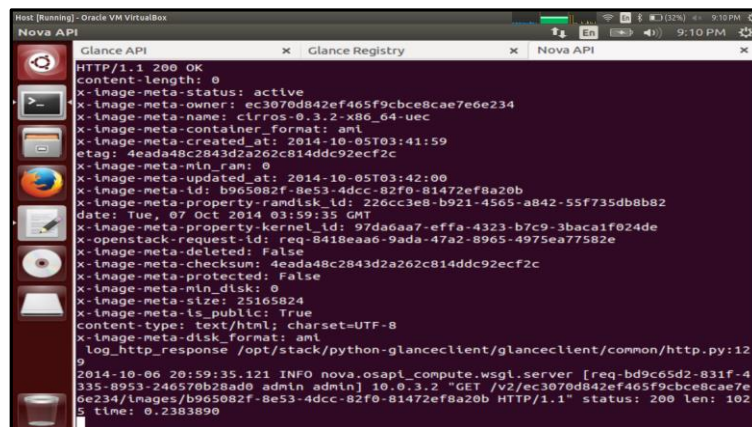
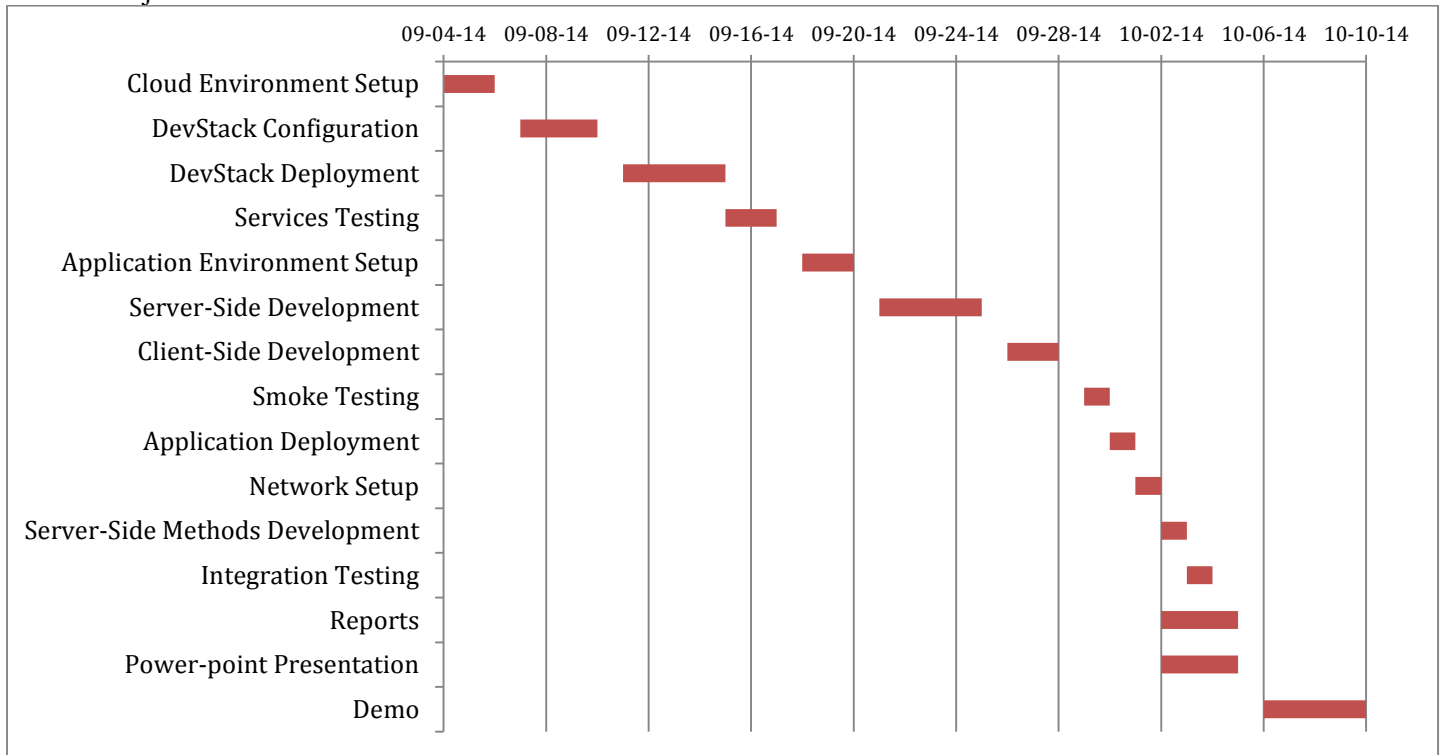


Figure 24 : OpenStack APIs running on VM

The following services could not be implemented due to unforeseen circumstances and short time frame:

- Object Storage Services
- Networking Services

## Project Timeline



## IV. RISK MANAGEMENT OF THE PROJECT

Risk Description	Likelihood	Impact	Priority	Mitigation Strategy
Network Connectivity	Medium	High	High	Virtual machines to use same network adapter
DevStack feature not supported by API	Low	Medium	Medium	Extend the API to integrate the required feature
Physical Machine Overload	High	High	High	Use a higher configuration machine or vLab

## V. CONCLUSION

We are attempting to provide a user friendly and intuitive web application to perform administrative tasks in a DevStack cloud platform. This will also provide the administrator with a remote access to the system.

The future scope for this project would be to extend this application to include a heterogeneous cloud infrastructure which can include different cloud environments like XenServer and XCP. Another future scope is to allow the application to connect to a multi-node environment. Porting the application for hand-held devices can be considered as another avenue to be implemented.

## VI. ACKNOWLEDGMENT

We sincerely thank Prof. Dr. Dijiang Huang for giving us this opportunity to understand and work on an open-source cloud platform. We would also like to thank our mentor, Chun-Jen Chung, for helping us out by clearing any doubts and explaining concepts.

## VII. REFERENCES

- [1] Oracle Corporation, (no date), Oracle VM VirtualBox User Manual. Available at: <https://www.virtualbox.org/manual/UserManual.html> (Accessed: 27th August 2014)
- [2] OpenStack Wiki, (no date), DevStack on a VirtualBox. Available at: <https://wiki.openstack.org/wiki/DevStackVirtualbox> (Accessed: 28th August 2014)
- [3] Joyent Inc., (no date), Node.js API Reference documentation. Available at: <http://nodejs.org/documentation/api/> (Accessed: 29th August 2014)
- [4] OpenStack Foundation, (no date), Chapter 4. OpenStack Architecture. Available at: <http://docs.openstack.org/training-guides/content/module001-ch004-openstack-architecture.html> (Accessed: 2nd September 2014)
- [5] OpenStack Foundation, (no date), Single Node DevStack Installation. Available at: <http://devstack.org/guides/single-machine.html> (Accessed: 1st September 2014)
- [6] OpenStack Foundation, (no date), OpenStack API Documentation. Available at: <http://developer.openstack.org/api-ref.html> (Accessed: 1st September 2014)
- [7] OpenStack Foundation, 2010, OpenStack Compute: An Overview. Available at: <http://www.openstack.org/downloads/openstack-compute-datasheet.pdf> (Accessed 1st September 2014)
- [8] OpenStack Wiki, (no date), OpenStack: HyperVisor Support Matrix. Available at: <https://wiki.openstack.org/wiki/HypervisorSupportMatrix> (Accessed: 29th August 2014)
- [9] OpenStack Docs, (no date), OpenStack Cinder Developer's Documentation. Available at: <http://docs.openstack.org/developer/cinder/> (Accessed: 1st September 2014)
- [10] OpenStack Docs, (no date), OpenStack Associate Training Guide, under Apache License 2.0. Available at: <http://docs.openstack.org/training-guides/content/> (Accessed on: 21 September 2014)
- [11] Node.js Project, (no date), Node.js Homepage. Available at: <http://nodejs.org/> (Accessed: 21st September 2014)
- [12] Tomislav Capan, (no date), Why the hell would I used Node.js? - A Case-by-Case tutorial. Available at: <http://www.toptal.com/nodejs/why-the-hell-would-i-use-node-js> (Accessed: 21st September 2014)
- [13] Isaac Z. Schlueter, 11th September 2010, Introduction to npm <http://howtonode.org/introduction-to-npm> (Accessed: 21st September 2014)
- [14] TJ Holowaychuk, (no date), Express.js Homepage <http://expressjs.com/> (Accessed: 21st September 2013)
- [15] Stephen Sugden, (no date), A short guide to Connect Middleware [http://stephensugden.com/middleware\\_guide/](http://stephensugden.com/middleware_guide/) (Accessed: 21st September 2014)
- [16] Oracle Corporation, (no date), What are RESTful services? <http://docs.oracle.com/javase/6/tutorial/doc/gijqy.html> (Accessed: 21st September 2014)
- [17] Alex Rodriguez, 6th November 2008, RESTful Web Services: The Basics <http://www.ibm.com/developerworks/library/ws-restful/> (Accessed: 21st September 2014)
- [18] OpenStack Foundation, (no date), Swift's Documentation. Available at: <http://docs.openstack.org/developer/swift/> (Accessed on: 18th September 2014)
- [19] SwiftStack, (no date), OpenStack Swift Overview. Available at: <https://swiftstack.com/openstack-swift/architecture/> (Accessed on: 18th September 2014)
- [20] Kenneth Hui, June 2013, OpenStack Compute. Available at: <http://cloudarchitectmusings.com/2013/06/26/openstack-for-vmware-admins-nova-compute-with-vsphere-part-2/> (Accessed: 19th September 2014)
- [21] OpenStack Foundation, (no date), Cinder Documentation. Available at: <https://wiki.openstack.org/wiki/Cinder> (Accessed: 20th September 2014)
- [22] Kenneth Hui, November 2013, OpenStack Cinder. Available at: <http://cloudarchitectmusings.com/2013/11/18/laying-cinder-block-volumes-in-openstack-part-1-the-basics/> (Accessed: 19th September 2014)
- [23] OpenStack Foundation, (no date), Neutron Documentation. Available at: [http://docs.openstack.org/security-guide/content/ch031\\_neutron-architecture.html](http://docs.openstack.org/security-guide/content/ch031_neutron-architecture.html) (Accessed: 19th September 2014)
- [24] Romil Gupta, iLearnStack, 23rd April 2013, Introduction to OpenStack. Available at: <http://ilearnstack.com/tag/glance/> (Accessed: 19th September 2014)
- [25] Panchaleswar Nayak, 20th March 2014, High Availability in the OpenStack Cloud. Available at: <http://panchaleswar.wordpress.com/2014/03/20/high-availability-in-the-openstack-cloud/> (Accessed: 21st September 2014)
- [26] PC Magazine, (no data), Definition of Port Forwarding. Available at: <http://www.pcmag.com/encyclopedia/term/49509/port-forwarding> (Accessed: 4th October 2014)
- [27] OpenStack Foundation, (no date), Configuring services to work with KeyStone. Available at: <http://docs.openstack.org/developer/keystone/configuringservices.html> (Accessed: 4th October 2014)
- [28] OpenStack Foundation, (no date), Authentication with KeyStone. Available at: <http://docs.openstack.org/developer/glance/authentication.html> (Accessed: 3rd October 2014)

- [29] OpenStack Foundation, (no date), Identity API v3 Reference. Available at:  
<http://developer.openstack.org/api-ref-identity-v3.html> (Accessed: 5<sup>th</sup> October 2014)
- [30] OpenStack Foundation, (no date), Managing Projects and Users. Available at:  
[http://docs.openstack.org/openstack-ops/content/projects\\_users.html](http://docs.openstack.org/openstack-ops/content/projects_users.html) (Accessed: 5<sup>th</sup> October 2014)
- [31] OpenStack Foundation, (no date), Introduction to OpenStack Compute. Available at:  
[http://docs.openstack.org/grizzly/openstack-compute/admin/content/ch\\_introduction-to-openstack-compute.html](http://docs.openstack.org/grizzly/openstack-compute/admin/content/ch_introduction-to-openstack-compute.html) (Accessed: 4<sup>th</sup> October 2014)
- [32] OpenStack Foundation, (no date), Configure the Image Service. Available at:  
[http://docs.openstack.org/havana/install-guide/install/apt/content/ch\\_glance.html](http://docs.openstack.org/havana/install-guide/install/apt/content/ch_glance.html) (Accessed: 5<sup>th</sup> October 2014)
- [33] OpenStack Foundation, (no date), Block Storage Service. Available at:  
<http://docs.openstack.org/havana/install-guide/install/apt/content/block-storage-service.html> (Accessed: 5<sup>th</sup> October 2014)