CHAPTER 7

EXPLORING THE PRIVATE CLOUD SETUP USING THE OPEN SOURCE CLOUD TECHNOLOGIES

7.1 INTRODUCTION

In the current economic climate where the expectations of efficiencies and cost savings are growing from IT organizations, private clouds provide a good opportunity to get started with cloud computing and reap the associated benefits of agility, cost savings, and on-demand services while meeting the stringent enterprise security, performance, and reliability requirements. Private clouds can be used to emulate a public cloud environment and can be used as a development and test platform while developing the applications to be deployed on the particular public cloud. Further, the same environment can also be used to test the developed applications for functionality as well as validations before it moves to the production at a public cloud.

The basic requirement for setting up a private cloud environment is not at all expensive. Private cloud has the potential to revolutionize the computing facilities of institutions in a cost effective way. There are different open source technologies which are available to build private cloud within the college, school campuses. This private cloud can offer students with desired computational facilities on demand without any expense. Now a day’s most of
the institutes are connected through high-speed internet with each other. Thus, centralized cloud infrastructure can satisfy all infrastructures related needs of those institutes on demand. It will create a common platform to be shared among faculties; students belonging to different institutions in a reusable fashion.

Public and private cloud are using the same underneath technologies like virtualization, but differed in the nature of demand, availability of resources and provisioning of resources. In the public cloud, the cloud infrastructure is owned by the cloud providers. The users can avail the services on demand through the internet on pay for use basis. The main benefit of the public cloud is no capital expenses for infrastructure, dynamic scaling, and no maintenance for resources. But the drawback is security; the resources are not present in the user’s site. Whereas in a private cloud, the cloud infrastructures are owned by the organization, hence the investment in infrastructure is not negligible. The access to the resource is restricted within the organization. The resources may exist on premise or off premise. The major benefits of private clouds are higher security, privacy, and more control.

Benefits and features of private cloud are:

- Allows the IT users to access the resources in a self-service manner
- Optimizes the resource utilization with the help of virtualization
- Manages and controls over the resources
- Cloud bursting is possible when there is spike in the demand

The remainder of this chapter is organized as follows. The next section discusses the open source cloud technology OpenStack, its components, setting a private cloud and leveraging services from the private cloud. Section 7.3 presents the components and setting up a private cloud.
using UEC. Section 7.4 gives the comparison of virtual machine provisioners in the open source cloud technologies. Section 7.5 discusses the chrome extension developed for the OpenStack for the benefit of cloud administrators.

7.2 OPENSTACK

Open source software for setting up a private cloud is widely available. OpenStack, Eucalyptus, and OpenNebula are the main competitors in the private cloud area. Among them, OpenStack has a more user-friendly Graphical User Interface (GUI). OpenStack is a cloud computing project for providing the services. It was founded by National Aeronautics and Space Administration (NASA) and RackSpace in 2010. It is a cloud operating System, a new management layer that adds automation and control for a pool of resources such as computation, storage, and network, empower administrators and users via self-service portals and empower developers to make applications cloud-aware via Application Programming Interfaces (APIs) (OpenStack 2013). The project is managed by the OpenStack Foundation, a non-profit corporate entity established in September 2012 to promote, protect, and empower OpenStack software and its community.

The major advantage of OpenStack is using the same API as Amazon, which makes cloud bursting or a hybrid cloud solution easily implementable. OpenStack is a cloud computing platform which is EC2 and Simple Storage Service (S3) compatible, and since its services can be reached using EC2 and S3 compatible APIs, any client tools written for Amazon Web Service (AWS) can communicate with OpenStack as well. It is customizable thus, OpenStack seems like a treat to researchers who are working in the cloud environment. The researchers experience the cloud, make studies on the algorithms used in cloud and exploit it as a real test bed for their work. They
can implement a private cloud in their laptop and do their work not depending on the public cloud which has all proprietary components.

### 7.2.1 Components of OpenStack

OpenStack is horizontally and massively scalable, hardware agnostic and hypervisor agnostic. It supports for Xen, Citrix XenServer, Microsoft Hyper-V, KVM, User Mode Linux, Linux Containers.

OpenStack has a modular architecture and it has various components. OpenStack (Essex) has five components namely Nova, Glance, Keystone, Swift, and Horizon. Among the current core projects of OpenStack, Nova project is the core of the cores. Nova contains nova-API, nova-compute, nova-volume, nova-network and nova-scheduler. For a virtual machine request, the infrastructure to the VM like the number of cores, RAM, virtual network is provided through the Nova component. Nova-compute hosts VMs, controls hypervisor and VMs when it receives commands on Message Queue. Nova-API is the public facing interface. Message Queue acts as a broker to handle interactions between services, currently based on RabbitMQ. Openstack refers to its cloud scheduler component using the name “nova-scheduler” (OpenStack). The central component that manages the allocation of virtual resources for a cloud infrastructure’s physical resources is known as the cloud scheduler. Nova-scheduler coordinates all services and determines the placement of new resources requested. The nova-scheduler chooses the host for the VM to be launched, according to the scheduling algorithm configured by the administrator. There are a rich set of scheduling algorithms available in the OpenStack. It is a customizable component; hence an organization may customize the scheduling algorithms according to their need. Nova-volume is a persistent storage; it can be attached or detached with any instance. Even if
the instance is terminated, the nova-volume exist. Nova-network provides networking to the virtual machines.

Glance keeps a database of metadata associated with an image for registering. It is used to retrieve and discover the image when needed. It is built on top of Swift where images are stored. Glance has two services namely Glance-API and Glance-registry. Glance-API is a public interface for uploading and managing images. Glance-registry is a private interface to the metadata database. Glance supports multiple image formats like International Organization for Standardization (ISO), Amazon Machine Image (AMI), and Virtual Desktop Infrastructure (VDI), etc. Keystone provides the identity and access policy services for all OpenStack components. This identity service includes authentication and authorization of all the components. All the components and users of OpenStack are registered with the KeyStone. Swift is the object storage device which provides service for storing and retrieving arbitrary data. It uses the native API and S3 compatible API. Horizon is the web interface for OpenStack. In the OpenStack dashboard, Horizon is used to provide a simple self-service user interface for end users. Figure 7.1 shows the architecture diagram of OpenStack.
7.2.2 Setting up the Private Cloud using OpenStack

The private cloud has been setup using the open source cloud operating system OpenStack. To implement and experience the private cloud infrastructure, minimum two dedicated systems are needed such as a front-end and one or more node(s) with minimum 2GB memory. Any number of compute nodes can be added up with the cloud controller to increase the resource availability of the cloud.
Figure 7.2 shows the implementation diagram of the private cloud set up. OpenStack (Essex) has five components like Nova, Glance, KeyStone, Swift, and Horizon. All the components should be installed in the cloud controller. It is enough that the compute nodes to have the Nova component. The nova component creates the virtualization environment in the compute nodes by the hypervisors.

To deploy a minimal cloud infrastructure, at least two dedicated systems are needed: a front end, one or more node(s) with minimum 2GB memory. The private cloud is set up using a cloud controller with three compute nodes. The scalability of the cloud can be extended through adding any number of compute nodes with the cloud controller. Table 7.1 describes the hardware details of the private cloud setup.

Table 7.1  Hardware used to setup the Cloud

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Cloud Controller (Server 1)</th>
<th>Compute Node 1 (S2)</th>
<th>Compute Node 2 (S3)</th>
<th>Compute Node 3 (S4)</th>
<th>Client</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>VT Enabled 64 bit Dual Core</td>
<td>VT Enabled 64 bit Dual Core</td>
<td>VT Enabled 64 bit Dual Core</td>
<td>VT Enabled 64 bit Dual Core</td>
<td>VT Enabled 64-bit Dual core</td>
</tr>
<tr>
<td>RAM</td>
<td>4 GB</td>
<td>16 GB</td>
<td>4 GB</td>
<td>4 GB</td>
<td>1 GB</td>
</tr>
<tr>
<td>Hard disk Size</td>
<td>320 GB</td>
<td>500 GB</td>
<td>320 GB</td>
<td>320 GB</td>
<td>320 GB</td>
</tr>
<tr>
<td>Network</td>
<td>1 Gbps</td>
<td>1 Gbps</td>
<td>1 Gbps</td>
<td>1 Gbps</td>
<td>1 Mbps</td>
</tr>
</tbody>
</table>
7.2.2.1 Cloud controller (server1) installation

1. In the cloud controller, Ubuntu 12.04 version has been installed as the base operating system. Server1 contains all nova- services (nova-compute, nova-API, nova-volume, nova-scheduler, nova-network), Glance, Swift, Keystone, and Horizon. Server1 has two Network Interface Cards (NICs).

2. The network configurations are set. The network configuration is restarted.

3. The file /etc/ntp.conf is edited to make sure that the time on the server stays in sync with an external server. If the Internet connectivity is down, the Network Time Protocol (NTP) server uses its own hardware clock as the fallback. The NTP server is restarted.

4. OpenStack allows us to use MySQL, PostgreSQL or SQLite for Nova and Glance. Databases for Nova, Glance, and Keystone are created and the privileges are given to the users.

5. Keystone is installed. Tenants are created such as admin and service. Users are created as nova, glance, admin, and swift. Roles are created as admin and member.

6. Glance component is installed by installing the packages glance-API, glance-client, glance-common, glance-registry, and python-glance.


8. Nova-volume is created using the following commands. sudo pvcreate /dev/sda6. sudo vgcreate nova-volumes /dev/sda6.

9. OpenStack dashboard is installed by installing the package openstack-dashboard.
10. Swift is installed. The primary components of swift are the proxy, account, container and objectservers.

7.2.2.2 Compute node installation

1. In the compute nodes (S2, S3, S4) Ubuntu 12.04 version has been installed as the base operating system. This node contains only the nova-services including nova-compute, nova-api, nova-volume, nova-scheduler, nova-network. It contains two NICs.
2. The network configurations like public Internet Protocol (IP) and private IP is set. The network services are restarted.
3. NTP package is installed and this compute node is synchronized with the cloud controller by editing the ntp.conf file. Then, the NTP client is restarted.
4. The nova-compute is installed and restarted.

Once Server1 has been installed, the “nova-manage service list” command lists all the nova services. If compute nodes are added up, it shows the compute node details also. By issuing the following command, the compute nodes available in the cloud are listed out. Figure 7.3 shows the list of all nova services installed on the system.
Using the private cloud setup, the following services such as Database as a Service (Oracle), Web service as Service (Xamp server), Compiler as a Service (C compiler), Platform as a Service (Ubuntu OS) are provided. All these software can be accessed by the authorized person using internal network. This eliminates the need for installing the software on each machine or in each lab and cut down the money spends for licensing of software.

This section describes creating a virtual machines in OpenStack and accesses it using the Horizon web interface.
Figure 7.4 shows the dashboard Horizon. Horizon is the default user interface for OpenStack. It checks for authentication. Authentication deals with the component Keystone. Keystone validates the user details using the database and allows the user to log in after validated.
Figure 7.5  Images in OpenStack

Figure 7.5 shows the list of images registered in OpenStack cloud. Image registration is done using the component glance.
Figure 7.6 Infrastructure Allocated to Virtual Machine

Figure 7.6 illustrates infrastructures such as virtual CPU, memory, and hard disk allocated to a virtual machine. There are some predefined flavors available in OpenStack, yet it allows the user to create their custom flavors. Before creating instances, keypair should be generated which contains a public key and private key. The private key is stored in the local directory. These keys are injected while the instance is being launched.
Figure 7.7  Instances Available in the Cloud Environment

Figure 7.7 depicts the list of created instances available in the OpenStack cloud environment.

Figure 7.8  Assigning Floating IP to Virtual Machine
Every instance is automatically assigned a private IP address. The user may optionally assign public IP addresses to instances. OpenStack uses the term "floating IP" to refer to an IP address (typically public) that can be dynamically added to a running instance. OpenStack Compute uses Network Address Translation (NAT) to assign floating IPs to virtual instances. Figure 7.8 shows the assigning floating IP to VM.

![Virtual Machine Accessed from Linux](ubuntu@sampleserver:~)

**Figure 7.9** Virtual Machine Accessed from Linux

Figure 7.9 depicts accessing the virtual machine from the ubuntu platform using the command `$ssh –i private key username@ipaddress`. 
Figure 7.10 Accessing the Virtual Machine in the Windows Environment

Figure 7.10 depicts accessing the virtual machine from the windows platform using putty. The private key should be converted from .ppk to .pem using puttygen. By specifying the IP address of the virtual machine with auth as the .pem file, the virtual machine can be accessed.

7.3 UBUNTU ENTERPRISE CLOUD (UEC)

Eucalyptus is an open-source and free Infrastructure as a Service private cloud framework for building private and hybrid cloud computing environments. Eucalyptus stands for Elastic Utility Computing Architecture for channelling the tasks to the resource pool (UEC 2013). It enables to scale up and down the various resources in the resource pool according to the utility changes dynamically. It contains five major components. They are (1) The
Cluster Controller (CC) gathers all the information about the different schedules inside the specific node (2) The Storage Controller (SC) controls the retrieving and posting of storage information of the virtual machine images and user data. (3) The Elastic Block Storage (EBS) (4) The Cloud Controller (CC) directs the request from the users and the administrators towards the Cluster Controller and makes the high-level decisions on the scheduling and finally (5) The Node Controller (NC) controls execution and termination of the virtual machine instances on the host machines.

7.3.1 Setting up Private Cloud using Ubuntu Enterprise Cloud

The UEC setup results in setting up an own private compute cloud with one controller "front-end" and one or several node(s) for running VM instances.

Server 1 - Installation

STEP 1: Prerequisites: To deploy a minimal cloud infrastructure, at least two dedicated systems are needed: a front end, one or more node(s) with minimum 2GB memory

STEP 2: Install the Cloud/Cluster/Storage/Walrus Front End Server

1. The 10.04 Server ISO is downloaded
2. The components are chosen, based on the chosen topology.
3. A name is given for the cluster.
   
   Name of the cluster, e.g. cluster1.

4. A range of public IP addresses on the local area network that the cloud can allocate to instances, e.g. 192.168.1.200-192.168.1.249.

Server 2 - Installation
STEP 1: Install the Node Controller(s)

1. Boot from the same ISO on the node(s)
2. “Install Ubuntu Enterprise Cloud” is selected
3. The partitioning scheme is confirmed
4. Installation is completed and the node(s) rebooted

STEP 2: Register the Node(s)

As of Ubuntu 10.04 LTS, all component registration should be automatic, assuming:

a. Public secure shell keys have been exchanged properly
b. The services are configured properly
c. The services are publishing their existence
d. The appropriate uec-component-listener is running
e. Verify registration.

STEP 3: Obtain Credentials

After installing and booting the cloud controller, users of the cloud will need to retrieve their credentials. This can be done either through a web browser, or at the command line.

STEP 4: Install an image from the store
Figure 7.11 Images in UEC

To add an image to UEC, should install it from the Image Store on the UEC web interface as shown in the Figure 7.11.

Figure 7.12 Installed Images in UEC
Once the image has been downloaded and installed, run the image by clicking the image button to view the command to execute to instantiate this image. The image will also appear on the list given on the Image tab. Figure 7.12 shows the installed images in the UEC private cloud setup.

**Installing the Client:**

The purpose of the client machine is to interact with the private cloud setup.

1. Ubuntu Desktop 10.04 version is installed on the client machine.
2. IP address are obtained through dynamic host configuration protocol.
3. KVM is installed so that VM images can be bundled and installed.

There are multiple ways to instantiate an image in UEC:

- Use the command line
- Use one of the UEC compatible management tools such as Landscape
- Use the ElasticFox extension to Firefox

**7.4 OPEN SOURCE CLOUD TECHNOLOGIES AND THEIR VIRTUAL MACHINE PROVISIONERS**

For a researcher, private cloud is a better option to experience the happening in a cloud environment. Because public clouds are all proprietary, even though services can be availed, the workings are not known. Private cloud, on the other hand, allows the user to customize the algorithms, work out their problems. This section explores the components used for resource management and monitoring available in the open source cloud technologies
such as Eucalyptus, OpenStack, and OpenNebula which have drawn the consideration of many people in the research community and the industry.

### 7.4.1 OpenNebula

OpenNebula is a free and open-source virtual machine provisioner, which is highly bendable and notably feature-rich (OpenNebula 2013). It manages the resources available in the resource pool and also takes care of the provisioning of the various resources to the virtual machines efficiently. It supports public, private, and hybrid implementations of the cloud. It acts as a scheduler which provisions the resources from the resource pool to the virtual machines. Also, it provides assorted schemes when it comes to provisioning a virtual machine with resources.

As mentioned earlier, OpenNebula is a free and open-source resource management toolkit, which is used to build IaaS, by provisioning virtual machines by separating the resource pool into clusters and datacenters. Cloud APIs like Representational State Transfer (REST), AWS EC2 API comes inherited within OpenNebula and it also holds into numerous hypervisors, Xen, KVM, VMware etc., The architecture of OpenNebula is a cell-based structure, wherein user’s scheduling algorithms can be integrated. Its frontier lies an intermediary physical machine which connects the chunks of clusters, which in turn contains a set of virtual machines. The default scheduler inside the OpenNebula provides a rank based scheduling scheme where the virtual machines are allotted resources based on ranks, however as the OpenNebula platform is flexible enough to accommodate the user’s scheduling schemes inside the clusters, the users can modify the scheduling schemes to suit their needs.
OpenNebula scheduling component is designed in a generic way, so it is highly modifiable and can be easily replaced by third-party developments. OpenNebula comes with a match-making scheduler. The main motive behind this Rank Based Scheduling Policy is to prioritize the resources which are more suited for the virtual machine. From the set of available hosts, the scheduler filtered out all the hosts those do not have enough resources to accommodate the virtual machines. Then on the set, it applies the RANK expression to evaluate the resources most suitable for the virtual machine. Resources with higher rank are used to allocate the virtual machines.

7.4.2 OpenStack

OpenStack is an open-source and free software which contains various components which support different users for their varied needs. These components usually are used to deploy as an Infrastructure as a Service solution where it provides a platform to deploy public and private cloud. OpenStack allows multiple instances of virtual machines which can handle the varied process on the fly enabling to scale-up tremendously by allowing many processes to carry out concurrently. This also leverages the cloud environment by supporting it in a way such that it can handle few or more users concurrently. Taking advantages of the fact that the OpenStack is open-source and free thousands of developers all over the world could collaborate and work on improving the framework thus by ensuring that the framework is the most secure, strong and contribute to a large community.

The schedulers available in OpenStack are FilterScheduler, ChanceScheduler, SimpleScheduler, and MultiScheduer. SimpleScheduler finds the least loaded host. ChanceScheduler randomly selects from the list of filtered hosts. Another option MultiScheduer is there which apply different schedulers for nova-compute and nova-volume. Inside OpenStack, the default
scheduler is the filter scheduler for provisioning virtual machines. Filter scheduler usually maintains a dictionary of unfiltered hosts for which the various filters will be applied and virtual machines are assigned. The various filter algorithms in practice are ComputeFilter-which filter the hosts according to the required compute units, RamFilter-filter the hosts according to the RAM required by the VM, CoreFilter- filters the host according to the number of cores required etc..

7.4.3 UEC

Scheduling in Eucalyptus determines the process by which virtual machines are allocated to the nodes. This is done to balance the load on the resource network consisting of numerous host machines. In Eucalyptus throughput, fairness in waiting time and response time are considered as important factors. Throughput means at a given time, the number of VM's that can be successfully allotted inside the various host. Fairness in waiting time denotes the time period that a VM has to wait in order for it to be scheduled with resources, Response time is the amount of time taken by the provisioner to provision the VM that has requested for the resources. The scheduling algorithm that is most popularly used in the cloud computing arena is the greedy algorithm which is a simple algorithm that tries to fit the requesting VM with the host machine that it first finds in the network. As greedy algorithm is one of the simplest algorithms when it comes to scheduling but, on the flip-side, it also has a very major drawback called underutilization of the resources in the resource pool. Based on the present study, Table 7.2 summarizes the comparison analysis of cloud solution platforms Eucalyptus, OpenStack and OpenNebula considering their general aspects.
<table>
<thead>
<tr>
<th></th>
<th>Eucalyptus</th>
<th>OpenNebula</th>
<th>OpenStack</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Installation Effort</strong></td>
<td>Difficult</td>
<td>Medium</td>
<td>Difficult</td>
</tr>
<tr>
<td><strong>Open Source License</strong></td>
<td>Linux Open Source</td>
<td>Apache 2.0</td>
<td>Apache 2.0</td>
</tr>
<tr>
<td><strong>Administration</strong></td>
<td>Web user interface, Command line interface</td>
<td>Web user interface, Command line interface</td>
<td>Web user interface, Command line interface</td>
</tr>
<tr>
<td><strong>Language</strong></td>
<td>Java, C, Python, Perl, Shell</td>
<td>C, Ruby, Shell Scripts, Java</td>
<td>Python, Shell Scripts</td>
</tr>
<tr>
<td><strong>Hypervisor Support</strong></td>
<td>Xen, KVM</td>
<td>Xen, KVM</td>
<td>KVM, XenServer, HyperV, VMWare (Focus on KVM)</td>
</tr>
<tr>
<td><strong>Cloud Interface</strong></td>
<td>Amazon EC2 API, Proprietary API, OCCI API</td>
<td>OCCI API, Amazon EC2 API, LibVirt API, Proprietary API</td>
<td>Amazon EC2 API, Proprietary API, OCCI API</td>
</tr>
<tr>
<td><strong>EC2 Compatibility</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes (Nova EC2 API)</td>
</tr>
</tbody>
</table>
### Virtual Network
- Static and Dynamic
- Static and Dynamic
- Static and Dynamic

### VM Placement Policies
- Greedy and round robin
- Match Making - Initial placement based on rank policy
- Filter Scheduler

<table>
<thead>
<tr>
<th>Virtual Network</th>
<th>Static and Dynamic</th>
<th>Static and Dynamic</th>
<th>Static and Dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM placement policies</td>
<td>Greedy and round robin</td>
<td>Match Making - Initial placement based on rank policy</td>
<td>Filter Scheduler</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VM Placement Configuration Policies</th>
<th>No</th>
<th>Can customize according to the user’s need</th>
<th>Can customize according to the user’s need</th>
</tr>
</thead>
<tbody>
<tr>
<td>Live Migration</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Load Balancing</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### 7.5 MANAGING RESOURCES IN CLOUD: CHROME EXTENSION TO OPENSTACK

The existing dashboard of OpenStack, Horizon shows the memory size and number of virtual CPUs allocated to each instance but does not give the summary of the resources consumed on the whole by the host. A chrome extension to OpenStack is developed that provides the state of the instances and the total amount of resources consumed in each host. This resource utilization information can be utilized by the administrator to do the dynamic consolidation to use the resources in an effective way, minimize the power consumption by migrating instances from less utilized hosts to other hosts. The developed chrome extension provides details of instances such as the state of each instance which shows whether the instance is in Build, Active, or Terminated state. Besides, it provides details such as CPU
utilization, memory utilization and disk utilization of each host which are not displayed in the default dashboard of OpenStack. This extension also provides an alert message to the administrator when there are over-utilization and under-utilization of resources in a host.

The chrome extension provides the state of the instances and total amount of resources consumed by each host. These details are displayed using chrome API. The information that is displayed in the chrome extension is list of instances with their state such as Build, Active, and Terminated. It also provides the amount of resources consumed by each host. The threshold value can be set up for CPU, memory, and disk resources. When the resource utilization of the particular host reaches the threshold value, chrome provides the alert message to the administrator to avoid over-provisioning of resources in a host. Chrome API communicates with nova API to display the above information.

![Sequence Diagram of Login Module](image)

**Figure 7.13  Sequence Diagram of Login Module**

In login module, admin provides the user name and password to the chrome API. Chrome API requests the nova API for authentication. If the
details are valid, nova API authenticates the admin. Figure 7.13 shows the sequence diagram of login module.

Figure 7.14 Sequence Diagram of Instance Description Module

In the instance description module, the user requests the instance status. Nova API provides the list of instances and their status to chrome API and it is displayed in the chrome extension. Figure 7.14 shows the sequence diagram of instance description module.
Nova API provides the total amount of resources provided for each host to chrome API and chrome calculates the percentage of resources consumed by each host and it is displayed in the chrome extension. Figure 7.15 gives the sequence diagram of host details module.

Google Chrome Extensions are browser extensions that modify the Google Chrome browser (Chrome Extension 2013).

The steps to do the chrome extension for OpenStack are as follows.

1. OpenStack chrome extension has a chrome API which communicates with nova API through Cross Height Request (XHR) which serves as an Hyper Text Transfer Protocol (HTTP) request and response message to get the resource information. XHR interacts with the built-in web services of nova to collect the information.

2. Nova API, in turn, communicates with the nova database to retrieve the information
3. The response message is being parsed in Java Script Object Notation (JSON) and is sent to the Chrome API

4. Chrome API calculates the amount of resources such as memory, disk, and CPU utilization based on the information gathered from Nova API.

5. Chrome API displays the resource utilization information in chrome extension and sets the threshold limit for each resource. An alert is given to the administrator when there is over-provisioning or under-provisioning of resources on the host.

Figure 7.16  Login Screen of Chrome Extension

Figure 7.16 shows the chrome extension, it is plugged with the chrome browser. When the administrator clicks, it will ask for the login information.
Figure 7.17 Resource Utilization of Each Host

Figure 7.17 shows the chrome extension, after the login the resource’s utilization information will be displayed for the administrators.

7.5.1 Alerts to Administrators for Resource Management

Chrome extension provides the list of hosts connected to the server. It also provides resource utilization information for each host which consists of the amount of CPU utilization, memory utilization, and disk utilization in percentage. The threshold value for each resource has been set and when any of this resource utilization by a particular host exceeds the threshold limit or goes below the threshold limit, chrome extension provides an alert message to the administrator to indicate the over-utilization or under-utilization of resources in a host. This helps administrator to control the over-provisioning and under-provisioning of resources in a host. Figure 7.18 shows the list of hosts and their resource utilization.
Figure 7.18  List of Hosts and their Resource Utilization

Figure 7.19  Setting up the Threshold Values
Figure 7.19 shows the screen when an administrator needs the alert message for overutilization or underutilization of resources in each host, she/he can set up the threshold values.

![Resource Alert](Image)

**Figure 7.20  Resource Alert**

Figure 7.20 and Figure 7.21 show the alert message given to the administrator.

![Resource Alert](Image)

**Figure 7.21  Resource Alert**
7.6 SUMMARY

The present system can be further enhanced by calculating the overall power consumed by each host and displaying it in the chrome extension. To minimize the power consumption, a scheduling algorithm can be proposed which allocates the incoming VM requests to the appropriate hosts. A daemon process should be made running as a background process to check the capacity of the hosts and the number of instances in it. When a host is detected as underutilized, the instances in the host can be migrated to some other active host and the host can be put in either idle or in a sleep state. This way of dynamic consolidation of VMs results in minimal power consumption. In future, this dynamic consolidation concept will be integrated with open source cloud technology OpenStack.