

Community Experience Distilled

Apache CloudStack Cloud Computing

Leverage the power of CloudStack and learn to extend the CloudStack environment

Navin Sabharwal
Ravi Shankar

[PACKT] PUBLISHING open source



Community Experience Distilled

Apache CloudStack Cloud Computing

Leverage the power of CloudStack and learn to extend the CloudStack environment

Navin Sabharwal
Ravi Shankar

[PACKT] open source*
PUBLISHING

Table of Contents

Apache CloudStack Cloud Computing
Credits
About the Authors
About the Reviewers
www.PacktPub.com
    Support files, eBooks, discount offers and more
        Why Subscribe?
        Free Access for Packt account holders
Preface
    What this book covers
    What you need for this book
    Who this book is for
    Conventions
    Reader feedback
    Customer support
        Errata
        Piracy
        Questions
1. Apache CloudStack Architecture
    Introducing cloud
        Infrastructure layer
            Computing resources
            Storage
            Networks and security
                Use of switches in cloud
        Management layer
            Automation
            Orchestration
            Task Execution
            Service Management
    Understanding CloudStack modules
        Cloud deployment model
            Zones

Pods
Clusters
Storage
Primary storage
Volumes
Secondary storage
CloudStack management server
API layer
Access control
Kernel
CloudStack operations
Security check
The virtual machine manager
Server resources
Installation
Job result
CloudDB
CloudStack networking architecture
Network service providers
CloudStack network offerings
Types of network in CloudStack
Physical network
Virtual network
Isolated networks
Shared networks
L3 network configuration in CloudStack
Access switches or L2 switches
CloudStack virtual router
Networking using CloudStack virtual router
Firewall and F5 Load balancer
Security groups
Summary
2. Installing Apache CloudStack
Pre-installation tasks
Requirements
Single node installation
Preparing the OS

- Management server installation
  - Building Deb packages
  - Setting up the apt repo
  - Adding the repository to the system
  - Building RPM
  - Creating yum repo
  - Adding the repository to the system
- Back to management server installation
  - Database installation and configuration
  - Preparing Network File System share for storage
  - Creating a separate NFS server
  - Preparing the system VM template
- Multinode installation
  - Management server installation
  - Installing and configuring CloudStack MySQL database
    - Installing additional management server
    - Preparing the system VMs
    - Keys and encryption
- Summary

3. Apache CloudStack Configuration
- CloudStack configuration
  - Management server console
    - Dashboard tab
    - Instances tab
    - Storage tab
    - Network tab
    - Templates tab
    - Events tab
    - Accounts tab
    - Domains tab
    - Infrastructure tab
    - Projects
    - Global settings
    - Administrators
  - Creating a domain
  - Creating an account
  - Service offerings

Compute offerings
Disk offerings
Network offerings
Infrastructure
Basic Zone configuration
Advanced Zone configuration
Creating a template
Creating a template using a running VM
Importing a VM to CloudStack
Creating a template using the Management console
Preparing the hosts
Summary
4. Apache CloudStack Networking
Zones and their types
Physical networks
Basic zone
Advanced Zone
Guest traffic
Isolated
Shared
Management traffic
Public traffic
Storage traffic
Virtual networks
Network offerings
Virtual router
System service offerings and virtual router
Network in cloud
Network services
Public IP addresses
Elastic IP address
Security groups
Using external devices with CloudStack
Load balancing rules
Elastic Load Balancer
Network Address Translation
Virtual Private Network

- CloudStack networking components
  - NetworkGuru
  - Network element
  - Network managers
  - Resources
  - CloudStack networking flows
- Summary

5. Apache CloudStack Storage
- Primary storage
  - System requirements and configuration
  - Adding a primary storage
    - Storage tags
- Secondary storage
  - Adding a secondary storage
  - Changing secondary storage IP address
  - Changing the secondary storage
  - Using OpenStack object storage (Swift)
- Volumes
  - Creating a new volume
  - Attaching a volume to Guest VM
  - Detaching a volume from an instance
  - Deleting a volume
- Snapshots
  - Creating a snapshot
  - Creating recurring snapshots
  - Creating a volume from a snapshot
  - Creating a template from a snapshot
  - VM storage migration
- Summary

6. Service Offerings and Virtual Machines
- Introducing service offerings and virtual machines
  - Compute offering
  - Disk offering
  - System service offering
- The complete process
  - Accessing the VM
  - Starting, stopping, rebooting, and destroying the VM instance

- Live migration of VMs between hosts
- CloudStack with different hypervisor
  - Citrix XenServer
  - Oracle VM
  - RedHat Enterprise Linux (KVM)
  - VMware vSphere
- Summary

7. Domains, Accounts, Projects, and Users
- Domain
  - Root domain
  - Domain and zones
  - Creating a domain
- Accounts
  - Creating an account
- Projects
  - Projects and accounts
  - Creating a project
  - Adding members to the project
  - Resource management in Projects
  - Invitation setup
  - Removing a member from a project
- Summary

8. High Availability and Scaling
- Ensuring high availability in CloudStack
  - CloudStack infrastructure high availability
  - CloudStack redundant virtual router
  - CloudStack storage high availability
    - Primary storage failure
    - Secondary storage failure
  - CloudStack and high availability
- CloudStack HighAvailabilityManager
  - HighAvailabilityManager – the Queue
- High availability of applications running on Cloudstack
  - CloudStack storage migration
- Scaling in CloudStack
  - Counters
  - Conditions

- Auto scale policy
- Auto scale VM profile
- Auto scale VM group
- Collector/Monitor
- Aggregator
- Trigger/alarm generator
- Trigger/alarm handler

Summary

9. Extending Apache CloudStack and Performance Tuning

Extending CloudStack

- Extending CloudStack networking

Integrating NetScaler with CloudStack

- Functional requirements
- Guest network with NetScaler load balancer
  - LB rule with public IP
  - Assigning a VM to the load balancer rule
  - Unassigning a VM from a load balancer rule
  - Deleting a load balancer from a zone
  - Load balancer with EIP in a basic zone
    - When the VM is created
    - When the VM is destroyed

CloudStack with Nicira NVP

- Integrating Nicira NVP to CloudStack
- Enabling the network service provider in CloudStack
- Using Nicira NVP

Integrating with OpenStack object storage (Swift)

- Customizing the CloudStack user interface
  - Changing the API path
  - Changing the session timeout
  - Single sign on integration
  - Integrating with LDAP for user authentication
    - LDAP user provisioning

The usage server

Performance tuning

- Increasing the management server maximum memory
- Database buffer pool size
- Setting and monitoring the hosts' capacity

Capping the resource usage
Summary
Index

Apache CloudStack Cloud Computing

Apache CloudStack Cloud Computing

Copyright © 2013 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the authors, nor Packt Publishing, and its dealers and distributors will be held liable for any damages caused or alleged to be caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

First published: May 2013

Production Reference: 1090513

Published by Packt Publishing Ltd.

Livery Place

35 Livery Street

Birmingham B3 2PB, UK.

ISBN 978-1-78216-010-6

www.packtpub.com

Cover Image by Artie Ng (<artherng@yahoo.com.au>)

Credits

Authors

Navin Sabharwal

Ravi Shankar

Reviewers

Shanker Balan

Lokesh Chanana

Kelcey Jamison-Damage

Piyush Pandey

Acquisition Editor

Mary Nadar

Lead Technical Editor

Sweny Sukumaran

Technical Editors

Vrinda Nitesh Bhosale

Nitee Shetty

Project Coordinator

Arshad Sopariwala

Proofreader

Lesley Harrison

Indexer

Tejal Soni

Graphics

Ronak Dhruv

Production Coordinator

Nilesh R. Mohite

Cover Work

Nilesh R. Mohite

About the Authors

Navin Sabharwal is an innovator, as well as a leader, author, and consultant in areas of Cloud Computing, Cloud Lifecycle Management, and Software Product Development.

He has been involved in identifying white spaces in areas of Information Technology and creating innovative products and services. He has taken ideas from their inception to revenue generation.

He has taken some of his ideas to develop innovative award winning products and solutions in the areas of Cloud Lifecycle Management, IT infrastructure management, IT processes, reporting analytics, and governance.

He works for HCL Technologies Infrastructure Services Division and leads the Automation and Cloud Computing Practice.

I would like to thank my family and friends, my co-author Ravi Shankar and the entire team working with me on Cloud Computing technologies. I would like to express my gratitude to my mentor Kalyan Kumar and HCL for giving me the freedom to innovate and experiment.

Special thanks to the entire Packt Publishing team who have worked hard with us all through the publication cycle of this book. Anish Ramchandani, Sweny Sukumaran, Mary Nadar, Wilson D'souza, Nitee Shetty, Vrinda Nitesh Bhosale, Arshad Sopariwala, and Yashodhan Dere a big thank you to all.

I would also like to thank Prof. Ravindra Dastikop who was instrumental in guiding us and motivating us to write this book.

Ravi Shankar holds a postgraduate degree in Information Technology from the Indian Institute of Information Technology and Management, Gwalior, India. He has been working on cloud-based technologies since the beginning of his career. He has been involved in development and implementation of Private Cloud as well as Hybrid Cloud. He has also worked on public clouds such as Amazon web services. He has worked extensively with open source technologies in the Cloud Computing space and on the Apache CloudStack platform.

He has also co-authored another book on Cloud Computing which is self-published on Amazon.com and Createspace.

I would like to thank my family and friends, my co-author Navin Sabharwal and my colleagues Piyush Pandey, Dheeraj Raghav and Lokesh Chanana for their guidance, mentoring, and continuous support. I would also like to thank Prof. Ravindra Dastikop, the team at Packt Publishing, Wilson D'souza, Mary Nadar, Yashodhan Dere, Anish Ramchandani, Nitee Shetty, Vrinda Nitesh Bhosale, Arshad Sopariwala, and Sweny Sukumaran for the support and guidance for writing this book.

About the Reviewers

Shanker Balan is managing consultant at ShapeBlue India, a globally leading consultancy, providing strategy, design, and implementation of IaaS/PaaS cloud platforms for service providers and enterprises. ShapeBlue has designed some of the worlds most high profile clouds and consulted on the go-to-market of cloud services, user experience, and process integration. Previously, he has also held technology leadership positions at Yahoo and InMobi and is involved with Apache CloudStack, Citrix CloudPlatform, and open source communities in India.

Lokesh Chanana is an engineering graduate from Maharishi Dayanand University, Haryana, India. Inclined with an extensive interest in virtualization and cloud automation, he has been working on various cloud-based technologies and their integrations since the beginning of his career.

He is currently working in HCL Technologies Infrastructure Services Division and is involved in the company's in-house cloud research team, working on various open source and enterprise-level cloud solutions and their integration with various monitoring and automation solutions. He has worked extensively on the OpenStack platform with a good amount of work on CloudStack and VMware VCloud Director Platform.

Kelcey Jamison-Damage is currently the Infrastructure Systems Architect at Backbone Technology, based out of Vancouver, BC Canada.

With close to 15 years of experience in various aspects of computer technology including sales, marketing, Internet services, support, administration, design, architecture, and business, Kelcey now focuses on helping people make the leap into Cloud Computing and provides consultation services primarily pertaining to Apache CloudStack. He is a leader in the Apache CloudStack community

and a committer to the project. He focuses on marketing and providing community support for new cloud builders.

He is currently involved in creating **RAAS** (**Redundant Array of Application Servers**) architectures and models for flexible service delivery and SaaS foundations.

I would like to thank the community at Apache CloudStack for helping us grow into one of the top IaaS cloud computing products, and in doing so, making books like this possible.

Piyush Pandey is currently working as a Track Lead in HCL Comnet. He looks after the service automation and Cloud Lifecycle Management Practice for HCL from India. He has an overall experience of 3.5 years in IT. He is responsible for designing automation solutions for enterprise IT infrastructure management. He is experienced in enterprise tools in the following tracks:

- Automation tools: BMC, HP, CA, Microsoft and open source tools such as Puppet, Chef, Cobbler, BMC Database Automation, BMC Application Release Management, BMC Cloud Lifecycle Management, and HP Cloud Service Automation
- Orchestration tools: BMC AO, HP OO, MS Opalis and VMWare orchestrator
- Monitoring tools: Nagios, Zenoss

He has worked to provide automation solutions for Fortune 500 customers such as Cummins, Ingram Micro, SGX, GulfStream, and Xerox. He holds B.Tech degree in computer engineering from NSIT Delhi.

www.PacktPub.com

Support files, eBooks, discount offers and more

You might want to visit www.PacktPub.com for support files and downloads related to your book.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at www.PacktPub.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at <service@packtpub.com> for more details.

At www.PacktPub.com, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.



http://PacktLib.PacktPub.com

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can access, read and search across Packt's entire library of books.

Why Subscribe?

- Fully searchable across every book published by Packt
- Copy and paste, print and bookmark content
- On demand and accessible via web browser

Free Access for Packt account holders

If you have an account with Packt at www.PacktPub.com, you can use this to access PacktLib today and view nine entirely free books. Simply use your login credentials for immediate access.

Preface

Apache CloudStack is an open source software for building public and private clouds. It is now a global success, and is developed and supported by scores of people around the globe as well as backed by some of the leading players in the cloud space today. This book is specifically designed to quickly help you get up to speed with Apache CloudStack and give you the confidence and understanding to roll it out in your own datacenters. From the installation of CloudStack to helping you implement production environments, this book covers a wide range of topics that help you get started with Apache CloudStack.

This book will show you:

- The architecture and core components of CloudStack along with the installation process to run an environment that can be managed and operated just like AWS, HP Cloud Services, and Rackspace.
- How to master the complete private CloudStack, from scaling out compute resources to managing object storage services for highly redundant and highly available storage.
- Practical, real-world examples of each service built upon in each chapter, allowing you to progress with the confidence that they will work in your own environments.
- Detailed screenshot-by-screenshot instructions on how to configure various features and use them.
- *Apache CloudStack Cloud Computing* gives you clear, step-by-step instructions to install and run your own cloud successfully. It is full of practical examples that enable you to use the latest capabilities of CloudStack and implement them.

What this book covers

Chapter 1, *Apache CloudStack Architecture*, introduces you to Cloud Computing, the architecture of Apache CloudStack, and the various components of Apache CloudStack along with its various deployment models.

Chapter 2, *Installing Apache CloudStack*, walks you through the installation steps of Apache CloudStack and the setting up of Apache CloudStack in an organization.

Chapter 3, *Apache CloudStack Configuration*, introduces you to the CloudStack management console and the setting up of the IT infrastructure and configuring CloudStack to provide cloud services.

Chapter 4, *Apache CloudStack Networking*, teaches you about the CloudStack networking components and shows you how to set up network offerings in CloudStack and various options available in CloudStack for setting up the networks.

Chapter 5, *Apache CloudStack Storage*, teaches you about the storage architecture in CloudStack, and the various kinds of storage options in the cloud, and also walks you through the steps of creating and managing various storage offerings in the cloud.

Chapter 6, *Service Offerings and Virtual Machines*, teaches you about the various service offerings available in CloudStack and also describes the virtual machine's life cycle in Apache CloudStack.

Chapter 7, *Domains, Accounts, Projects, and Users*, teaches you about the management of domains, accounts, projects, and users in Apache CloudStack.

Chapter 8, *High Availability and Scaling*, teaches you about high availability and scaling configuration options in Apache CloudStack for setting up Apache CloudStack in HA mode as well providing

services in HA. It also introduces you to the various components of Apache CloudStack to maintain high availability.

Chapter 9, *Extending Apache CloudStack and Performance Tuning*, teaches you about the various options and modules of CloudStack so as to extend its functionality and also walks you through the steps to tune the performance of Apache CloudStack.

What you need for this book

To use this book, you will need access to computers or servers that have hardware virtualization capabilities. To set up the lab environments you will need any hypervisor (VMware, XenServer, KVM) installed on at least two servers. You will also need access to an Ubuntu 12.04/ RHEL/ CentOS 6.3+ 64 bit for installing and configuring Apache CloudStack.

Who this book is for

This book is aimed at cloud enthusiasts, cloud architects, system administrators, and technical architects moving from a virtualized environment to cloud environments who are familiar with cloud computing platforms. Knowledge of virtualization and managing Linux environments is expected. Prior knowledge or experience of Apache CloudStack is not required, although it is beneficial.

Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.

Code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles are shown as follows: "The invitations can be enabled in CloudStack by setting the parameter `project.invite.required` to `true` in the global settings page."

A block of code is set as follows:

```
<servlet-mapping>
<servlet-name>apiServlet</servlet-name>
<url-pattern>/api/*</url-pattern>
</servlet-mapping>
```

Any command-line input or output is written as follows:

```
#service cloud-management restart
#service cloud-usage restart
```

New terms and **important words** are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: "The **Storage** tab allows the administrators to create, view, and manage volumes and snapshots".

> ## Note
>
> Warnings or important notes appear in a box like this.

Tip

Tips and tricks appear like this.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply send an e-mail to <feedback@packtpub.com>, and mention the book title via the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on www.packtpub.com/authors.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in the text or the code—we would be grateful if you would report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting http://www.packtpub.com/submit-errata, selecting your book, clicking on the **errata submission form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded on our website, or added to any list of existing errata, under the Errata section of that title. Any existing errata can be viewed by selecting your title from http://www.packtpub.com/support.

Piracy

Piracy of copyright material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works, in any form, on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at <copyright@packtpub.com> with a link to the suspected pirated material.

We appreciate your help in protecting our authors, and our ability to bring you valuable content.

Questions

You can contact us at <questions@packtpub.com> if you are having a problem with any aspect of the book, and we will do our best to address it.

Chapter 1. Apache CloudStack Architecture

To understand Apache CloudStack, it is important to have an understanding of the basic building components of the cloud. In this chapter we will introduce you to the following:

- The concept of cloud computing
- The core components of the cloud
- The basic components of CloudStack—management servers, types of storage, networking architecture, and so on
- Various deployment models of CloudStack

Introducing cloud

Before embarking on a journey to understand and appreciate CloudStack, let's revisit the basic concepts of cloud computing and how CloudStack can help us in achieving our private, public, or hybrid cloud objectives.

Let's start this chapter with a plain and simple definition of cloud. **Cloud** is a shared multi-tenant environment built on a highly efficient, highly automated, and preferably virtualized IT infrastructure where IT resources can be provisioned on demand from anywhere over a broad network, and can be metered. Virtualization is the technology that has made the enablement of these features simpler and convenient. A cloud can be deployed in various models; including private, public, community or hybrid clouds. These deployment models can be explained as follows:

- **Private cloud**: In this deployment model, the cloud infrastructure is operated solely for an organization and may exist on premise or off premise. It can be managed by the organization or a third-party cloud provider.

- **Public cloud**: In this deployment model, the cloud service is provided to the general public or a large industry group, and is owned and managed by the organization providing cloud services.
- **Community cloud**: In this deployment model, the cloud is shared by multiple organizations and is supported by a specific community that has shared concerns. It can be managed by the organization or a third party provider, and can exist on premise or off premise.
- **Hybrid cloud**: This deployment model comprises two or more types of cloud (public, private, or community) and enables data and application portability between the clouds.

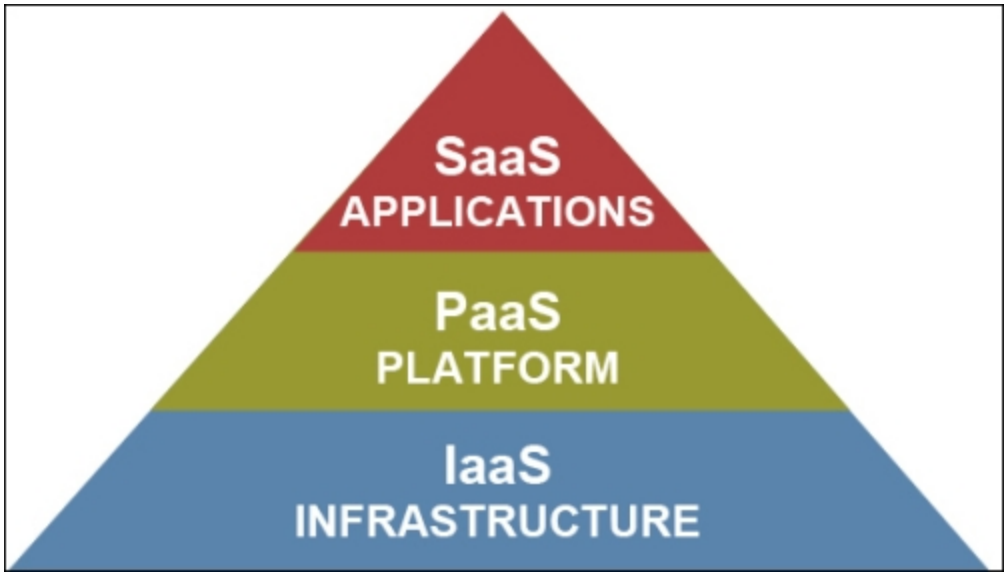
A cloud—be it private, public, or hybrid—has the following essential characteristics:

- On-demand self service
- Broad network access
- Resource pooling
- Rapid elasticity or expansion
- Measured service
- Shared by multiple tenants

Cloud has three possible service models, which means there are three types of cloud services that can be provided. They are:

- **Infrastructure as a service (IaaS):** This type of cloud service model provides IT infrastructure resources as a service to the end users. This model provides the end users with the capability to provision processing, storage, networks, and other fundamental computing resources that the customer can use to run arbitrary software including operating systems and applications. The provider manages and controls the underlying cloud infrastructure and the user has control over the operating systems, storage and deployed applications. The user may also have some control over the networking services.

- **Platform as a service (PaaS)**: In this service model, the end user is provided with a platform that is provisioned over the cloud infrastructure. The provider manages the network, operating system, or storage and the end user has control over the applications and may have control over the hosting environment of the applications.
- **Software as a service (SaaS)**: This layer provides software as a service to the end users, such as providing an online calculation engine for their end users. The end users can access these software using a thin client interface such as a web browser. The end users do not manage the underlying cloud infrastructure such as network, servers, OS, storage, or even individual application capabilities but may have some control over the application configurations settings.



As depicted in the preceding diagram, the top layers of cloud computing are built upon the layer below it. In this book, we will be mainly dealing with the bottom layer—Infrastructure as a service.

Thus providing Infrastructure as a Service essentially means that the cloud provider assembles the building blocks for providing these services, including the computing resources hardware, networking hardware and storage hardware. These resources are exposed to the consumers through a request management system which in turn

is integrated with an automated provisioning layer. The cloud system also needs to meter and bill the customer on various chargeback models. The concept of virtualization enables the provider to leverage and pool resources in a multi-tenant model. Thus, the features provided by virtualization resource pooling, combined with modern clustering infrastructure, enable efficient use IT resources to provide high availability and scalability, increase agility, optimize utilization, and provide a multi-tenancy model.

One can easily get confused about the differences between the cloud and a virtualized Datacenter; well, there are many differences, such as:

- The cloud is the next stage after the virtualization of datacenters. It is characterized by a service layer over the virtualization layer. Instead of bare computing resources, services are built over the virtualization platforms and provided to the users. Cloud computing provides the request management layer, provisioning layer, metering and billing layers along with security controls and multi-tenancy.
- Cloud resources are available to consumers on an on demand model wherein the resources can be provisioned and de-provisioned on an as needed basis. Cloud providers typically have huge capacities to serve variable workloads and manage variable demand from customers. Customers can leverage the scaling capabilities provided by cloud providers to scale up or scale down the IT infrastructure needed by the application and the workload. This rapid scaling helps the customer save money by using the capacity only when it is needed.
- The resource provisioning in the cloud is governed by policies and rules, and the process of provisioning is automated.
- Metering, Chargeback, and Billing are essential governance characteristics of any cloud environment as they govern and control the usage of precious IT resources.

Thus setting up a cloud is basically building capabilities to provide IT resources as a service in a well-defined manner. Services can be provided to end users in various offerings, depending upon the amount of resources each service offering provides. The amount of resources can be broken down to multiple resources such as the computing capacity, memory, storage, network bandwidth, storage IOPS, and so on. A cloud provider can provide and meter multiple service offerings for the end users to choose from.

Though the cloud provider makes upfront investments in creating the cloud capacity, however from a consumer's point of view the resources are available on demand on a pay per use model. Thus the customer gets billed for consumption just like in case of electricity or telecom services that individuals use. The billing may be based on hours of compute usage, the amount of storage used, bandwidth consumed, and so on.

Having understood the cloud computing model, let's look at the architecture of a typical Infrastructure as a Service cloud environment.

Infrastructure layer

The Infrastructure layer is the base layer and comprises of all the hardware resources upon which IT is built upon. These include computing resources, storage resources, network resources, and so on.

Computing resources

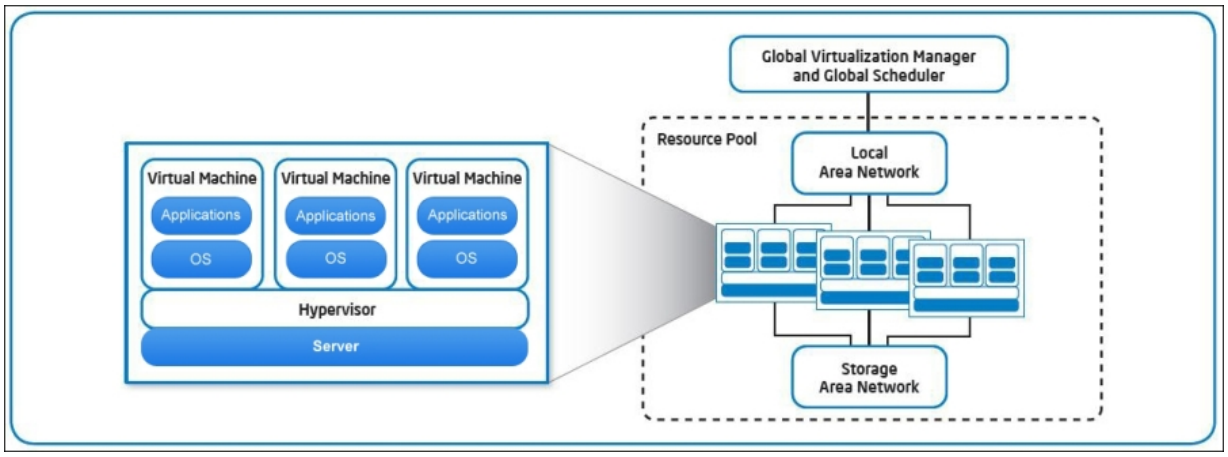
Virtualization is provided using a hypervisor that has various functions such as enabling the virtual machines of the hosts to interact with the hardware. The physical servers host the hypervisor layer. The physical server resources are accessed through the hypervisor. The hypervisor layer also enables access to the network and storage. There are various hypervisors on the market such as VMware, Hyper-V, XenServer, and so on. These hypervisors are

responsible for making it possible for one physical server to host multiple machines, and for enabling resource pooling and multi tenancy.

Storage

Like the Compute capacity, we need storage which is accessible to the Compute layer.

The Storage in cloud environments is pooled just like the Compute and accessed through the virtualization layer. Certain types of services just offer storage as a service where the storage can be programmatically accessed to store and retrieve objects.



Pooled, virtualized storage is enabled through technologies such as **Network Attached Storage** (**NAS**) and **Storage Area Network** (**SAN**) which helps in allowing the infrastructure to allocate storage on demand that can be based on policy, that is, automated.

The storage provisioning using such technologies helps in providing storage capacity on demand to users and also enables the addition or removal of capacity as per the demand. The cost of storage can be differentiated according to the different levels of performance and classes of storage.

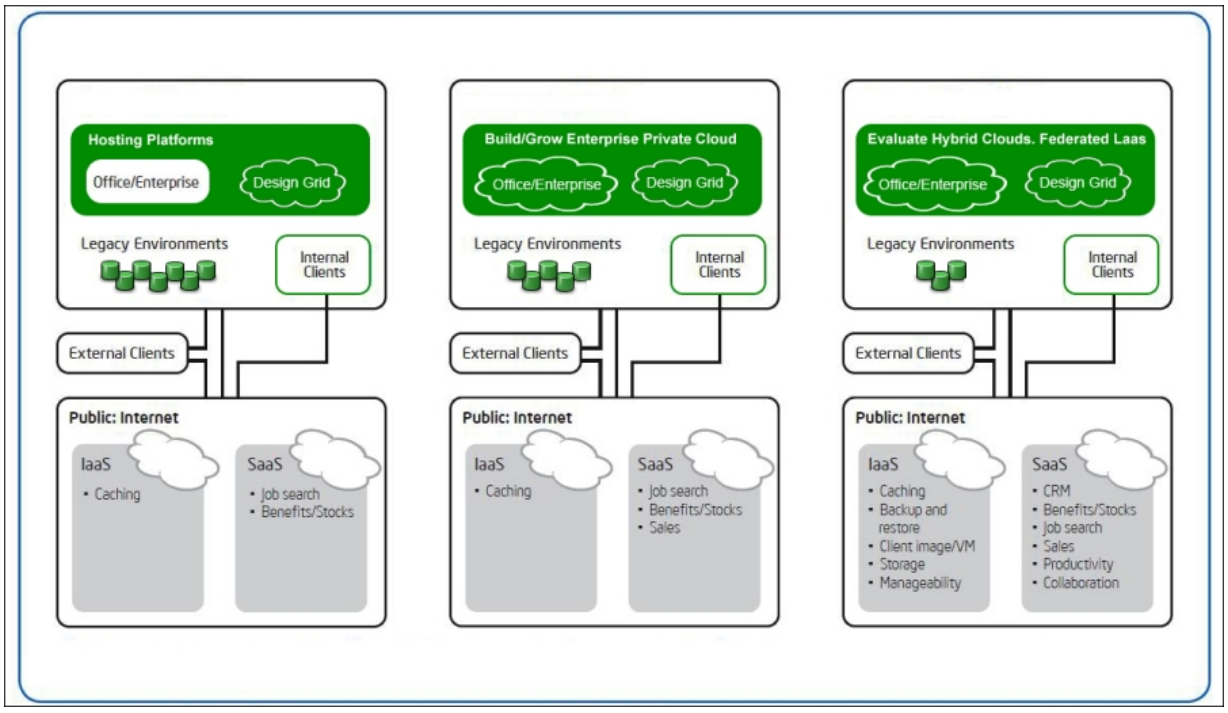
Typically, SAN is used for storage capacity in the cloud where statefulness is required. **Direct-attached Storage** (**DAS**) can be

used for stateless workloads that can drive down the cost of service. The storage involved in cloud architecture can be redundant and prevent the single point of failure. There can be multiple paths for the access of disk arrays to provide redundancy in case connectivity failures.

The storage arrays can also be configured in a way that there is incremental backup of the allocated storage. The storage should be configured such that health information of the storage units is updated in the system monitoring service, which ensures that the outage and its impact are quickly identified and appropriate action can be taken in order to restore it to its normal state.

Networks and security

Network configuration includes defining the subnets, on-demand allocation of IP addresses, and defining the network routing tables to enable the flow of data in the network. It also includes enabling high availability services such as load balancing. Whereas the security configuration aims to secure the data flowing in the network that includes isolation of data of different tenants among each other and with the management data of cloud using techniques such as network isolation and security groups.



Networking in the cloud is supposed to deal with the isolation of resources between multiple tenants as well as provide tenants with the ability to create isolated components. Network isolation in the cloud can be done using various techniques of network isolation such as VLAN, VXLAN, VCDNI, STT, or other such techniques.

Applications are deployed in a multi-tenant environment and consist of components that are to be kept private, such as a database server which is to be accessed only from selected web servers and any other traffic from any other source is not permitted to access it. This is enabled using network isolation, port filtering, and security groups. These services help with segmenting and protecting various layers of application deployment architecture and also allow isolation of tenants from each other.

The provider can use security domains, layer 3 isolation techniques to group various virtual machines. The access to these domains can be controlled using providers' port filtering capabilities or by the usage of more stateful packet filtering by implementing context switches or firewall appliances. Using network isolation techniques such as VLAN tagging and security groups allows such

configuration. Various levels of virtual switches can be configured in the cloud for providing isolation to the different networks in the cloud environment.

Networking services such as NAT, gateway, VPN, Port forwarding, IPAM systems, and access control management are used in the cloud to provide various networking services and accessibility. Some of these services are explained as follows:

- **NAT**: Network address translation can be configured in the environment to allow communication of a virtual machine in private network with some other machine on some other network or on the public Internet. A NAT device allows the modification of IP address information in the headers of IP packets while they are transformed from a routing device. A machine in a private network cannot have direct access to the public network so in order for it to communicate to the Internet, the packets are sent to a routing device or a virtual machine with NAT configured which has direct access to the Internet. NAT modifies the IP packet header so that the private IP address of the machine is not visible to the external networks.
- **IPAM System/DHCP**: An IP address management system or DHCP server helps with the automatic configuration of IP addresses to the virtual machines according to the configuration of the network and the IP range allocated to it. A virtual machine provisioned in a network can be assigned an IP address as per the user or is assigned an IP address from the IPAM. IPAM stores all the available IP addresses in the network and when a new IP address is to be allocated to a device, it is taken from the available IP pool, and when a device is terminated or releases the IP address, the address is given back to the IPAM system.
- **Identity and access management**: A access control list describes the permissions of various users on different resources in the cloud. It is important to define an access control list for users in a multi-tenant environment. It helps in

restricting actions that a user can perform on any resource in the cloud. A role-based access mechanism is used to assign roles to users' profile which describes the roles and permissions of users on different resources.

Use of switches in cloud

A switch is a LAN device that works at the data link layer (layer 2) of the OSI model and provides multiport bridge. Switches store a table of MAC addresses and ports. Let us see the various types of switches and their usage in the cloud environment:

- **Layer 3 switches**: A layer-3 switch is a special type of switch which operates at layer 3—the Network layer of the OSI model. It is a high performance device that is used for network routing. A layer-3 switch has a IP routing table for lookups and it also forms a broadcast domain. Basically, a layer-3 switch is a switch which has a router's IP routing functionality built in.

  A layer-3 switch is used for routing and is used for better performance over routers. The layer-3 switches are used in large networks like corporate networks instead of routers. The performance of the layer-3 switch is better than that of a router because of some hardware-level differences. It supports the same routing protocols as network routers do. The layer-3 switch is used above the layer-2 switches and can be used to configure the routing configuration and the communication between two different VLANs or different subnets.

- **Layer 4-7 switches**: These switches use the packet information up to OSI layer 7 and are also known as content switches, web-switches, or application switches. These types of switches are typically used for load balancing among a group of servers which can be performed on HTTP, HTTPS, VPN, or any TCP/IP traffic using a specific port. These switches are used in the cloud for allowing policy-based switching—to limit the different amount of traffic on specific

end-user switch ports. It can also be used for prioritizing the traffic of specific applications. These switches also provide forwarding decision making like NAT services and also manages the state of individual sessions from beginning to end thus acting like firewalls. In addition, these switches are used for balancing traffic across a cluster of servers as per the configuration of the individual session information and status. Hence these types of switches are used above layer-3 switches or above a cluster of servers in the environment. They can be used to forward packets as per the configuration such as transferring the packets to a server that is supposed to handle the requests and this packet forwarding configuration is generally based on the current server loads or sticky bits that binds the session to a particular server.

- Layer-3 traffic isolation provides traffic isolation across layer-3 devices. It's referred to as **Virtual Routing and Forwarding** (**VRF**). It virtualizes the routing table in a layer-3 switch and has set of virtualized tables for routing. Each table has a unique set of forwarding entries. Whenever traffic enters, it is forwarded using the routing table associated with the same VRF. It enables logical isolation of traffic as it crosses a common physical network infrastructure. VRFs provide access control, path isolation, and shared services. Security groups are also an example of layer-3 isolation capabilities which restricts the traffic to the guests based on the rules defined. The rules are defined based on the port, protocol, and source/destination of the traffic.
- **Virtual switches**: The virtual switches are software program that allows one guest VM to communicate with another and is similar to the Ethernet switch explained earlier. Virtual switches provide a bridge between the virtual NICs of the guest VMs and the physical NIC of the host. Virtual switches have port groups on one side which may or may not be connected to the different subnets. There are various types of virtual switches used with various virtualization technologies such as VMware Vswitch, Xen, or Open Vswitch. VMware also provides a distributed virtual switch which spans multiple

hosts. The virtual switches consists of port groups at one end and an uplink at the other. The port groups are connected to the virtual machines and the uplink is mapped to the physical NIC of the host. The virtual switches function as a virtual switch over the hypervisor layer on the host.

Management layer

The Management layer in a cloud computing space provides management capabilities to manage the cloud setup.

It provides features and functions such as reporting, configuration for the automation of tasks, configuration of parameters for the cloud setup, patching, and monitoring of the cloud components.

Automation

The cloud is a highly automated environment and all tasks such as provisioning the virtual machine, allocation of resources, networking, and security are done in a self-service mode through automated systems.

The automation layer in cloud management software is typically exposed through APIs. The APIs allow the creation of SDKs, scripts, and user interfaces.

Orchestration

The Orchestration layer is the most critical interface between the IT organization and its infrastructure, and helps in the integration of the various pieces of software in the cloud computing platform.

Orchestration is used to join together various individual tasks which are executed in a specified sequence with exception handling features. Thus a provisioning task for a virtual machine may involve various commands or scripts to be executed. The orchestration engine binds these individual tasks together and creates a

provisioning workflow which may involve provisioning a virtual machine, adding it to your DNS, assigning IP Addresses, adding entries in your firewall and load balancer, and so on.

The orchestration engine acts as an integration engine and also provides the capabilities to run an automated workflow through various subsystems. As an example, the service request to provision cloud resources may be sent to an orchestration engine which then talks to the cloud capacity layer to determine the best host or cluster where the workload can be provisioned. As a next step, the orchestration engine chooses the component to call to provision the resources.

The orchestration platform helps in easy creation of complex workflows and also provides ease of management since all integrations are handled by a specialized orchestration engine and provide loose coupling.

The orchestration engine is executed in the cloud system as an asynchronous job scheduler which orchestrates the service APIs to fulfill and execute a process.

Task Execution

The Task execution layer is at the lower level of the management operations that are performed using the command line or any other interface. The implementation of this layer can vary as per the platform on which the execution takes place. The activity of this layer is activated by the layers above in the management layer.

Service Management

The Service Management layer helps in compliance and provides means to implement automation and adapts IT service management best practices as per the policies of the organization, such as the **IT Infrastructure Library** (**ITIL**). This is used to build processes to

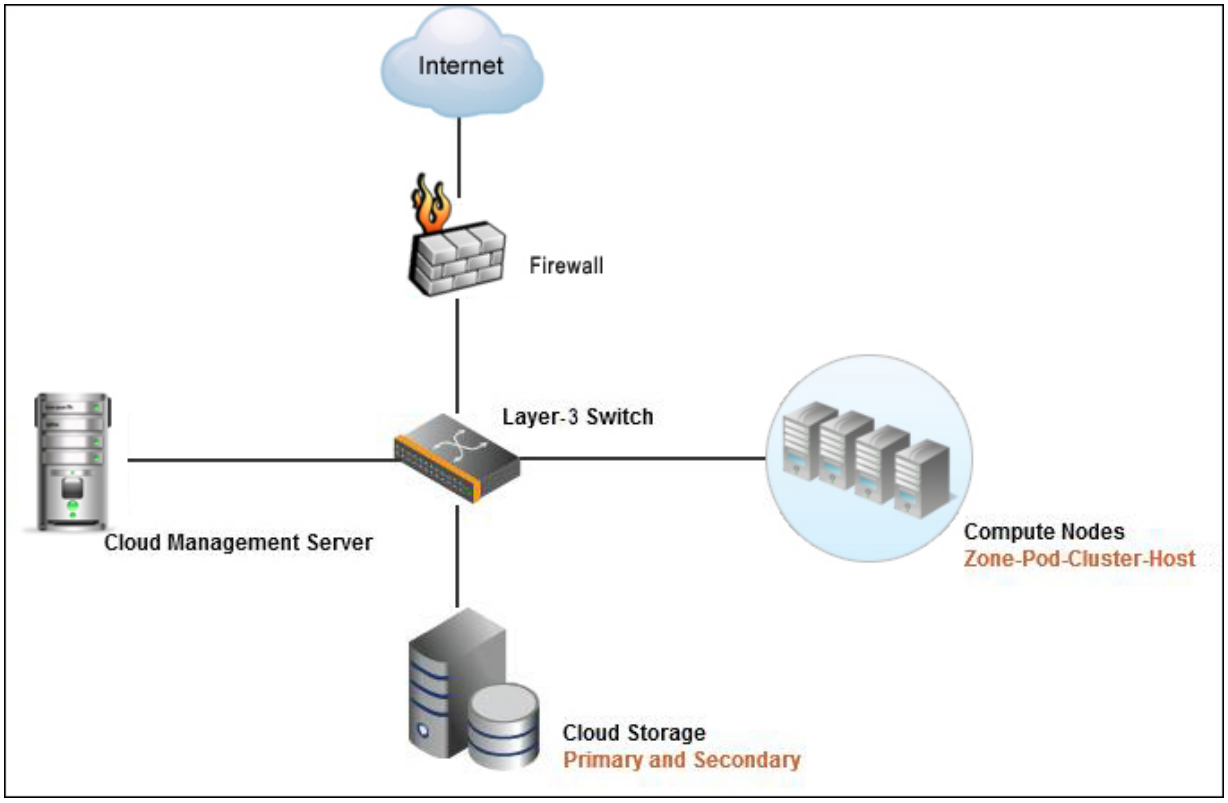
implement different types of incident resolutions and also provide change management.

The self service capability in cloud environment helps in providing users with a self-service catalog which consists of various service options that the user can request and provision resources from the cloud. The service layer can be comprised of various levels of services such as basic provisioning of virtual machines with some predefined templates/configuration, or can be of an advanced level with various options for provisioning servers with configuration options as well.

Understanding CloudStack modules

Having gone through the various components in the cloud setup, let's look at the CloudStack platform that offers a complete platform to set up a Private or Hybrid cloud for enterprises.

CloudStack is a solution or a platform for IT infrastructure as a service that allows to pool computing resources which can be used to build public, private and hybrid IaaS cloud services that can be used to provide IT infrastructure such as compute nodes (hosts), networks, and storage as a service to the end users on demand.



In this section we are going to discuss the underlying architecture of CloudStack that makes it possible to set up on demand computing services which can be leveraged as a public, private, or hybrid cloud

and makes requesting, provisioning, configuring, and managing the IT infrastructure possible.

The architecture of CloudStack follows a hierarchical structure, which can be deployed in order to manage thousands of physical servers from a single management interface. Users can either define a basic networking zone or can configure an advanced networking zone with the desired networking features and capabilities. CloudStack also allows integration with public cloud AWS by using the APIs of some of its services.

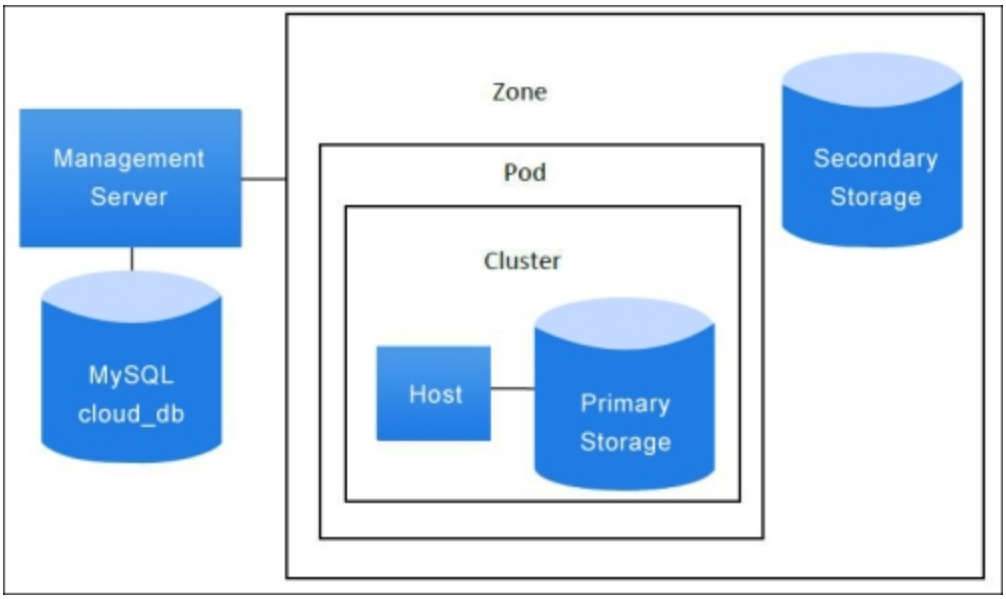
The basic CloudStack deployment, commonly known as **dev-cloud**, consists of a single management server with or without CloudDB in a machine that can be used to manage a group of hypervisor host(s).

The management server controls or collaborates with the hypervisor layers on the physical hosts over the management network and thus controls the IT infrastructure.

Let's look at the architecture of private cloud infrastructure that the CloudStack management platform will manage. The physical servers in the various datacenters that are to be managed using CloudStack platform are segregated into various levels of logical partitions that help in managing them easily. Partitioning of the cloud into various zones, pods, and clusters makes them manageable. We will first look at the architecture, how we will set up the private cloud, and then focus on how CloudStack controls the underlying cloud infrastructure.

Cloud deployment model

The CloudStack deployment model covers the basic components of CloudStack, using which CloudStack provides all the functionalities; it also covers the logical segregation of resources to help better manage them.



Zones

Zones are the highest level of hierarchy in which the datacenter of an organization is logically partitioned to provide physical isolation and redundancy. The zone may be a complete datacenter that has a geographic location with its own power supply and network uplinks. The zone can also be distributed across various geographic locations. Typically a datacenter contains a single zone but it can also contain multiple zones. The logical division into zones enables the instances and the data stores at a specified location to be compliant with an organization's data storage policies, and other compliance policies, and so on.

A zone is divided into various other logical units. A zone comprises of following:

- At least one or more pods. A pod is the second level of hierarchical unit discussed later which consists of one or more clusters. We will discuss this in detail next.
- Secondary storage, this storage is shared by all the pods in a zone.

End users can select a zone while requesting a guest VM.

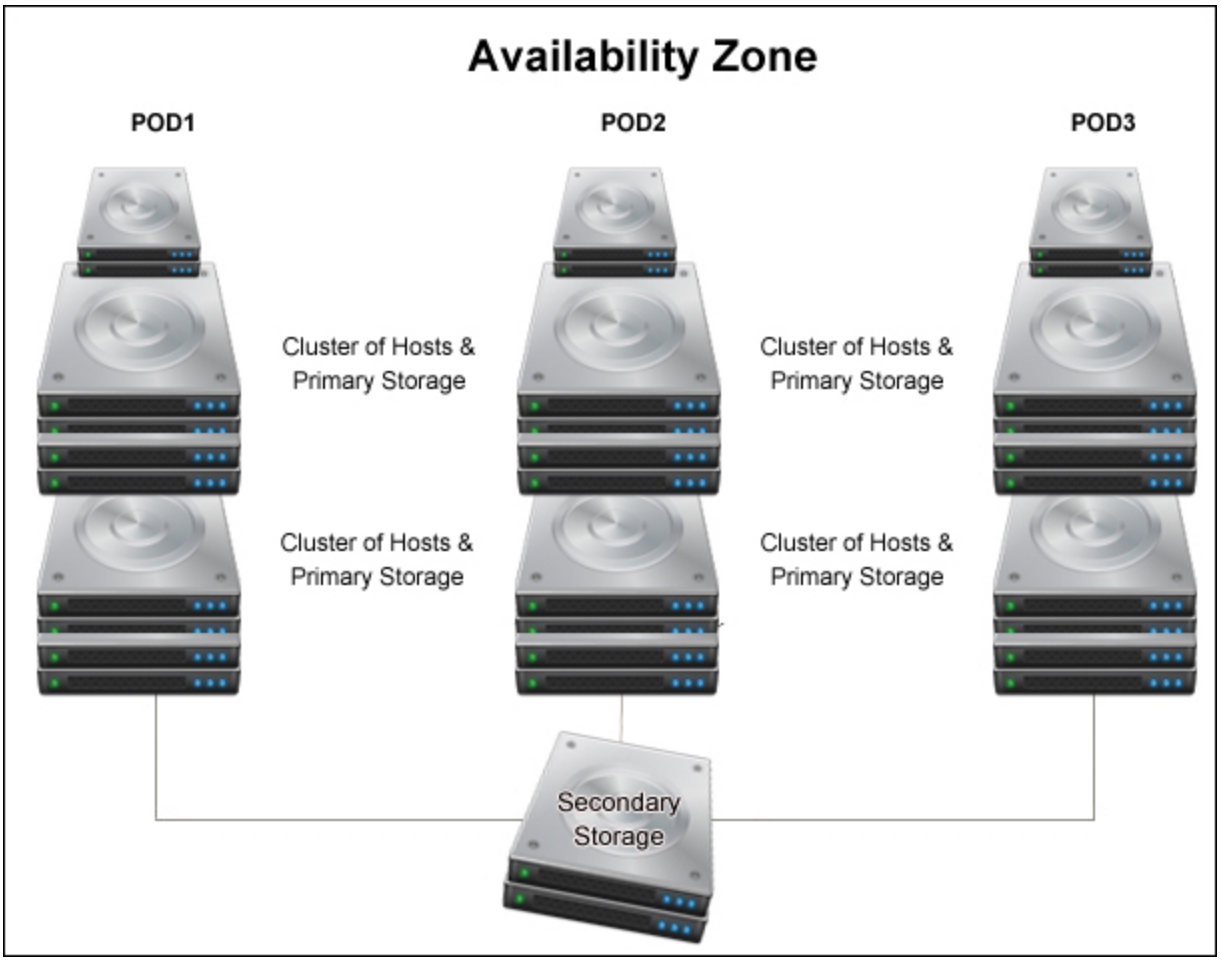
A zone can be associated with domains; a zone can be public or private. A public zone is visible to all the users whereas private zones can be reserved for some specific domain which means that the users that belong to that domain or its subdomain have the permissions to create guest VMs in that private zone.

The pod(s) in a zone consists of one or more clusters which in turn contains host(s).

The hosts in a zone can access and be accessed by other hosts in that zone if there's no restriction from any firewall and a flat network is defined, but if a host in a zone tries to access hosts in other zones, they have to follow VPN tunnels which are configured with firewall rules. Among all the other traffic, only key management traffic is allowed in between the hosts in different clusters.

The zones must be configured by the cloud administrator with the number of pods in the zone, number of clusters in a pod, and number of hosts in each of the clusters. The clusters also contain primary storage servers so the cloud administrator must also decide upon the number and the capacity of primary storage servers that are to be placed in each of the clusters. As a minimum configuration, each zone must have at least one pod, each pod must have at least one cluster with at least one host, the cluster must have at least one primary storage. The capacity of the secondary storage that is used by all the hosts in all the pods of a zone is also to be configured by the administrator.

Once these components are configured they are consumed by the management server to provide cloud services to the customers. A zone is hypervisor specific—a zone will only consist of hosts with the same hypervisor only. Hosts with different hypervisors are added to different zones.



Pods

A pod is the second level of hierarchy in the deployment of the CloudStack. A zone is further logically divided into one or more entities known as pod. A pod can be assumed to be like a physical rack in a datacenter in which all the hosts reside in the same network subnet. The pod defines the management subnet of the system VMs (discussed later), this network is used by the CloudStack management server communication. A pod contains one or more clusters and a cluster in turn contains one or more hosts. All the hosts that are inside a pod are configured to be in the same subnet. There are one or more primary storage servers in a pod. A pod is invisible to users—the users don't know which pod their machine is being provisioned in to. The logical division into pods is for administrative purposes only. As with the zone, all the hosts in a pod

will have hosts with same hypervisor type as defined during the creation of zone and these hosts are in the same subnet.

Clusters

A cluster is the third level of hierarchical division in the deployment of CloudStack inside pods. The hosts are grouped into a logical group called clusters. This cluster can be a XenServer server pool, VMware cluster that is preconfigured in the VCenter, or a set of KVM servers.

Hosts within a cluster are accessible to each other and guests residing on a host in a cluster can also be "live migrated" to another host in the same cluster using a shared storage device. The live migration occurs without any interruption to the users and thus provides high availability option. In order to support live migration of VMs between hosts in a cluster, those hosts should have similar properties such as using the same hypervisor version and the same hardware. They should also have access to the same primary storage servers that are shared among the hosts in a cluster. A cluster contains at least one primary storage server.

Storage

In this section we will discuss the various storage options available in the CloudStack system.

Primary storage

A cluster contains at least one primary storage server that is shared among all the hosts in that cluster. This storage server is among the critical component of the cluster and is used to host the guest virtual machine instances. The primary storage is unique to each of the clusters inside the pods.

The primary storage can be a SAN such as iSCSI, NAS such as NFS or CIFS, or it can be a DAS such as VMFS or EXT file systems. The primary storage should be supported by the underlying hypervisor. Building primary storage on high performance hardware

with multiple high speed disks increases the performance. The primary storage should also be reachable from all the hosts in the cluster. This storage is used to host the guest virtual machines in the cluster stores. The volume of these virtual machines and the allocation of guest virtual disks to the primary storage managed by CloudStack.

The primary storage server is basically a machine with a large quantity of disk space, the capacity of which is dependent on the users' need. Primary storage can be a shared storage server or local storage, and hosts the storage for guest virtual machines.

The primary storage pool can be created using any of the technologies mentioned earlier for any given cluster. If it is created using iSCSI LUN or an NFS share, the CloudStack management server asks each of the hosts' hypervisors in the cluster to mount the NFS share or the LUN. The hypervisor then communicates with the primary storage pool as it is presented as a datastore in case of VMware, storage repository in case of a XenServer or a mount point in case of KVM.

The hosts are recommended to have a dedicated management interface for communication with the primary storage pool. The mechanism for making an additional interface for the host for primary storage traffic is to create a management interface. The management interface of the hosts and primary storage interfaces must be in the same CIDR.

The primary storage pool provides storage for the root volumes and the disk spaces for the guest VMs. When a guest virtual machine is created, its root volume is automatically created from this primary storage. When the VM is deleted, the data volumes associated with it are disabled and this VM can also be recovered afterwards. It thus provides security by ensuring no data is shared or made available to a new guest in a multi-tenant environment by deleting the data on deletion of a virtual machine.

These storage servers provide volume storage to the running virtual machines. The volumes can be the root device or other data volumes attached to the machine. These extra volumes can be attached to the virtual machine dynamically. Except for Oracle VM hypervisor, data volumes can be attached to the guest dynamically only when the machine is in stopped state. After the VM instance is destroyed, the VM instance is disabled from the start action, the VM instance is still in the database and volume is not yet deleted. The destroyed VM can also be recovered. After the destroyed state, there is an expunged state which signifies permanent deletion of the volume. The time for expunging is stored as 86400 seconds by default, but this can be changed. The administrator sets up and configures the iSCSI on the host not only the first time but also during the recovery of the host from a failure—whenever there is a host failure, the administrator has to set up and configure the iSCSI LUNs on the host again.

In case of XenServer, it uses clustered logical volume manager to store the VM images on iSCSI and fiber channel volumes. In this case CloudStack can support over provisioning only if the storage server itself allows that, otherwise over-provisioning is not supported. In case of KVM, shared mount point storage is supported but the mount path must be same across all the hosts in the cluster. The administrator must ensure that the storage is available otherwise CloudStack will not attempt to mount or unmount the storage.

With NFS storage, CloudStack manages the over provisioning of storage independent of the hypervisor.

The primary storage can also be a pool of local storage in case of vSphere, XenServer, OVM, and KVM. CloudStack supports multiple primary storage pools in a cluster. It also supports dynamic addition of storage servers as your requirements increase.

As primary storage is a critical component, its capacity must be monitored by the administrator and additional storage space must be attached to it when needed. This can be done by creating a storage pool that is associated with the clusters. Additional capacity can be

added by adding additional iSCSI LUN to the storage when the primary storage is iSCSI or an additional NFS server can be added to the primary storage when the first one reaches its size limit. Thus CloudStack supports multiple storage pools in a cluster as well as a single SAN or a storage server can also be used to provide primary storage to multiple clusters.

Volumes

Volumes are the basic unit of storage in CloudStack. All the storage space that is provided to the guest instance is provided in the form volumes. These volumes are created from the primary storage servers that are described as above.

The additional storage space and the storage space for the root disk drives of the VMs are provided by volumes. These volumes are dependent upon the type of hypervisor, because the disk image format for different hypervisors are different. So the volume that has been created for a guest of a hypervisor type cannot be attached to a guest VM using another type of hypervisor.

The guest VMs' root disk is provided storage in the form of volumes from the primary storage that contains all the files for booting the OS or additional storage for storing data. There can be multiple additional volumes mounted to a guest VM. The users are provided with multiple disk offerings (discussed later) that are pre-created by the administrator and which users can select to create different types of volumes. In addition, a volume can be used to create templates. A volume can be detached or attached to any instance but they must be of same hypervisor type.

The volumes are provided to the virtual machines from the primary storage. CloudStack doesn't provide the functionality of backing up the primary storage but it does provide the functionality for backing up of individual volumes in primary storage using snapshots.

Secondary storage

This storage space is used to store the templates, ISO images and snapshots that can be used to deploy IT infrastructure using CloudStack. Every zone has at least one secondary storage server and this secondary storage(s) is shared by all the Pods in that zone.

CloudStack also provides the functionality to automatically replicate the secondary storage across zones so that one can implement a disaster recovery solution by backing up the application across zones, allowing easy recovery from a zone failure. Thus, CloudStack provides a common storage solution across the cloud. Unlike primary storage, secondary storage only uses **Network File System** (**NFS**) as it is to be accessed by all the hosts in the clusters across the zones irrespective of the hypervisors on the hosts.

The secondary storage is used to store templates that are basically OS images that are used to boot VMs with some more additional configuration information and installed applications.

Secondary storage also stores ISO images that are disk images used for booting operating system and disk volume snapshot, used for backing up the data of VMs for data recovery or for creating new templates. These items are available to all the hosts in one zone.

A secondary storage device can be an OpenStack object (swift) with at least 100 GB of space, an NFS storage appliance or a Linux NFS server. The secondary storage and the VMs that it is serving must be located in the same zone and should be accessible by all the hosts in that zone. The scalability of a zone is dependent upon the capacity of the secondary storage device. A disk with high read:write ratio, larger disk drives, and lower IOPS than primary storage is best suited for secondary storage.

The administrators can change the secondary storage server afterwards or it can be replaced with a new one after implementation; to achieve this one just needs to copy all the files from the old one to the new one.

CloudStack management server

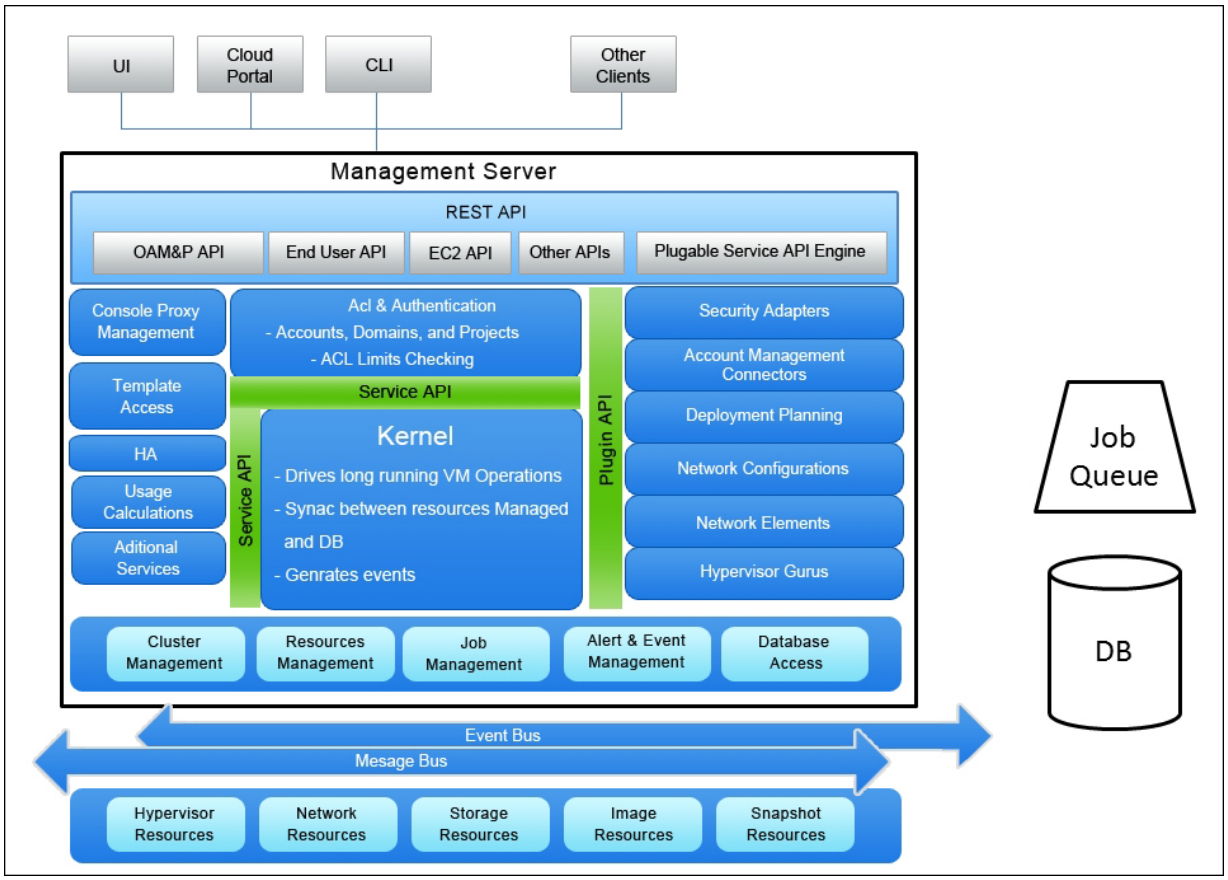
The CloudStack management server helps us to manage the IT infrastructure as defined in the previous sections. The CloudStack management server provides a single point of configuration for the cloud. Basic functionalities of the management server include:

- The web user interface for both the administrator and the users is provided by the management server
- The APIs for the CloudStack are also provided by the management server
- The assignment of guest VMs to particular hosts is also provided by it
- The public and private IP addresses to particular accounts are provided by management server
- Storage to guests as virtual disks is allocated by management server
- The functionalities such as snapshots, templates and ISO images, and their replication across multiple datacenters is also managed by the management server
- It acts as a single point of configuration for the cloud

The management server provides an easy to use web user interface for administrators and the users who can leverage it to manage and request IT infrastructure on demand. It also provides the users and administrators with CloudStack APIs. The management server can be configured to be highly available to prevent single point of failure. This is achieved by deploying it in a multi-node configuration and placing it behind a load balancer so that multiple requests must be served by multiple nodes acting as the management server. High availability can be ensured for the CloudDB by setting up a MySQL cluster. Setting up the CloudStack in this fashion, with a highly available management server and highly available CloudDB provides no single point of failure.

Even though the management server is configured in a multi-node configuration, the operation of the guest VMs is not impacted due to its outage, but the creation of new VMs and management of existing VMs through the management server cannot be done and all the

interfaces along with dynamic load balancing and HA will also stop working. The CloudStack Management server is comprised of various parts which handle the functionalities of the cloud.



The basic units of CloudStack management server are:

- **Interface**: The User Interface and Application Program interface. The management server provides different types of interfaces for both administrators and users. The user interface is the Management server console which the admins and end users use for configuring, requesting and provisioning IT infrastructure. The API is used for programmatic access to the server's functionalities.
- **Business Logic**: This part takes care of the business logic and sits below the interfaces. When the user or admins requests any kind of operation, it is processed through the business logic and then passed down to the Orchestration

engine to perform the operation. For example, if a user requests a VM using the API or UI, the request is passed through the business logic section to process the necessary information such as the host the VM is to be deployed to, the workflow process, the required user authentication to perform that operation, the availability and/or applicability of required network configuration, and so on. The request is then passed down to the orchestration engine that performs the operation to create that virtual machine.

- **Orchestration Engine**: The orchestration engine is critical to the CloudStack deployment and is used for configuring, provisioning and scheduling any operations. After the request or command is processed by the business logic, the request is passed to the orchestration engine for processing. The orchestration engine is responsible for performing that action, for example provisioning virtual machine and configuring them, allocating storage and so on. The orchestration engine also helps in scheduling operations.
- **Controllers**: The controllers are basic underlying parts of the CloudStack management server that talks to the underlying hypervisor or hardware for compute, network, and storage resources. These controllers, also known as "Gurus" or "Providers" help in provisioning resources that are requested by the admin or the user and are used to build the guest virtual machine as per the request of the user.

On a more granular level, the basic functions of the management server are divided among the server's various modules, which are described as follows.

API layer

The API layer is the top-most layer of the CloudStack management server that the management server listens to. It is basically the call to the functional components of the CloudStack. It passes on this call to the concerned component of the CloudStack. The various API calls can be among OAM&P API, EC2 API, End User API or it can be

any other pluggable service API engine. The translation of third-party APIs such as Amazon Web Services is done using the CloudBridge (discussed later). These API calls are fired as per the request by the users or the admin for the execution of some task; the CloudStack management server is responsible for executing the given task as per the request.

Access control

The access control component of the management server is responsible for the access control and authentication of the users requesting services. This layer is the second layer, just beneath the API layer, which cross-checks the authorization of the users requesting the action. The user must authenticate, and the access control component maps the users to the domain, project, and other groups to which the user belongs. The request action should always have an authentication token that authorizes the user for the action and also specifies the permissions, which indicate whether he has the rights to perform the action that he is requesting. This is also recorded in the logs of the actions performed.

Kernel

The kernel is made up of several different components performing tasks in silos. It is the central component for distributing, integrating, and handling the tasks and operations going in and out. The kernel distributes the tasks among the various other components of the CloudStack and drives long-running VM operations, performs sync between the resources, and the database, generates events that are caught by different components and performs actions based on it.

- **Virtual Machine Manager**: The virtual machine manager is responsible for the management of the virtual machines in the CloudStack environment. All the virtual machine requests, requirements, and states are handled by the virtual machine manager. It is responsible for the management of the resources allocated to the virtual machines in the CloudStack environment. It also manages the live migration, and other

actions that are to be performed on the virtual machines such as start, stop, delete, assign IP addresses, and so on. In addition, it ensures that resources are allocated to the virtual machines, as per their needs or specifications.

- **Storage Manager**: This component of the management server is responsible for the management of storage space attached to the CloudStack as resource. It creates, allocates, or deletes the storage volumes or space as per the end users' request. The storage manager is responsible for all the actions that are concerned with the storage from the users or virtual machines. The storage manager evaluates requests from users and performs the specific action to the storage resources or generates an error if necessary.
- **Network Manager**: Network manager handles all the networking of the virtual machines in the CloudStack environment. The network manager is responsible for managing the network configurations of the virtual machines and any other resources in the environment. It has functions such as IP address management, load balancing, firewall configuration, and others that are performed as per the user's request. These configuration operations are predefined in the services that the user chooses. The users are unaware of the operations that are being performed in the back end by such managers.
- **Snapshot Manager**: The snapshot manager is the component responsible for managing the snapshot of the virtual machines or any other resources in the environment. When a user requests an action for creating a snapshot or any other operations based on snapshots, such as creating a virtual machine using a snapshot, this component takes care of the request. Snapshots are used for backups and restoration. They are taken on the primary storage and then moved onto the secondary storage. There are basically two types of snapshot; **incremental snapshot**, where the snapshot of only modified data is taken since the last snapshot and **full snapshot**, where full snapshot of the service is taken every time.

- **Async Job Manager**: The jobs that take place in the CloudStack can be synchronous or asynchronous. This component manages the asynchronous jobs that are requested and are to be performed. Commands are usually executed asynchronously; the manager schedules the jobs as per the priority.
- **Template Manager**: The template manager is responsible for handing templates and their operations. Whenever there is a request for creating template, creating VM from a given template, deleting template, and such other tasks. The template manager is notified of the same and it handles all the operations pertaining to it.

Below all the components lie some more core components that provide the end-to-end interaction possible in CloudStack. Some of these are discussed as follows:

- **Agent Manager**: Agents are very critical resource to the Cloud architecture. Agents are deployed in all the resources that are managed in the CloudStack environment. They provide communication channel between the management server and the resources. They provide information, as well as assist in performing operations on the resources.
- **Resource Layer**: The resource layer is the layer which provides fuel to our engine, i.e. the resources. The resources can be of many kinds such as XenServer resources, KVM resources, VSphere resources, F5 resources, and so on.

The CloudStack management server is the core component that is responsible for managing the actions that make the whole deployment a success. Let's take a close look into the process flow within the CloudStack—when a new request comes in, how is it fulfilled?

CloudStack operations

The user is presented with a number of options with respect to the interface through which he or she can submit his/her request or demand. There are user interfaces such as CloudPortal, Command Line Interface, API calls, or any other clients. The user submits the request using such a console.

When a request is submitted by the user, that request is authenticated and the access rights are checked to confirm the user's right to perform the specified request. If the user's authentication fails, the access control layer of the CloudStack management server denies further processing of the request. If the user's request is successfully authenticated, the request is passed by the Access layer to the kernel.

Security check

After all the security checks are passed, the request is passed down to the kernel and the kernel distributes the tasks to its different components for execution one by one. Let's take an example of a user requesting a new virtual machine service with some software packages installed on it. After passing through the security checks, the request is passed to the virtual machine manager.

The virtual machine manager

The virtual machine manager is responsible for the deployment plan for provisioning the virtual machine such as the host to which this machine is to be created on, to which cluster the host belongs, the pod and the zone of the cluster. The virtual machine manager then starts with the creation of virtual machine by allocating the resources from the host to the virtual machine.

The virtual machine manager initiates the creation of NICs that have to be attached to the virtual machine and assigns this task to the network manager. The network manager takes care of the preparation of NICs and the assignment of IP addresses that are to be attached to the virtual machine that is being created.

The virtual machine manager also triggers the storage allocation to the virtual machine, and requests the storage manager to create volume specified. This volume is then attached to the new VM.

If the virtual machine is to be created using a template, then the template manager is contacted for the details and other resources such as the OS, packages, and other resources to be associated with the request.

Server resources

After all the resources are allocated, the request is passed on to the deployment planner and the server resources. The server resource helps in translation of the CloudStack commands to the resources' APIs. The resource APIs perform tasks as per the instructions and create the virtual machine.

Installation

Once the virtual machine is created, the OS and the software packages are installed on it. The software packages also include the agent that is to be deployed inside the virtual machine, which helps the virtual machine and the CloudStack management server to communicate with each other.

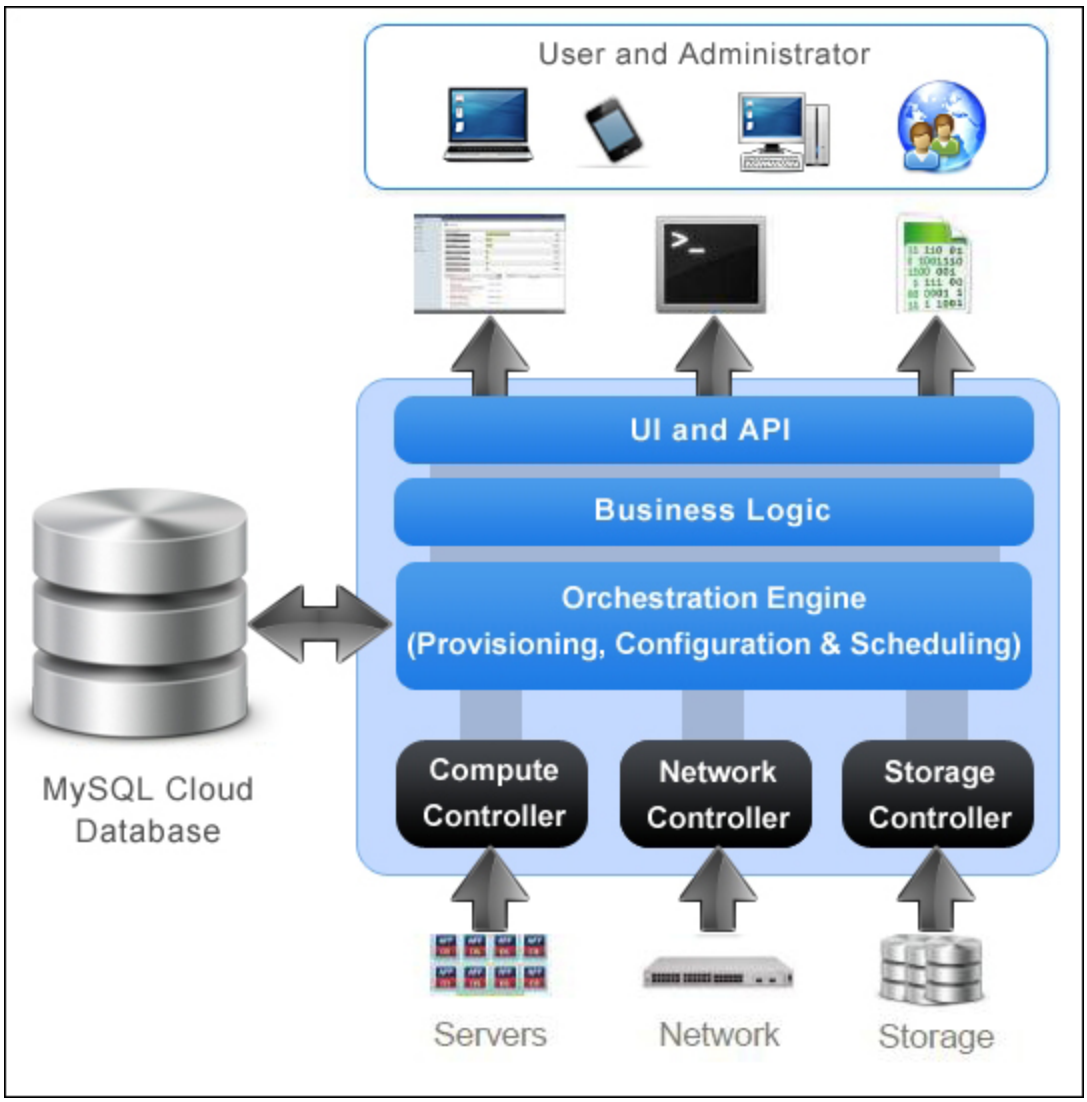
Job result

After the request is provisioned, the job status is reported back to the user. The user gets the details of the virtual machine that has been newly created upon his request. He can now log in to the virtual machine and use it as he wishes. The job results and the properties of the virtual machine are stored in the databases. The logs of the process are stored for reference and the database records for the resource are updated.

CloudDB

The CloudDB is the primary MySQL database that is used by management server in the CloudStack deployment for storing all the configuration information. The CloudDB can be installed on the same server or on a different server. The CloudDB can also be set up as a MySQL cluster for high availability. The CloudStack management server communicates with the master database and the replicas are in sync with the master continuously. The administrator must configure the MySQL database or CloudDB with a username and password for security.

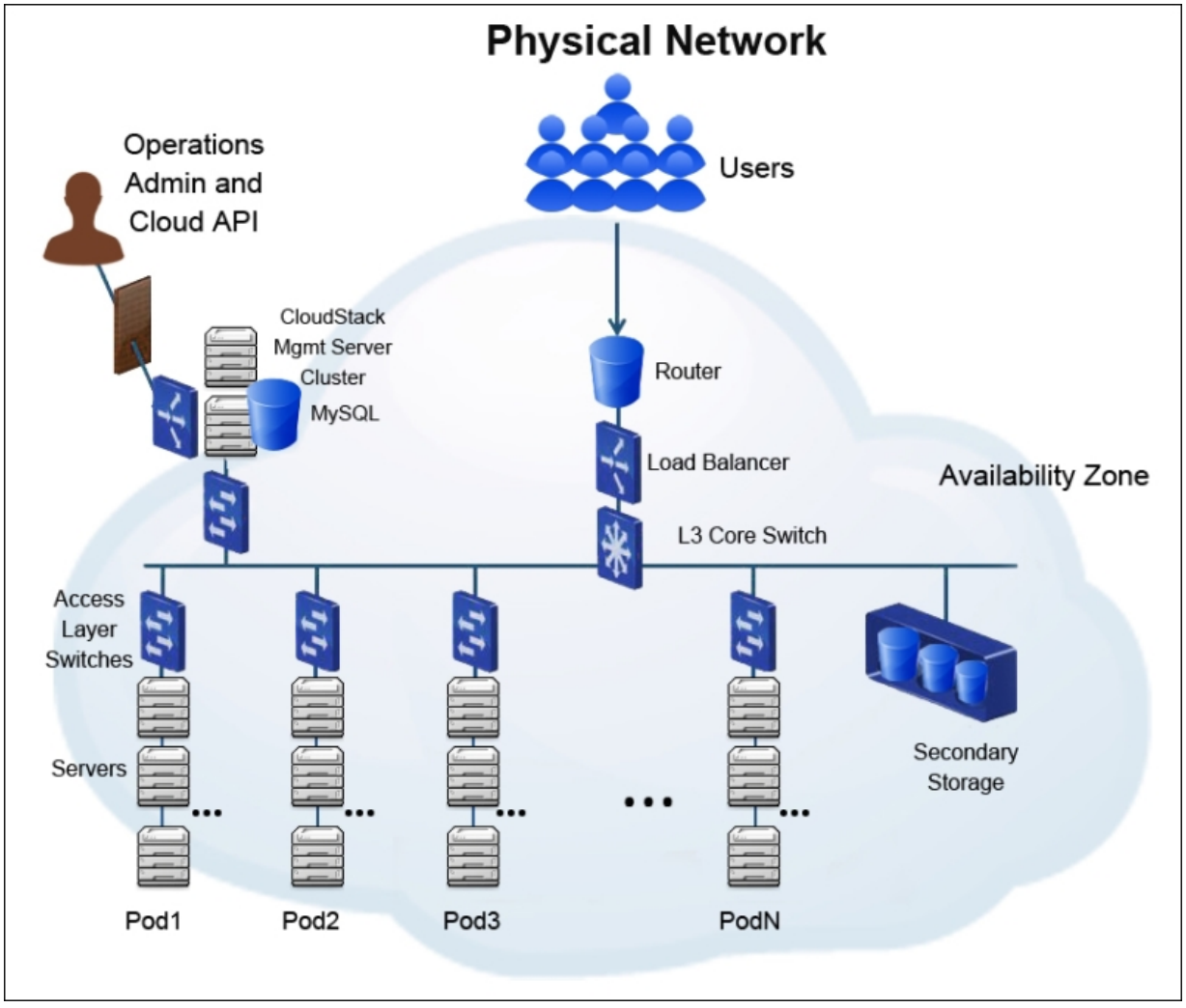
The administrator can also provide MySQL replication in order to provide manual failover in case of database failure.



CloudDB is a critical component for the working of CloudStack as it contains information about the offering, hosts' profiles, accounts credentials, configuration information, network information, and so on. The database is accessed for information on an on-demand basis, which shows the criticality of the management server and database and the need to configure them properly.

CloudStack networking architecture

There are two main types of network configurations that can be deployed using CloudStack: the basic and advanced types.



The basic type is similar to AWS level 3 isolation using security groups. The security groups ensure the isolation and the egress and

ingress of the traffic. The IPAM system manages the IP addresses and tenants don't usually get a contiguous IP address or subnet, the assignment of IP addresses to tenants is basically random from the IPAM system. This configuration has the capability to scale to configure millions of virtual machines on thousands of hosts.

The advanced type offers full level 3 subnets where security and isolation are provided using VLANs, **Stateless transport tunneling** (**STT**), and **Generic Routing Encapsulation** (**GRE**). Other features such as NAT, VPN, and so on can also be configured.

Network service providers

CloudStack network services are made possible with the help of a network service provider, which is basically a network element, hardware, or a virtual appliance. It can be a Cisco or Juniper device(s) that provide(s) firewall services in the same physical network or a F5 load balancer which provides load balancing for the virtual machines registered with it, it can also be a CloudStack virtual router which provides networking configuration for VLANs or overlay network, which helps in the division of a network into multiple tiers.

There can be single or multiple network service providers which are used to provide network services for a single network. There can be multiple instances of the same service provider in a single network. In the case where various network service providers are configured to provide network services, the users have the option to select from the several network offerings that are created by the administrator.

CloudStack network offerings

CloudStack provides various network offerings. These network offerings are a group of network services such as firewall, DHCP, DNS, and so on, that are provided as an offering to the users. These network offerings also provide the specifications of the service providers and are tagged to specific underlying network.

The cloud administrator can define new network offerings which can be segregated based on tags. It is up to the administrator to determine the network offering they want to provide throughout their entire cloud offering. The users are allowed to access the network offering based on their tags. The network offerings group together a set of network service such as a firewall, DHCP, and DNS.

The administrator can also choose specific network service providers to be provided as an offering. The network offerings can be any of the three states—**Enable**, **Disable** or **Inactive**. By default they are in the Disable state when created. Some of the network offerings are used by the system only and the users don't have their visibility. The tags that are used with each network offering cannot be updated but the physical network tags can be updated or deleted.

CloudStack is deployed with three default network offerings for the end users, a virtual network offering, a shared network offering without a security group, and a shared network offering with security group. Furthermore new network offerings can be created by the administrator to suit the environment needs and include various networking services. These network offerings include different networking services based on the configuration defined.

The shared network offerings are created when the user provisions a VM using that network. The users can also create networks from these network offerings. A set of network offerings can be DHCP, DNS, Source NAT, Static NAT, port forwarding, load balancing, firewall, VPN, or any other optional service provider based network offering Some of these services are provided using third party hardware equipment such as Juniper or Netscaler.

Types of network in CloudStack

CloudStack provides various types of network services for end users. CloudStack supports multiple network services from third parties. It helps with providing complex network configurations in the cloud.

Physical network

A zone in the CloudStack deployment can be associated with one or more physical networks. A physical network can be used to carry one or more types of network traffic. A zone can use the basic network configuration or advanced network configuration, which will decide the type of network traffic that flows through the physical networks.

In a zone with basic network configuration, only one physical network can be present. There are basically three types of network traffic that are allowed. They are:

- **Guest Network traffic**: This is the traffic flowing over the guest network for communication between the guest VMs when they are running. All the guest networks which are of type isolated share the same subnet which is set at the zone level. Guest traffic of a VM within one zone is carried in one network, VMs in different zones cannot communicate with each other. In order for the VMs in different zone to communicate, they must do it via a router through a public IP address.
- **Management traffic**: This traffic is generated by the internal resources of CloudStack. This basically comprises of the traffic between the hosts in the clusters, system VMs (these VMs perform various tasks by CloudStack in the cloud). The administrator must configure the IP ranges of the system VMs. This type of network traffic is usually untagged. The management traffic is should be isolated from the other traffic. The management traffic contains all the UDP traffic for heartbeats. It is highly recommended to isolate the management traffic from the other network traffic.
- **Storage traffic**: This traffic is the traffic flowing between the primary and secondary storage servers. These can be the VM templates which are placed on the secondary storage and when the user requests to create a VM based on some template, that template data has to flow from secondary storage server to the primary storage server. Another

example would be when a user creates a snapshot; the snapshots are stored in the secondary storage, so this snapshot data has to flow to the secondary storage. The storage network traffic is generally configured to be on a separate NIC to ensure better performance. In a zone with advanced network traffic types, there are additional network traffics that flow apart from the traffic flow in zone with basic network traffic. In the basic type of zone, VM traffic is publicly routable by default, whereas in advanced zone type, public label network traffic is exposed.

- **Public network traffic**: This kind of traffic flows between VMs and the Internet; this requires the VM to have a public IPs which can be assigned to the VM through the UI. In the case of an advanced network zone, one public IP is assigned to per account to be used as the source NAT. Using hardware devices such as Juniper SRX firewall, a single public IP can be used across all the accounts. Users can also request additional public IPs. This public IP can also be used to implement and configure a NAT instance between the guest and public networks.

All these types of network traffics can be multiplexed in the same underlying physical network using VLANs. It is up to the admin how he configures the network traffic and maps these network types to the underlying physical network and configures the labels on the hypervisor. These can all be done using the admin user interface of the CloudStack. Once the zone is created, the traffic labels can be changed from the user interface, whereas if we need to change the physical networks, some database entries are to be changed as well.

Virtual network

In order to enable multi-tenancy on a single physical network, the physical network has to be logically divided into several logical constructs, each logical construct is known as virtual network. All the information about the virtual networks and their setting are configured and stored in CloudStack. These settings are activated only when the first VM is started and assigned to this network and

the virtual network is also deleted or garbage collected when all the VMs are removed from that network. Thus, CloudStack helps in preserving the network resources and optimizing wastage. CloudStack allows the virtual network to be shared or isolated. The various types of virtual networks are discussed in the following sections.

Isolated networks

These networks, as the term suggests, are isolated and can be accessed only on virtual machines of a single account except for the domain administrators. The resources such as VLAN are allocated to these types of networks and the garbage collection is done dynamically. The isolated network can be upgraded or downgraded only if it is done for the entire network because it is unique for the entire network.

Shared networks

As the name suggests, a shared network can be accessed by the VMs of different accounts. But if one wants to attain isolation on this type of network, it can be achieved by using security groups as per CloudStack 4.0. These networks are created by the administrator who can also designate the shared network to a certain domain. The administrator has the responsibility to designate the network resources such as VLAN and the physical network it is mapped to. This network should be pre-created before the guest VM is provisioned on it.

CloudStack also supports the creation of guest network with the isolation type set to "STT". When configuring a zone, the administrator should create a new physical network for guest traffic and select "STT" as the isolation type.

L3 network configuration in CloudStack

The core switches provide L3 network isolation. The core switch connects to the various access switches in various pods. The

various features provided in the L3 network configuration in CloudStack are:

- **IPAM**: The IP address management systems which provides the IP address management for the network. The virtual router configured can act as the DHCP server to provide IP addresses to the guest VMs in the environment. The tenants are provided with the facility to enable more than one NIC and assign multiple IP addresses to the guest VMs, so an instance can lie in different networks at one time.
- **Gateway**: The gateway provides routing between multiple subnets. This means if two subnets are to be connected, the traffic flows through the gateway.
- **Remotely accessible VPN**: CloudStack allows configuration of remote access VPN between multiple remote sites to be connected over a public network. This is facilitated by security features such as IPSec which uses PSK.
- **Firewall**: Firewall can be configured in CloudStack which is based on source **Classless Inter-Domain Range** (**CIDR**) IP range and egress and ingress of network traffic to the instances.
- **NAT**: There can be two types of NAT services defined in a CloudStack network—source NAT, and static NAT. The source NAT can be configured per network where the virtual router provides the NAT service to the guest VMs connected to it. Static NAT can be configured by assigning an Elastic IP address to any instance in the environment.
- **VPN**: The VPN facility can also be configured where a device like the virtual router or core switch can be configured to provide site to site VPN access to connect to remote sites including security features such as IPSec.

The core switches also allow multiple protocol label switching, which enhances the speed of communication and prevents unnecessary lookups in the route table. It encapsulates packets of various network protocols and data packets are assigned labels which are used for forwarding the packets without even examining the packets.

Access switches or L2 switches

The access switches used in CloudStack provide L2 network isolation and are present at the POD level. There can be one or more access switches in a POD. The access switches are connected to the L3 core switch present in the zone that connects to all the pods' access switches in that zone. Core switches provide routing and connectivity between the access switches in the various pods in a zone. The access switches facilitate various network traffics such as management traffic, storage traffic, VM traffic, and the public traffic. The access switch has different types of networks connected to its different port groups and thus provides L2 network isolation to these networks.

It also allows configuring an overlay network and VLANs to configure various guest private networks over a single network. The access switches also provide physical isolation based on network labels. This type of configuration allows multiple NICs to be attached to a particular instance so that an instance can belong to more than one network at any time. These multiple NICs can be attached or detached from the VMs over time as per the need.

The administrator can also configure the monitoring of the various traffics that helps in providing updated information of the traffic congestion and the resource utilization. The access switches provide the facility of traffic monitoring of various network traffic and also provides various security configuration such as anti-spoofing and other features such as broadcast as well as multicasts suppression.

CloudStack virtual router

The virtual network in the deployment of CloudStack consists of various virtual networks which can be configured as per the demands by the administrator. This is achieved by using the CloudStack virtual router. Virtual routers are deployed in the basic type of networking where they are used as a shared service among the multiple tenants and provides features such as DHCP, DNS, and

so on. By default, only one virtual router can be deployed per network in an account in the advanced type of networking where there is one virtual router unique to the isolated guest private network. The administrator can raise this limit by increasing the quota.

A virtual router has three network interface cards, one is connected to the isolated guest network used for advanced VLAN, and is assigned the first IP in the CIDR range and will also act as DHCP, DNS, and gateway for instances in the private guest network, the second NIC is used for a local link network (only for KVM and XenServer), the management network, and for configuration of the virtual router. The link local interface is created on all the VMs on a host for dedicated communication between the guests and the host, but it is not supported in case of VMware. The third and the last NIC of the virtual router resides on the public network and is assigned a public IP that is used to provide NAT services to the guest VMs connected to it. The source NAT service is already configured on the virtual router in the default isolated mode which forwards the outbound traffic for the entire guest VMs and manages the incoming traffic as per the rules defined by the users.

Networking using CloudStack virtual router

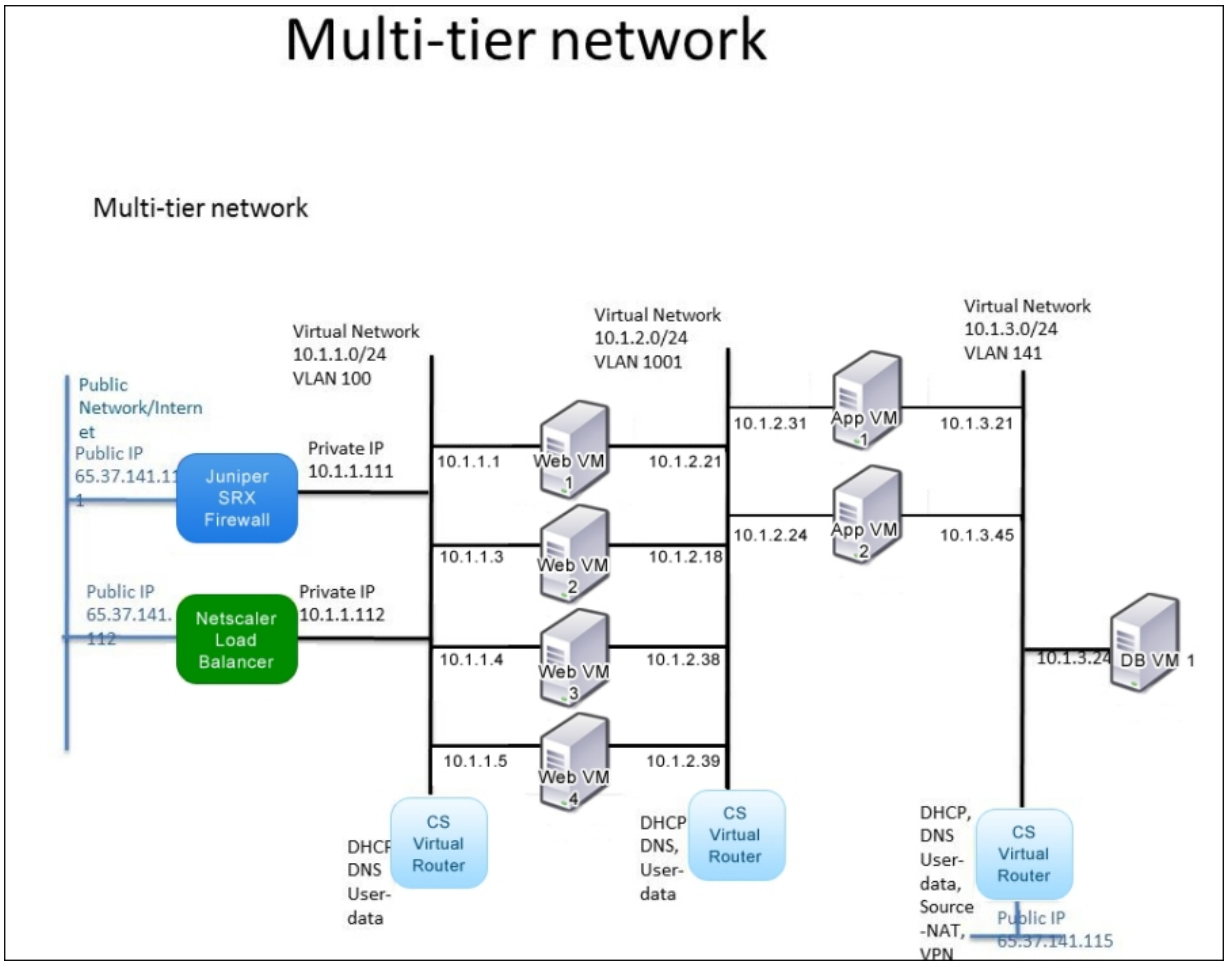
The virtual router is used by CloudStack to provide various networking services. The machines connected to guest virtual networks are connected to the outside world using public IP addresses through the virtual router and communicate with the other guests using the virtual router. All the communications from these servers behind the virtual router to the outside world are routed by the virtual router. The CloudStack virtual router is a device which is actually a virtual machine that provides networking services such as DHCP server which assigns IP addresses to the machines behind it, a DNS server which manages the host name to IP address mapping of the machines, NAT, a VPN gateway and many other.

There can be multiple guest virtual networks in the CloudStack environment. These networks are created using some network offering. Most of the networking services that are provided in the network offerings used in the creation of a network are provided by the CloudStack virtual router. The machines can have more than one network interface cards that can be assigned multiple IP addresses to the guest machines. This means that a virtual machine connected to one of the guest virtual network can have two NICs attached to it. One of the NICs has a private IP address and the other is connected to another guest virtual network of different subnet facilitated by another CloudStack virtual router.

The machine with two NICs, one attached to the guest virtual network 1 can be used to host the web server which has incoming public traffic of requests and the other NIC dedicated to the communication to databases server which can be present on the another VM attached to the second guest virtual router, and it can be accessed only through the web servers and does not have connectivity to the outside world.

The guest virtual network 2 has different subnet or a different VLAN tag and is private to the guest virtual network 1, which means any communication to the guest virtual network 2 has to be routed through the guest virtual network 1. This type of networking architecture is known as a multi-tier network.

In a multi-tier network, there can be multiple CloudStack virtual routers present between different networks of different VLAN tags or different overlay networks, and the communication between these private networks is facilitated by the CloudStack virtual router that acts as the gateway to the communication between the two or more networks, this router can also act as the DHCP server, DNS server, NAT, and for site-to-site VPN access.



Firewall and F5 Load balancer

The device between the access switch and the Vswitch of the hypervisor hosts can also be a firewall which provides the firewall protection to the virtual machines connected to the guest virtual network. The CloudStack administrator can install the Juniper SRX firewall between the public network and the guest virtual private network.

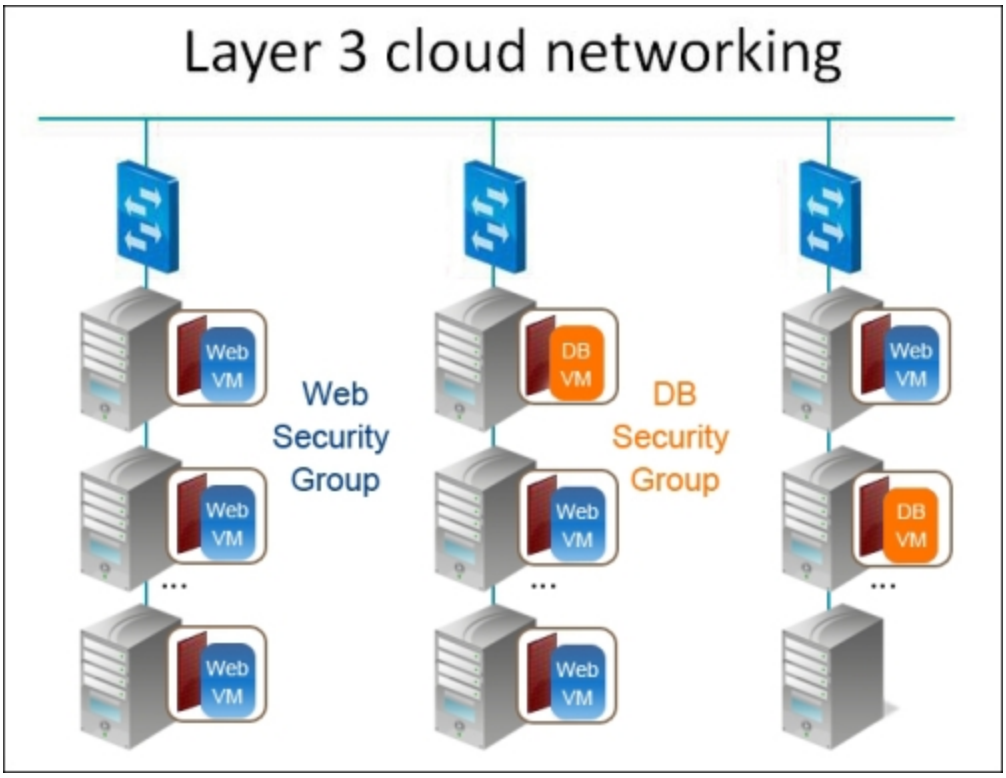
It can also be a F5 load balancer that provides load balancing services to the application hosted on machines attached to the guest virtual private network. Administrator installs the static route, which can be manual or automated configuration to point to his routing VM. The routing VM also provides VPN connectivity between two sites and this is configured directly on routing VM, not by CloudStack. The

load balancer configuration allows architecting a highly available solution where there are multiple instances on different servers hosting same application and registered to the load balancer. In case one of the instances goes down, the other continues to serve users' request. The load balancer also provides the distribution of requests to the different instances registered to it which helps in providing better performance.

Security groups

Security groups can be attached to any particular instance in the CloudStack environment. The security groups act as the firewall to allow or deny the egress and ingress of network traffic. The rules defined in security groups decide whether communication of some protocol, to some port of the instance from a particular source can be allowed or denied. The security groups can also be used to define rule for the outgoing traffic.

The security groups offer an extra level of security or firewall that can be applied to the instances for restricting the incoming and outgoing traffic. For example, the web servers on one VLAN can have security group to allow traffic from anywhere on the Internet to its particular port serving the users' requests, whereas a backend database server can have security group configured to allow traffic only from the security group of the web servers to its database port and deny any other traffic from anywhere. Thus, the security of the database server is maintained by restricting the access only from the web server. Security groups provide firewall configuration at the instance level. There can be one security group with a particular setting attached to multiple instances. Security groups allow a scalable network configuration in cloud over VLANs, where the numbers of VLANs that can be created on a Vswitch are restricted.



CloudStack is built using the above defined components and these components are used for different functionalities in cloud. The users can configure these components differently to configure the cloud as they want. CloudStack can be used to deploy a public cloud where people can access it over the Internet on the go, from anywhere; or it can be used to configure a private cloud that is private to an organization and can also be extended to be used as a hybrid cloud where it offers limited public access.

Summary

We introduced you to the building blocks of cloud and the architecture of cloud computing in this chapter. We discussed the various deployment and the service models of cloud computing. We saw the various layers that are essential components of a cloud solution. We also described the architecture and the several components in Apache CloudStack along with their functionalities.

We can now proceed with the next chapter, where we will cover the details about the different types of CloudStack deployments, the differences between them, the deployment scenarios, and the steps to accomplish them.

Chapter 2. Installing Apache CloudStack

In Chapter 1, *Apache CloudStack Architecture*, we discussed the core components of IT infrastructure that build up the cloud and how are they used to provide uninterrupted automated service to cloud users. We also discussed the various components of CloudStack and the deployment models of CloudStack. In this chapter, we are going to walk through the following:

- The different types of installations of CloudStack
- The use cases for different types of deployment of CloudStack
- How CloudStack can be set up in your organization's environment

Pre-installation tasks

After having gone through the architecture of CloudStack, let's start with the installation of the CloudStack environment.

As we have already seen in the previous chapter, the CloudStack deployment architecture basically consists of two major parts; they are Management Server and CloudDB. CloudStack provides an option to deploy both of them on the same server, or one can choose to deploy them separately on different servers. CloudStack deployment can be done in two different ways popularly known as single node installation and Multinode installation.

- **Single node installation**: In this type of installation, there is only one instance of the management server; the database server can be on the same node or a different node.
- **Multinode installation**: This type of installation comprises of multiple management servers and one or many database

servers. These management servers are placed behind a load balancer to balance the load across all the servers.

The installation of Apache CloudStack comprises a sequence of steps that includes the installation of various other components such as the management server, database, agent and other optional components such as the usage server.

Before proceeding to the installation, it is good practice to review the deployment architecture, requirements, and network and storage setup. The users can choose from the two deployment scenarios as per the needs of the organization and the resources that are to be managed using the CloudStack.

> **Note**
>
> The installation steps mentioned in this chapter are related to Apache CloudStack Version 4.0.0 and may change for any other upcoming version.

Requirements

Before we move on to the installation steps, first let us consider the basic requirements that are needed for the CloudStack deployment. The recommended hardware configuration of the machine hosting the Management Server is as follows, starting with the operating system:

The operating system for the installation of CloudStack 4.0 is recommended to be RHEL 6.3 + 64 bit that is available for download at https://access.redhat.com/downloads or one can use CentOS 6.3+ 64 bit that can be downloaded from https://isodirect.centos.org/centos/6/isos/x86_64/.

Ubuntu 12.04 LTS can also be used for this purpose.

There are builds available for Centos/RHEL and Ubuntu online which make the installation simple.

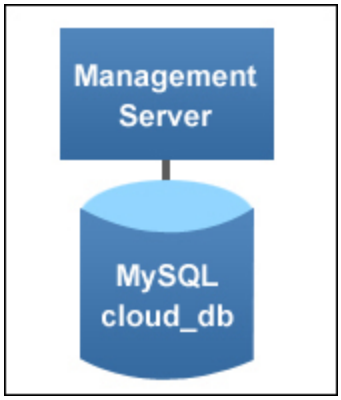
The CloudStack system can be deployed on physical or virtual machines.

The basic hardware configuration for installation is:

- **CPU**: It should have 64-bit X86 CPU. The performance is dependent upon the number of cores the processor is of, but at least a quad core is recommended. The CPU capacity is dependent on a lot of factors including the number of devices and users accessing the system.
- **Memory**: There should be at least of 4 gigabits of memory. This is the basic requirement. The actual production instances may require a higher amount of RAM based on usage.
- **Hard disk**: It is recommended for the local disk to have at least 36 GB of space. More storage space leads to better performance. It is recommended to use 500 GB of hard disk space. We will be setting up a dev cloud in this document which can be used as the Management server and the CloudDB as well as to provide primary and secondary storage using a local disk or via Network File System and placing the management server on a virtual machine. However these specifications are only for testing purposes and for a production environment one should consider a multinode installation with SAN /NAS storage.
- **Network**: The server must have at least one network interface card and there should be a static IP address associated with it. The machine should ideally also have a fully qualified domain name. The hostname entry should be done in the hosts file. For production instances, dual network cards and two HBA cards to access SAN storage are recommended.

Single node installation

In a single node installation, there is only one Management server. For simplicity's sake we are also installing the MySQL database on the same server as the Management Server. We will look into the single node installation first. The following diagram shows the setup of a single node installation:



The steps for the installation of CloudStack on a single server are as follows:

- Preparing the operating system
- Management server installation
- Installation and configuration of the database
- Prepare Network File System share for storage
- Prepare the system VM template

Preparing the OS

A list of supported operating systems is given in the *Requirements* section. We must prepare the OS before we proceed to installation of CloudStack Management server. The steps required at operating system level before we proceed with the installation of CloudStack are discussed as follows:

1. Log in to the operating system as the root user.

2. We need to set a **Fully Qualified Domain Name** (**FQDN**). We can set something like `cs01.packt.com` by editing the hosts file in `/etc/hosts` or if the machine is in a network with DNS configured so that the hostname is resolvable.
3. Let's verify the hostname by using the `hostname` command. The output is shown as follows:

```
[root@localhost ~]# hostname cs01.packt.com
[root@localhost ~]# hostname
cs01.packt.com
[root@localhost ~]#
```

4. We also need to set SELinux to be permissive by default. This will ensure that every operation is allowed and that the CloudStack agent runs smoothly. First we need to check whether SELinux is installed on the machine or not. To check if you have SELinux installed, use the following command based on the operating system you are using:

   For RHEL or CentOS, though SELinux is installed by default on the machine, anyways you can verify that it is installed by using the following command:

```
#rpm -qa | grep selinux
```

   For Ubuntu, you can verify that it is installed by using the following command:

```
#dpkg -list 'selinux'
```

5. Now, if SELinux is installed on the machine, we need to set it to permissive by default. This can be done in the following way:

   - In case of RHEL:

```
#setenforce 0
```

```
[root@cloudstack ~]# setenforce 0
[root@cloudstack ~]#
```

- In case of Ubuntu, we can use:

```
#setenforce permissive
```

- And in case of CentOS/RHEL, we can use:

```
#setenforce permissive
```

6. Next, we need to edit the file `/etc/sysconfig/selinux` in case of RHEL and CentOS, The file looks like the following screenshot, we need to set `SELINUX=permissive` as shown in the following screenshot:

```
[root@cloudstack ~]# vim /etc/sysconfig/selinux
[root@cloudstack ~]# cat /etc/sysconfig/selinux

# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=permissive
# SELINUXTYPE= can take one of these two values:
#     targeted - Targeted processes are protected,
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted

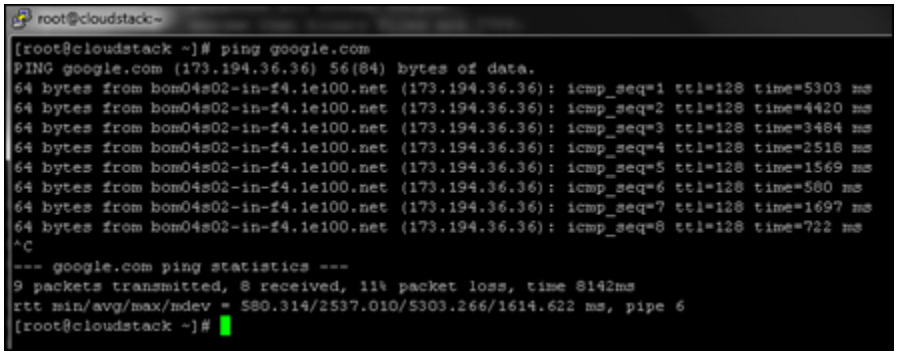

[root@cloudstack ~]#
```

- In case of Ubuntu, we need to run the following command:

```
# selinux-config-enforcing permissive
```

By doing this we ensure that the permissive settings will be maintained even after the system reboots.

7. Before we begin installation, we need to make sure that the server is connected to Internet. This is required for downloading packages from the Internet. To do so, we need to check whether we can ping a site on the Internet. In our example we have pinged `google.com`:



Note

In case of CentOS, if everything (Management Server, Database, and KVM hypervisor) is to be installed on a single machine that is only recommended for test scenarios, we need to ensure that network configuration file `/etc/sysconfig/network-scripts/ifcfg-<YourPhysicalDeviceName>`; for example, `etc/sysconfig/network-scripts/ifcfg0` is present because without this the cloud platform will not be able to create the bridge.

8. As most of the packages that will be used for installing CloudStack are already present on the installation disc, we need to create a local Yum repository. In case you are using a VM, you would first need to attach the CD ROM and mount it before creating a local yum repo. We can create a local repo by creating a repo file under `/etc/yum.repos.d/` with the details of the repo, as shown in the following screenshot:

```
[root@cloudstack ~]# tail -n 9 /etc/yum.repos.d/local.repo

[local]
name=CentOS-$releasever - Media
baseurl=file:///media/cdrom/

gpgcheck=1
enabled=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-6
```

9. Next we need to ensure that the NTP server is installed and configured on this machine. The NTP server configuration will help to synchronize the time for all the servers in environment. This can be done using following steps:

    - For RHEL and CentOS:

    ```
    # yum install ntp
    ```

    - For Ubuntu:

    ```
    #apt-get install ntp
    ```

    We need to edit the NTP configuration file to locate the NTP server under `/etc/ntp.conf`. We need to add one or more servers with the names of the NTP servers that can be used for locating the NTP server, like the example given below:

    ```
    server 0.north-america.pool.ntp.orgserver
    1.north-america.pool.ntp.orgserver 2.north-
    america.pool.ntp.orgserver 3.north-
    america.pool.ntp.org
    ```

    After these steps are done we need to restart the NTP client by using the following command:

    ```
    #service ntpd restart
    ```

The services are not set to auto start, which is needed so, we need to make the services auto start.

For RHEL or CentOS:

```
# chkconfig ntpd on
```

On Ubuntu:

```
# chkconfig ntp on
```

Management server installation

After preparing the operating system, we can now begin with the installation of the management server and MySQL in single node installation. The installation process mentioned here assumes the compilation of the CloudStack package is done on a Linux system that uses RPMs or DEBs. Now, we will proceed with the installation of Apache CloudStack by building it from the source.

1. As the first step we need to download some packages and files from the Internet.

   i. Get the Apache CloudStack tar ball from http://www.apache.org/dyn/closer.cgi/dist/incubator/cloudstack/releases/4.0.0-incubating/apache-cloudstack-4.0.0-incubating-src.tar.bz2. or you can get the tarballls from the download page http://incubator.apache.org/cloudstack/downloads.html

   ii. To verify the authenticity of the release, we need to get a detached cryptographic signature from http://www.apache.org/dist/incubator/cloudstack/releases/4.0.0-incubating/apache-cloudstack-4.0.0-incubating-src.tar.bz2.asc.

iii. To verify the validity of the release download, we need a MD5 hash of the release which can be downloaded from http://www.apache.org/dist/incubator/cloudstack/releases/4.0.0-incubating/apache-cloudstack-4.0.0-incubating-src.tar.bz2.md5.

iv. To help in the verification of the validity of the release download we also need a SHA512, this can be downloaded from http://www.apache.org/dist/incubator/cloudstack/releases/4.0.0-incubating/apache-cloudstack-4.0.0-incubating-src.tar.bz2.sha

2. After we have downloaded all the files from the Internet, we need to the check the integrity and the validity of the release that can be done using the following command. This is done by the series of steps mentioned as follows:

   i. First step is to get the keys from http://www.apache.org/dist/dev/incubator/cloudstack/KEYS in order to verify the GPG signature.

   ii. The next step is to import the downloaded keys that can be done by using the following command. If the signature is valid, the output will be "Good Signature".

```
# gpg --import KEYS
```

   iii. Now you can check the signature that you have downloaded, using the following command.

```
# gpg --verify apache-cloudstack-4.0.0-
incubating-src.tar.bz2.asc
```

   iv. After the signature validity check, we will use the cryptographic hashes to assure the validity of the download release which can be done by the following command.

```
# gpg --print-md MD5 apache-cloudstack-
4.0.0-incubating-src.tar.bz2 | diff -
apache-cloudstack-4.0.0-incubating-
src.tar.bz2.md5
```

If this is successful, there will be no output. In case there is any kind of output, this means that there is difference between the hash generated locally and the hash that was downloaded from the Internet.

i. Apart from the above MD5 hash check, CloudStack also provides a SHA512 to validate the release that is downloaded, it can be done using the following command and the SHA512 hash that you downloaded:

```
# gpg --print-md SHA512 apache-cloudstack-
4.0.0-incubating-src.tar.bz2 | diff -
apache-cloudstack-4.0.0-incubating-
src.tar.bz2.sha
```

If the package that you have downloaded is correct, there will be no output, else we will get some output, which means there is a difference between hash generated by you and the hash that was downloaded.

3. There are some basic requirements that must be met in order to install CloudStack. These are as follows:

   i. Ant

   ii. Maven 3 is recommended

   iii. Java (Java 6/open JDK 1.6)

   iv. Rpmbuild or dpkg-dev

4. Now, let's begin with the extraction of the package. To extract the package, we will issue the following command:

```
# tar -jxvf apache-cloudstack-4.0.0-
incubating-src.tar.bz2
```

5. The command will extract the tar ball to the folder, `apache-cloudstack-4.0.0.-incubating-src`.
6. We will move into the extracted directory for the ease of installation.

```
# cd ./apache-cloudstack-4.0.0-incubating-
src
```

7. Now depending upon the Linux environment, we need to build the packages and create repositories so that it can directly bootstrap dependencies and these can be directly used to install Apache CloudStack.

Building Deb packages

To build the packages on Ubuntu, it is recommended to use Maven 3 or newer, but is not available with Ubuntu 12.04. The user may need to download the package from the Internet and install it.

1. The following is to be executed in sequence to install various packages required for Apache CloudStack such as Java, Ant, and python-software-properties.

```
# sudo apt-get update
# sudo apt-get install python-software-
properties
# sudo add-apt-repository
ppa:natecarlson/maven3Available Packages
# sudo apt-get update
# sudo apt-get install ant debhelper
openjdk-6-jdk tomcat6 libws-commons-util-
java genisoimage python-mysqldb libcommons-
codec-java libcommons-httpclient-java
liblog4j1.2-java maven3
```

2. Though we have installed a number of packages above, still we need to install some of the other packages as build time

dependencies for the Apache CloudStack, which we will try to deal with using maven. CloudStack uses Maven for dependency resolution. In the same extracted directory of the CloudStack tar ball, we need to execute the following command to resolve the dependencies of CloudStack.

```
# mvn3 -P deps
```

3. After successfully resolving all the dependencies for the building process, we can issue the following command to build the process.

```
# dpkg-buildpackge -uc -us
```

4. After this command is executed, there will be 16 completely built packages. The list comprises of the following packages:

```
cloud-agent_4.0.0-incubating_amd64.deb
cloud-agent-deps_4.0.0-incubating_amd64.deb
cloud-agent-libs_4.0.0-incubating_amd64.deb
cloud-awsapi_4.0.0-incubating_amd64.deb
cloud-cli_4.0.0-incubating_amd64.deb
cloud-client_4.0.0-incubating_amd64.deb
cloud-client-ui_4.0.0-incubating_amd64.deb
cloud-core_4.0.0-incubating_amd64.deb
cloud-deps_4.0.0-incubating_amd64.deb
cloud-python_4.0.0-incubating_amd64.deb
cloud-scripts_4.0.0-incubating_amd64.deb
cloud-server_4.0.0-incubating_amd64.deb
cloud-setup_4.0.0-incubating_amd64.deb
cloud-system-iso_4.0.0-incubating_amd64.deb
cloud-usage_4.0.0-incubating_amd64.deb
cloud-utils_4.0.0-incubating_amd64.deb
```

Setting up the apt repo

Once all the packages are successfully built, we can add them to a repo or a system which enables them to be accessed over the HTTP, this can be a different machine from the management server.

These packages can be added to the repository and the installation can be done directly using the apt-get install command, which will also install the packages resolving all the dependencies. We will add the built packages to the repository using the commands given as follows:

We need the `dpkg-dev` package to be installed on the system, if it is not, we can do that using the following command:

```
# sudo apt-get install dpkg-dev
```

Now we need to set up a repository which can be accessed over HTTP by the machine which needs to install Apache CloudStack, so we need to copy all the packages to a folder that will be accessed over HTTP, this can be any folder of your choice, we will choose `/var/ww/cs/repo`.

```
# sudo mkdir -p /var/www/cloudstack/repo/binary
# sudo cp *.deb /var/www/cloudstack/repo/binary
# sudo cd /var/www/cloudstack/repo/binary
# sudo dpkg-scanpackages . /dev/null | tee
Packages | gzip -9 > Packages.gz
```

After the successful run of all the preceding commands, the binary packages of Apache CloudStack and Packages.gz are available over HTTP.

Adding the repository to the system

Now that we have all the packages and binaries available over HTTP, we need to add the address of the repository to the system where want to download the packages and install them.

1. For doing this, we need to add a repository file in the path `/etc/apt/sources.list.d` with the content given as follows:

```
# deb http://server.url/cloudstack/repo
binary/
```

2. After adding the repo file, you may want to update the apt-get so that it knows the location of the Apache CloudStack packages.

```
# sudo apt-get update
```

Building RPM

Now that we have seen the complete building process of the Deb packages for Ubuntu machines with Ubuntu, we will now proceed with the building of RPM packages for Centos/RHEL machines.

As already mentioned, Apache CloudStack requires Maven for resolving the dependencies, once you have maven installed on your system, we can resolve the dependencies by using the following command:

```
# mvn -P deps
```

Once the dependencies are resolved, we can start the package building process using the following command:

```
#./waf rpm
```

Creating yum repo

Now that all the RPMs are built, we can proceed with creating a YUM repository which can be used to install Apache CloudStack directly by yum.

The packages can be copied to another folder which can be hosted on some other server over HTTP and used as the repository, this

machine can be some other machine apart from the Management server, so after copying all the files to the folder to be used as repo, in this case we are using the directory where the packages are built, we can create the repo using the following command:

```
# cd artifacts/rpmbuild/RPMS/x86_64
# createrepo ./
```

Adding the repository to the system

Once we have created the repository as defined above, we can proceed with addition of the repo path to our system so that we can use the repository to install the Apache CloudStack. This can be done by creating a file `/etc/yum.repos.d/cloudstack.repo` with the following contents in it.

```
[apache-cloudstack]
name=Apache CloudStack
baseurl=http://webserver.tld/path/to/repo
enabled=1
gpgcheck=0
```

Back to management server installation

Now that we are done with building of the packages, we can continue with the installation of management server.

Installing the management server on a Centos/RHEL instance can be done using yum and this can be done using the apt-get in Ubuntu.

For Centos/RHEL:

```
# yum install cloud-client
```

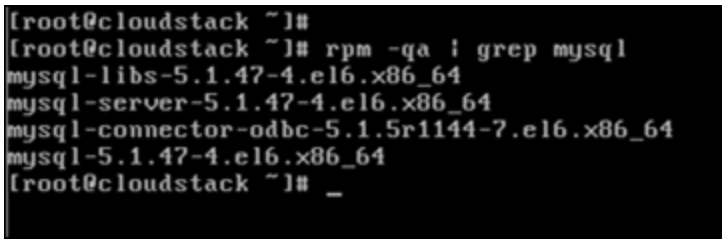
For Ubuntu:

```
# sudo apt-get install cloud-client
```

Database installation and configuration

Now that we have successfully installed the management server on this node, we need to install CloudDB; that is, a MySQL database on it.

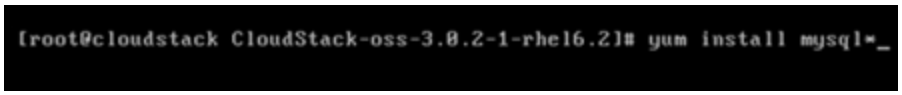
1. Let us first check whether we already have MySQL installed on our server:



> ## Note
>
> We need to take care over the version of MySQL installed with the CloudStack version. It is recommended to use MySQL version 5.1.58 or later with CloudStack 4.0.0. In case you have MySQL of earlier version than 5.1.58 then, you need to remove that and proceed with the installation of Version 5.1.58 or later.

One can also use a yum repository to install mysql for RHEL or CentOS, using the following command:

```
#yum install mysql
```



2. After we are done installing the MySQL database on our system, we need to make some changes in the configuration, so that it can run properly with our CloudStack management server. To do so, we need to edit the `/etc/my.cnf` or `/etc/mysql/my.cnf` file. We need to insert some new additional lines in this file as per our deployment under the `[mysqld]` section that is given as follows:

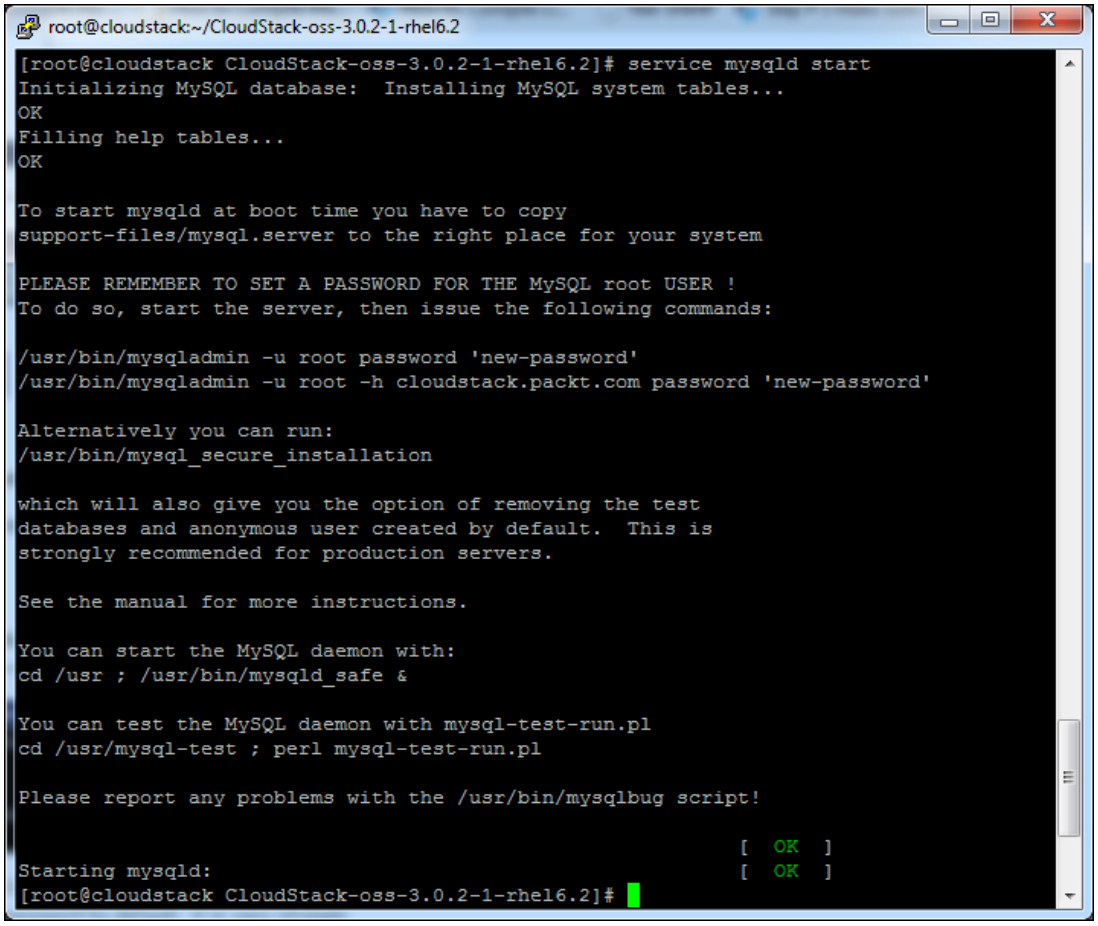
```
innodb_rollback_on_timeout=1
innodb_lock_wait_timeout=600
max_connections=350
log-bin=mysql-bin
binlog-format = 'ROW'
```

   The `max_connection` parameter we are setting here can be around 350 multiplied with the number of management servers we are deploying in our environment. Here, we are only deploying one management server, therefore we are using 350. Please note that this number is an estimate and ideally should be based on the hardware assigned to the management server.

   Only MySQL with version equal or above 5.1 supports `binlog-format` variable.

3. After performing these steps, we need to restart the MySQL server using:

```
#service mysqld restart
```



4. After the MySQL service is restarted, we will have to configure the password for the root user of the database. You can use the following commands:

    - For RHEL or CentOS:

```
#mysql –u root
```

    - For Ubuntu server:

```
#mysql –u root –p <password>
```

    Note that in RHEL and CentOS, MySQL doesn't set the password by default. You must change the default password and configure a new root user password for security reasons.

    You can use the following command to change the password:

```
mysql> SET PASSWORD = PASSWORD('your-
password')
```

After you have configured the password, you will have to use the following command to log in to the MySQL server:

```
#mysql -u root -p <your-password>
```

5. When all of this is done, you may want to restart the MySQL service for the changes to be effective. Use the following commands to do this:

    - For RHEL or CentOS:

    ```
    #service mysqld restart
    ```

    - For Ubuntu:

    ```
    #service mysql  restart
    ```

6. Now, if we have used a different server for database server, we need to open the database port on the firewall of our management server machine to allow communication with our database server. We can do that using the following command:

    ```
    #iptables -I INPUT -p TCP -dport 3306 -j
    ACCEPT
    ```

7. And then you will also have to edit the `iptables` file under `etc/sysconfig/` to make sure that changes in firewall are effective after every restart. You will need to add the following line to the file:

    ```
    -A INPUT -p TCP -dport 3306 -j ACCEPT
    ```

We are now done with the installation of the MySQL database for our CloudStack.

8. Now the database and user needs to be created. The following command sets up the database for CloudStack. It will also create a cloud user on the database.

```
# cloud-setup-databases cloud:
<dbpassword>@localhost --deploy-as=root:
<password> -e <encryption_type> -m
<management_server_key> -k <database_key>
```

The database host is provided as localhost, because the management server and the database are on the same host. The command uses certain parameters, which are listed as follows:

- `dbpassword`: This is the password which will be associated with the Cloud user which is being created.
- `deploy-as`: The username and password of the user who is deploying the database is passed via this parameters. In this example, the user is root who deploys the database.
- `encryption-type`: This is optional. This value depends upon the technique which has been used for encryption of database password. It can be either "file" or "web".
- `management_server_key`: This is optional. This parameter provides the key which has been used for encrypting all the confidential parameters in the CloudStack properties file. This parameter defaults to the value "password".
- `database_key`: This is optional. This key is used to encrypt all the confidential parameters in the CloudStack database and it also defaults to `password`, which must be changed for security reasons.

Preparing Network File System share for storage

As we have already seen in Chapter 1, *Apache CloudStack Architecture*, there are basically two types of storage—primary and secondary storage associated with CloudStack deployment.

These two types of storage have different purposes in the environment. These two storage servers can be NFS shares. We will now look into the setup of NFS server before adding the servers as storage to the CloudStack environment.

Depending upon the need, we can set up the primary storage using any of the two technologies NFS or iSCSI. If we need to prepare the server for primary storage, you can use iSCSI also instead of NFS.

The primary storage server(s) is associated with a cluster, which is accessible by all the hosts in that cluster and is used to store VMs' disk volumes. There should be at least one primary storage server in each cluster and should be accessible from all the hosts. There are some standards-compliant iSCSI and NFS servers that are supported to be deployed in CloudStack platform. Few of them are as follows:

- Dell EqualLogic for iSCSI
- Network Appliances filters for NFS and iSCSI
- Scale Computing for NFS

For a server to be used as primary storage, it should meet the following hardware requirements:

- The hardware should be one from the list of standards-compliant iSCSI or NFS servers, which must be supported by the underlying hypervisor used.
- Basically, the capacity of the primary storage depends upon the need. We need to deal with this in the planning phase, about how big our cloud environment is going to be. The

primary storage should be calculated based on the number and size of Virtual Machines that are going to be deployed onto the cloud platform. The primary storage can be increased to accommodate increase in demand.

- The storage server(s) which is/are to be added as primary storage in CloudStack must be provisioned as a shared storage system if it is not an iSCSI storage, so that more volumes can be created and attached to the guest VMs on hosts in a cluster. If the primary storage is not a shared storage system, we will have to set the `system.vm.local.storage.required` flag to be set to true, this parameter is found in the list of global configuration parameter, which we will discuss when we introduce you to the user interface. This will enable the VM to be functional.
- You must first add the host(s) to the cluster before adding primary storage.

For a server to function as secondary storage, it should meet the following hardware requirements:

- The secondary storage could be a NFS storage appliance or a Linux NFS server.
- The capacity of the secondary storage also depends upon your needs but it is recommended to be a minimum of 100 GB.
- The secondary storage, as we have already seen in Chapter 1, *Apache CloudStack Architecture*, is common to Zone and is shared by all the hosts in all the clusters in that zone, and thus, serves the VMs that are located in that zone and must be available to all the hosts in that zone.
- You can also use OpenStack ObjectStorage commonly known as Swift for secondary storage (For more details about Swift, go to http://swift.openstack.org).

Now that we know the minimum requirements for preparing a NFS server that can be used as primary or secondary storage, let's take a look at steps involved in preparing a separate NFS server.

This node is different than the server on which we installed the management server and the database. We will discuss about using management server as NFS server after this.

Creating a separate NFS server

A secondary storage in CloudStack can be a NFS storage appliance or a Linux NFS server. Here we are going to discuss the creation of a Linux NFS server separated from the management server. In order to create a separate NFS server, first we need to create an NFS share for secondary storage and export it using the options `rw`, `async`, and `no_root_squash`. These commands are dependent upon the operating system. A working example is shown as follows:

1. We need to edit the `exports` file, open it using the following command:

```
#vi /etc/exports
```

2. Add the following line:

```
/export *(rw,async,no_root_squash)
```

3. Now export the `/export` directory using:

```
#exportfs -a
```

4. After doing this, we need to go to the management server and create a mount point for the exported Linux NFS share:

```
#mkdir -p /mnt/secondary
```

5. Now on the management server, mount the NFS server:

```
#mount -t nfs
nameoftheNFSserver:/nfs/share/secondary
/mnt/secondary
```

6. You need to replace `nameoftheNFSserver` with the NFS server name and `/nfs/share/secondary` with the NFS share path in the preceding example. In this way the NFS share path `/nfs/share/secondary` on the NFS server `nameoftheNFSserver` will be mounted at the `/mnt/secondary` on the management server.

As shown in the preceding example, the NFS share server is set up to be used as secondary storage.

Optionally, if you want to use NFS for primary storage, you will need to carry out the same steps with a different server. An iSCSI storage server can also be used for primary storage, but for that the primary storage is added using the CloudStack user interface. We will discuss this configuration and adding primary and secondary storage servers to the CloudStack deployment in detail in the next chapter.

Till now, we have created a separate NFS server to be used as primary and secondary storage and have mounted the secondary storage to the management server. The following steps show you how to use management server as a NFS server to be used as primary and secondary storage.

1. First we need to create two separate folders for primary and secondary storage in the `export` folder as shown in the following commands:

```
# mkdir -p /export/primary
# mkdir -p /export/secondary
```

2. After we have created the folders for primary and secondary storage, we need to configure them as NFS exports. This can be done by inserting a few lines in the /etc/exports file. So you need to carry out the following steps:

```
#vi /etc/exports
```

- Now insert the following line in the file.

```
/export * (rw,async,no_root_squash)
```

- After doing this, export the export directory using:

```
# exportfs -a
```

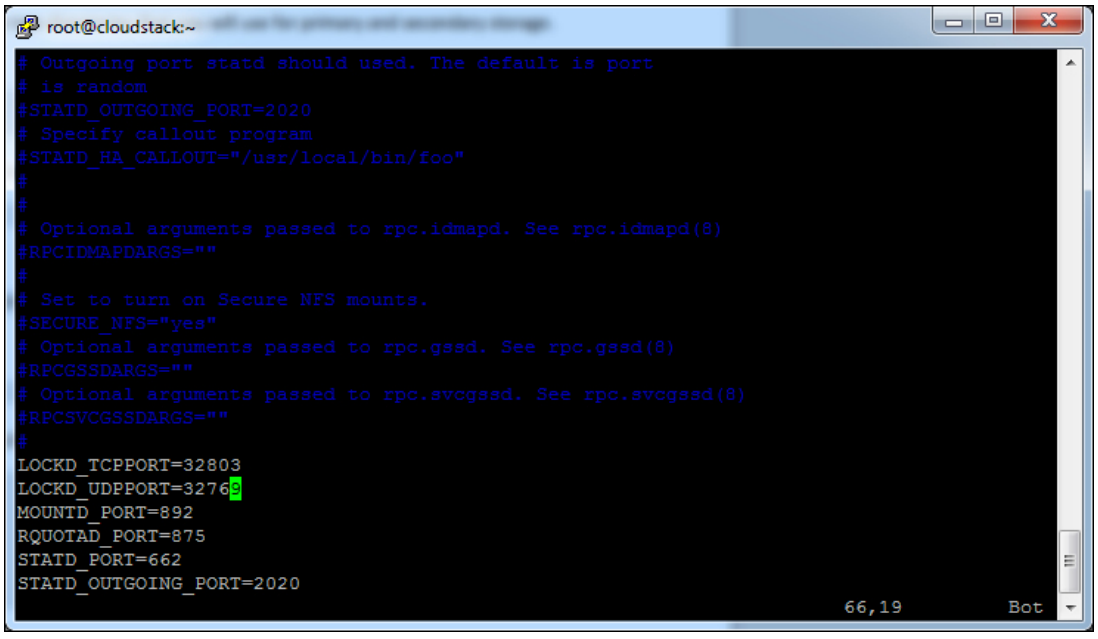
3. Once we have exported the two folders as NFS shares we need to make some changes in the `/etc/sysconfig/nfs` file.

- Open the file using:

```
#vi /etc/sysconfig/nfs
```

- Next uncomment the following lines in the file.

```
LOCKD_TCPPORT=32803
LOCKD_UDPPORT=32769
MOUNTD_PORT=892
RQUOTAD_PORT=875
STATD_PORT=662
STATD_OUTGOING_PORT=2020
```



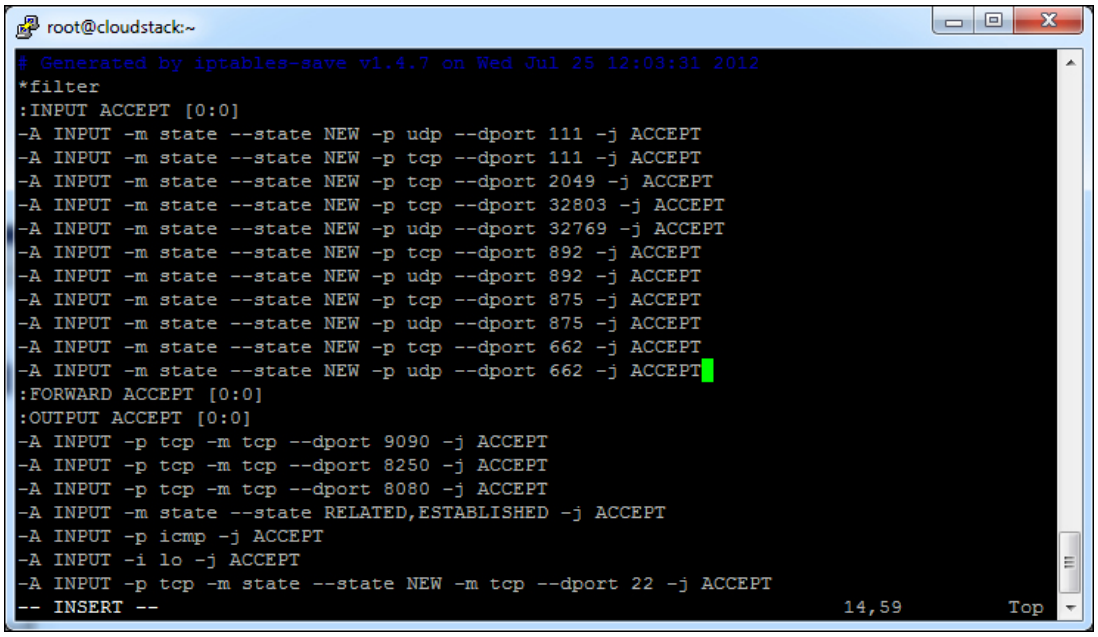
4. Now you need to edit the IP Tables to open the ports for accepting the incoming traffic.

    - So open the file `/etc/sysconfig/iptables` using the following command:

      `#vi /etc/sysconfig/iptables`

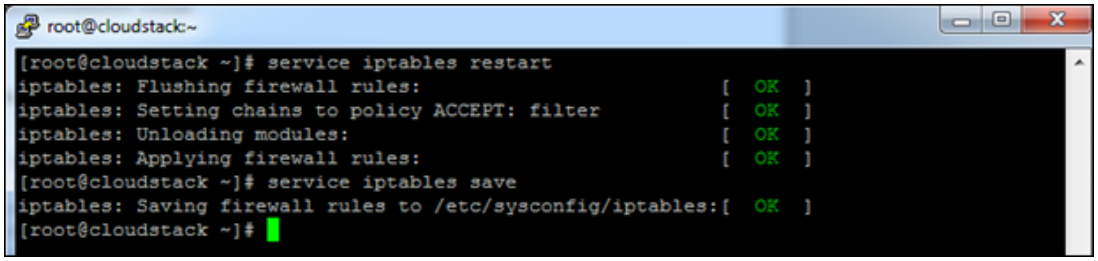
    - Next add the following lines at the beginning of the `INPUT` chain.

```
-A INPUT -m state --state NEW -p udp
--dport 111 -j ACCEPT
-A INPUT -m state --state NEW -p tcp
--dport 111 -j ACCEPT
-A INPUT -m state --state NEW -p tcp
--dport 2049 -j ACCEPT
-A INPUT -m state --state NEW -p tcp
--dport 32803 -j ACCEPT
-A INPUT -m state --state NEW -p udp
--dport 32769 -j ACCEPT
-A INPUT -m state --state NEW -p tcp
--dport 892 -j ACCEPT
-A INPUT -m state --state NEW -p udp
--dport 892 -j ACCEPT
-A INPUT -m state --state NEW -p tcp
--dport 875 -j ACCEPT
-A INPUT -m state --state NEW -p udp
--dport 875 -j ACCEPT
-A INPUT -m state --state NEW -p tcp
--dport 662 -j ACCEPT
-A INPUT -m state --state NEW -p udp
--dport 662 -j ACCEPT
```



5. And then restart the `iptables` service:

```
#service iptables restart
#service iptables save
```



6. We have successfully created NFS share and opened the ports for the incoming communication.
7. If you have used NFS v4 for communication between the client and server, you must add your domain to `/etc/idmapd.conf` on the management server as well as the hypervisor host. Follow these steps:

    - Open the following file:

      ```
      #vi /etc/idmapd.conf
      ```

    - Uncomment the domain line and replace the value with your domain name:

```
Domain = cloudstack.packt.com
```

8. After restart of the server you will have two separate NFS shares called `/export/primary` and `/export/secondary`.
9. As a final step, let's test our installation before moving further:

    - Log in to the hypervisor host that might have a domain name. We will use the domain name `hypervisor.packt.com`.
    - If the OS that you are using is RHEL or CentOS, you will have to make sure that NFS and `rpcbind` are running. To check them, you can use the following commands (which may vary depending upon the OS).

```
# service rpcbind start
# service nfs start
# chkconfig nfs on
# chkconfig rpcbind on
```

    - After we have checked the services are running properly, we can use these exported directories as the storage and add them while creation of a zone (this will be discussed in the next chapter).
    - You will have to replace the name of your management server with `<management-server-name>`.

Now we are done setting up the NFS server on the management server itself. The two folders that we created in the initial steps act as NFS shares for primary as well as secondary storage. Let's begin with preparing the system VM template.

Preparing the system VM template

Apache CloudStack has another component known as the system VM template that is an essential part in managing the infrastructure resource and providing services. The system VM template is present in every CloudStack zone and helps in tasks such as template processing that includes downloading and uploading templates, uploading ISOs, console proxy, virtual router, and so on. The system

VM templates are required to be put into the secondary storage so that CloudStack can use them to provision system VMs. There are several types of system VMs that are used to perform various system tasks in the cloud. The CloudStack is responsible for managing these VMs. For the VMware-specific system VMs, it creates them and starts or stops them as per the need. If it needs to scale up then these VMs are started and stopped when it needs to scale down.

There is a single system VM in a zone in a CloudStack environment, which is used to create machines for processing tasks such as template downloading, and template and ISO uploading.

Note

In case of VMware, there can be multiple system VMs supported by CloudStack that helps in performing tasks specific to VMware such as load balancing.

These VMware specific system VMs are launched by the CloudStack Management server as per the need. The CloudStack Management server performs the task of load balancing between these system VMs in the environment by monitoring and weighting all the commands sent to these system VMs. The management server also provisions and decommissions servers based on the load.

The system VMs are provisioned one for each zone to manage tasks and also helps in providing services like Console Proxy and Virtual router, using which a lot of networking services are provided to the users in network.

We need to prepare the system VM template on the secondary storage which will be used by CloudStack Management server to provision system VMs. The system VMs are deployed using a single template, below are the characteristics of the system VM templates:

- Debian 6.0 ("squeeze"), 2.6.32 kernel which is updated with the latest security patches (downloadable from Debian security APT repository).
- The template also has a minimal set of packages installed on it, which reduces the risk of attack surface.
- The template has a 32-bit CPU which is recommended for enhanced performance on hypervisors such as Xen/VMware.
- The template also has a pvops kernel with Xen PV driver, virtio drivers for KVM, and VMware tools that help in optimum performance for all the hypervisors. The Xen tools installed in the template include performance monitoring.
- The templates are updated with latest versions of HAProxy, iptables, IPsec and Apache, all from the Debian repository and latest version of JRE from Oracle/Sun, which ensures improved security and makes the system fast.

Now that we know the requirements for the system VM template, let's begin with the preparation of our system VM template. The steps to prepare the system VM templates are as follows:

Log on to the management server and run one or more of the following `cloud-install-sys-tmplt`. These commands will retrieve and decompress the system VM template and should be run for each type of hypervisor that will be installed on the hosts in a zone. Internet access from the server should be available as these will be downloaded from the Internet.

Note

The setup process requires around 5 GB of free space on the local file system and takes some time for each run.

Also note that these steps should be repeated for each of the secondary storage servers that you want to add in your environment.

You will need to carry out different steps according to the hypervisors you are using on your hosts in the zone. The users can also have multiple template packages on a single secondary storage server that helps in using the same secondary storage for various hypervisor type clusters.

- For VSphere:

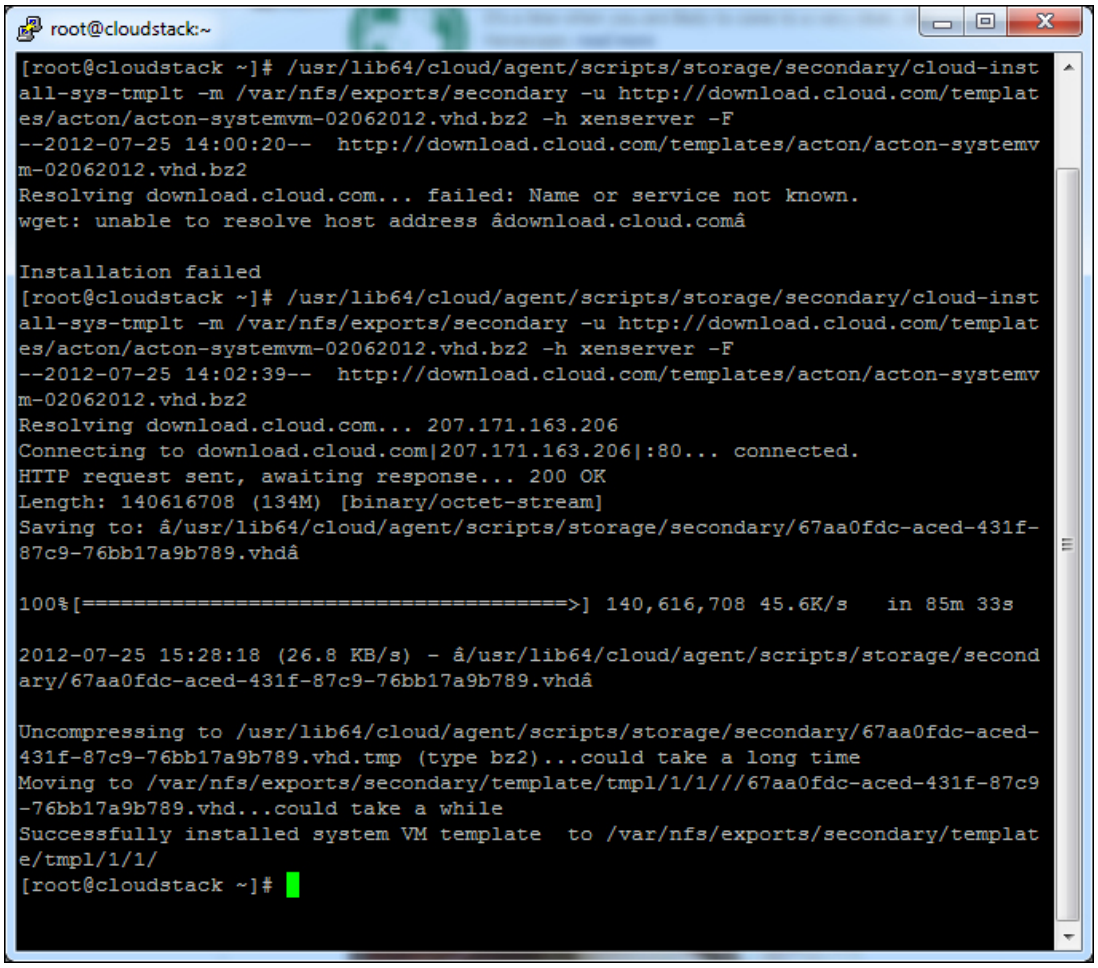
```
#/use/lib64/cloud/agent/scripts/storage/sec
ondary/cloud-install-sys-tmplt –m
/mnt/secondary –u
http://download.cloud.com/templates/acton/a
cton-systemvm-02062012.ova –h vmware –s
<optional-management-server-secret-key> -F
```

- For KVM hypervisor:

```
#/use/lib64/cloud/agent/scripts/storage/sec
ondary/cloud-install-sys-tmplt –m
/mnt/secondary –u
http://download.cloud.com/templates/acton/a
cton-systemvm-02062012.qcow2.bz2  -h kvm –s
<optional-management-server-secret-key> -F
```

- For XenServer:

```
#
/usr/lib64/cloud/agent/scripts/storage/seco
ndary/cloud-install-sys-tmplt -m
/mnt/secondary -u
http://download.cloud.com/templates/acton/a
cton-systemvm-02062012.vhd.bz2 -h xenserver
-s <optional-management-server-secret-key>
-F
```



- After running the above scripts, the secondary storage can be unmounted and the directory must be removed. This is done using the following commands:

```
#umount /mnt/secondary
#rmdir /mnt/secondary
```

Note

You will need to replace the mount point of the secondary storage with `/mnt/secondary` and you must use the `-s <optional-management-server-secret-key>` option if you

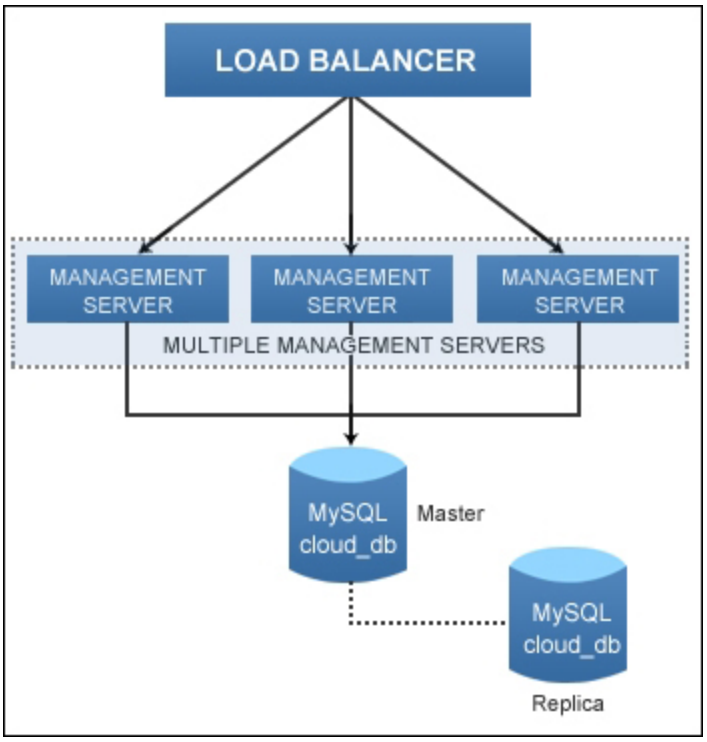
have set the CloudStack database encryption type to `web` while setting up the database.

This completes our single node installation of CloudStack where there is only one management server. This installation prepares the management server and CloudDB in a single server that is not recommended in a production environment. It is always recommended to use a highly available architecture in production. Note that the single node installation of CloudStack can comprise of multiple database servers but has only one management server.

Multinode installation

In a multinode installation, the management server(s) is installed on a separate node than database server. In our multinode installation of CloudStack, we will set up the CloudStack management server in a highly available fashion such that there are multiple management servers that link up to the database server(s) and these management server can serve request from different users and thus, you get a better performing load balanced Cloudstack environment. Not only this, the CloudStack database can also be installed on multiple nodes of database queries. These different database servers are basically replicas of the master database server and are synchronized with the master server. The database read replicas can be used to balance the load for the read requests whereas all the write requests are served by the master node. The slave nodes are in continuous sync with the master node.

Multinode installation can be visualized in the figure below. Let's look into the steps to install a Multinode management server. The management server requirements are the same as defined earlier in a single node installation.



The multinode installation process is very much similar to the single node installation process. You need to carry out steps 1 to 6 from the single node installation of CloudStack. Once you are done preparing the OS for the installation of the CloudStack management server. We can proceed with the installation of the first nodes of the management server.

Management server installation

Assuming that we have built the packages from the source as described above, and we have also set up a repository so that the built packages can be accessed over HTTP. We have to add the repository entry in each of the machines so that the provider `yum` in Centos/RHEL or `apt-get` in case of Ubuntu knows the path of the repository from where to fetch the Apache CloudStack packages along with the dependencies.

For installing management server on multiple nodes, we need to carry out the steps for installation as described above in the case of single installation on all of the machines.

Next we can continue with the installation of CloudDB that will act as the database server.

The installation steps of MySQL database are also similar to the installation that we did for single node installation. But here we need to configure the database so that it can be connected to multiple management servers. Let's see how to achieve this.

Installing and configuring CloudStack MySQL database

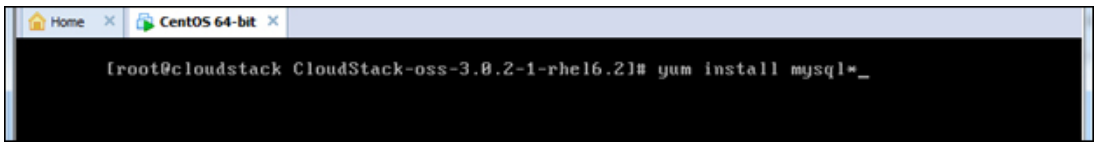
The users can install MySQL from a separate package depending upon the server's OS; or, if you already have a version of MySQL installed, you must check whether there is an upgrade required or not. The MySQL version that has been tested with CloudStack and found to be stable is 5.1.58.

In case you have already got MySQL installed on your system, you must consider upgrading it to the required version (5.1.58) or you can also simply uninstall it and re-install the latest version.

The following steps show the installation of a MySQL server from the `yum` repository. These steps are for the Master database server in case you are going to create replicas.

1. Log in to the server as the root user that is going to be the database server of your environment.
2. Use the following command:

```
#yum install mysql*
```



Note

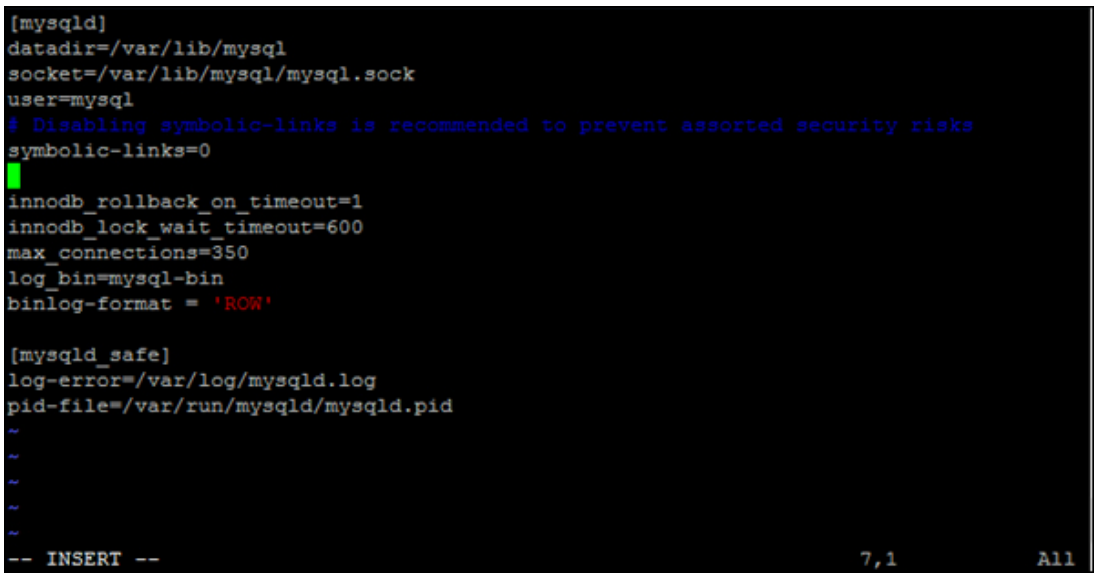
That we used `mysql*`, it will resolve all the dependencies of the package and install them with the available plugins.

3. Use the following command to put MySQL daemon as the system service of run levels 3 and 5.

```
#chkconfig –level 35 mysqld on
```

4. After you have done this, we have to edit the MySQL configuration file located at `/etc/my.cnf` or /`etc/mysql/my.cnf` that may vary depending upon the OS you are running. We need to add some extra lines to the file as a part of our configuration of database server. You will need to add the following lines to the `[mysqld]` section.

```
innodb_rollback_on_timeout=1
innodb_lock_wait_timeout=600
max_connections=700
log-bin=mysql-bin
binlog-format = 'ROW'
```



5. Once this is done, we need to restart/start the MySQL server in order to run it with the configurations done.

    - Use the following command in case of RHEL or CentOS:

```
#service mysqld restart
```

    - In case of Ubuntu, you can use:

```
#service  mysql restart
```

6. We also need to set the password for the root user of MySQL, use the following command to log in to the MySQL server in case of RHEL or CentOS:

```
#mysql –u root
```

    In case of Ubuntu, use the password you have set while installation:

```
#mysql –u root –p <password>
```

7. Next, we need to set the password in case of RHEL or CentOS, because it is not set during the installation process. You can use the following command to set the password:

```
mysql> SET PASSWORD = PASSWORD('password');
```

```
[root@cloudstack CloudStack-oss-3.0.2-1-rhel6.2]# mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.1.47-log Source distribution

Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL v2 license

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement
.

mysql> SET PASSWORD = PASSWORD('password');
Query OK, 0 rows affected (0.00 sec)

mysql> exit
Bye
[root@cloudstack CloudStack-oss-3.0.2-1-rhel6.2]#
```

From now on, whenever you start the MySQL server, you need to do it using the `mysql -p` command and it will prompt you to enter the password for the same.

8. Once all of this is done, you need to grant permissions to remote users so that it can be accessed remotely. Use the following command to do this:

```
mysql>GRANT ALL PRIVILEGES ON *.* TO
'root'@'%' WITH GRANT OPTION;
```

9. Restart your MySQL service as in step 5.
10. You must open the `mysql` ports (3306) on the firewall of the DB server just created to allow the communication to the server. For doing this you will have to enter the command in the shell:

```
# iptables -I INPUT -p tcp --dport 3306 -j
ACCEPT
```

11. You should also make this change permanent, by editing the file at `/etc/sysconfig/iptables`.

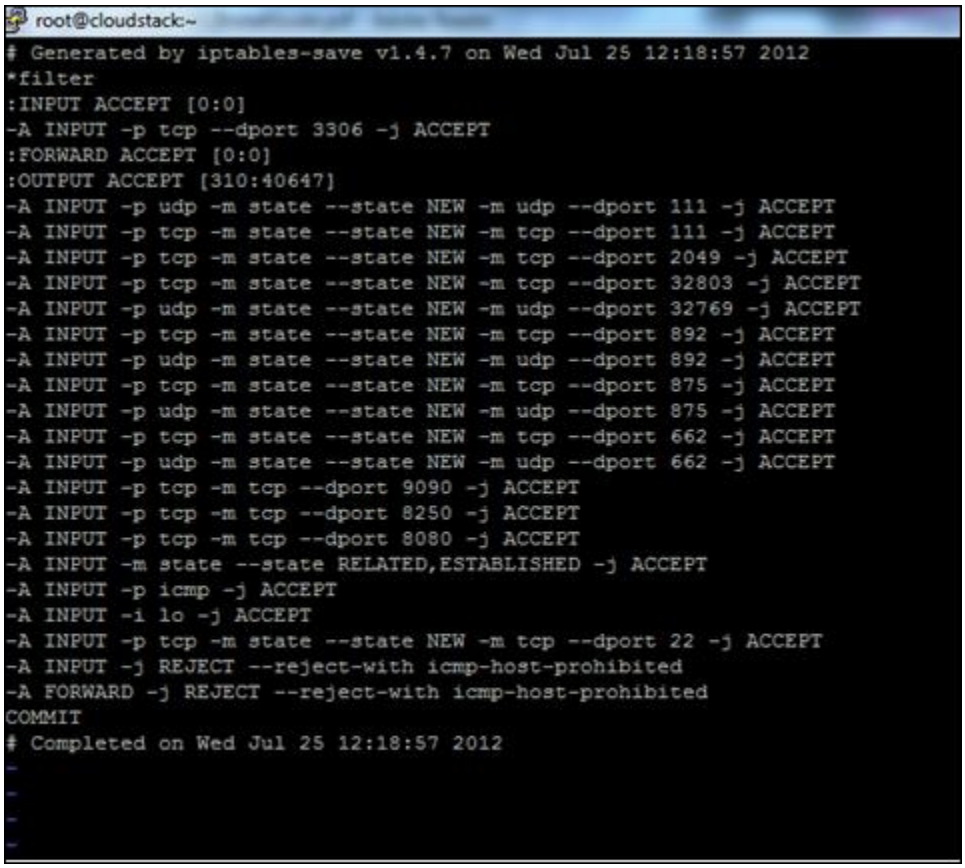
    - Open the file using:

```
#vi /etc/sysconfig/iptables
```

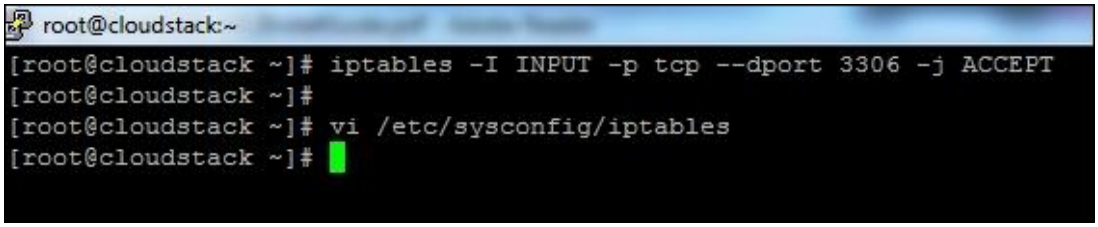
- Enter the line in the `INPUT` section of the file:

```
-A INPUT -p tcp --dport 3306 -j
ACCEPT
```

The contents of the file will be as shown in the following screenshot:



12. Save the file and exit.



Now, we are done setting up the database server and we can return to our first management server.

1. We need to set up the database for the CloudStack management server so that the CloudStack management server and the database are connected. For doing this we will go through a series of steps, running some commands on the management server. Log in to the shell of management server and run the following command:

```
# cloud-setup-databases cloud:
<dbpassword>@<dbhost> --deploy-as=root:
<password> -e <encryption_type> -m
<management_server_key> -k <database_key>
```

2. You will need to replace `dbpassword` with the password you want to set for the database (leave it blank if you don't want any password) and `dbhost` with the hostname of the CloudStack database server. The `cloud-setup-database` script takes some additional parameters such as:

    - **Encryption type**: This includes `env`, `web`, and `file` (by default value) and is stored in the database properties file `db.properties` in the value of `db.cloud.encryption.type`.
    - **Management server secret key**: This is the key used to connect to the management server and defaults to password. This is stored in the environment variable as `CLOUD_SECRET_KEY`.
    - **Database secret key**: This key is used to encrypt and decrypt the data values and is stored in one of CloudStack's internal properties file.

    We will discuss more about keys in general later in this chapter.

    When the user doesn't provide any of these parameters, the default value is taken for each of them but it is highly recommended that we change these default values for security concerns.

3. It is also not necessary to deploy the database as root user; it can be any combination of username and password.
4. Optional fields:

    - If you are using any encryption type, you can provide that information or else leave it blank.
    - You may also pass the Management Server key that helps in encrypting the confidential information in the CloudStack properties file. You must also replace the password with the default password with a new one for security reasons.

```
# cloud-setup-database
:mypassword@databasenode.cloudstack.
packt.com –deploy-as=root:password
```

We are done configuring the CloudStack database in the management server. If there are multiple management servers, you may need to follow the configuration steps as above once again.

You must set the `iptables` rules for enabling the communication between the management server and database node as well and also add the services to the system services so that they are started by default whenever the system changes its state (started or stopped) using `chkconfig` and then you can start the management server. This all can be done using the following script:

```
# cloud-setup-management
```

Now, we can proceed on to the setting up of the NFS share for the primary and secondary storage as we did after completing the single node installation.

For a production environment, it is highly recommended to use multiple management servers so that there is load balancing for all the requests. For this kind of setup, you must also set up a load

balancer configured to send requests based on some algorithm. To balance the load across the management servers you can use various options available in load balancers such as Round Robin with sticky sessions.

Installing additional management server

For installing an additional management server, follow these steps:

The steps are similar to the installation of the first management server as done earlier.

1. Prepare the OS as we did earlier in the single node installation.
2. Add the repository to the server you want to install the Management server on.
3. Install the Cloud-client.

    - For Centos/RHEl

    ```
    #yum install cloud-client
    ```

    - For Ubuntu

    ```
    #sudo apt-get install cloud-client
    ```

> **Note**
>
> Refer to the installation steps carried out above for more detailed description of each steps mentioned as we have carried out the same steps in case of single node and while installing first management server in multinode installation.

4. Now that we have completed the installation of an addition management server, we need to map it to the address of the database server. To do so, run the following command on the management server you just installed.

```
# cloud-setup-databases cloud:
<dbpassword>@<dbhost> -e <encryption_type>
-m <management_server_key> -k
<database_key>
```

5. The installation of additional Management server is now complete; you can now configure and start this server so that it can be put into service. Use the following command:

```
# cloud-setup-management
```

This management server points to the same database server we had set up in the multinode installation. Note the absence of the `--deploy-as` argument in the command.

If you want to add more management servers in the highly available environment, carry on the steps 1 to 5 above.

In CloudStack, multiple database servers can also be installed for replication to provide high availability of the database server. MySQL server supports replication. In this kind of setup, various slave databases can be installed who remain in real-time synchronization with the master database server using the binary logs. These slaves can only be queried by the master database server; the management servers cannot run any query on these slave database servers. The automation process of promoting one of the slave servers as the master server when the master database server is down can be done using Linux HA heartbeat and Distributed Replicated Block Device. Thus, we can configure multiple database servers as well as multiple management servers in our environment. We are done with the installation of highly available CloudStack setup. The next thing remaining in this section is setting up the system VMs as we did in the single node installation steps.

Preparing the system VMs

We need to follow the same procedure as we did in the case of a single node management server installation.

Keys and encryption

We have used various types of keys in this chapter while installation. Let's discuss them in detail. Keys and encryption are features provided by CloudStack and are recommended to for ensuring security of the installation. CloudStack allows various secret keys that enable high end security features. These key values are encrypted by default. The following are the keys which are available in the CloudStack environment:

- **Database secret key**: This key is used to encrypt and decrypt the data values and is stored in one of CloudStack's internal properties files.
- **Database Password**: This is also stored in the internal properties file `db.properties` with the database secret key in the variable `db.cloud.encrypt.secret`. It is also stored in file and is encrypted using the management server secret key.
- **SSH keys** : These keys are used for logging in to the servers.
- **Compute Node root password**: This is the root password.
- **VPN password**: This key is used in case of VPN connection for security association.
- User API key which is used for API calls.
- VNC password.

Java Simplified Encryption (**JASYPT**) library is used for encryption. The keys mentioned above are stored in CloudStack internal database.

All the passwords and the secret keys to be used in CloudStack are either stored in text format in the database or on the management server.

The management server secret key will be used for encrypting the text values. This secret key as mentioned earlier is stored in a file `/etc/cloud/management/key` or can also be provided by the admin on management server start up. Once the key is available to the management server, the database password in `/etc/cloud/management/db.properties` will be decrypted and thus the management server will be able to connect to the database server.

Due to security reasons, the database secret key is not shared, hence another key is provided to the management server by the administrator, or it reads it from a file. This is configured while configuring the CloudStack database that lets CloudStack know which one to use.

If you set the encryption to `file`, the key must be present in a file whose location must be configured and if the encryption type is set to `web` then the key is provided to the management server over a port by running the utility `com.cloud.utils.crypt.EncryptionSecretKeySender`. This is configured while setting up the database using the following command:

```
#cloud-setup-databases
```

The preceding command takes parameters to the options, by default; the values are `file`, `password`, and `password`. But it is recommended to change them.

Summary

In this chapter we covered the following:

- All the steps that are involved in installing CloudStack in both the styles, single node and multinode
- It describes in details the prerequisites of both hardware and operating system to ensure the proper installation and functioning of CloudStack

Now that we have seen the various components of CloudStack and their installation, we can now begin to configure the cloud. This includes:

- Adding infrastructure to CloudStack
- Configuring CloudStack
- Details of different types of offering with their creation and modifications
- Creating users, domains, projects, and so on

Chapter 3. Apache CloudStack Configuration

In the first two chapters we covered the following:

- Installation of CloudStack
- Various options available to us for the deployment of CloudStack

Now that we have successfully installed CloudStack in our environment; let's take the next steps towards the configuring it. We will introduce to you the following in this chapter:

- Introduction to the management console
- The process of configuring the CloudStack environment
- Adding IT infrastructure to CloudStack

CloudStack configuration

After the installation is complete, we can verify the IP Address of the Management Server by using the following command line:

```
cloud-management-setup
```

The output tells us the IP address that is used to bind the management server, in case you have multiple network interfaces. You can open the user interface in a web browser using the following URL:

```
http://<cloudstack-management-server-ip-address>:8080/client
```

The user interface can be accessible using the IP address of the management server or the FQDN, If you are using the FQDN of the

management server, the domain name should be resolvable from the host that you are accessing. It can be done using a pointer record in the DNS server of the network or through an entry in the host file.

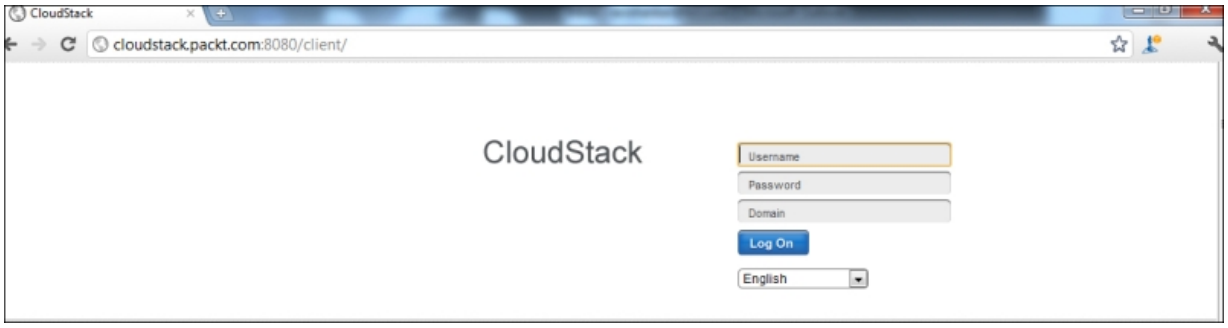
You will need to use the management server's IP address or management server's hostname in place of the `<cloudstack-management-server-ip>`. For using the hostname, you must have the DNS entry in the host file of the machine from where you are trying to access the UI. One can use common browsers such as IE7+, Firefox 3.5+, and Safari 4 and Safari 5 for viewing the UI.

This interface is for both end users as well as for administrators. The end users can use this UI for viewing, using and managing their cloud resources such as virtual machines, templates, ISO, data volumes, their snapshots, networks, and IP addresses.

The UI can be used by CloudStack administrator for various purposes such as:

- Provision, view, and manage the cloud infrastructure
- Create, update, manage, or delete new domains
- Add new user accounts, delete, or manage existing users
- Create, update, or manage projects

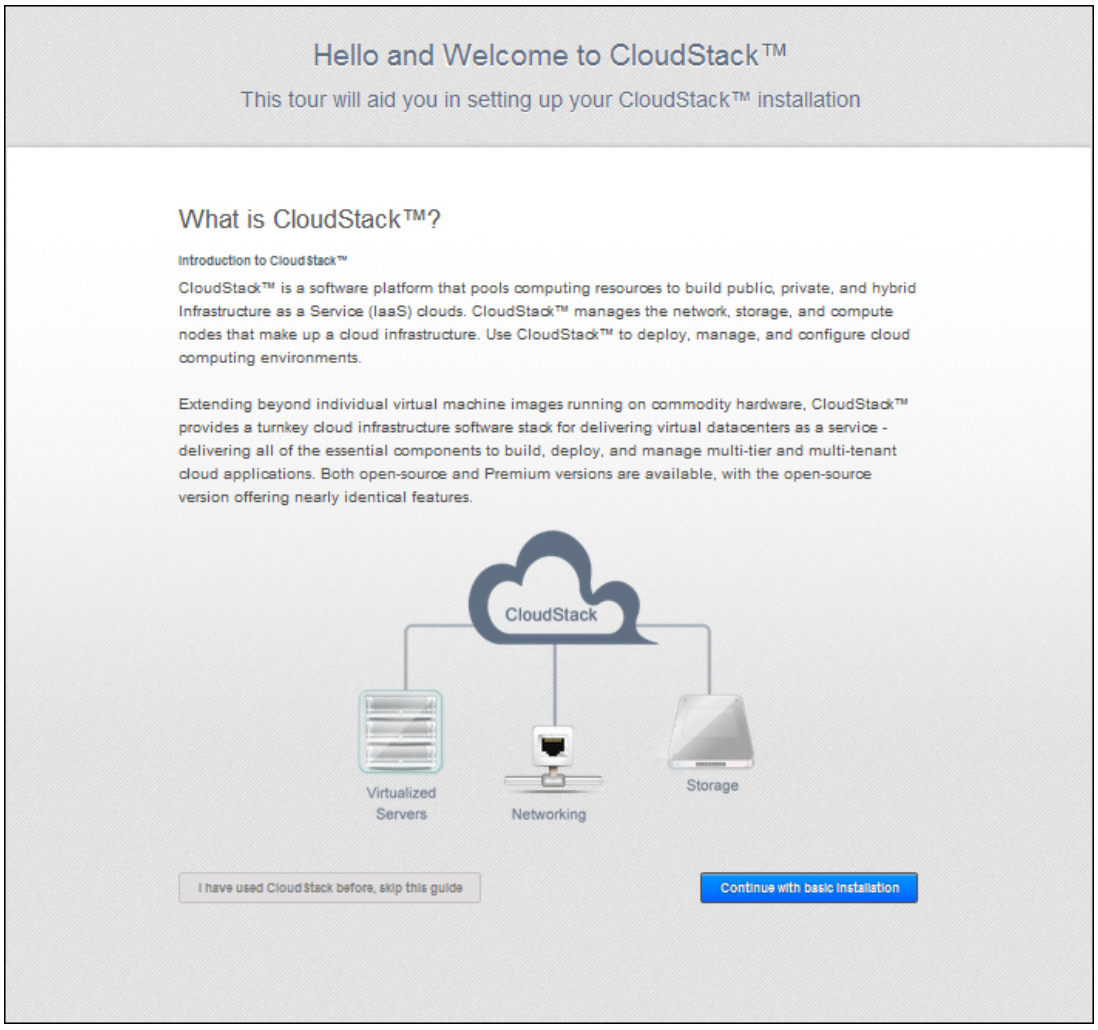
When you enter the URL as provided above to access the user interface, you will get a screen as follows. The default username and password for the CloudStack management server user interface login is `Admin` and `password`. The users log in to the User Interface using the default credentials the first time and should change the default password after that. Later on, when the domains are configured, the domain users (which is discussed later) can also log in using their credentials along with the domain name. The users in the root domain don't need to provide the domain name while logging in.



On the first root/administrator login after the installation of CloudStack, you will be given an option to take a guided tour to CloudStack to provision your cloud infrastructure. On the next login attempt you will get the dashboard instead.

On the first login attempt, you will encounter a screen which has two options as follows:

- **Take a guided tour for CloudStack provisioning**: CloudStack makes it easy for administrators to use the system by providing a guided tour of the provisioning process in CloudStack. As an administrator you can choose this option when you want to take a guided tour of CloudStack, this will include an introduction to the basic configuration to get you started with CloudStack. It will walk you through the steps of creating a basic zone containing one Pod, containing one Cluster and one host with NFS servers to provide primary and secondary storage, and a running virtual machine under XenServer hypervisor or KVM hypervisor and a shared public network.

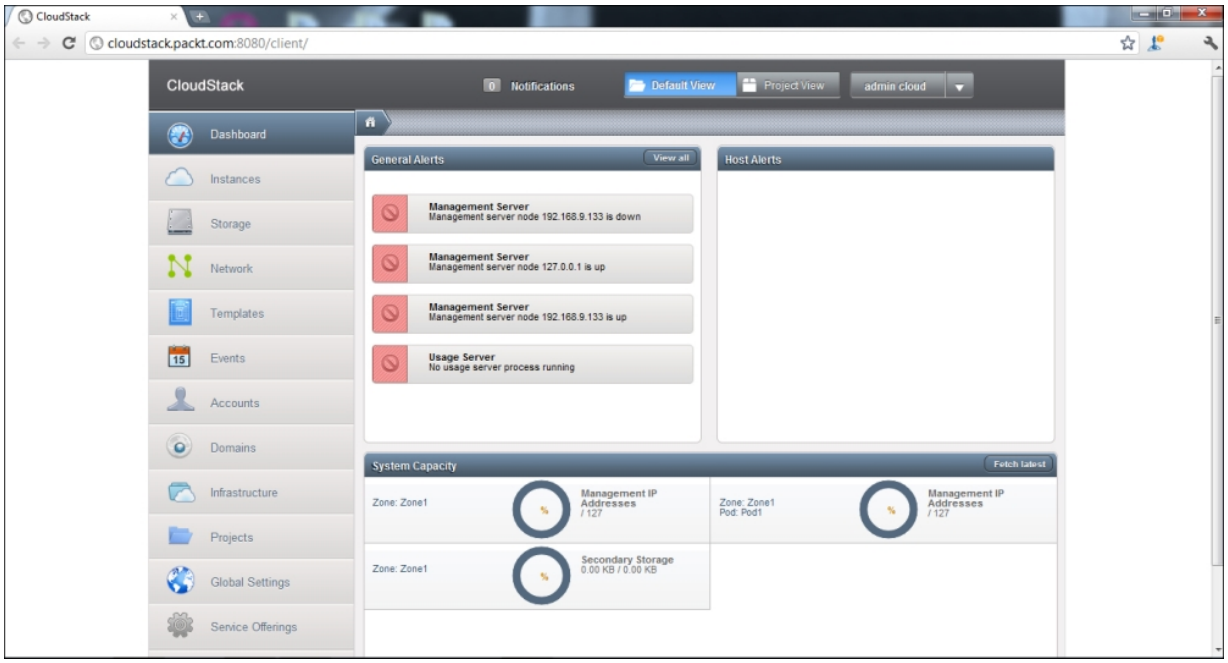


- **I have used CloudStack before**: This option should be chosen when you have already gone through the CloudStack deployment and designing. The user interface is used by the administrators to configure the cloud by allowing them to create offerings of different types so that they can meet their design and deployment plan. It will take you to the console which will be used to create, add, and modify various configurations so that you can add support for powerful features such as Advanced VLAN networking, high availability, load balancers, firewalls, and multiple hypervisors such as Citrix XenServer, KVM, and VMware vSphere.

You have the option of changing the default admin password at the installation time. If you haven't done it yet, it is recommended that you change the admin password first, it will be prompted to you

when you choose the basic guided tour option or if you have chosen the experienced user, then you should first change the admin password before continuing for security purposes.

We will continue with the second option on the screen and will configure the CloudStack environment so that we can manage our infrastructure. After logging in as the root user, you will get a screen as follows:



This screen is the Dashboard of the CloudStack Management Server which shows the status of all the Management servers if you have chosen a multinode installation.

The screen has two kinds of views:

- **Default view**: This provides the resources and options corresponding to the link to overall deployment.
- **Project view**: This filters the resources as per the project selected. When you click on the project view, you get to select the project from the various projects created. Once you select a particular project, you will be able to view the resources and configuration specific to that project.

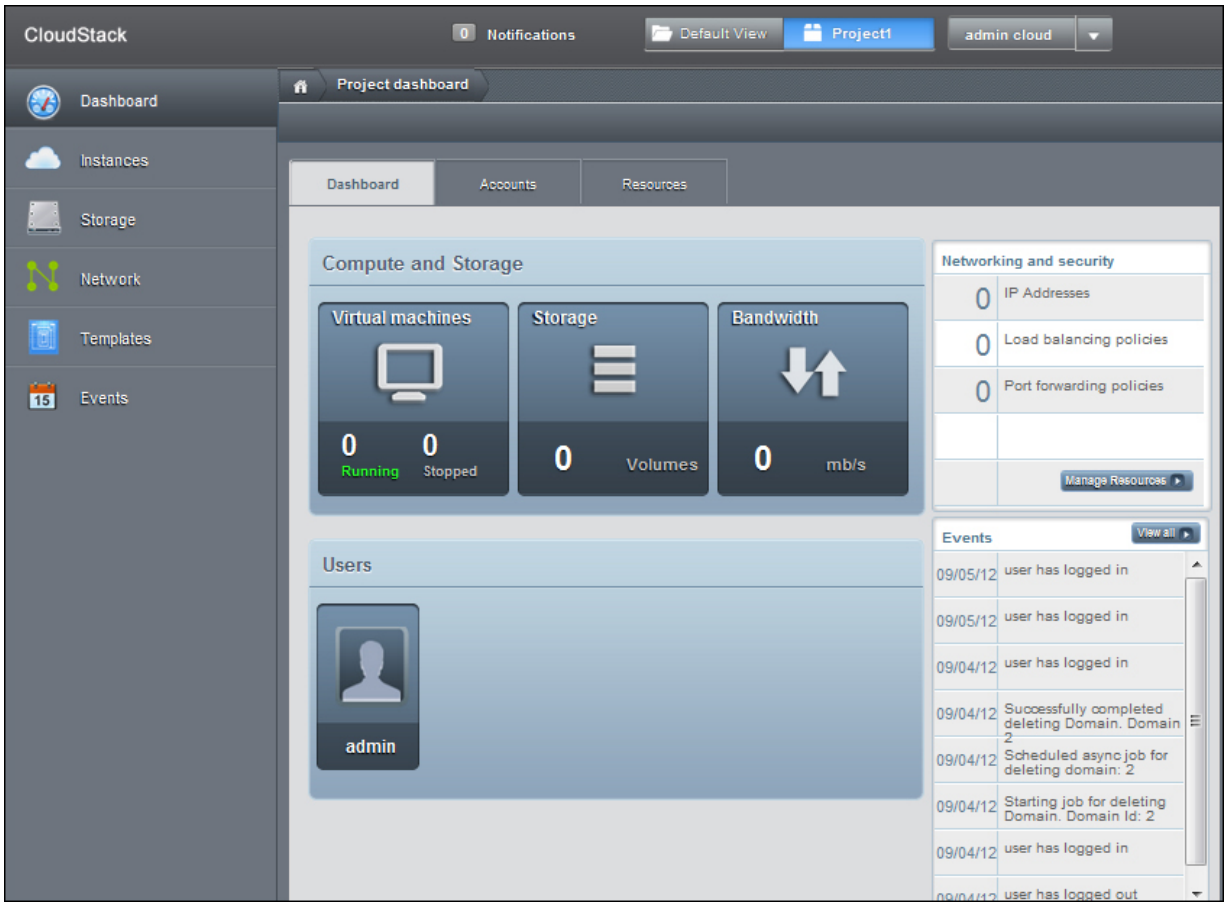
We will start with a brief overview of the options available on the left menu. These links on the side panel provide access to a variety of administrative functions.

Management server console

The management server console is used by the administrator to configure the cloud and to create various types of offerings as per the design and deployment plan. It is used by the users to request resources from the cloud by selecting from the offerings designed by the administrators. Some of the panel options that are available in the UI are discussed in the following sections.

Dashboard tab

The dashboard is the page where you get a brief overview of the health of various services registered in CloudStack as well notifications for various events triggered by the CloudStack system. In the default view, one can see the alerts that are triggered, where as in a project specific view, the dashboard shows the virtual machines, volumes, users, events, network settings, and so on that are specific to a project. The project dashboard screen looks as follows:



From the dashboard, the user can perform a number of functions by navigating to the page directly.

There are total of 3 tabs available which guide to other operations, they are:

- **Dashboard**: The dashboard screen highlights the number of VMs, storage volumes, bandwidth, and users that are configured with CloudStack.
- The administrator can also manage network resources by using the link given on the right-hand side. On clicking the **Manage Resources** link on the right side menu, the admin can add new guest network to the CloudStack project, can acquire new IP addresses, and can set up load balancing and port forwarding rules. The administrator can also use the **Network** tab on the left to manage the network.

- **Accounts**: The administrator can view and manage members of the project selected. The administrator can also add new members to the project, remove the existing members as well as change the role of a user. There are two roles available by default: user and admin. There can only be one admin user at a time in a project and a user can belong to only one project and one account.
- **Resources**: On the resources tab, the user can view the limits to various resources on the project. The project administrator can also lower the resources also.
- **Invitation**: This tab can also appear if invitations are enabled in the global configuration settings (discussed later). Using the **Invitations** tab, the users can view and manage the invitations which have been sent to new members of the project selected.

CloudStack can be configured in such a way that the users are directly added to it or it can be a necessity for a user to accept the invitation to join cloud. The members to whom the invitations are sent can either accept their invitations or reject them. The pending invitations will be listed here until the timeout specified. The user can also cancel the invitations. Once the invitation is accepted, the user is added to that project.

The invitations can be enabled in CloudStack by setting the parameter `project.invite.required` to `true` in the global settings page. There are various other parameters that are used to customize the request.

Instances tab

The **Instance** tab of the left panel will display the instances that are provisioned using CloudStack. In the default view, all the instances in the CloudStack environment will be displayed, and in the project view, various virtual machines that are provisioned under the selected project will be displayed. The various details of the instances are displayed for all the instances.

Storage tab

The **Storage** tab allows the administrators to create, view, and manage volumes and snapshots. These volumes are provided by the storage configured in CloudStack set up. The various details of volumes are also provided on the page. In the project view the volumes available in the project selected are displayed as well the users who can create, view, and manage storage volumes and snapshots.

Network tab

The **Network** tab provides us options to view, add, and manage the guest networks present in CloudStack. The users can acquire IP addresses, set up the load balancers, and create port forwarding rules.

Templates tab

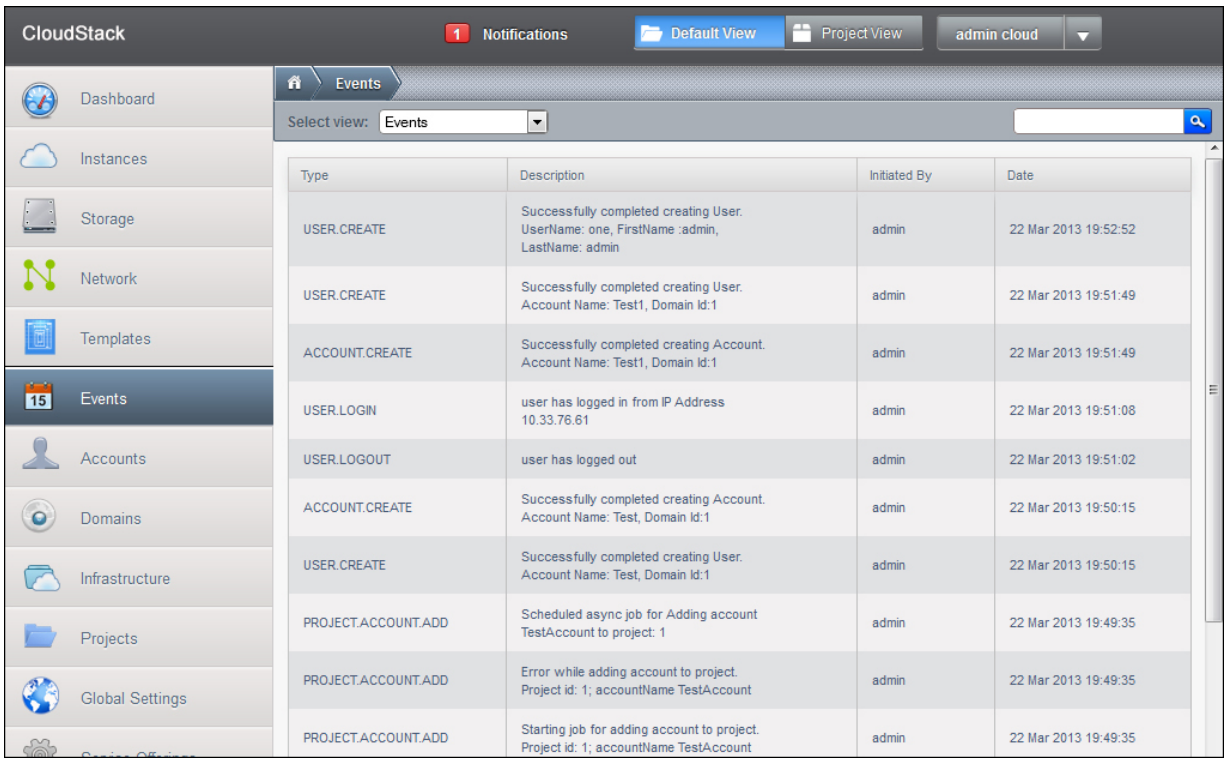
The **Templates** page displays the various existing templates. The user can also create new templates by providing the description, selecting from the list of hypervisors, or selecting the OS type, and so on.

The templates can be created by the CloudStack administrator as well as any of the cloud users. The templates are the OS images using which the guest instances are created. These templates contain the configuration information about the OS type, software packages, and so on and they are associated with particular hypervisor. While creating an instance, the user specifies the template, based on which the instance is created.

Events tab

The **Events** page shows the various events in the CloudStack deployment. It basically acts as the logging place where one can see

the log of activities performed in CloudStack. The following screenshot displays a typical event page displaying all the events.



Accounts tab

The **Accounts** tab enables the admin to create, view, and manage accounts in various domains. Accounts created in cloud basically represents customer of the service provider or it can also be used to distinguish departments in a large organization. There can be many users in a single account. The resources in the cloud can be shared between users of the same account whereas the users in different accounts are isolated from each other. An account can have any of the two roles—user or admin.

Domains tab

Domains are basically referred to a group of accounts—a domain can contain multiple accounts. The domain is used to group some accounts that are related to each other. A domain can also contain

various sub domains. Domains can be created for resellers by the service provider.

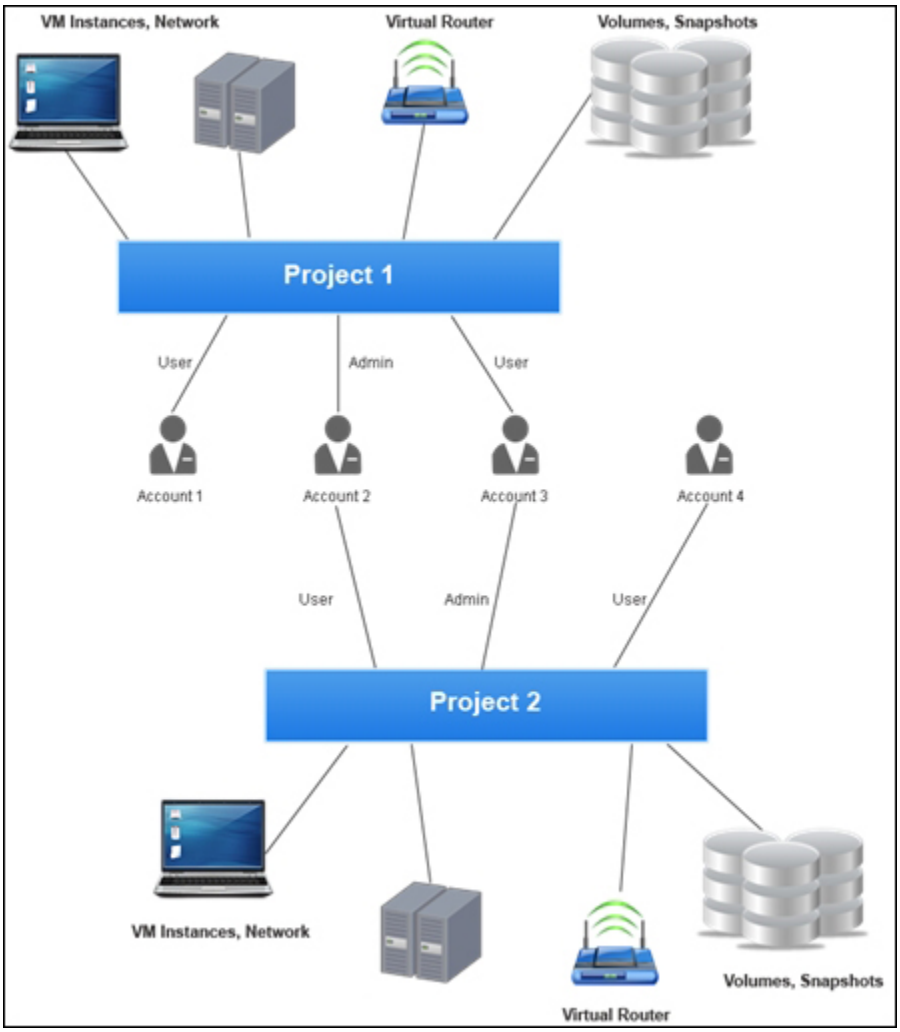
In the domains page, the user can create, view, and manage domains in the CloudStack environment. The administrator can view various accounts in a domain. The administrator can update the domain and also view the details of an existing domain; for example, the limits on various resources such as volume, IP, and network as well as view the details of number of volumes and networks.

Infrastructure tab

The **Infrastructure** tab provides details about the CloudStack infrastructure such as number of zones, pods, clusters, and hosts, number of pods, number of clusters, and the number of hosts. The user can navigate to the any of the infrastructure component to create, view, and manage them such as the user can navigate to the zones and view all the existing zones, add new zone by providing various required details.

Projects

The **Projects** tab displays the projects details. The administrator can view all the projects that exist in the CloudStack environment and a user can view the project details of their projects. The creation of projects allows administrators to create groups for dividing their workforce; these groups share resources in the CloudStack environment. An organization can create various projects with different groups of people to share the resources in the CloudStack environment. The project administrators have the privileges to set limits on the amount of resources such as VM instances, snapshots, templates, and volumes. The resources created by a user in a project belong to that project and cannot be used outside that project. The user can also create resources outside a project, but this resource belongs to the user's account and is not a part of any project. A single user may belong to multiple projects and will have access to resources based on the projects assigned to them.



The administrators can also create a project specific network to isolate the project related traffic in the CloudStack environment and can also provide project specific network services such as port forwarding and NAT. CloudStack also provides shared resources which can be shared across projects as well as used by individual users. The options for making resources dedicated to a project or keeping them shared are a part of adding resources.

Global settings

The global settings page provides the settings for various parameters in the cloud. These parameters control many aspects of cloud. The CloudStack administrator is recommended to check the

global settings of parameters. The global settings page mainly has two options, which are:

- **Global settings**: The configuration of various parameters. This lists the various parameters that define the settings and configuration in cloud.
- **Hypervisor capabilities**: This displays various hypervisors with their respective versions that are supported by CloudStack. The administrator can click on the hypervisor to edit as well as view that hypervisor capability; for example, the administrator may choose to increase the maximum guest VM limit on that hypervisor.

Now that we have completed the introduction to the user interface and the various options available, we will begin with the process of setting up the infrastructure and users to the cloud. Before that, let's go through some of the keywords which will be used very frequently in the chapter.

Administrators

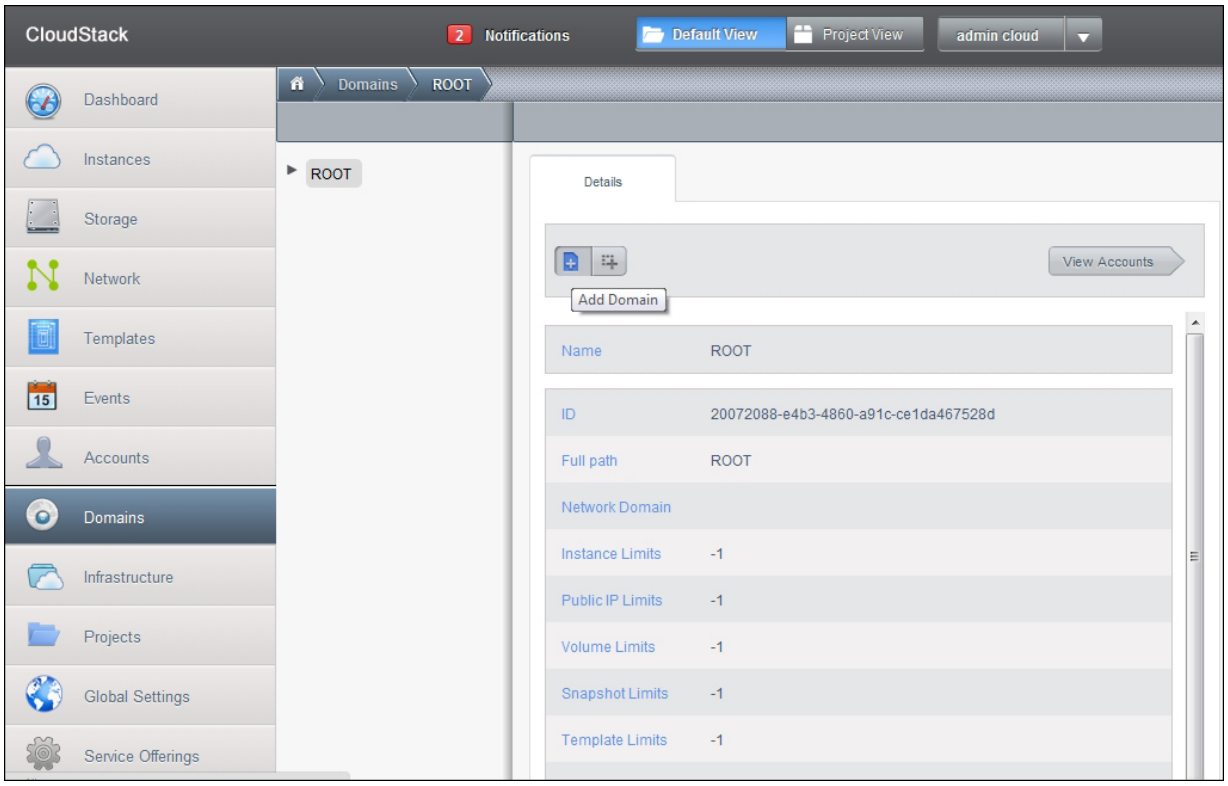
There can be various types of administrators:

- **Cloud administrator**: There can be cloud administrators who have complete access to the system who can manage templates, service offerings, projects, users, accounts, domains, create, and delete other administrators.
- **Project administrator**: The project administrators have access to the resources in the project only, they can add, view, delete, and modify the existing users, accounts, which belong to the project. They don't have the privileges to access the physical resources or any other projects.
- **Domain administrator**: The domain administrators have access to the resources in that domain only and cannot perform actions on resources other than their domain.

You can also integrate an existing external LDAP with CloudStack like Microsoft Active Directory services or ApacheDS. The integration

of CloudStack can be done with LDAP or ADFS using API provided by CloudStack. CloudStack searches external LDAP directory tree structure for matching common values for users such as their e-mail address and name. We specify the base directory in our LDAP directory tree and CloudStack will start the search from that directory and return the name of **Distinguished Name** (**DN**) of the matching user. The user can use this DN along with their password to log in to the CloudStack. The integration of CloudStack with external LDAP services can be done using the CloudStack API.

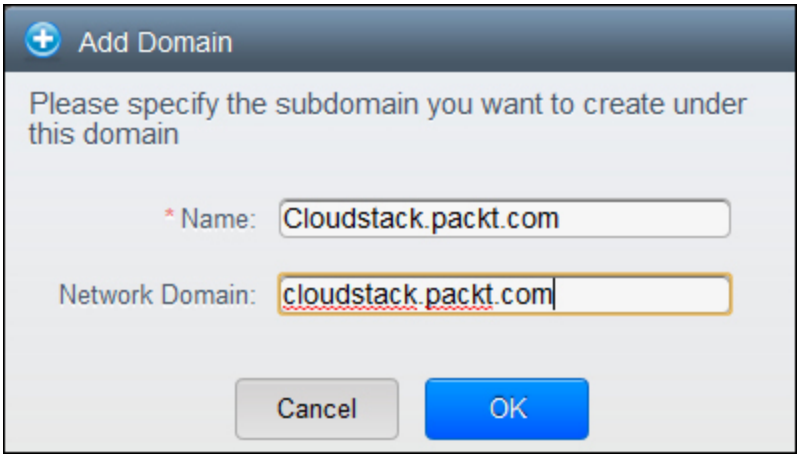
Now, as we have explained that there can be various administrators in CloudStack, let's have a look into what are these domains, accounts, and projects. We will go through a step-by-step process of creating users, accounts, domains, and projects. When you first install CloudStack, a domain name `Root` is already created for you. You can create new domains by going to the **Domain** tab and clicking on **Add Domain** as shown in the following screenshot:



Creating a domain

You will have to enter the details of the domain that you are creating now:

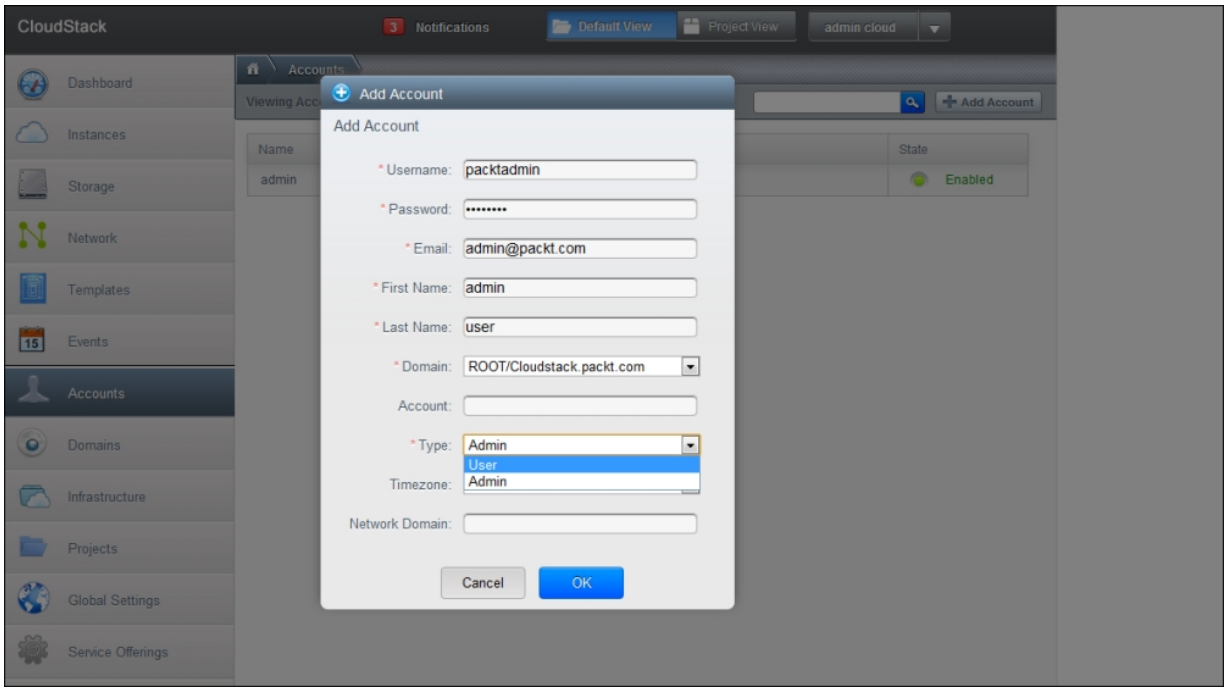
- **Name**: This is the name of the new domain that you are creating.
- **Network Domain (Optional)**: This is the custom DNS suffix that you may want to assign to the network in this domain.



On clicking the **OK** button, a new domain of name `Cloudstack.packt.com` will be created. Now you can add new accounts under this domain from the **Account** tab and clicking on **Add New Account**.

Creating an account

The new Account creation wizard is opened where you will have to enter the details of the account.



This operation will create an account of the name `packtadmin` as well as user of name `packtadmin`. This user that we are creating with the account can be the user in that domain or the Admin of the domain.

New users can also be added, removed, or modified from the **Account** tab by navigating to the account into which you want to create the new user.

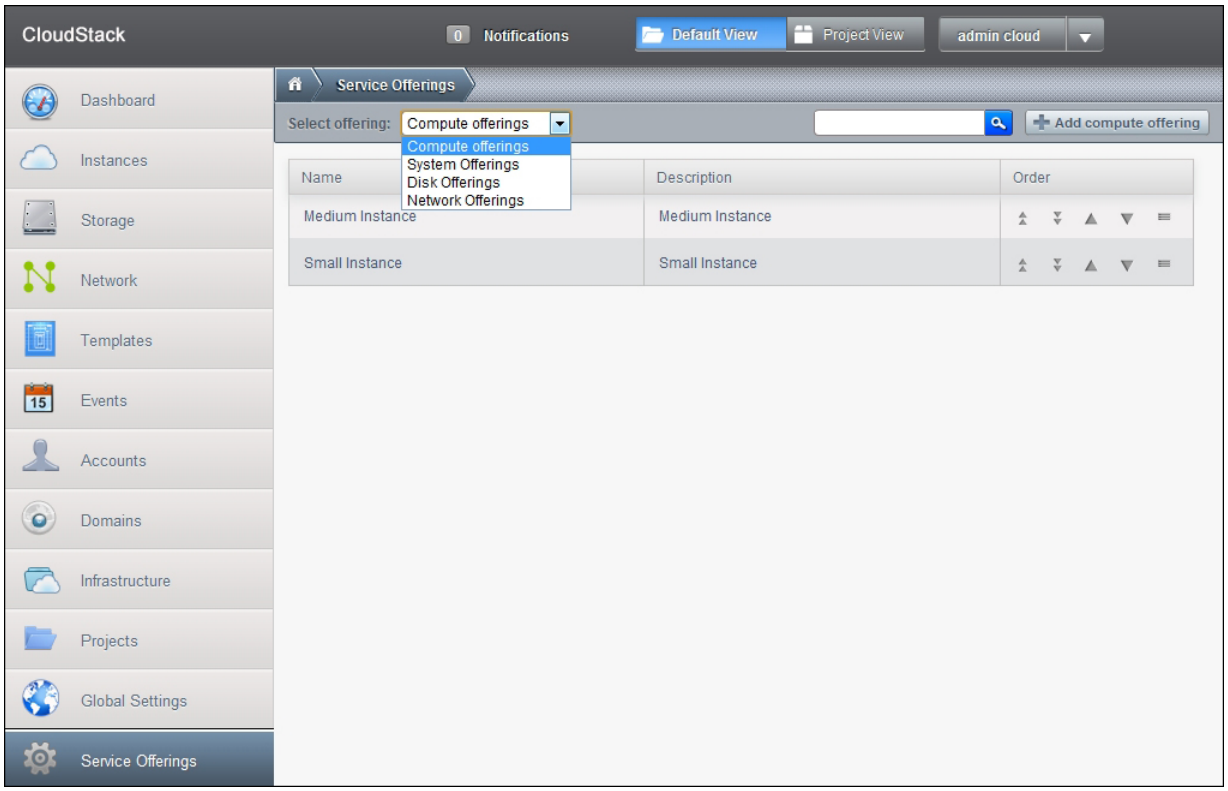
Service offerings

Let's move on to adding offerings in the cloud which we will be providing to the users. The users as well as the admin can go to the Management server console and request any one of the offerings from the list which we will be adding and configuring in the cloud as a cloud administrator. There are various types of offerings that can be created from the cloud console, they are described below.

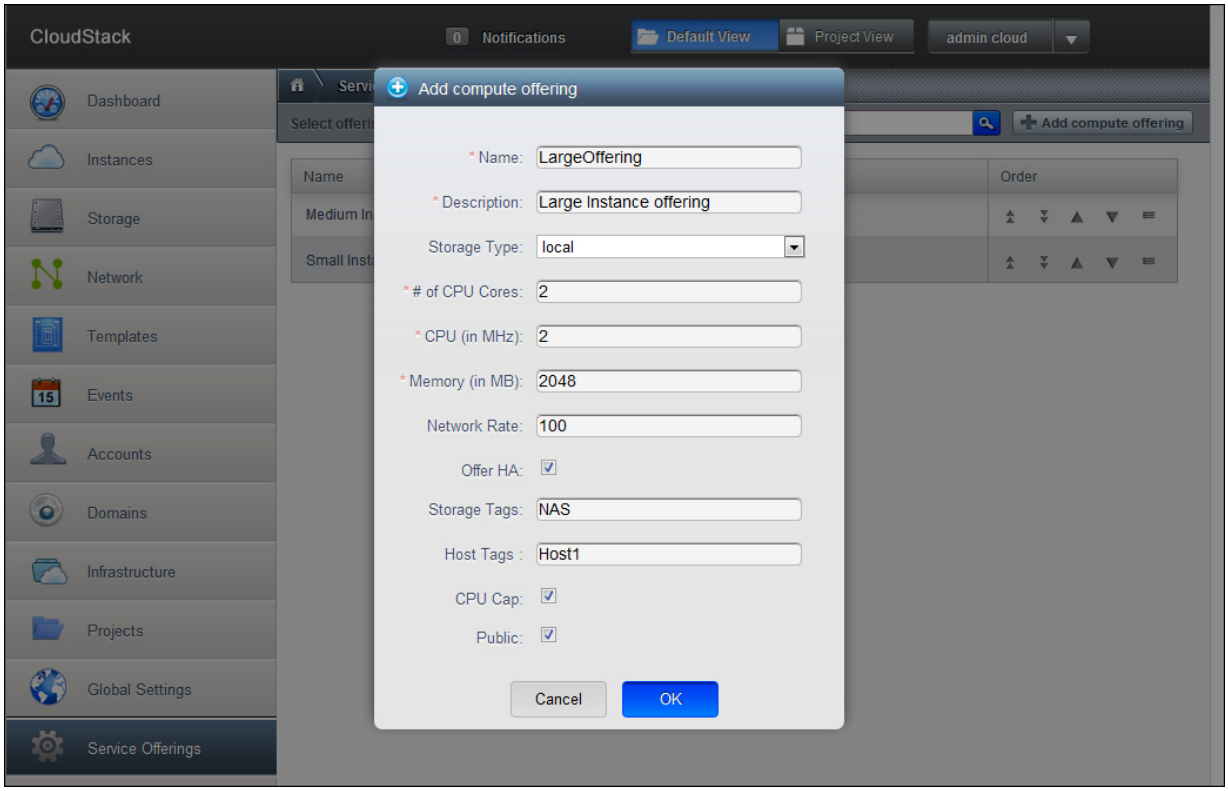
Compute offerings

These offerings are basically details about the compute offerings such as the choice of CPU speed, number of vCPUs, RAM size, root

device tags, storage type and storage tags, network rate, and other compute choices. To add a new compute offering, we need to move to the **Service offerings** tab and select compute offering from the drop-down menu at the top, as given in the following screenshot:



To add a new compute offering, click on the **Add Compute Offering** tab, and provide the appropriate entries as shown in the following screenshot:



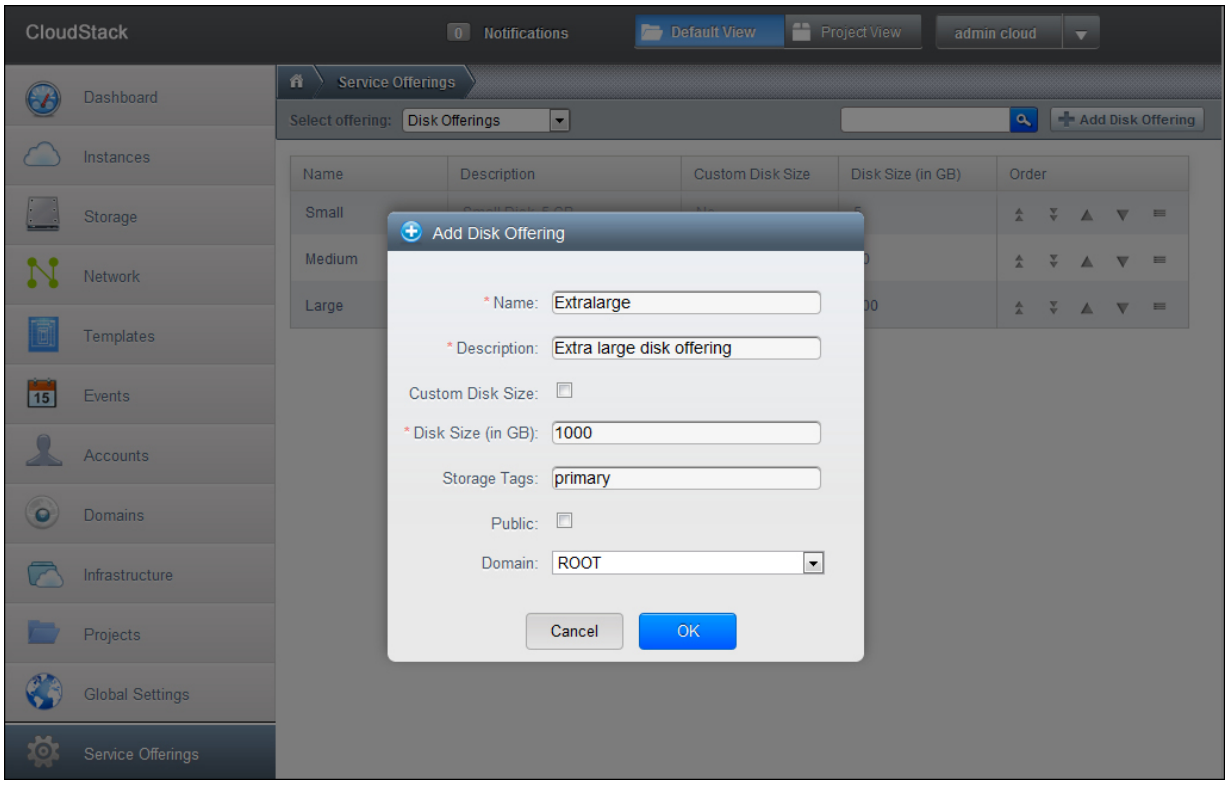
The various fields in the screen are described below:

- **Name**: This will be the name of the Compute offering we are going to create.
- **Description**: Description of the compute offering we are creating.
- **Storage type**: This field describes the type of storage that will be provided in this offering. It can be local storage or shared storage. Local storage is the storage from the host on which the hypervisor is connected and the shared storage is the storage that is accessible by NFS, network attached storage.
- **# of CPU cores**: The number of CPU cores that will be available in with this offering.
- **CPU (in MHz)**: This is the speed of the CPU clock which will be offered to the instance in this offering.
- **Memory**: The amount of memory available with this offering in MB.

- **Network rate**: The network data transfer rate in this offering in MBPS.
- **Offer HA**: This is an option provided to the user, whether he wants to monitor the instance and make it as highly available as possible.
- **Storage tags**: These are the storage tags that should be associated with the primary storage for providing disk storage to the instances.
- **CPU cap**: This option is to limit the amount of CPU available to the user to the assigned value, even if there is spare capacity available.
- **Public**: This option is provided for the visibility of the service offering. Public means that this offering will be available to all the domains in the cloud. If you don't want this offering to be public, CloudStack will let you configure the scope of the visibility of the offering. We can define the scope of the visibility of the offering to a subdomain. CloudStack will prompt to select one of the subdomains from the drop-down list.

Disk offerings

Using Disk offering, we can specify the choice of disk size that the users can request from the Primary storage. For creating a new Disk offering, you need to select **Disk offerings** from the drop-down menu on the **Service Offering** page and click on the **Add Disk Offering** button in the upper-right corner of the screen. You will get a screen like this:



You need to enter all the details in the form as per the offering that you want to create. The fields are described as follows:

- **Name and Description**: The name and description of the Disk offering that you are creating. This will appear in the service request page when the user requests services.
- **Custom Disk Size**: This option allows the user to specify the volume size that he wants for the disk. If this is not checked the administrator will provide the size of the volume and the user will get that much amount of the volume. If it is not checked, you will have to enter the size in GB for the offering.
- **Storage tags**: This option provides the tag that is to be associated with the primary storage for this offering. CloudStack uses this tag to match with the tag of the primary storage volume. The disk is offered to the user from the primary storage whose tag matches the tag provided here. If CloudStack finds that there is no primary storage available with a tag name, the provisioning fails.

- **Public**: This field specifies the permissions defined for the disk offering. This value determines whether the offering will be available to all the domains in the cloud or only one particular domain.

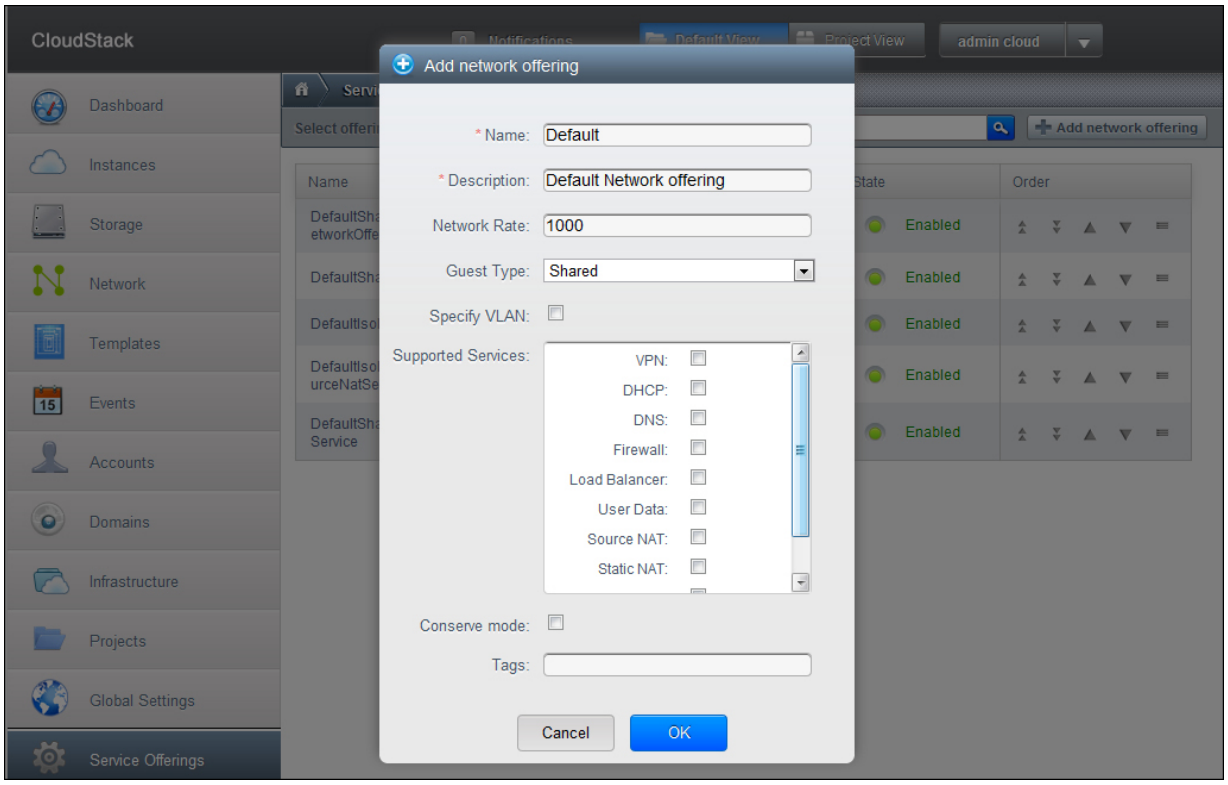
Network offerings

The network offerings are the set of features that will be available to the users as offerings from the virtual router or any other external network devices on the guest network he is on.

There are various network offerings that you can select from the list, they are as follows:

VPN, DHPC, DNS, Firewall, Load Balancer, User Data, Source NAT, Static NAT, Port Forwarding, and Security Groups.

For adding a new Network Offering, you need to select the Network Offering from the drop-down list on the Service offering page and click on **Add new Network Offering** button on the right top. You will get a page as shown in the following screenshot, where in you will have the option to select for the network offerings.



The options that are provided while configuring the network offering are:

- **Name and description**: The name and the description of the Network offering that we are creating.
- **Network Rate**: This is the maximum network data transfer rate that is allowed in the offering.
- **Guest Type**: The network offering can be Share or Isolated offering. These have different characteristics as described in Chapter 2, *Installing Apache CloudStack*.
- **Supported Services**: These are the services that can be provided with this offering such as the Load Balancer, VPN, DHCP, and DNS. The options available with different services will be as per applicability of the configurations. As an example for a load balancer, you can select CloudStack Virtual router or any other load balancer which is configured in cloud.
- **Conserve Mode**: This defines whether the conserve mode is to be switched on or off, when this mode is enabled, the

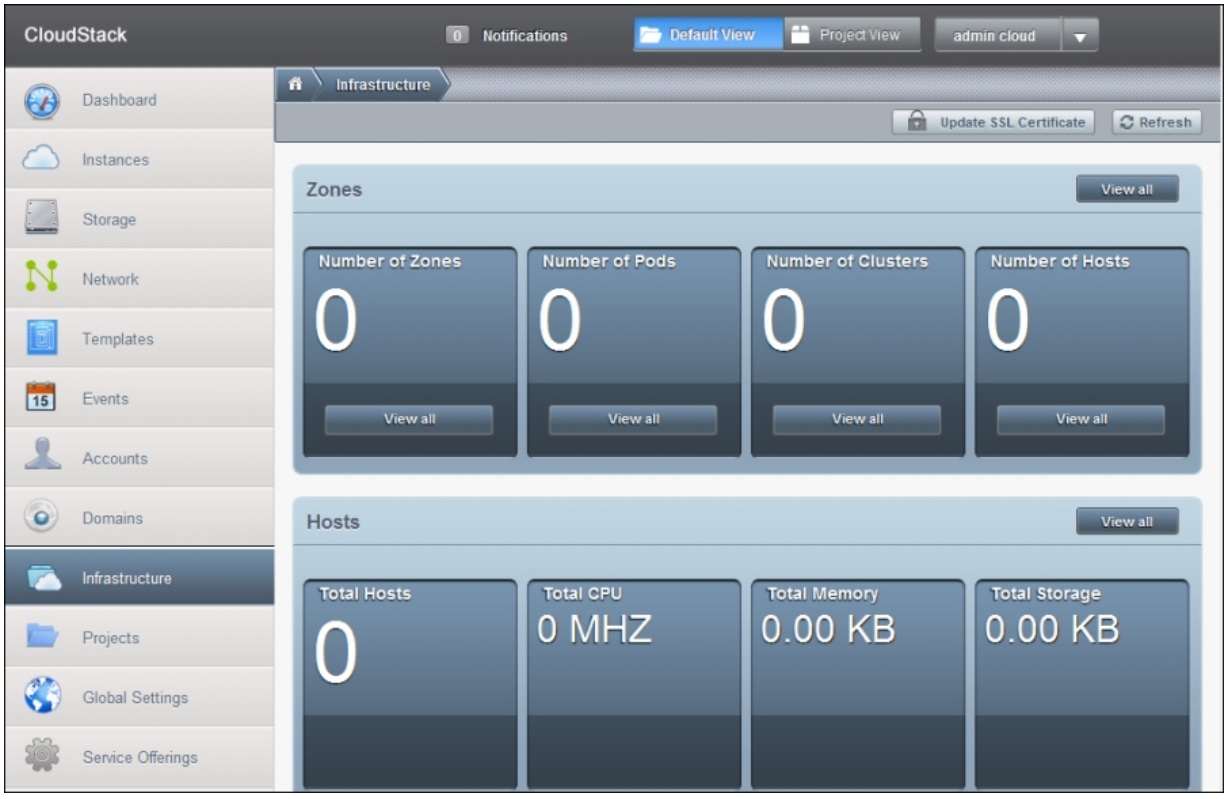
network offering will be provided only when the first virtual machine is started in the network.
- **Tags**: This tag defines the physical network to be used with this network offering.

On these pages, after the creation of various offerings, you can also edit the offerings such as disabling them, editing some properties.

Now that we have got our hold on creating different types of offerings, let's move to adding some infrastructure to the cloud. After which we will be able to create service offerings which will be provided to the end users to request.

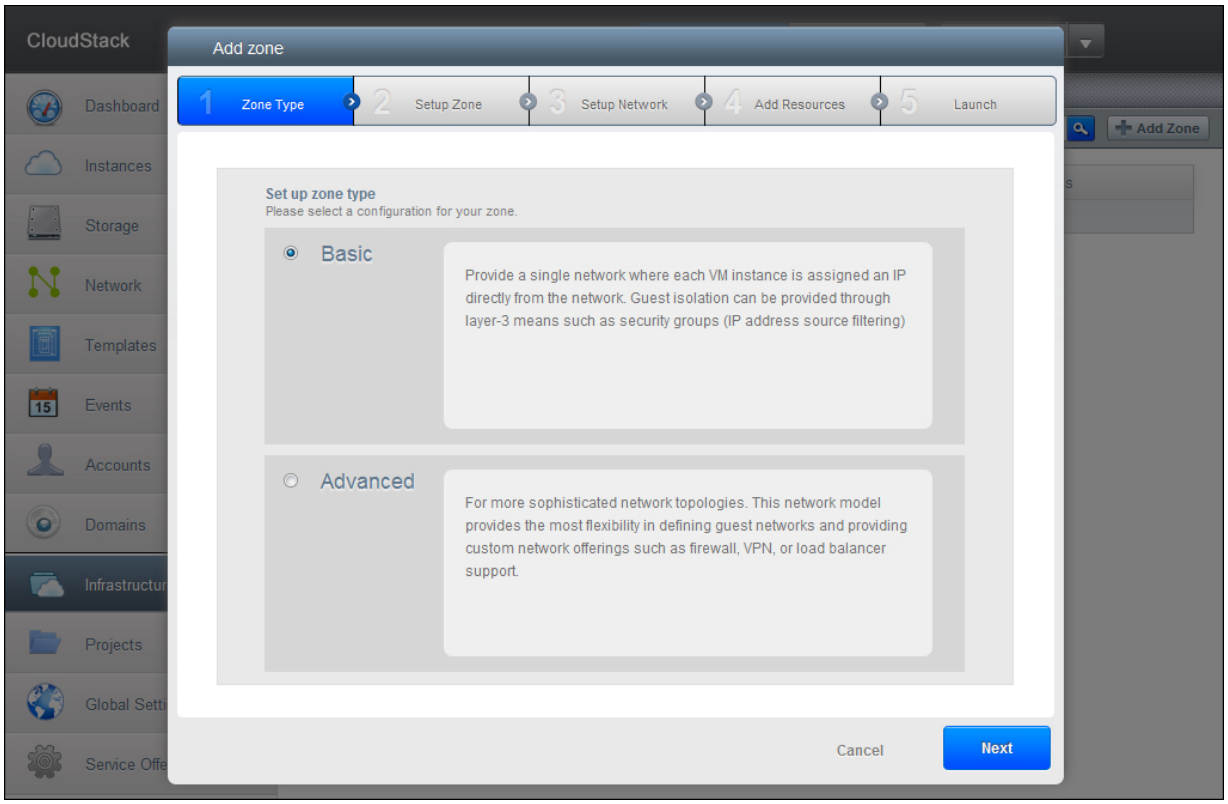
Infrastructure

When we navigate to the Infrastructure page, it displays the overall capacity of Infrastructure that has been added to the cloud. The screenshot of the page is given as follows:



All the information about the number of Zones, pods, Clusters, and Hosts is provided on the screen. You can click on any button of **View All** to get the details about each component of the infrastructure.

We will start by adding a **New Zone** to our environment. For this we will have to move to the zone page and click on the **Add New Zone** button on the top right. The screen provides us with options to configure the zone we are about to add.



There are basically two types of configuration given as follows:

- **Basic**: This type of zone is used for creating a zone with a single network in it. The virtual machines in this type of zone will be assigned IP address directly from that network. In this type of zone, guest isolation is provided using Layer-3 using security groups, that provides IP filtering based on the source IP address.
- **Advanced**: This type of zone is recommended to be created for more sophisticated network topologies where the

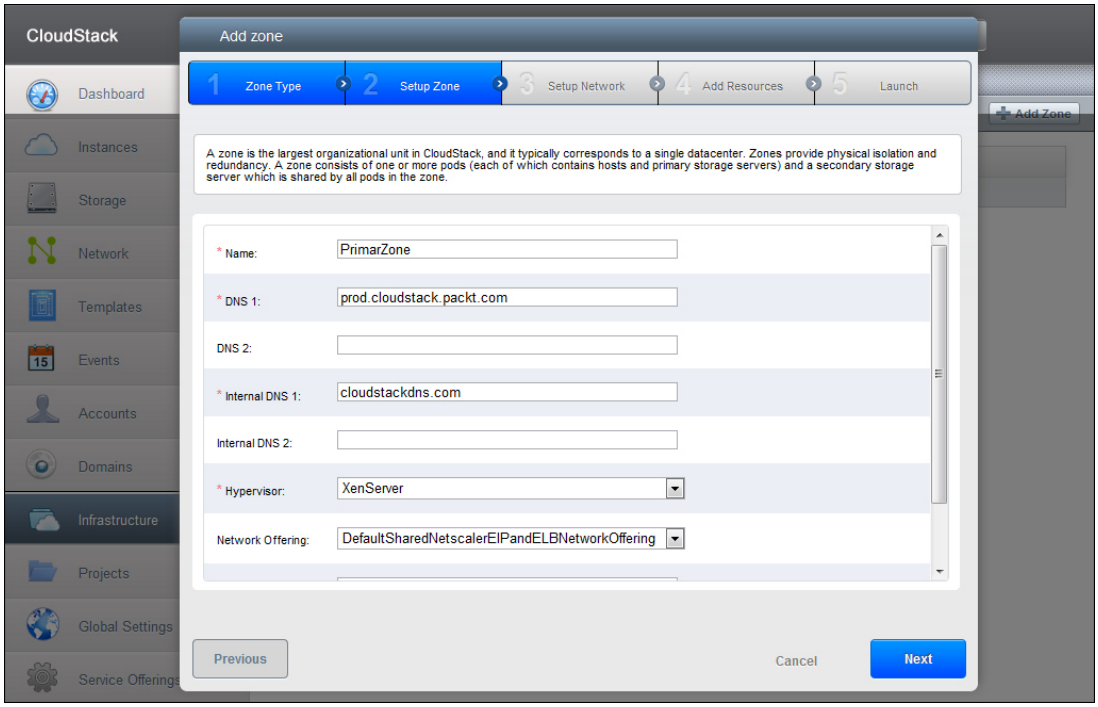
administrator can define guest networks and also provide custom network offerings such as firewalls, VPNs, or load balancer support.

Basic Zone configuration

We will first provide an overview of adding a basic type of Zone setup.

On selecting the Basic Zone type, move on to the next screen and provide details about the zone that we are creating. The configuration will involve the setting up and configuration of the zone properties such as:

- Zone type
- Network configuration of the zone—Public Network and traffic, configuration of one Pod, Guest traffic, and storage traffic
- Addition of resources to the zone—one cluster, one host, one primary, and one secondary storage



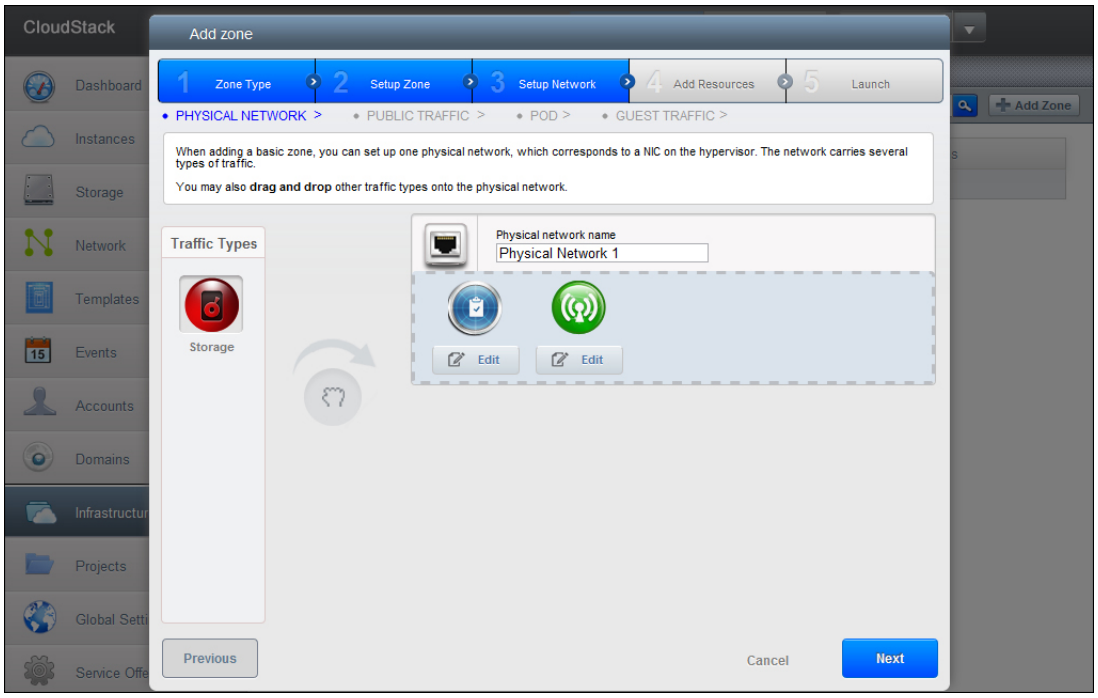
The parameters provided for adding a new basic zone are as follows:

- **Name**: This is the name of the new Zone that we are creating.
- **DNS1**: This is the primary DNS server that will be available in the Zone.
- **DNS2**: This is an optional field. This is the second DNS server that will be used when the primary DNS server is not available.
- These DNS servers are used by guest virtual machines in the zone and should be accessible by the public network that will be added to the zone at a later stage. The zone will also have some public addresses; the DNS servers must be accessible from these public addresses.
- **Internal DNS1**: This is the address of the primary internal DNS server.
- **Internal DNS2**: This is the address of the secondary internal DNS server, which will be used by the cloud only when the primary internal DNS server is down.
- These internal DNS servers will be used by the system virtual machines in the zone. This DNS server will be accessed by the management traffic of the cloud. These DNS servers must be accessible by the private IP addresses in the pods of the zone that we are adding.
- **Hypervisor**: Here we have the option of choosing the type of hypervisor that will be used in this Zone. We have the option to choose from the list, like XenServer, KVM, VMware, Baremetal, KVM, or OVM. The hypervisor that we select here defines the zone and is unique for the complete zone, i.e. only host with the same hypervisor can be added to this zone. If we need to add hosts with another type of hypervisor, we need to create a different zone for that hypervisor type.
- **Network offering**: In this list are available various network offerings that we have created in the previous section of this chapter. The choice of the network offering will determine the type of the network services that will be available in the Zone

for the guest virtual machines. These network offerings are defined in the network offering tab of the service offering page. The options available here are:

    - **DefaultNetworkofferingWithSGServer**: This network offering has security groups enabled for the guest virtual machines which will provide guest isolation to them. This network offering has other network services such as UserData, DHCP, and DNS.
    - **DefaultSharedNetworkOffering**: This network offering doesn't offer the support for the security group, so if you don't want the security group, choose this offering.
    - **DefaultSharedNetscalerEIPandELBNetworkOffering**: This network offering should be chosen when you have a Citrix NetScaler appliance as a part of the zone. This enables the usage of the Elastic IP address and Elastic Load balancer features provided by that appliance. You can create a basic zone with security groups enabled to offer 1:1 static NAT and load balancing.
- **Network Domain**: The network domain can also be assigned here if you want a special domain name to be assigned to the guest VM. This DNS suffix will appear in the hostname of the guest VMs.
- **Public**: This option specifies the scope of availability of the Zone to the users in the cloud. If it is selected, then all the users across all the domains will be allowed to create guest VMs in this zone. Otherwise, you can limit this zone scope to a particular domain, in this case only users belonging to that domain will be able to create guest VMs in this zone.

  After all the options have been filled out accordingly, we will move to the next screen where we have to configure the physical network corresponding to the physical network interface card of the hypervisor.



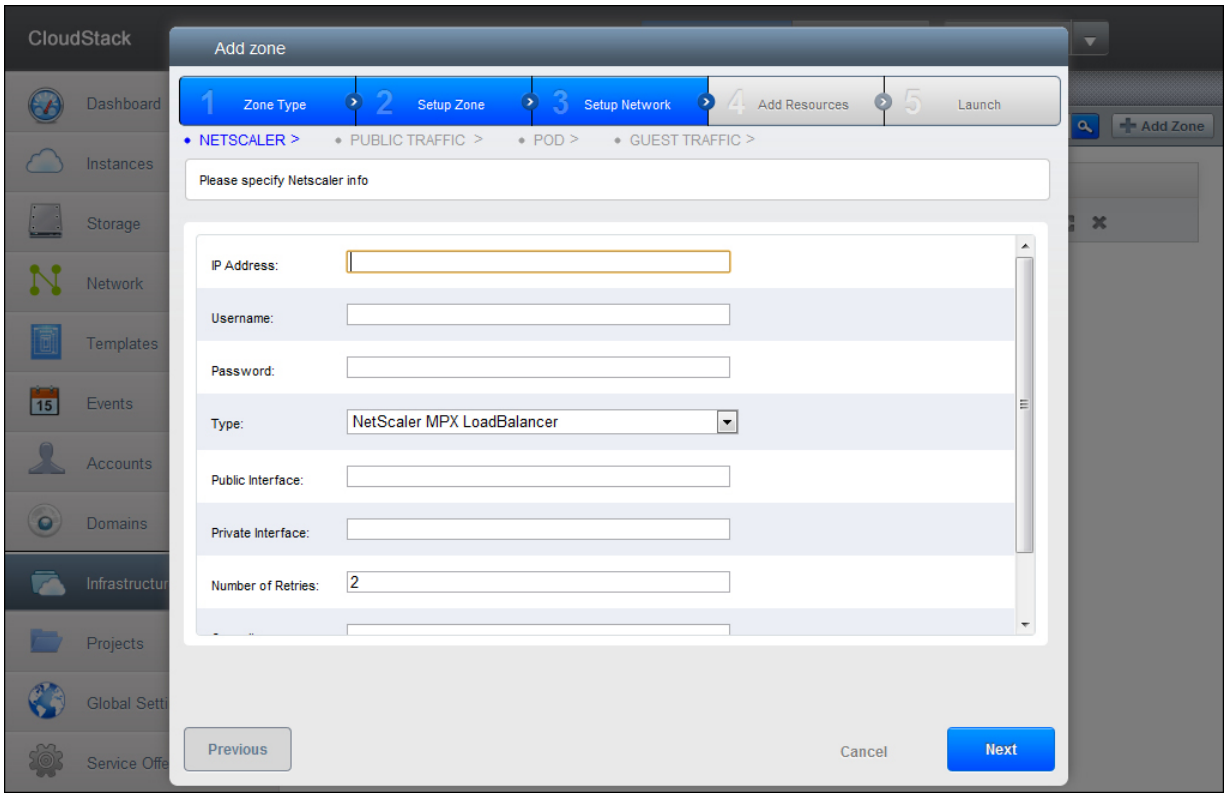
You will have to set up one physical network which will carry multiple traffics. We can add the network traffic that will flow across this network. The network traffic can be any combination of the three types of network:

- **Management Traffic**: This is the traffic between the CloudStack's internal resources. All the communication from any resources that communicates to the Management Server such as Hosts and System VMs is categorized as Management traffic.
- **Traffic between CloudStack's guest VMs**: The communication of guest VMs in the CloudStack is categorized as this kind of traffic.
- **Storage traffic**: All the traffic between the secondary storage VM and the secondary storage such as VM templates and snapshots is categorized as the storage traffic.

We can edit the labels to be used for these individual traffics by clicking on the **Edit** button. These labels are text strings that the CloudStack matches for different kinds of traffic with the traffic label that is already defined on the hypervisor host. These labels are

defined just for the first hypervisor host that will be selected for the first cluster, for any other hosts, labels can be added once the zone has been created.

The users can also select different Network offerings such as `DefaultSharedNetscalerEIPandELBNetworkOffering`, in which case the users will have two extra screens to fill in that consists of the NetScaler device configuration. This network offering has various services supported with it as User Data, DNS, load balancer, and Elastic IP, and these options are for configuration of the network carrying traffic between end users' virtual machine and internet—public traffic. In this case you must enter the configuration details for the NetScaler device, as shown in the following screenshot:

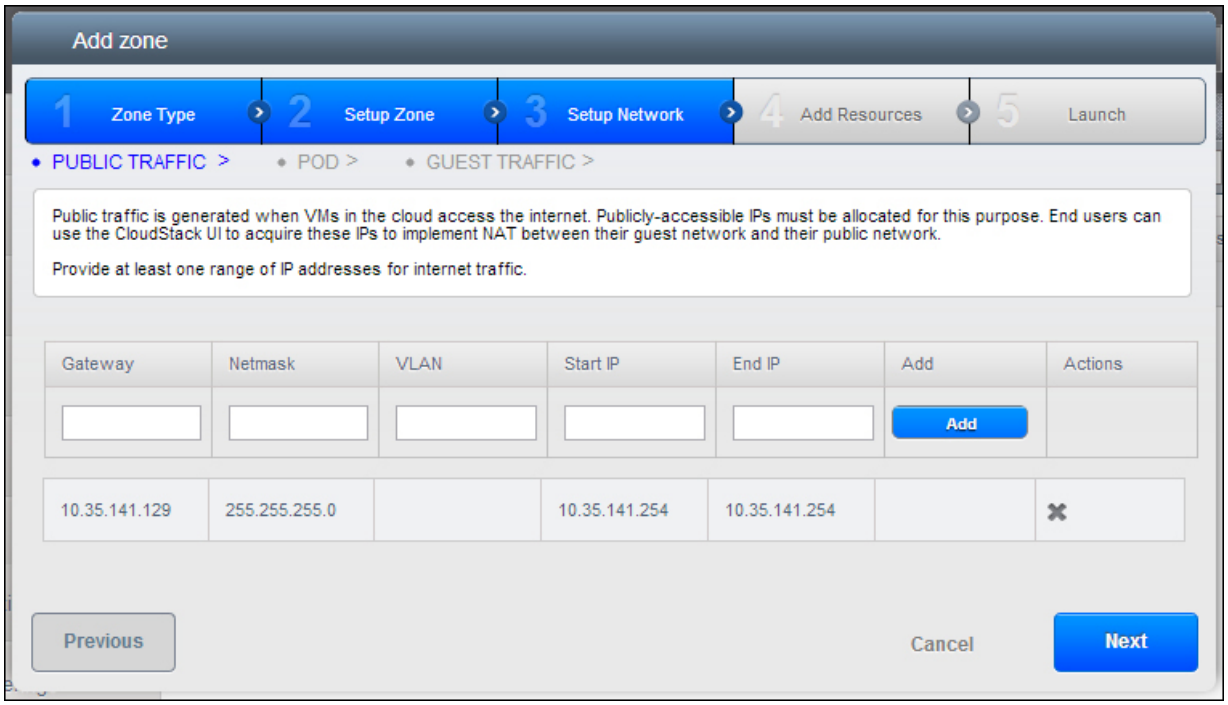


The options displayed while adding a NetScaler device are as follows:

- **IP address**: The IP address of the NetScaler device (NSIP)
- **Username**: Username for the login of NetScaler device

- **Password**: Password corresponding to the username specified above for the Netscaler device
- **Type**: The NetScaler device type which you are adding, It is the list of values such as NetScaler VPX, NetScaler MPX, or NetScaler SDX
- **Public Interface**: The interface of the NetScaler which is a part of the public network
- **Private Interface**: The interface of the NetScaler device which is configured to be a part of the private network
- **Number of retries**: This is the number of times a command should be retried before it is considered to have failed
- **Capacity**: This specifies the number of guest networks/accounts that will share this NetScaler device
- **Dedicated**: This option makes the device dedicated to a single account. If this option is selected, the value in the capacity field is taken to be "1" irrespective of what is specified

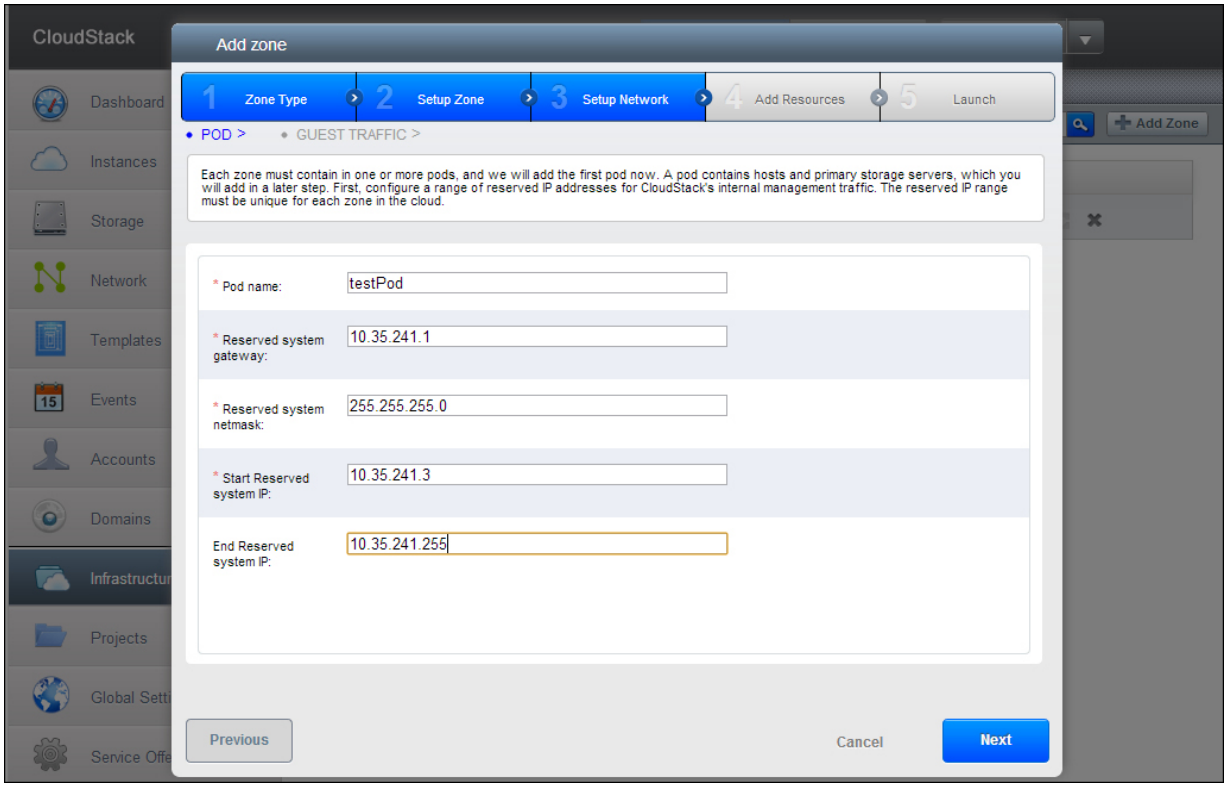
After you have configured the NetScaler device details, you will be asked about the details of the public traffic—the IP range for the public traffic is to be provided. These IP addresses will be used to provide static NAT capability. These capabilities are enabled while selecting the Network offering for NetScaler with EIP and ELB. The screen on the next page looks like this:



The options for setting up the public traffic in the network setup of a basic zone are:

- Gateway: The address of gateway to be used for these IP addresses
- Netmask: The netmask associated with this IP range
- VLAN: The VLAN which is to be used for public traffic
- Start IP/End IP: This is the range of the IP addresses that are Internet facing and are to be allocated for access to the guest VMs so that they can access the Internet

After we have configured the physical network and NetScaler device configuration, the next step is to add pods to this zone. A zone can contain multiple pods. For more details, refer to Chapter 1, *Apache CloudStack Architecture*.

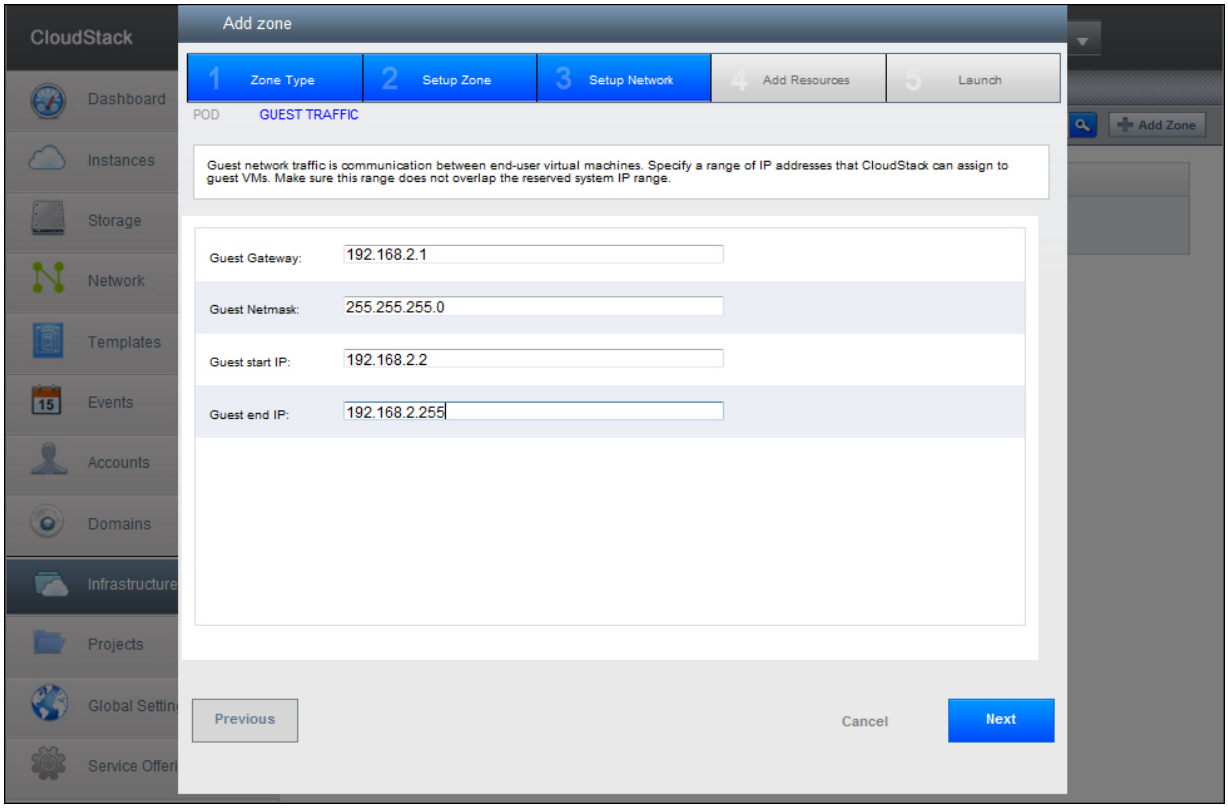


We will have to add the first Pod where we will configure a range of reserved IP addresses for CloudStack's internal management traffic which should be unique for each zone in the cloud. Later, after the creation of zone, we can add more pods.

- Name: This is the name of the Pod which we are adding.
- Reserved System Gateway: This is the gateway for the hosts in this Pod.
- Reserved System Netmask: This is the network prefix which will define this Pod's subnet. We have to define an IP range here using CIDR notation.
- Start/End Reserved System IP: This is the IP range which will be used for the Management traffic. This IP range will be used by CloudStack's Management server for managing various System VMs, secondary storage VMs, Console Proxy VMs, and DHCP over the Management traffic.

After adding the first Pod by providing the required values, we will proceed to configure the guest traffic. Clicking on the **Next** button

brings up the next screen for configuring guest traffic:

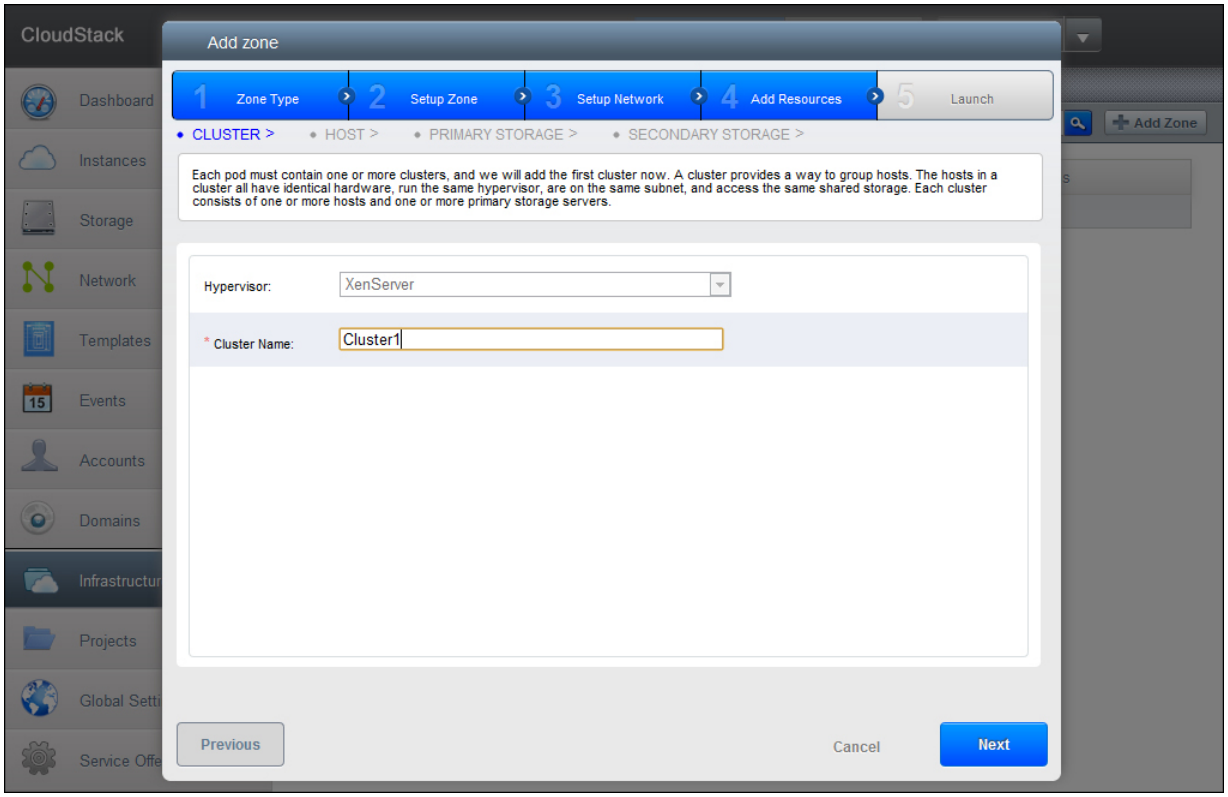


The Storage traffic options are:

- Guest Gateway: This is the gateway that the guests use.
- Guest Netmask: This is the netmask of the subnet which will be used by the guests.
- Guest Start/End IP: This is the first and the last IP addresses of the IP address range which will be assigned to the guests by CloudStack.

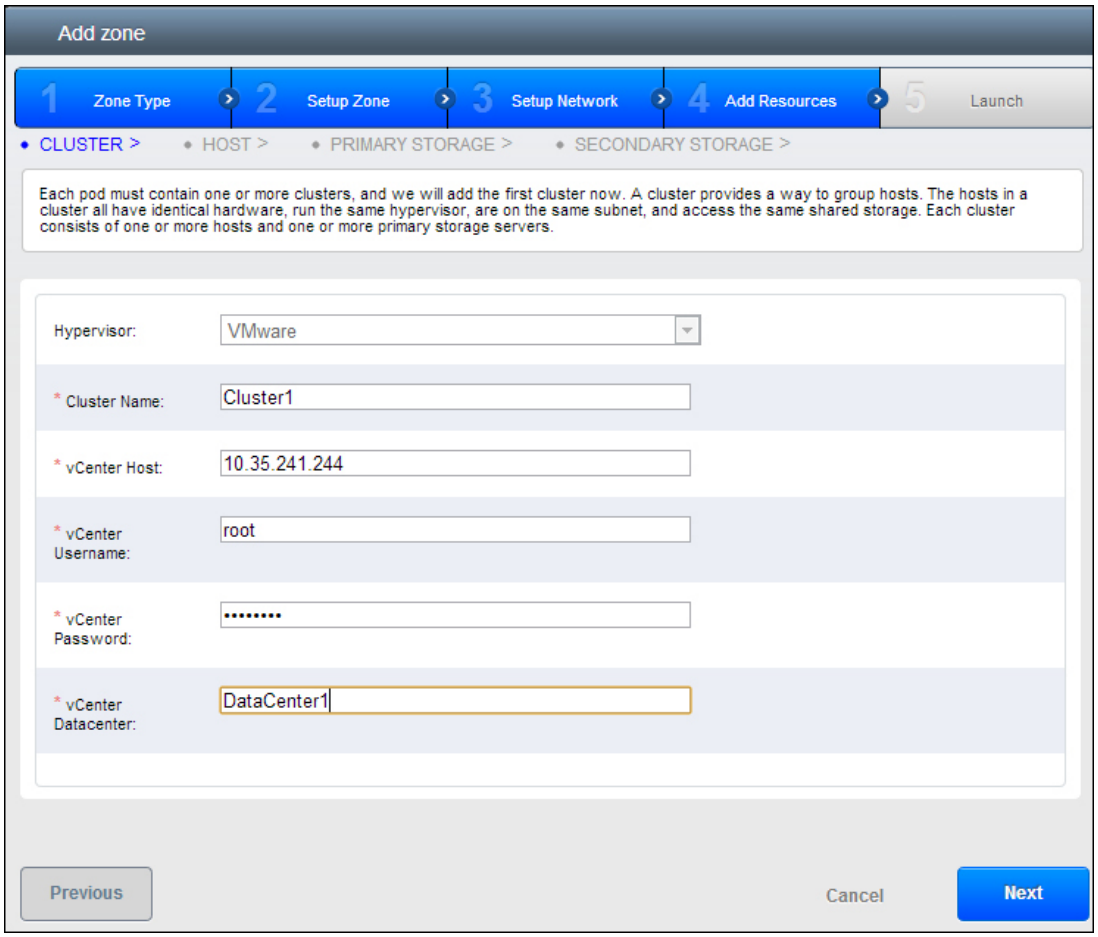
CloudStack recommends using multiple NICs which allow the NICs to be in different subnets. In case you use only one NIC, these IP addresses must be in the CIDR range of the Pod.

After the guest traffic has been set up, we will add the first clusters of the hypervisor type we have defined earlier. On clicking the **Next** button, we get the following screen:



The Cluster details are entered in the options given as follows:

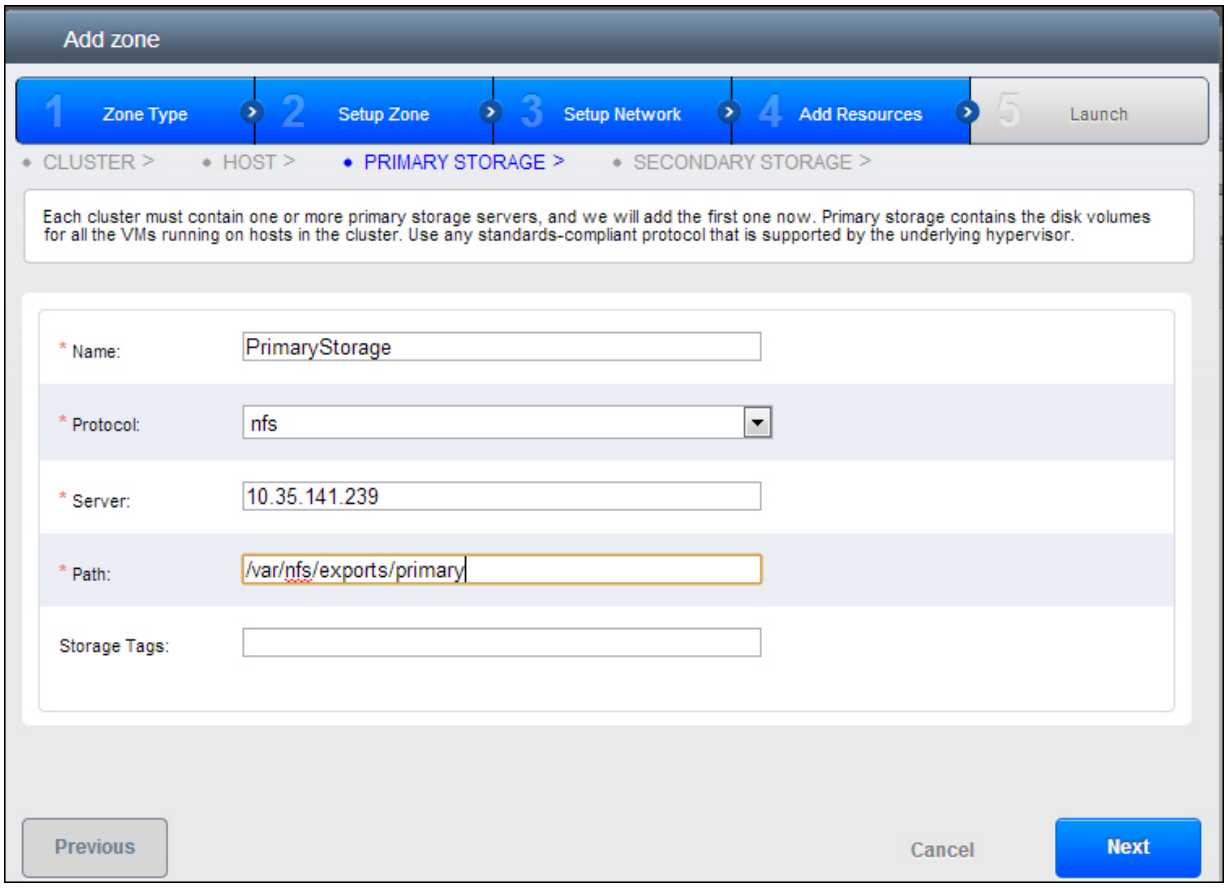
- Hypervisor: This is the type of the hypervisor which we selected earlier. This is the hypervisor which all the hosts in the cluster will use. Different fields may need to be filled in based on the selection of the type of hypervisor. In case we are using VMware as the hypervisor, it is recommended to create the cluster in vCenter and then add it here.
- Name: This is the name of the first cluster that we are configuring here. This is the first cluster that we are adding to the cloud, if you want to add more clusters, you can do that after the creation of the zone. After this first cluster is added, we have to add hosts to the zone. A cluster can have multiple hosts of same hypervisor type which hosts the guest VMs. We will add the first host here and for any more addition of hosts, we can do that after the creation of this zone. The host we are adding must have a hypervisor running on it with an IP address configured and must be connected to the Cloudstack Management server.



The host configuration values are provided in the options given as follows:

- Hostname: This is the DNS name of the host.
- Username: This is the username of the host, usually "root".
- Password: This is the password for the user specified in the field above.
- Host Tags (optional): This can be any tag which will be used to categorize the hosts for the sake of maintenance.

This will add the first host to the Zone. The next screen on clicking **Next** lets us add the primary storage server. Each host has at least one primary storage server.

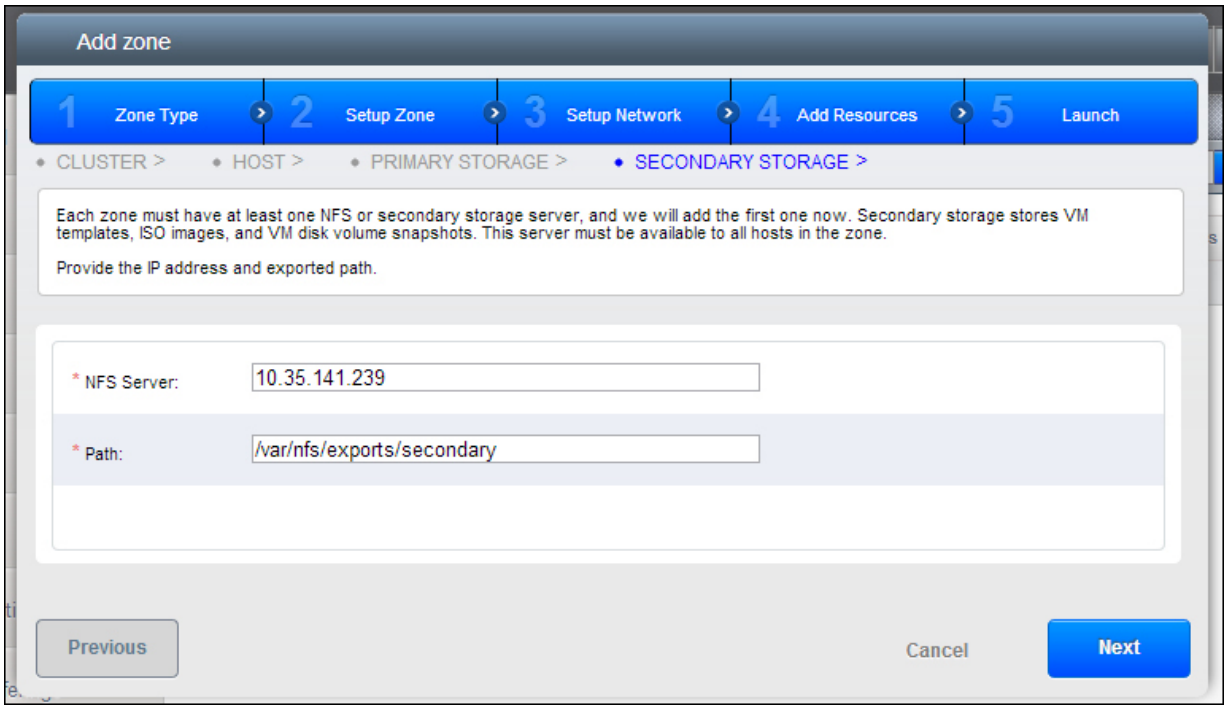


The options for adding primary storage in the zone are:

- Name: This is the name of the first primary storage server that we are adding here.
- Protocol: This field is dependent upon the type of hypervisor we are using. In the case of XenServer, we have the option to choose from NFS, iSCSI, or PreSetup. In case of KVM, we can choose the protocol for the Primary storage server as NFS or SharedMountPoint. For vSphere, we can use either VMFS, which can be iSCSI or FiberCannel or NFS. Depending upon the selection of this field there can be other fields in this screen.

If we select an NFS server, the path is the location of the Primary storage on the server.

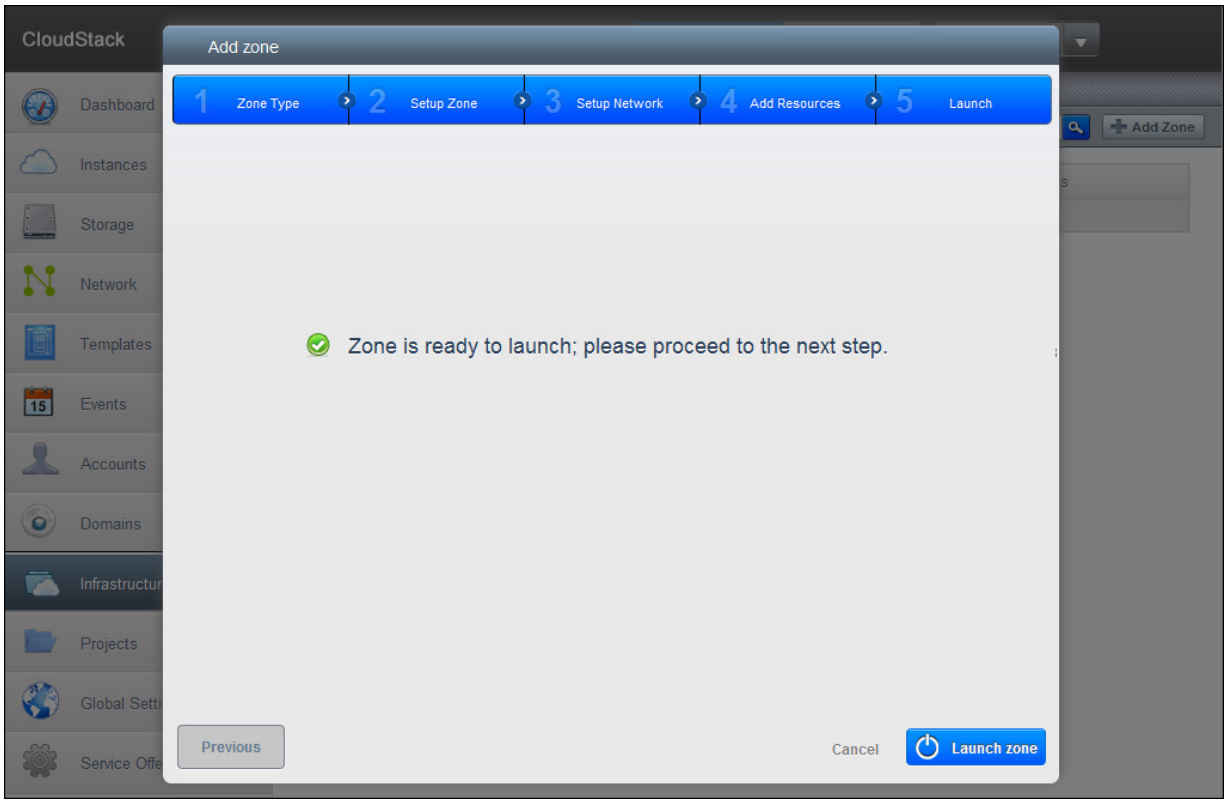
Next we have to add the secondary storage to the zone. The secondary storage can be accessed by all the hosts in all the cluster of the Zones and is used to store templates, ISO images, snapshots, and so on. On moving to the next screen we get:



The options include:

- NFS server: This is the NFS server hosting the secondary storage.
- Path: This is the exported path from the server.

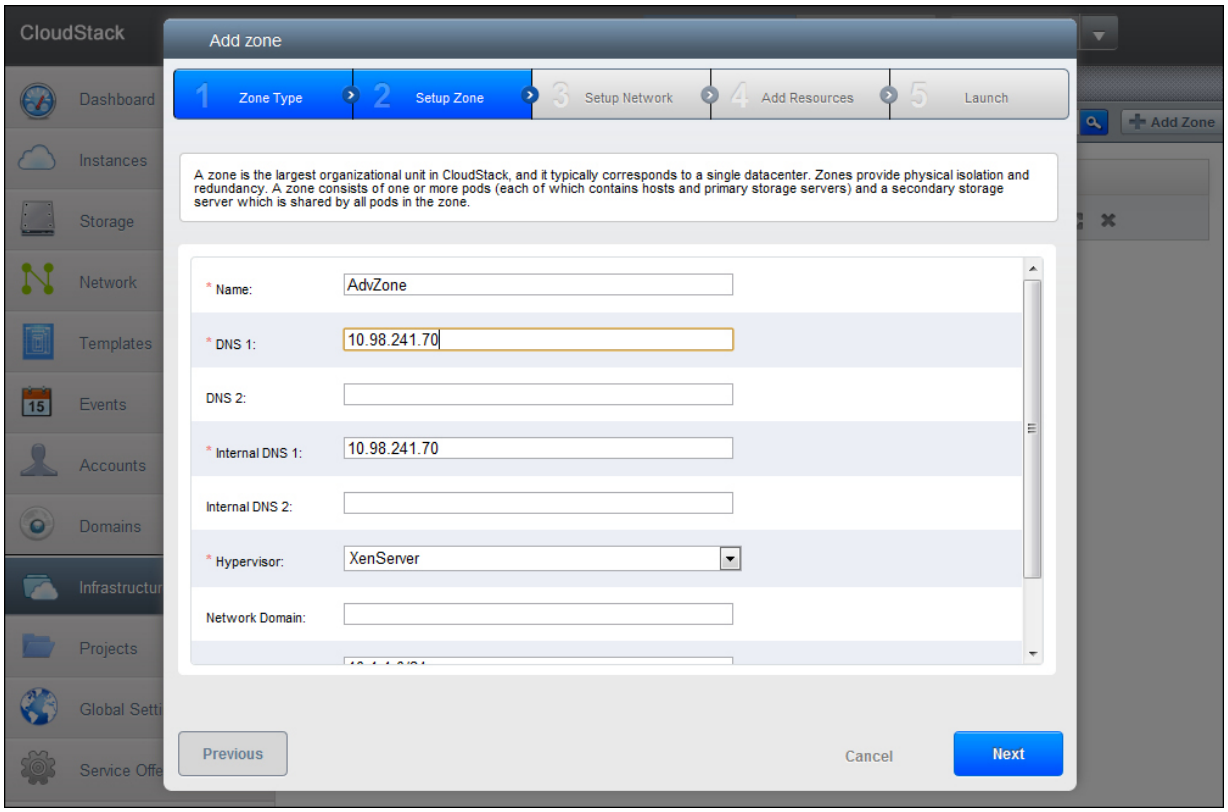
After the addition of secondary storage, we are done with the configuration of the zone, and we can launch it.



This was the configuration of the Basic Zone type. Let's see how it is different from the Advanced Zone configuration and what extra configuration inputs are required in order to create a zone with advanced network topologies.

Advanced Zone configuration

When you select the option of creating a zone of Advanced type in the first screen of the Zone, you will get a screen like the one shown below.



The basic information of the advance zone is provided for the following parameters:

- **Name**: Name of the advanced zone that we are creating.
- **DNS1/DNS2**: This is the address of the primary and secondary DNS server to be used by the guest VMs in the zone. These servers should be accessible by the public network that we add later.
- **Internal DNS1/Internal DNS2**: These are the DNS servers that are used by the system VMs in the zone.
- **Network Domain**: This is an optional field and should be specified if you want a special domain name for the guest VM network.
- **Guest CIDR**: This field is the CIDR IP range that describes the IP addresses to be used by the guest virtual networks in this zone. It is recommended that the guest CIDR IP range should be different for different zones.
- **Hypervisor**: This is the list of hypervisor for the first cluster in this zone. Once the zone is created, you can add multiple

clusters with hosts of different hypervisors. A cluster is a group of homogenous hypervisor hosts, i.e. all the hosts in a cluster must have the same type of hypervisor.

- **Public**: This field specifies the permissions for the zone, whether this zone is to be assigned to all domains or a particular domain. If this zone belongs to one domain, then users who belong to that domain, will only be able access it.

After configuring the initial details, you have the option to add multiple physical networks and configure which traffic can flow across them. In this case, you will have to define labels to distinguish different kinds of network traffic.

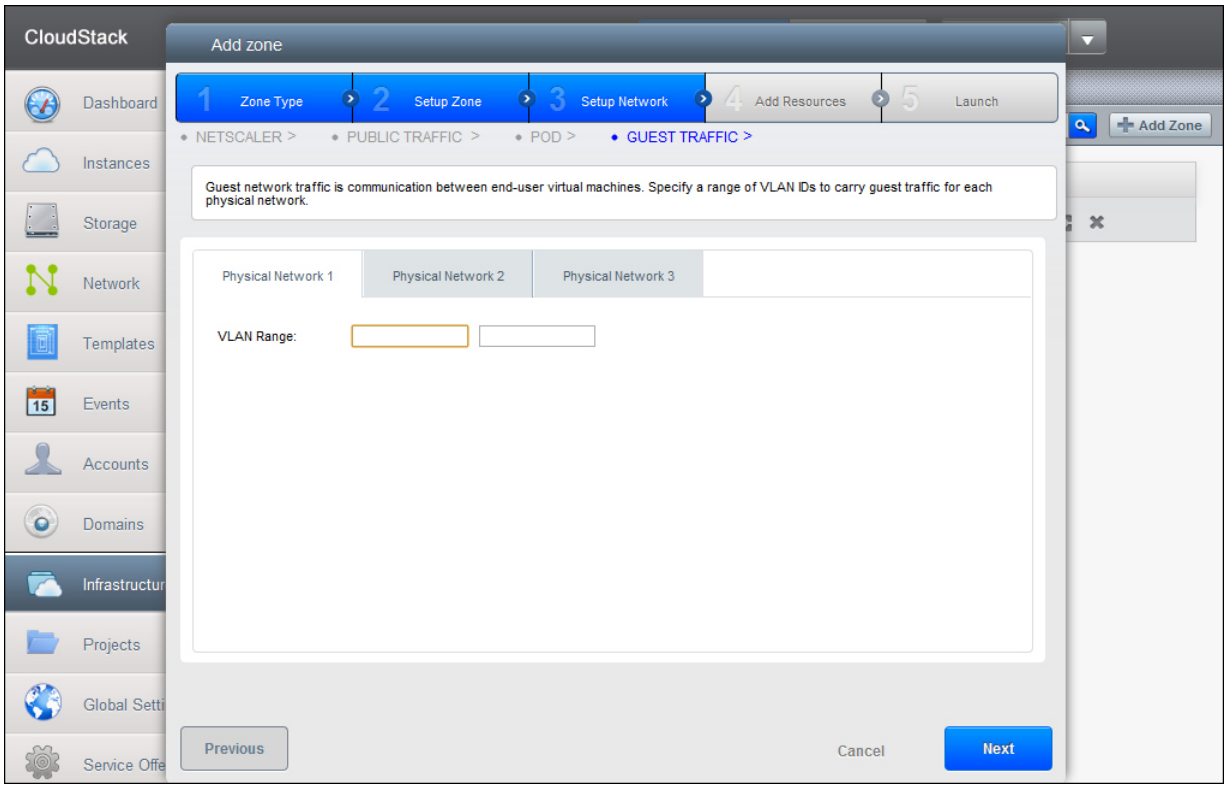


As you can see in the preceding screenshot, there are three traffic types in the first Physical Network and then there can be number of physical networks which allow traffic between the guest VMs. Each of these traffic types must have a label associated with them in order to distinguish between the different types of network traffic.

These labels must match the labels that you have defined on the hypervisor host. Only one physical network can host the storage traffic, internet traffic and Management traffic. This configuration is done only for the first hypervisor host that is added to the zone, for any other host it can be done after the zone has been created.

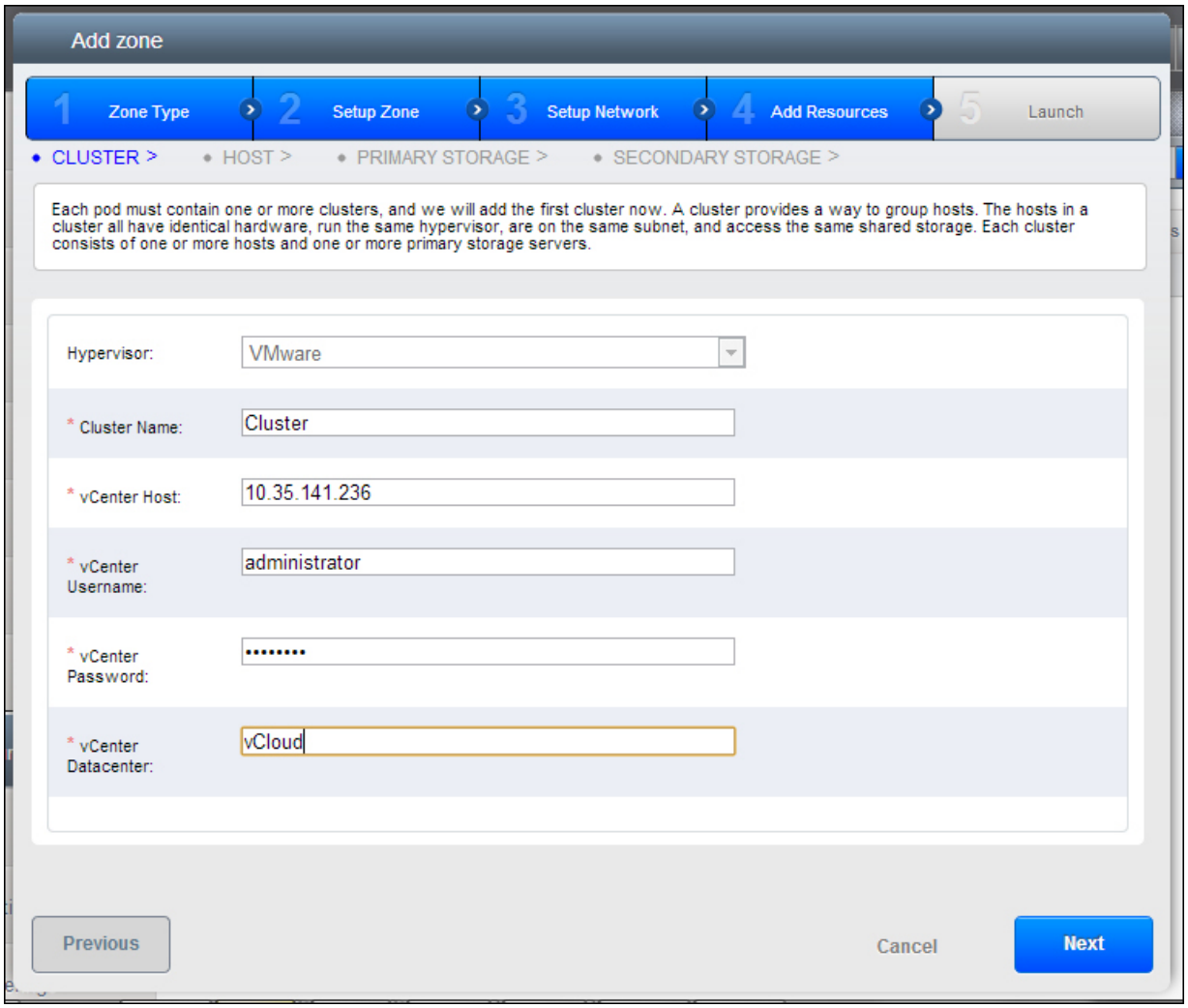
After you have assigned various traffic types to the different physical networks, you will have to configure the NetScaler device in the zone as per the Screen 15 of the Basic Zone configuration in the previous section.

You will have to follow the steps from screen 15 to screen 17 to add the Pod in the CloudStack environment as defined in the Basic Zone configuration. After adding the Pod to the environment, you will be asked for configuring the VLAN range for the various physical network that you have defined for carrying the guest traffics for each physical network.



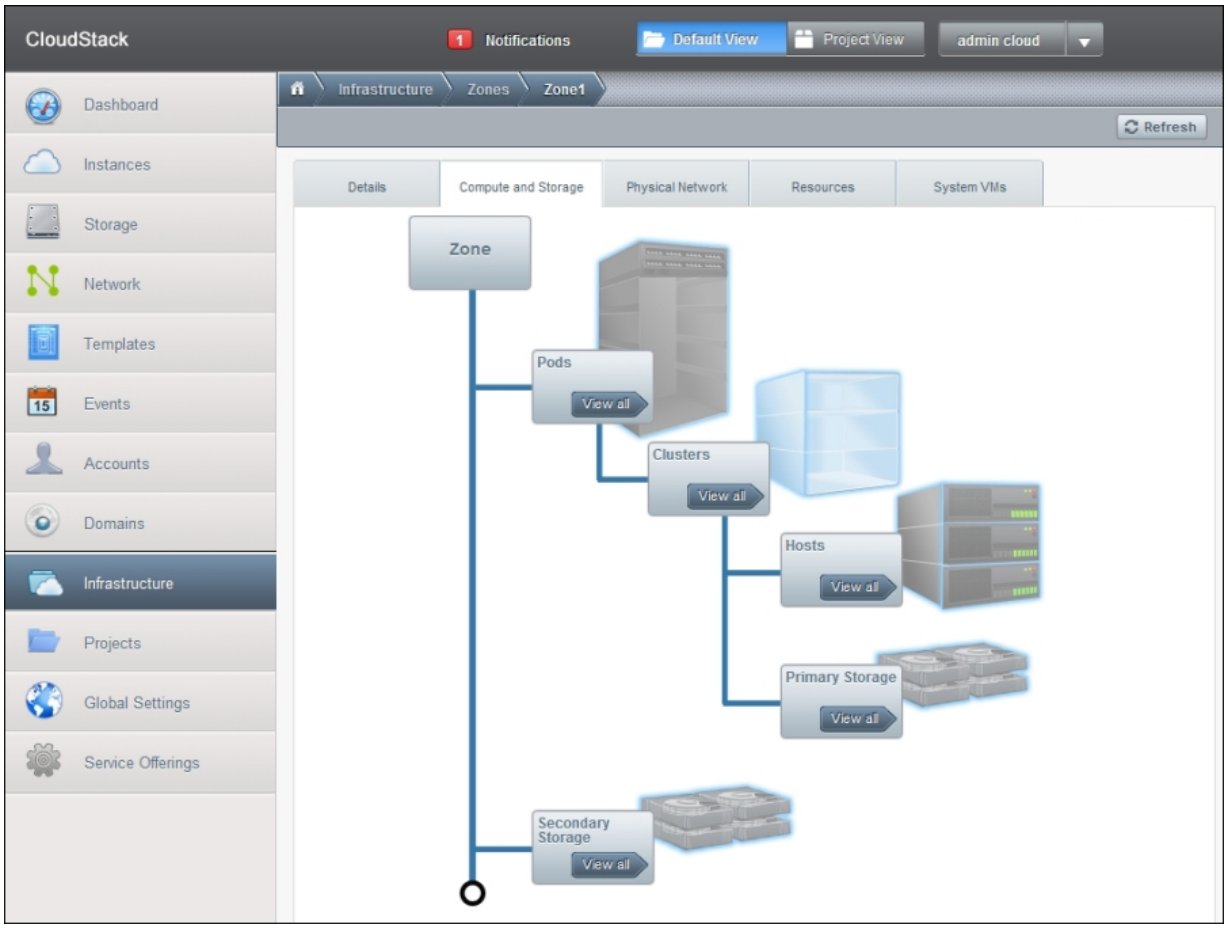
The screen for defining the VLAN range for the various physical networks in this zone is given above. After defining various VLAN ranges for different kinds of traffic, you can proceed to the configuration of pods, Clusters and Hosts in the zone same as described for the Basic Zone.

There can be various other configuration parameters depending upon any other network offerings than those defined in this chapter or it can depend upon the type of hypervisors such as if you select VMWare, you will be asked to enter the details of the VMware VCenter server while defining the details of the cluster (A cluster consists of homogenous hosts—the hosts must have the same type of Hypervisor), as depicted in the following screenshot:

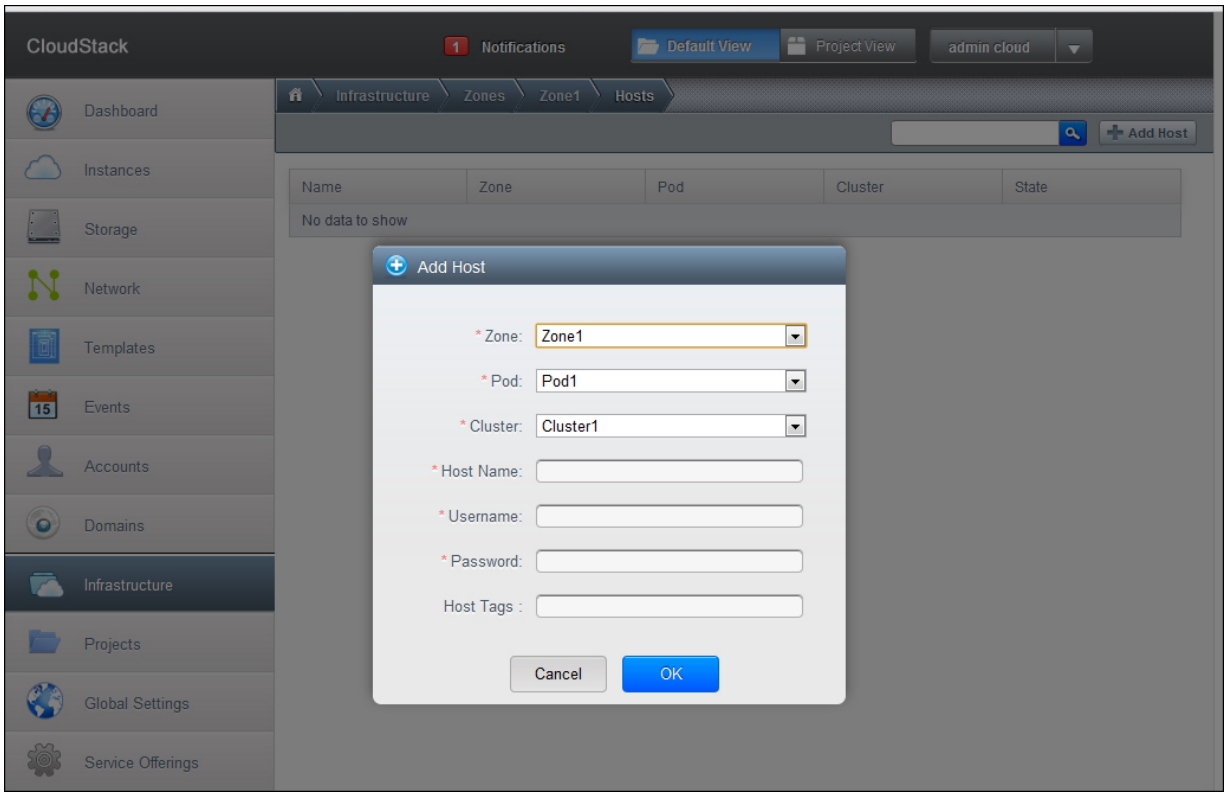


For VMware, it is recommended that you have the cluster defined at the VCenter server before adding it to the CloudStack environment.

After the successful creation of the zone, you can view the details of the same on the zone page and also edit some of the options. The page displaying the details about the zone will look somewhat like this:



You can also add more infrastructure resources to the zone or create another zone after this zone is created. For example, lets say you want to add another host to the zone, it can be done by moving to the host page and clicking on **Add Host** which will provide you with a screen to enter the details about the new host.



The options for adding another host in the zone are:

- **Zone**: This is the zone to which we are adding our new host to.
- **Pod**: The pod in the zone selected above in which the host should lie.
- **Cluster**: The cluster within the pod which the host is a part of.
- **Hostname**: The IP address or hostname of the host which is to be added.
- **Username**: The username of the host.
- **Password**: The password of the host for the username provided above.
- **Host Tags**: This can be any tag to distinguish the host for ease of maintenance.

Similarly we can add more clusters or pods to the existing zone by navigating to the desired resource page. We have added the infrastructure to the cloud; we can proceed to adding other resources such as templates and storage.

Creating a template

A template can be created by a number of ways. We are going to discuss them below.

Creating a template using a running VM

We can create a template from a running virtual machine in our environment by following these steps:

1. Create a VM instance with the Operating system that you want and make the configuration changes as per your need.
2. Stop the running virtual machine.
3. Now, the volume of this virtual machine is also in the stopped state, convert this volume into a template.
4. Or, create a snapshot of this volume and then create a template from that snapshot.

Importing a VM to CloudStack

The other way to create a template is to import a virtual hard disk from any other environment into the CloudStack environment.

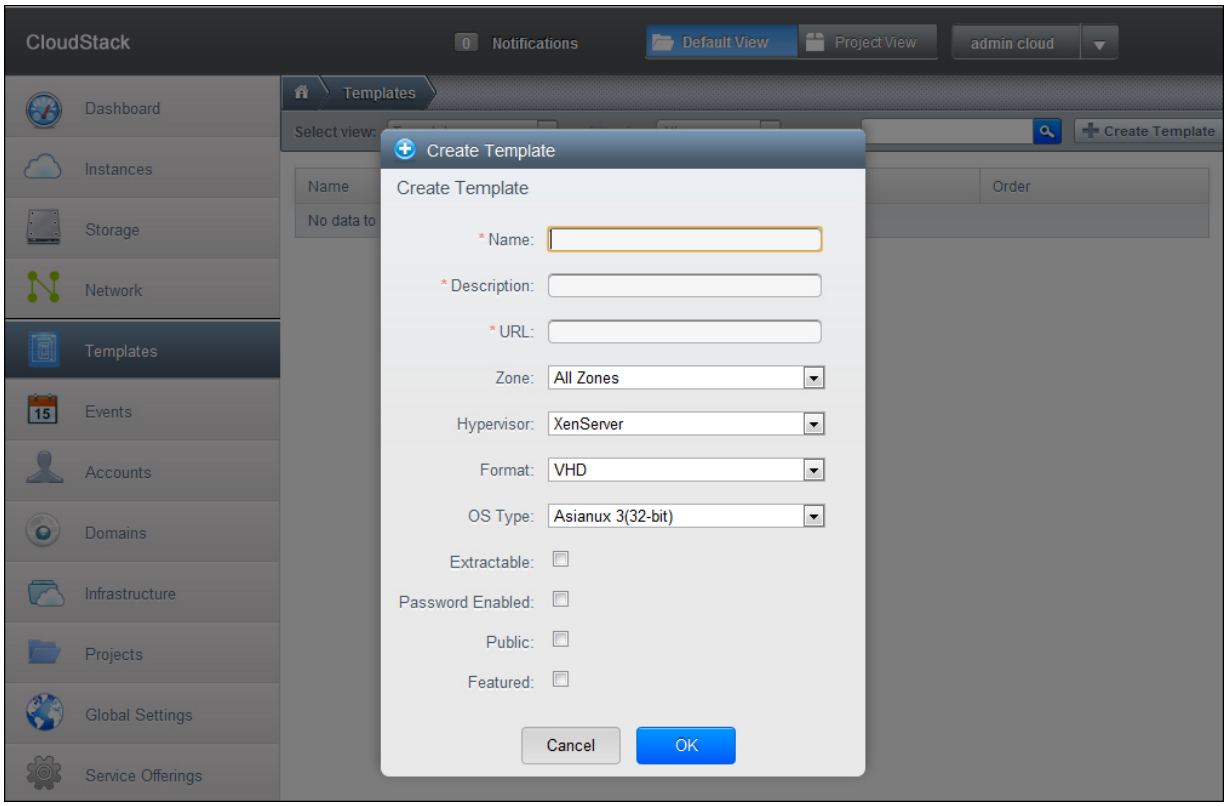
There are some basic requirements while creating a template from any other environment which enable some functionalities such as the console view and guest shutdown. The requirements are as follows:

- If you are planning to import a XenServer virtual machine, install the PV drivers or Xen tools on each of the machine which you are going to import into CloudStack and convert into a template. The PV drivers or the Xen tools help manage the virtual machine and also provide performance data for them.
- If the virtual machine is part of a VMware environment, install the VMware tools.

We will discuss the steps for importing a template of different OS and different environment later in this book.

Creating a template using the Management console

We can as well create templates using the Management console by navigating to the **Templates** tab of the left menu and clicking on the **Create Template** button. The screen looks as follows:



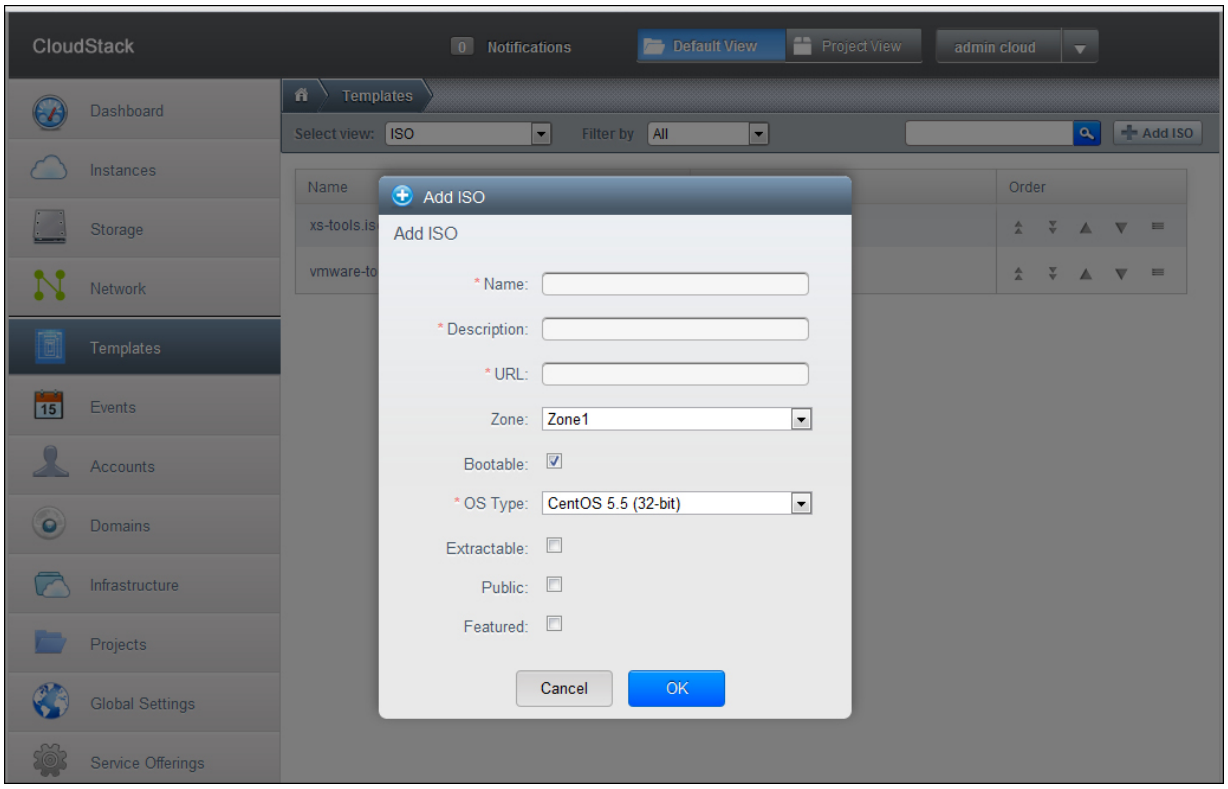
The options while creating a new template are:

- Name and Description: This contains the name and description of the template that we are creating. This will be visible to the end users while creating VMs from the template.
- URL: This asks for the URL of the file (which can be a VHD or a compressed file) which is going to be used to create template and then guest virtual machine from. This file is uploaded to the CloudStack environment. HTTP and HTTPs are supported protocols for this.
- Zone: This is a drop-down list of zones in the environment, you can either create template to be used in all the zones or for some specific zone.

- Hypervisor: Select the hypervisor to which the file belongs.
- Format: This is the format of the template file that we are providing URL for. The format of the template depends upon the type of hypervisor we chose for it.
- OS Type: The operating system of the template we are creating. This will help CloudStack to perform a set of operations and will also help in improving the performance of the guest VM created from this template. In case the OS type is not present, you can choose **Other**.
- Extractable: Specified whether the template is extractable or not. If it is extractable, the full image can be available for download by the end users.
- Password Enabled: Check this if the template supports the password reset feature.
- Public: Specifies whether the template is to be made public or not.
- Featured: It should be checked only if you want to make this template a featured template more prominent for users to select from the list.

By providing the valid values, the template will be created and you can create new guest VMs based on this template.

After adding a template from the console, you can also add the ISO to CloudStack. CloudStack also supports adding ISO images to the guest VMs by uploading them to the CloudStack environment. To add a new ISO, select the ISO from the drop-down menu on the **Templates** page and click on the **Add ISO** button. This will open a pop-up box to allow you to fill in the details shown as follows:



The options available while adding an ISO are as follows:

- Name and Description: The name and description of the ISO that we are adding.
- URL: The ISO that we add in CloudStack is uploaded, so they must be available on a URL. HTTP/HTTPs is the supported protocol for this.
- Zone: Choose the zones to which the ISO must be available.
- Bootable: This property specifies whether the ISO is bootable or not. This property specifies whether the guest VMs can boot off this ISO image or not.
- OS Type: The type of the OS that the ISO has. You can choose other if the OS is not listed in the list. This helps CloudStack to perform certain operations as well as improves the performance of the guests.
- Extractable: This property specifies whether or not the ISO can be extracted. This means that the ISO can also be available for extraction.
- Public: Specifies whether to make the ISO public or not.

- Featured: This option must be selected if you want this ISO to be more prominent, so that users select it from the list.

These templates and the ISOs that we add in the CloudStack environment can be used to create instances. The end users, while requesting a new instance, specify the template or the ISO which is to be used to create the instance in the zone.

We are done creating projects, domains, accounts, and users in our CloudStack environment and as we have added infrastructure to our cloud. Now, if the end users request an instance based on some template or ISO, they will specify the zone in which the instance should be created along with the service offerings and networks. These options are listed according to our configuration in the cloud. For example, the compute offerings that you have added will be listed for the user to select from.

The administrator can also perform all the operations that the user is entitled to. The admin or the user can request storage volumes based on the disk offering that we have added to our CloudStack environment, these volumes are provided to the users from the primary storage of the hosts that we add in our infrastructure. These volumes can be attached to the guest anytime.

Preparing the hosts

When we add a host to our infrastructure, CloudStack installs an agent on the hosts, so that it can communicate with and control the environment. We can install an agent on the host using the following command line:

- For Rhel/Centos:

```
#sudo yum install cloud-agent
```

- For Ubuntu:

```
#sudo apt-get install cloud-agent
```

The installation of the agent on the host is the part of the host preparation tasks that we carry out so that the hosts can be added to our environment and the guest VMs can be provisioned on these hosts.

The hosts added to the Cloudstack environment provide the following functionalities:

- They provide compute—CPU, Memory, storage and networking resources to the guest VMs hosted on them.
- These hosts also provide interconnectivity to the Internet using high bandwidth TCP/IP network.
- These hosts can be residing across multiple data centers but are grouped under a logical entity called **cluster**. A cluster is a group of homogeneous hosts. The hosts in a cluster must have same hypervisor installed on them.

Summary

In this chapter, we have covered the configuration of CloudStack, which covers the below points:

- Working with the management console
- Infrastructure resources which can be added in different style of configuration
- Introduction to various logical segregations of physical resources and how they can be used to benefit the end users

In the next chapter, we will introduce you to:

- The concept of networking in CloudStack
- It will give you the details about the importance of networking in Cloud
- Details about the various types of configuration that can be done as per the needs of your organization

Chapter 4. Apache CloudStack Networking

Networking is one of the most important aspects in the deployment of a cloud solution. Maintaining the privacy and isolation of the users' data in a multi-tenant environment is of prime importance. CloudStack allows administrators to define network topologies at various levels, which enables users to build complex application environments on the cloud.

Project or domain administrators can also define private networks which will be available only within that project or domain. Projects are used to group users to work together and the resources can be managed together, whereas Domains are a group of accounts connected with some logical relationship. Projects and domains are explained in detail in Chapter 7, *Domains, Accounts, Projects, and Users*. Administrators can create various kinds of network offerings which can include different types of network services that the users can leverage.

We have already given an overview of creating networks while creating zones in the previous chapter.

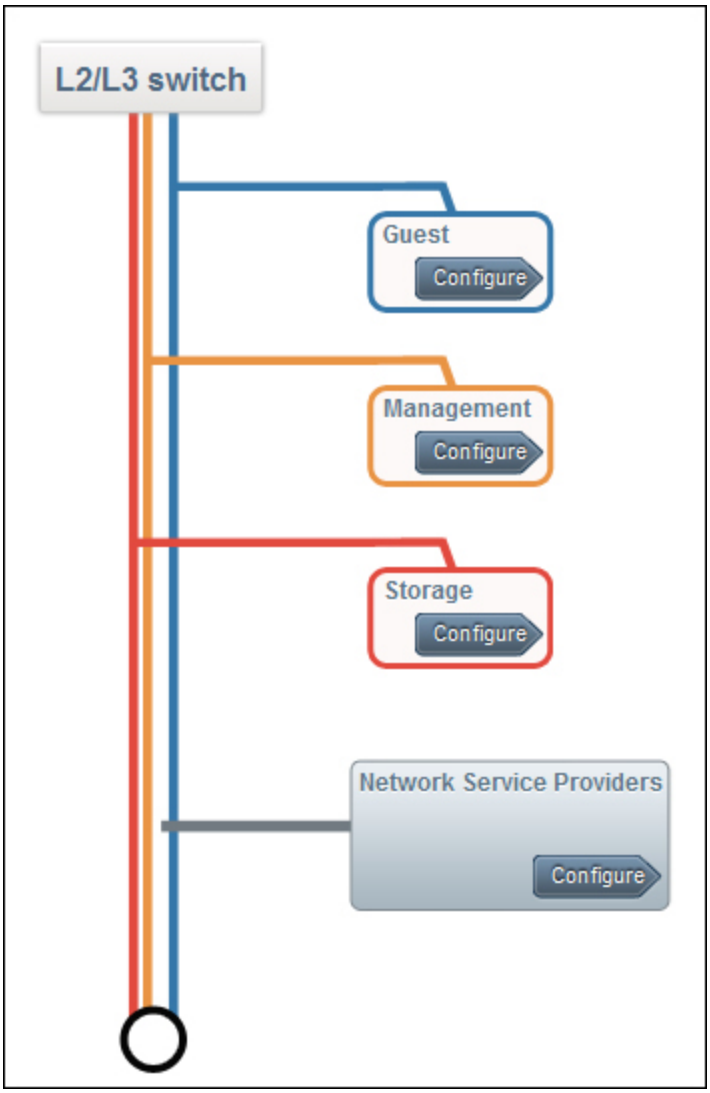
We are going to dive in to more details about what happens when you are creating different types of networks in CloudStack, and what other services are supported to be deployed in CloudStack.

As we already know that while defining networks in CloudStack, administrators have the options to define two types of networking methodologies, they are basic and advanced. The choice of networking model depends upon your requirements.

Zones and their types

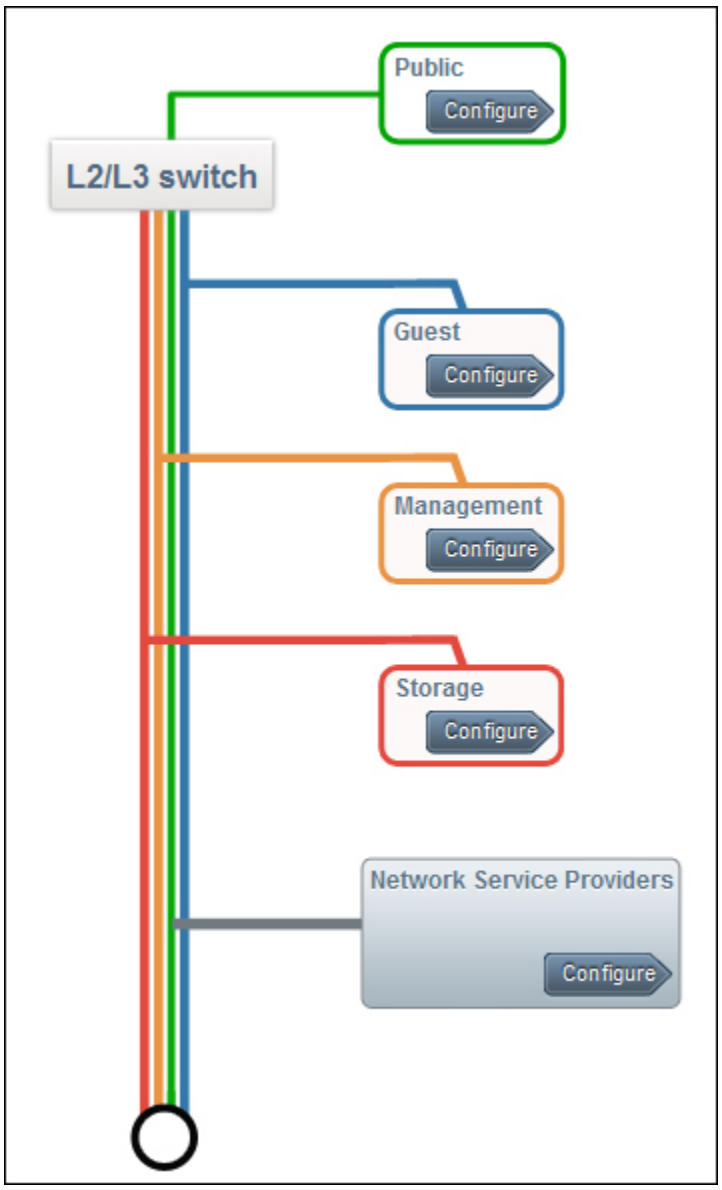
Let's start by taking a look the types of zones:

- **Basic Zone**: If you want to configure basic networking with only one physical network then you should go for the basic network configuration. In basic network configuration, the guest instances can be assigned an IP address that is the same as the underlying CloudStack hosts. You should provide an IP address range sufficient for the CloudStack infrastructure as well as the guest instances. You can provision up to a maximum of a /16 subnet which includes all the zones in your deployment, that is, if you specify an IP range such as 192.168.0.0/20 for each zone, there will be 4096 IP addresses available in that zone. From this set of IP addresses assigned to the zone, a subset of IP addresses is assigned to each of the pods in the zone, and the guests get their IP address from this range. In any case, each of the components of CloudStack infrastructure such as System VMs and Hosts has a unique IP address. The pods in a zone are assigned a different set of IP addresses as it is a broadcast domain and must not have same IP address as any other POD in the same zone for the guest network. The administrator must configure IP ranges for each POD. This zone provides a single physical network where each guest VM is assigned an IP address directly from the network and the guest network isolation can be provided using layer-3 techniques such as Security groups, which is basically IP address source filtering.



- **Advanced Zone**: If you are deploying a cloud with complex networking in mind with multiple physical networks, guest networks and advanced networking services, then you should choose the **Advanced Networking** option. This option provides isolated networks for tenants. In advanced networking option, you can deploy various physical networks for various types of network traffic such as public communication between the guest instances and the Internet, or guest instances traffic for communication between the guest instances. Each of the guest networks that you create inside of the zone with the advanced networking option has a VLAN ID associated with it. This type of zone must be used when a more complex network topology is to be defined

providing flexibility in defining the guest networks and network services such as firewall, VPN, or load balancers.



The guest IP address ranges within a zone are the same by default for all the zones, the administrator can also change it to use their own range of IP addresses. There are different VLANs assigned, the guest instances are a part of one of these VLANs, and these instances communicate with other instances in the same VLAN and the virtual router. If these instances are to communicate with other instances in other VLANs, they can do this by using a layer 3 switch or router.

We will discuss more details about the two different types of networking options later in this chapter. First let us discuss the different types of networks in cloud. There are basically two broad types of networks available in CloudStack.

Physical networks

The physical network corresponds to a NIC on the hypervisor host and can carry one or more types of network traffic; the choice of traffic depends upon the type of network that you choose in the zone —basic or advanced.

Basic zone

The various types of network traffic available in a zone with basic networking are as follows:

- **Guest traffic**: This is the traffic generated by the guest VMs in the cloud when the guest virtual machines communicate over their local area network. The administrator configures the IP address range which is assigned to the guest virtual machine within a POD. Each POD must have a different set of IP address ranges in a zone as it is a broadcast domain.
- **Management traffic**: This traffic is generated when the internal resources of CloudStack such as hosts, system VMs, or any other component communicate with each other or with the CloudStack management server. and it also includes the service network which carries the heartbeat traffic associated across systems configured for high availability.
- **Storage traffic**: This is the traffic which is generated when the primary and secondary storage servers communicate with the hosts. It is generated on operations such as creating an instance from VM templates, creating snapshots in case of secondary storage and data read/write operations in case of primary storage.

While configuring a zone with basic network configuration, the administrator must assign different IP address ranges for different types of traffic. This kind of configuration is very straightforward and the IP range is used for all the traffic as well as the CloudStack infrastructure such as hosts—the hosts and the guests can also be in the same VLAN. The guest VMs receive a public IP address on each of their single NICs and the IP addresses provided to the pods are generally used for the management traffic.

Advanced Zone

In case of a Zone with advanced networking type, the administrator can configure multiple physical networks in a zone. These networks can be configured to carry one or more traffic types as per the configuration done by the administrator.

Guest traffic

This is the network traffic generated by the communication between the guest VMs. This traffic flows over the guest network and it can be shared or isolated.

Isolated

In case the guest network is an isolated one, the guest instances will be provisioned in a network isolated from other networks based on VLANs. There can be multiple accounts in each domain and each account can contain multiple networks. The network isolation is provided for different networks. In this type of isolated guest network, the administrator needs to provide different VLAN ranges so that isolation is provided to each network in a CloudStack account.

Shared

In the case of a shared network, the guests are provisioned in a single guest network and the guest VMs can be isolation from each other using Layer-3 isolation techniques such as security groups. In the shared network the users can define the communication ports

from a source which is to be allowed—the IP addresses and the ports for communication are defined. These shared networks can be VLAN tagged or not.

These networks created by the administrator can span the zone, multiple pods, and thus be available to multiple accounts or the administrator can configure it to be scoped to only a single account wherein the users of only that account will be able to create guests and attach them to this network.

There can be thousands of different networks provisioned which are defined using different VLAN ID, IP range, and gateway. The guests in the networks are provided IP addresses from the range specified for the network they are being provisioned into.

> **Note**
>
> Note that Security Groups in advanced networks have many dependencies which may be resolved in the upcoming version of CloudStack—4.1.

Management traffic

As defined above in the basic network type zone, this network traffic is generated when the internal resources of CloudStack such as hosts and system VMs communicate with each other or with CloudStack Management server.

The administrator must configure a unique range of reserved IP addresses to be used for the management network that is to be used for management traffic. There cannot be two resources of CloudStack infrastructure with same the IP addresses even if they are in different zones. The private IP address range is provided to the POD by the administrator and the infrastructure components of CloudStack such as the hosts, console proxy, and secondary storage system VMs are allocated private IP addresses from this range.

The Compute nodes where the Management server and the hypervisors run must be in the same subnet as of the Management network IP address to enable direct communication between them. Thus, the guest VMs, hosts and the management server do not have overlapping IP addresses with the system reserved IP range. For example, the administrator can have a 192.168.20.0/24 subnet from which he can assign 192.168.20.2 to 192.168.20.7 as the system reserved IP addresses and then he can assign the rest of the IP addresses in 192.168.20.0/24 network to the management server and the hypervisor hosts.

- The administrator provides an IP address range to the pod. CloudStack recommends using one private IP per host in the POD. In case of KVM the host hypervisor and the management server may not be in the same subnet. So, in this case, if you plan to grow the strength of the POD, you must provide sufficient IP address to accommodate the number of hosts. In this case, where XenServer or KVM is running on the nodes in the POD, link-local IP addresses can be used to accommodate the large number of hosts. These link-local addresses are private IP addresses that can be assigned to machines by the CloudStack for internal communications only within a segment of a network and are defined as 169.65.0.0/16 in /16 subnet, so it provides around 65000 IP addresses which should be sufficient to accommodate any number of hosts or any CloudStack virtual router.
- In case of VMware vSphere, the number of private IP addresses must be enough to accommodate the total number of customers in the cloud. The IP addresses assigned in this case lie within the subnet defined by the administrator which has around 255 IPs per POD in case you specify 24 as the size of the subnet. The IP addresses in this range are used for the hosts, guest virtual router, and other cloud infrastructure resources, so if you plan to add more hosts in the future, you must specify a greater range of IP addresses with CIDR as 20 which provides around 4000 IP addresses or

you can create multiple pods with their own subnets of 24 size that provides *255 * 10 = 2550* IP addresses.

Public traffic

This traffic is unique in the zone with the advanced network type. This is the traffic which is between the guest VMs and the Internet. This traffic requires IP addresses that are publicly routable.

The users can request these IP addresses from the CloudStack UI and can also configure NAT server to allow communication between the guest VMs and the Internet.

The public communication between the guests and the Internet is made possible using the public IP addresses. CloudStack provisions public IP address up to the maximum limit in the account. The account admin/users can use these public IP addresses to set up a NAT instance so that other instances in that account can also have access Internet access. The other instances when try to access the Internet will be routed to the NAT instance, which in turn provides Internet access to them. These IP addresses can also be used to provide load balancing and VPN services.

CloudStack also allows us to access Internet using a single IP address to provide NAT interface for all the accounts within it. This can be done using the Juniper SRX firewall which significantly reduces the consumption of public IP addresses (this is mostly used in private cloud scenarios) enabling the use of single public IP address to provide Internet accessibility to multiple machines.

The administrator can define one or more ranges of public IP addresses to be used by CloudStack, which can be requested by users.

Storage traffic

As described above, this traffic is the traffic generated by the communication between the hosts and the Primary storage, and

hosts and the secondary storage. The administrator is asked to configure the storage network while defining the Zone.

Administrator can configure multiple physical networks in the zone and these physical networks can carry any single or group of network traffic. When a user creates a zone with advanced networking, he has the option to create multiple networks and add multiple types of traffic to them, as shown in the following screenshot:

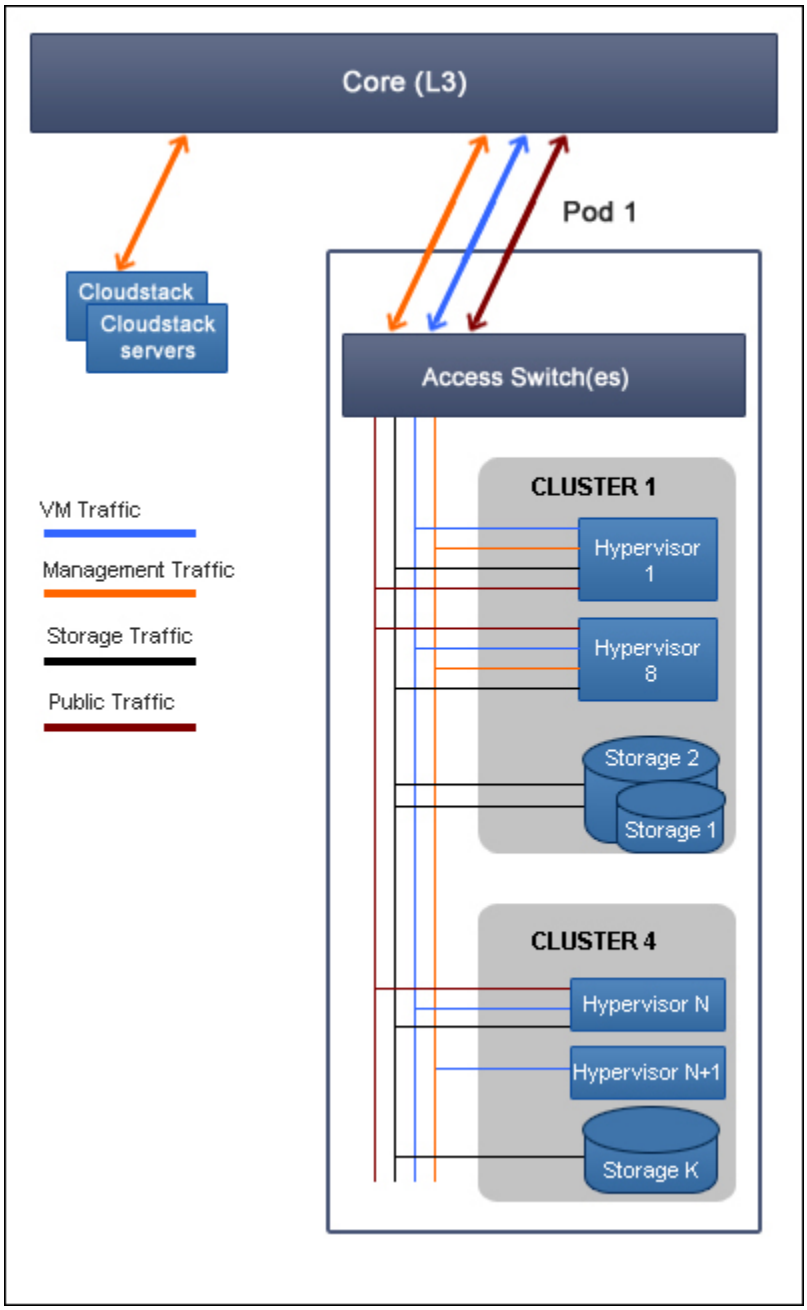


In the screen as you can see, we have added multiple physical networks such as Physical Network 1, Physical Network 2, and Physical Network 3. In Physical Network 1, we have all kinds of

traffic. Whereas the Physical Networks 2 and 3 only have Guest Traffic configured.

Now that we have specified multiple physical networks in the zone, we need to configure labels for the different traffic in different Physical Networks. This can be done by clicking on the **Edit** button below. These labels are used to distinguish and map different traffic with the underlying physical network label.

These networks are used for different kinds of traffic, as depicted in the following diagram:



Virtual networks

CloudStack also supports virtual networks, which is basically a logical construct and helps in providing multi-tenancy on a physical network. Lets' say you need a private guest network for your guests, isolated from the other guests on the physical network, then you need to create a virtual network. This virtual network in CloudStack can either be shared or isolated.

When the administrator creates the virtual network, the information is only stored in CloudStack. CloudStack actually activates the resources and creates the virtual network once the first guest virtual machine starts in the network, and when there is no virtual machine in the network, all the network resources are garbage collected, thus it helps in conserving the network resources. The types of virtual networks in CloudStack are:

- **Isolated virtual network**: The isolated network is private to a single account, that is it cannot be accessed by guests from other accounts. In this type of network, the resources like VLANs are assigned as well as garbage collected dynamically. It has only one network offering for the entire network, which can be upgraded and downgraded for the entire network.
- **Shared virtual networks**: By the name, you must have got an idea that this virtual network can be shared among multiple accounts and projects in CloudStack. The shared virtual network is designed to provide inter-connectivity between the guest VMs. The guest VMs can control the communication among themselves in this type of shared network using Layer-3 isolation such as security groups. These networks created by the administrator are assigned to a certain domain. The administrator provides network resources such as VLANs and physical networks that this virtual network is mapped to.

Network offerings

Network offerings defined by the CloudStack administrator are a set of network services that are provided to the users. The network offerings include network services such as virtual router, DHCP, DNS, and Load Balancer. Network offerings can also be used when you want to extend your networking services. While creating a network offering, administrators have a number of service options that they can add. The list of services supported by CloudStack 4.0.1 is as follows:

- VPN
- DHCP
- DNS
- Firewall
- Load balancer
- User data
- Source NAT
- Static NAT
- Port forwarding
- Security groups

Administrators can choose from this list of services to offer while creating network offerings.

There are some network offerings already present with the installation of the CloudStack. The administrators can choose from the default network offerings or they can create different network offerings which can be custom apart from the standard default network offerings available by CloudStack. This creation of different types of network offerings enables the cloud provider to offer different classes of services to the users in a single multi-tenant physical network. It allows different tenants on the same physical network to have different kinds of services such as firewall and NAT.

The administrator selects the network services to be added into the network offering being created from the list given above and they are tagged to map them to the underlying physical network.

While creating a guest instance, users can chose which offering they want to use from the list of offerings, this choice of offerings determines which networking services the guest can use. By default there are three network offerings that are available in CloudStack and there are other network offerings also available which are used by the CloudStack system and are not visible to the end users.

There are various states of network offering such as **enabled**, **disabled**, or **inactive**. The network offering when disabled cannot be used to create any network.

You might have noticed that we have used the term virtual router a lot of times, let us discuss a little bit more about what a CloudStack virtual router is.

Virtual router

CloudStack virtual router is a service provided by CloudStack. It is a type of System Virtual Machine. The tenants do not have access to this router. The administrator can access this router. The administrator can log in to this router and change the configuration and can also troubleshoot the router. The administrator may also need to restart the router after some configuration changes. The tenants can take certain actions such as configuring port forwarding on these routers but cannot log in to these machines by any means.

The tenants can ping this virtual router to ensure its availability for testing purposes.

CloudStack virtual routers provide network services and are provisioned along with system offerings whenever required. CloudStack creates a virtual router when the guest network is created using the default settings which are defined in the system service offering associated with the network offering, but this virtual router's capabilities can also be upgraded by choosing a different system service offering.

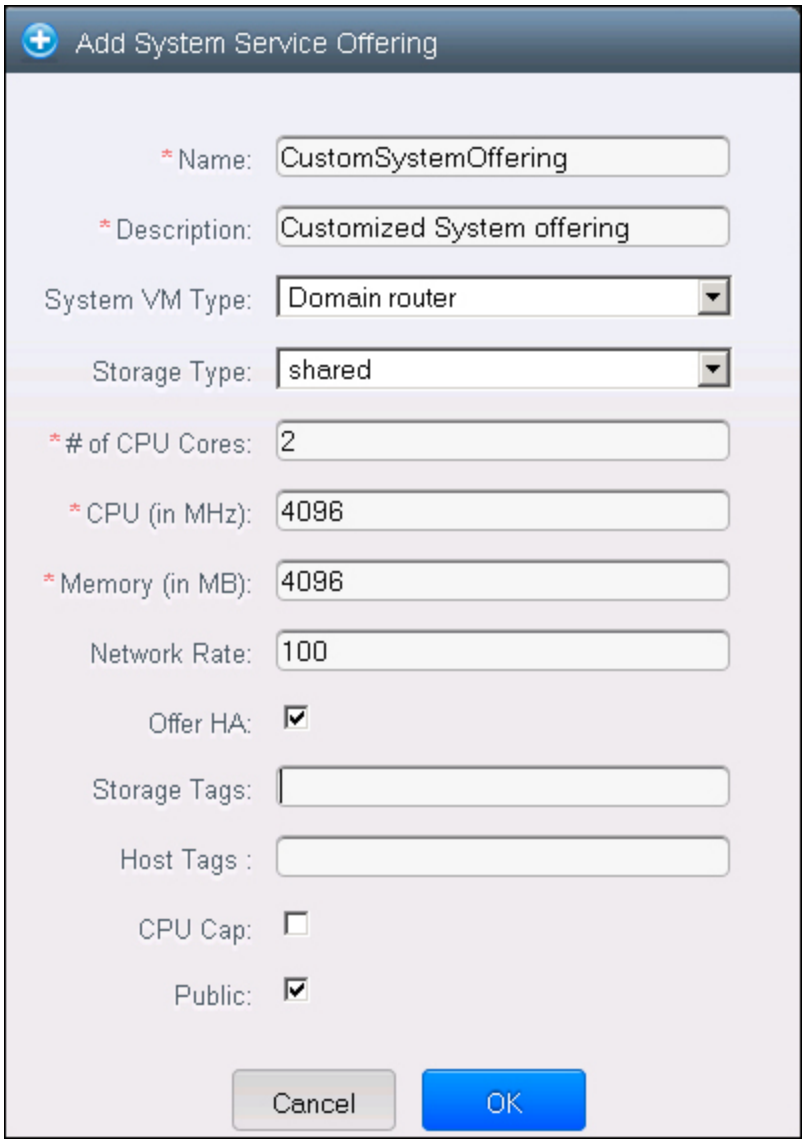
The users can set the some of the features of a virtual router such as:

- IP range to be assigned to the virtual router
- Network services that are supported by the virtual router
- Default domain name for the network serviced by the virtual router
- IP address of the gateway
- The administrator can also set the frequency of fetching network usage statistics from CloudStack virtual routers. This can be done by setting the global configuration parameter `router.stats.interval` which is set to `0` when there is no virtual router present in the network.

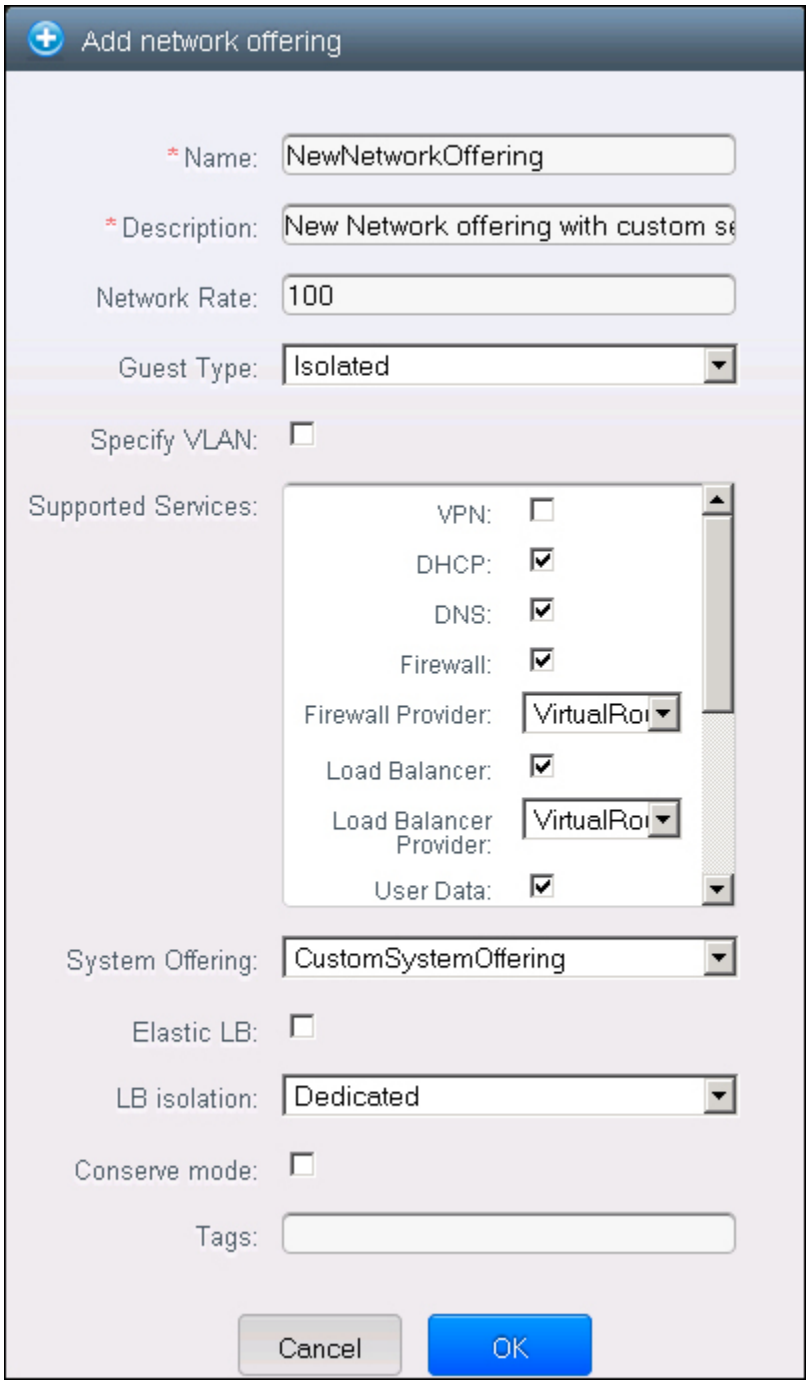
System service offerings and virtual router

CloudStack creates a virtual router when a new network is created; the virtual router provides the network services to the network. CloudStack uses default settings which are defined in the system service offering that is unique for all the virtual routers in the same network. The administrators also have the ability to upgrade the virtual router by applying custom system service offerings. The steps are given as follows:

1. Create a custom system service offering as per your requirements. While creating a system service offering chose **Domain router** in the System VM type.



2. Once this new system service offering is created we can use this system service offering with a new network offering. This new network offering can then be used while creating a new network. When creating a network offering select the network services that are to be included in this network offering and then select the new system offering just created.

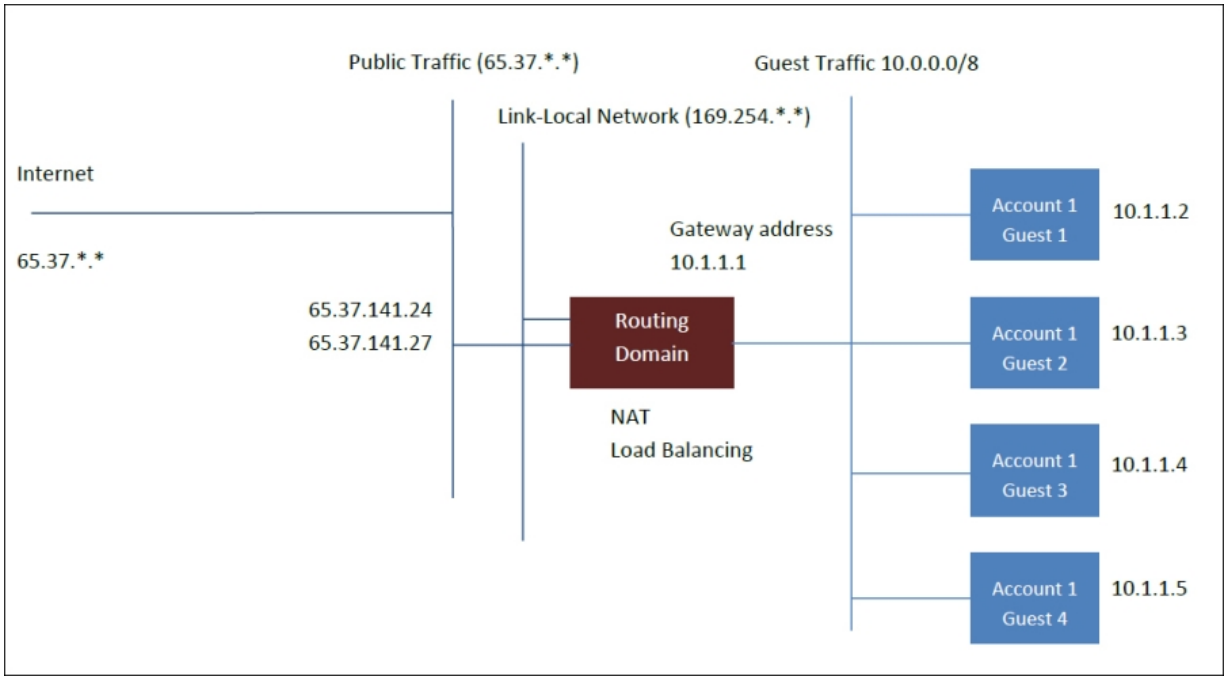


3. Now, users can create a new guest network based on this new network offering. The virtual router created in this network will use the system service offering that you created in step 1.

As already mentioned you can configure multiple network services in a network offering as specified in the list above. CloudStack provides

the network services using the virtual router. For example, when you are adding DHCP, DNS, Port Forwarding, User Data, VPN, Firewall, Load Balancer, Source Nat, and Static NAT, you will be asked which system service offering you want to use with them. The system service offering listed here are those which are configured as CloudStack virtual router. While you are adding services such as firewall, load balancer, source NAT, static NAT, port forwarding. CloudStack provides the option to use Hardware Juniper SRX or use CloudStack virtual router.

CloudStack virtual router is the main component of CloudStack which provides most of the network features as discussed above. We will see how it can be used to provision guest instances in a multi-tier network. CloudStack virtual router is automatically provisioned by CloudStack on the hosts for each of the networks that is created using some network offering that has a System service offering using CloudStack virtual router. The virtual router deployed by CloudStack has multiple Network interface cards associated with it which has different IP addresses to serve different purposes. An example is shown in the following diagram:



As explained in the preceding diagram, there are different types of traffic that flow in the cloud and each network is given a separate range of IP addresses. The guest traffic is given a range of IP addresses (10.0.0.0/8), from which the IP addresses 10.1.1.2-5 is given to the guest VMs in the network with the gateway address as 10.1.1.1. These guests can reach Internet using the NAT services provided the virtual router as shown in the preceding diagram. This is how multiple guest VMs can reach internet with one public IP address. The public traffic is shown distinctively in the figure. The link-local IP address is used by the virtual machines to communicate with the hosts.

CloudStack virtual router has 3 NICs with 1 IP address each. These IP addresses are used for different purposes.

- The first NIC has the IP address 10.1.1.1 and it serves as the gateway for the guest traffic. All guest traffic going outside the guest network passes through the gateway.
- The second NIC is used by the CloudStack system to configure the virtual router.
- Now, in this kind of network, DNAT is configured, which allows the guest instances to connect to the Internet, so all the instances access the internet using the NAT instance—the virtual router. The third NIC of the virtual router has a public IP address which has direct access to the Internet. This IP address provides Internet access to all the instances.

If you chose to create a new network with a network offering with network service like DHCP, then CloudStack will create a virtual machine for the use of CloudStack virtual router which serves as the DHCP server for the network and provides IP addresses to all the machines which are created in this network. You can also configure your guest virtual machine not to use this service—it can have an IP address manually configured as well, though CloudStack reserves an IP address from the pool for each guest VM.

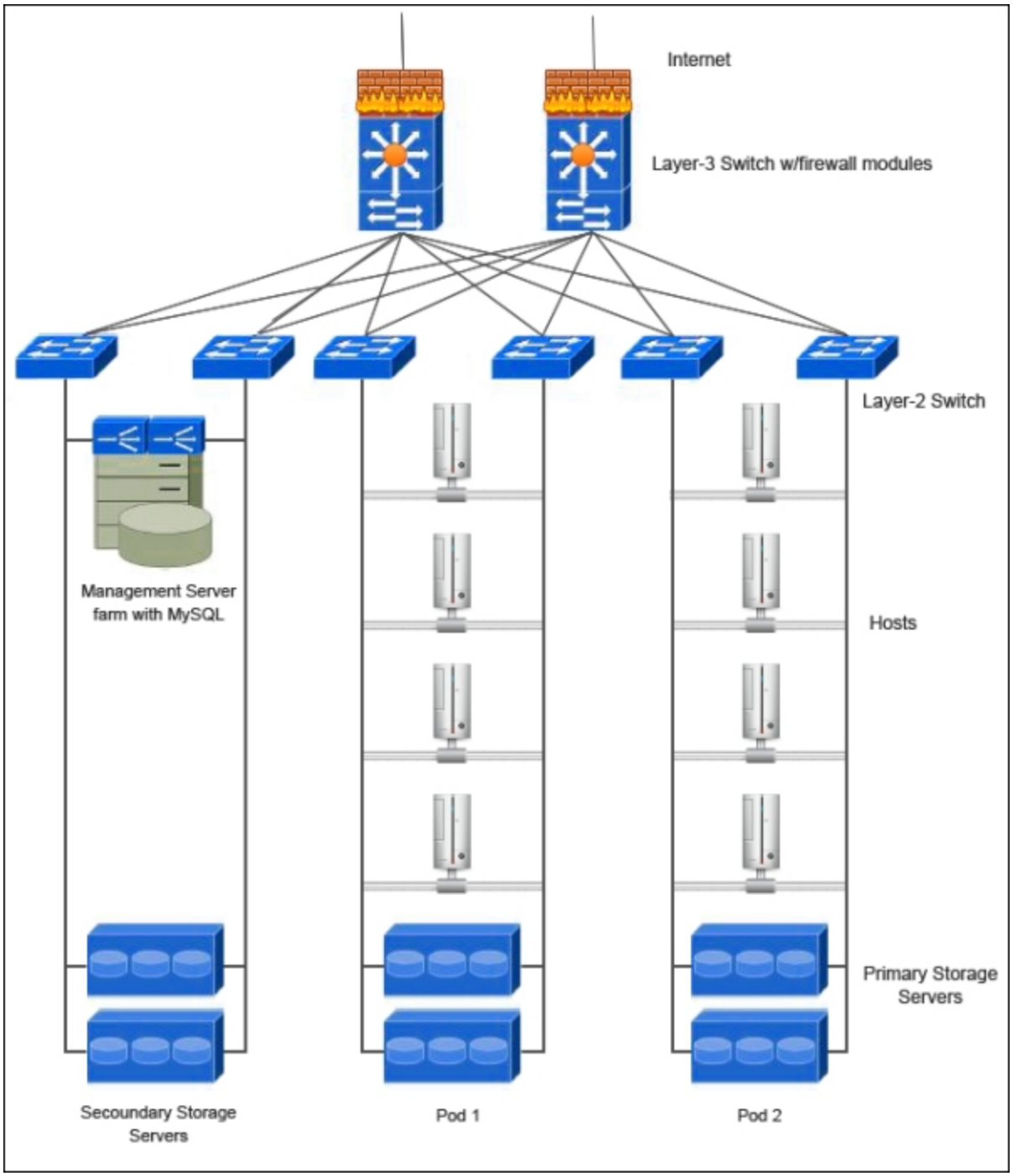
Network in cloud

CloudStack allows configuring of networks such that you can isolate each of the customers from each other as well provision different networks for different projects, domains, and accounts which can be private to each of the entities or can be made public so that they can be shared among different groups.

When you create a zone with a basic network configuration, it is pretty straight forward when you have to configure only one guest network to carry guest traffic generated by the guest virtual machines the steps of which were described in the previous chapter. The guest instances are provided IP addresses from the CIDR range of the POD in which the guests are provisioned and they are in the same VLANs as the hosts.

Whereas when you configure a zone with an advanced network configuration, you can create multiple guest private networks and the guest instances are provided IP addresses and network configurations as per the configuration of the network they are provisioned into.

Let's consider the following network configuration of the zone, as shown in the following diagram:



In this network configuration of the zone, there are different pods with different VLANs. Each pod has two layer-2 switches, which provide isolation to the hosts and guests on them and the management server, and the secondary storage is placed on a different network from the hosts. The layer-2 switch functionality is provided by the CloudStack virtual router and all the traffic such as

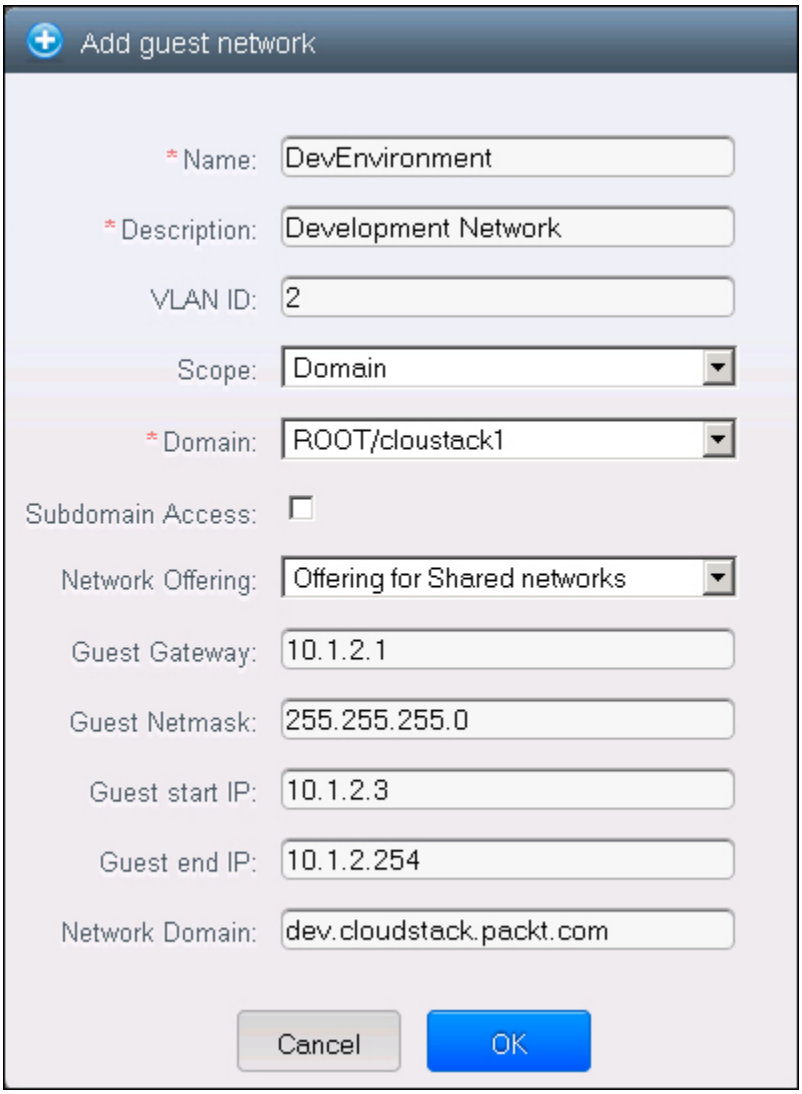
management traffic, guest traffic, and public traffic are all routed by these switches.

In a zone with advanced network configuration, additional networks for guest traffic can also be added at any time once you have created the zone. For creating an additional guest network, follow these steps:

Note

These steps can also be initiated from the **Infrastructure** tab, in the **Network** section.

1. Click on the **Network** tab in the left panel.
2. Click on **Add Network**; the following screen is displayed:



In order to create a new guest network, we need to fill in the information as per our requirements, as follows:

- **Name / Description**: Name and description of the new network that you are creating
- **VLAN ID**: This is an optional field for tagging the network for referencing purposes
- **Scope**: This option defines the scope of the network that you are creating; it can be domain-specific or account specific
- **Domain/Account**: Depending upon the choice of the scope, select the domain name or the account for which this network is defined

- **Network Offering**: Select the network offering that you want to choose for this network
- **Guest Gateway**: This is the gateway that the guests should use
- **Guest Netmask**: This is the netmask that is used on the subnet which the guest will use
- **Guest Start IP/ Guest End IP**: This is the start and end IP address that define a range of IP addresses which will be assigned to the guests in the network
- **Network Domain**: This is an optional field which specifies the Domain name or the DNS suffix that you want to assign to this network

That's it; you have created a new guest network that can now be used by the guest VMs. The network the VM is associated with is defined at the creation time. It cannot be changed after the VM has been created, the user can at most change the IP address associated with one of the NICs. The virtual machine that is created always has one default network; the users can add multiple non-default networks to a guest machine apart from the default network. The network that is being used is configured by the administrator.

A network can be made available to an entire zone or specific to a domain or an account. The guests in the same network can communicate with each other. The user who has access to the scope of the network, i.e. domain/account/zone can create a VM with access to that network. The networks that are zone-wide provide little or no isolation between the guests. For strong isolation, it is recommended that you create networks which are account specific. The VLANs are not enough for providing the security to the guest VMs, we can also configure security groups to provide Layer -3 isolation which helps in restricting traffic by port and source IP addresses.

Network services

Networking is built using various types of network services. In CloudStack administrators can use one or many types of services to be included in the type of networking they want to establish in the cloud. These network services provide different services such as public communication, firewall protection, load balancing, and so on.

Public IP addresses

The administrator can add at least one public IP address for the Internet traffic. This public IP address can be used to provide source NAT. If you are using a Juniper SRX firewall, then CloudStack can instead use a single IP address as an interface NAT for all the existing accounts, which reduces the number of IP addresses consumed. For adding public IP addresses, follow these steps:

1. Click on the **Network** tab in the left panel.
2. In the public node of the diagram, click on **Configure**.
3. Click on the IP ranges tab.
4. Enter the details given as follows:

    - **Gateway**: The gateway that will be used for these IP addresses
    - **Netmask**: The netmask that is associated with this IP range
    - **VLAN**: This is an optional field to specify the VLAN to be used for public traffic
    - **Start IP/End IP**: This is the range of the IP addresses that are accessible from the internet and will be allocated to the guest networks

Elastic IP address

CloudStack provides a public IP address which can be associated with any instance using a **NetScaler** appliance. This NetScaler appliance can provide both Elastic Load Balancing as well as Elastic IP addresses. There must be a public VLAN to be present and the zone must be security group enabled. The public IP address is

allocated from this range of the IP addresses. A user can request the public IP address using the API call `associateIp` and can configure static NAT. These IP addresses can be associated with any instance within the same zone.

CloudStack provides Static NAT (1:1) service in a basic Zone using Elastic IP addresses. When a user deploys a VM, it can access the Internet using the static NAT service between its private IP address and a public IP address, this public IP address can be attached to or detached from any running instance in a zone.

EIP cannot be moved between zones. The users can acquire new public IPs when they want to and it is provided to their account and it can be associated with any instance in that account.

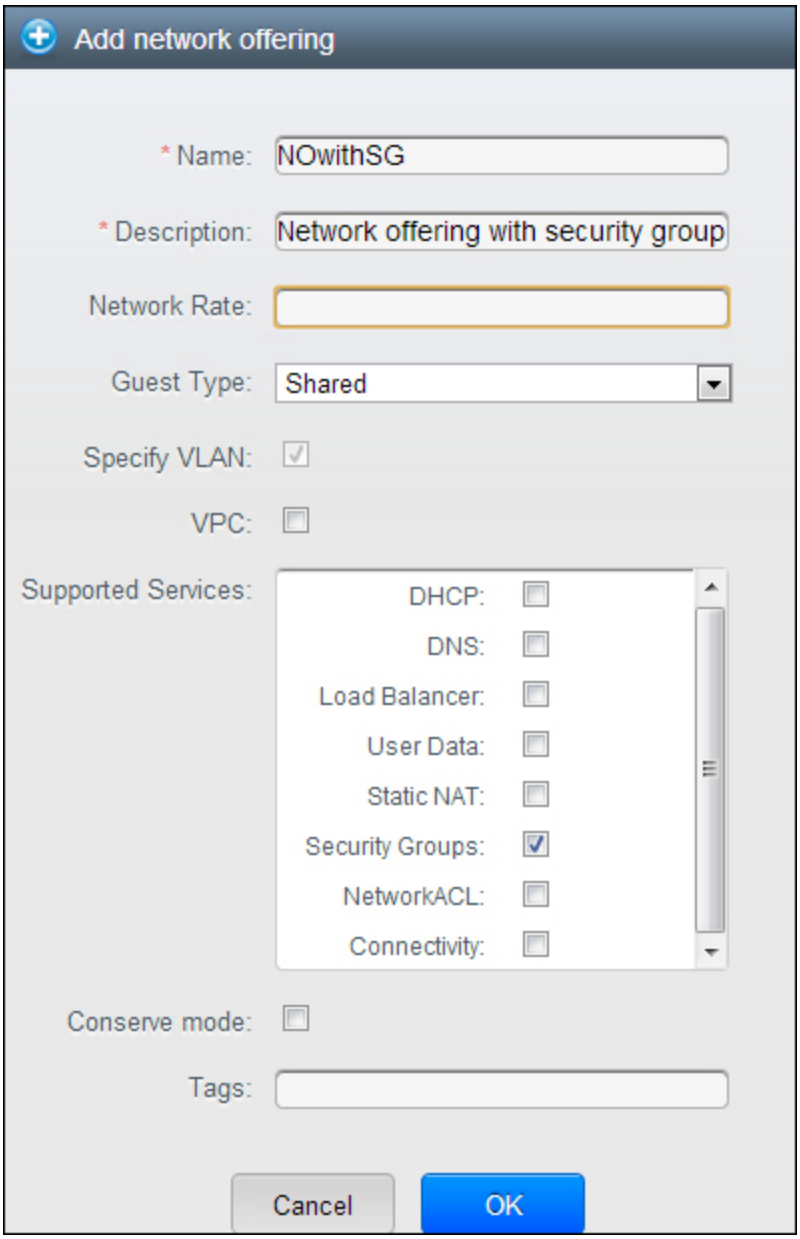
Security groups

CloudStack provides isolation of traffic to VMs by the use of security groups. The security group is basically a group of VMs that provides filtering of the incoming and outgoing traffic as per the rule configuration. We can set the ingress and the egress rules for the network using security groups. The network traffic is filtered according to the rules configured in the security groups. As we have already discussed, a zone with basic network configuration has isolation implemented using security groups. These security groups are very useful in a zone with basic network configuration as all the guest VMs are present in a single network.

The security groups provide IP address filtering and have rules defined within it which define the incoming and outgoing traffic characteristics, like for example, the protocol, port and the source of the communication which is to be allowed or denied. CloudStack provides a default security group which has some rules predefined with itself and it can be modified whenever required. This default security group has rules defined to deny all incoming traffic and allow all outbound traffic. Users can modify this security group or define new ones as per their requirements. An instance is associated with a security group when it is created and it cannot be changed

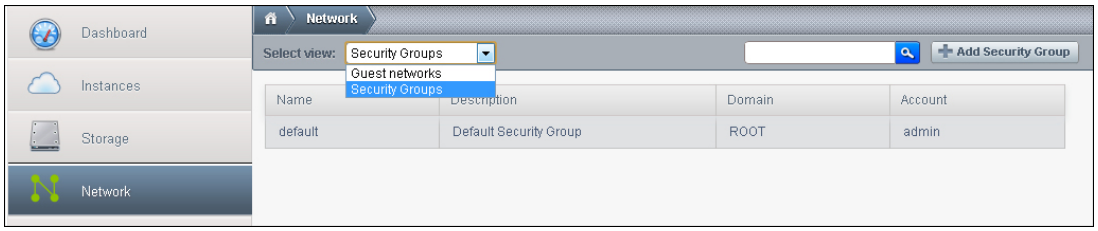
afterwards, this security group can be the default one or any other security group created by the users. The security group associated with the instance defines the traffic that will be allowed or denied for that instance and it can be modified at any point of time and the new configuration will define the traffic rules from that point irrespective of the VM's state (stopped/started).

The zone has a security group feature that must be enabled before the security group functionality can be used in the zone. The enablement of a security group in a zone is defined by the administrator when creating the zone. As we have seen, while creating a zone, the administrator selects the type of network offering, the security group feature is defined in the network offerings and if the administrator wants the security group to be enabled in the zone, he must choose a network offering with security group functionality. The security group feature is provided by the CloudStack virtual router. The network offerings can be created to include services such as the security groups as described earlier. We can do this by selecting the Security Groups check box in the list below while creating a new network offering.

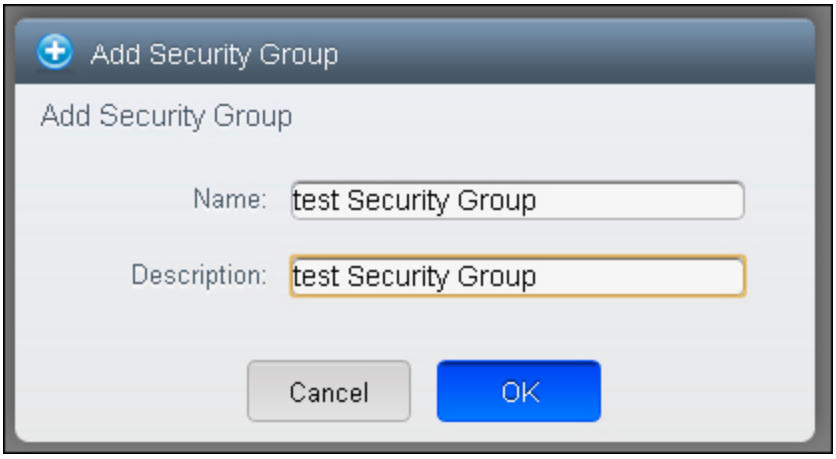


To add a new security group, follow these steps:

1. Click on the **Network** tab in the left pane.
2. Select one of the networks from the list. The selected network must have an offering with a security group.



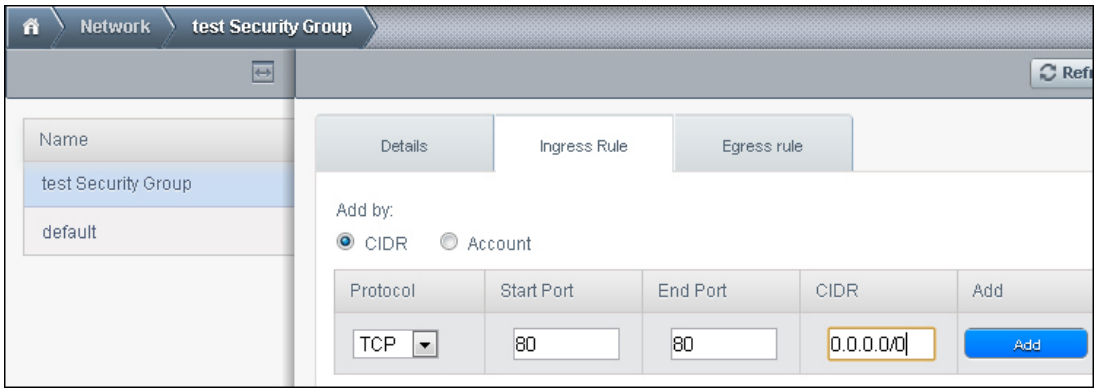
3. Select choose security groups and click on **Add Security Groups**.
4. **Name** / **Description**: Provide the name and a description of the security group.
5. Click on **OK**.



The newly created security group appears in the list of security groups.

Once you have added a security group, you can add or delete new rules to it. If a security group has no rules defined in it, no traffic is allowed to it. Add the Ingress and Egress rules as follows:

1. Click on the security group you want to add or delete the rules to.
2. Select the rule that you want to add—if you want to add an Ingress rule, click on the **Ingress Rules** tab, and fill the fields in the following screenshot:



Fill in the information as follows:

- **Add by CIDR/Account**: This field defines whether the source of the traffic will be defined by an IP range or by a security group already present in a CloudStack account. If you want to allow the traffic from a particular existing account, then select "Account". This is done to allow traffic from one account to another. For example, if this security group is for a Database server and you want to allow traffic from the WebServer then you must select the WebServer account. Or if you want to allow the traffic defined by an IP range, then you must select the IP range. For example, if this security group is to be associated with a WebServer, and traffic from all over the internet is to be allowed, then select CIDR range.
    - **Protocol**: Select the networking protocol from the list (TCP, UDP, ICMP, etc.). The traffic from the source using this protocol will be allowed.
    - **Start Port, End Port**: This field is for the TCP and UDP protocol in the protocol field. This is the range of the listening ports where the incoming traffic will be sent for this rule.
    - **ICMP Type, ICMP Code**: This is only for ICMP rule. It is the type of ICMP message or error code that should be accepted as per this rule.

- **CIDR**: If you have selected CIDR in the first option, enter the CIDR range(s) or IP address(es) from which the traffic should be allowed. For example, if you want all the traffic from all over the internet to be allowed for the WebServer instance, specify 0.0.0.0/0 or if it is internal application to be used only for private set of users, enter their IP ranges, like 192.8.0.0/24.
- **Account/Security Group**: This option is when you select the account option. Enter the name of the existing CloudStack account or an existing security group for which this rule is defined. If you have a security group for WebServer instances and this security group is for the Database server, then enter the WebServer instances' security group.

3. Click on **Add.**

Follow the same steps for adding an Egress rule to the security group.

Using external devices with CloudStack

CloudStack provides most of its network services using its virtual router but in some of the services, the administrators have the option to select whether they want to use a CloudStack virtual router or an external device for providing that service. These services are firewall, load balancer, source NAT, static NAT, port forwarding where the administrator can use CloudStack virtual router or an external device while defining the network offering.

CloudStack allows using an external Juniper SRX device instead of virtual router and also allows using external NetScaler or F5 load balancer for gateway and load balancing services. But for using these external devices for firewall and load balancer we need to have some initial setup in place before we could proceed further.

CloudStack programs the firewall and the load balancer as soon as the first guest machine is set up in CloudStack's account. The configuration done by CloudStack on these devices are listed as follows:

When using external firewall:

- A new logical interface is created on the firewall device to connect it to the CloudStack's account private VLAN. The IP address assigned to this logical interface is the first IP address of the account's private subnet.
- A source NAT rule is defined using the public IP address as source address to provide internet access and forwards all the outgoing traffic from the account's private VLAN to the public Internet.
- A counter known as firewall filter counts the number of bytes of outgoing traffic for the account.

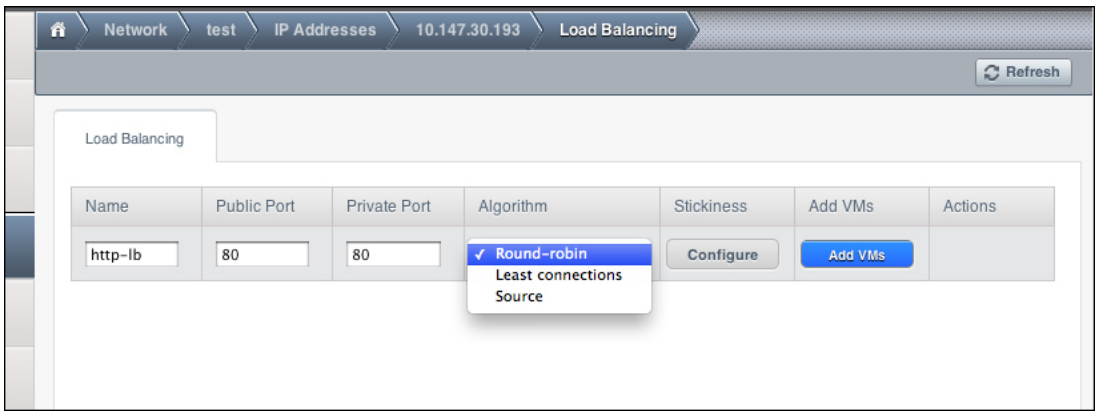
When using external load balancer:

- When an external Load Balancer is used, a new VLAN is created on the device which matches the account's provisioned zone VLAN.
- The VLAN has an IP address reserved for it as self IP address which is always the second IP address of the account's private subnet.

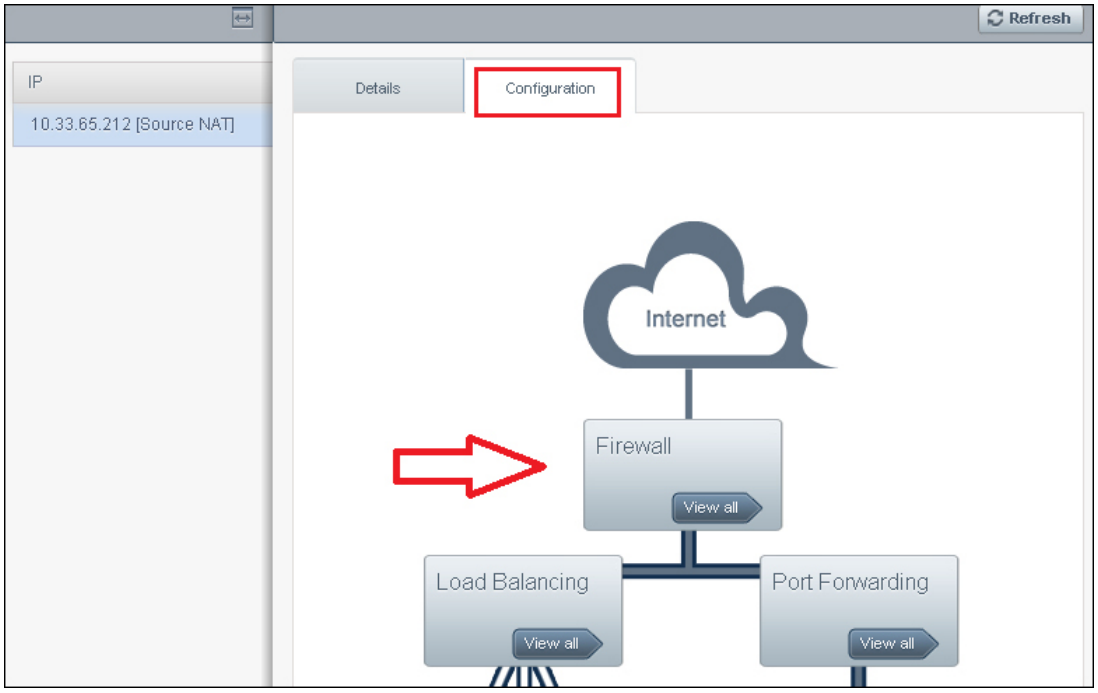
Load balancing rules

For adding new rules to the load balancer, follow these steps:

1. Click on the **Network** tab on the left pane.
2. Click on the network where you want to load balance the traffic.
3. Click on **View IP address**.
4. Click the IP address for which you want to create the rule, and go to the **Configuration** tab.



5. On the load balancing diagram, click on **View All**.



Fill all the information as follows:

- **Name**: Name of the load balance rule that you are creating.
- **Public Port**: The port address which is receiving the traffic that needs to be load balanced.
- **Private Port**: The port of the guest instance which will receive the traffic.
- **Algorithm**: From the list, chose the load balancing algorithm using which the load should be balanced.

- **Stickiness**: This is an optional field which specifies the algorithm for the stickiness policy which ensures the availability of the information across the numerous requests in a user's session. It is most commonly used in Web applications. Let's say there are multiple instances hosting a website under a load balancer. If a user has a session in one of instance, when he returns back the request will be directed to the server having his session instead of sending it to some other instance using the load balancing algorithm, thus the users' session remains as it is. The load balancer rule specified in CloudStack can have a stickiness policy associated with it. These policies are defined by three fields, they are Name: name of the policy, Method and Parameters. The Method of the stickiness policy can be load balancer-generated cookie, or an application-generated cookie, in which cookies are used or it can be source-based in which the source IP address is used.

6. Click on **Add VMs** and select the guest instances among which the incoming traffic load is to be balanced and then click on **Apply**.

CloudStack allows you to add multiple load balancing rules to one IP address.

Elastic Load Balancer

CloudStack allows the usage of the Elastic Load Balancer. This service is provided by CloudStack using NetScaler device. This load balancer balances the load among various instances in different networks within a zone and we can add/remove the instances registered with this load balancer at any time. The Elastic Load Balancer when created returns an IP address and you can also add a security group to this load balancer after it has been created to control the traffic.

The Elastic Load Balancer has many instances in the account registered to it. The ELB helps in dividing load of requests to these guest instances. Whenever there is a request on the ELB, it forwards them to each of the instance based on some algorithm which allows the instances to serve the requests easily. The ELB also does some periodic health check ups for these instances and constantly monitors the status of each of the instances.

CloudStack allows adding NetScaler device for configuring Elastic Load Balancers and it also supports protocols such as TCP, HTTP, and UDP and it also provides session persistency based on source IP address and cookie. The session persistency allows the request with same IP address and cookies to be forwarded to the instance which served its last request so that the in memory data can be used. It also allows the algorithms such as Round Robin, Least Connections, and Source to be configured when creating an ELB.

The IP address allocated to the Elastic load balancer is a public IP that was provided to the NetScaler appliance when it was registered. After the ELB is created, the account owner adds instances to the ELB among which the load will be distributed. These instances can be in running or in stopped state but there must be a valid Instance ID. The ELB can also be deleted by the Account owner at any time even when there are instances running registered to it.

Both the services of ELB and EIP are provided by CloudStack using the NetScaler device and must have at least one public VLAN provisioned in CloudStack from where the public IP addresses are assigned to them. This is essential for communication between CloudStack and NetScaler.

Network Address Translation

If you want to allow any of the guest instances to have access to the Internet, CloudStack allows it to be done using NAT. The static NAT rule is created, which maps a static Public IP address to the private IP address of a VM and all the Internet traffic is allowed into the VM considering the firewall ports are configured to allow them.

To enable or disable static NAT, follow these steps:

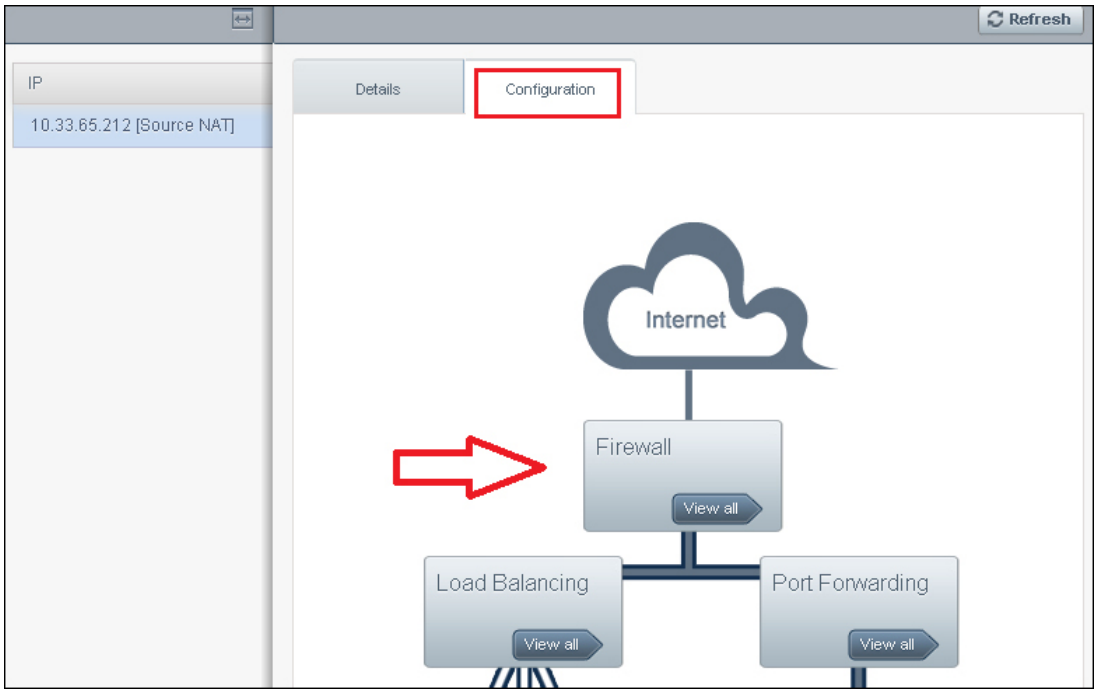
1. Click on the **Network** tab on the left pane.
2. Click on the name of the guest network in which you want the NAT to be configured.
3. Click **View IP Address**.
4. Click on the IP address that you want to enable NAT on.
5. Click on the **Static NAT** button to enable or disable. When you enable it a dialogue box appears which asks you to add instances/ destination guest machines.

By default, the firewall doesn't allow any incoming traffic to the public IP address and all the outgoing traffic from the guests are allowed via the NAT. CloudStack allows you to configure firewall to allow income traffic. Users are allowed to set up firewall rules or port forwarding rules.

Firewall cannot be configured for the Elastic IP address but one can use security groups define the ingress and egress rules in case of Elastic IP addresses.

To create a Firewall rule, you will have to set the global configuration parameter `firewall.rule.ui.enabled` to true. This setting will show the **Firewall** tab on the management server UI. Follow these steps:

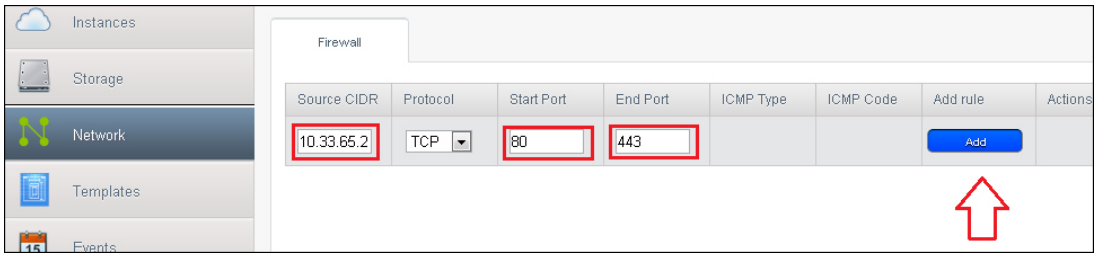
1. Click on the **Network** tab on the left pane.
2. Click on the name of the network where you want the configuration to be done.
3. Click on **View IP Address**.
4. Click on the **Configuration** tab which will provide the screen shown as follows:



The fields should be filled as per the requirements and the information given as follows:

- **Source CIDR**: This is an optional field and specifies the IP address range from which the traffic should be allowed.
- **Protocol**: This specifies the communication protocol to be used on the opened port(s).
- **Start Port/ End Port**: The ports which should be opened on the firewall. In case you want to open only one port, use the same port number on both fields.
- **ICMP Type & ICMP Code**: This field appears if you have specified the ICMP protocol. Provide the type and the code of the ICMP header that should be allowed in the firewall rule.

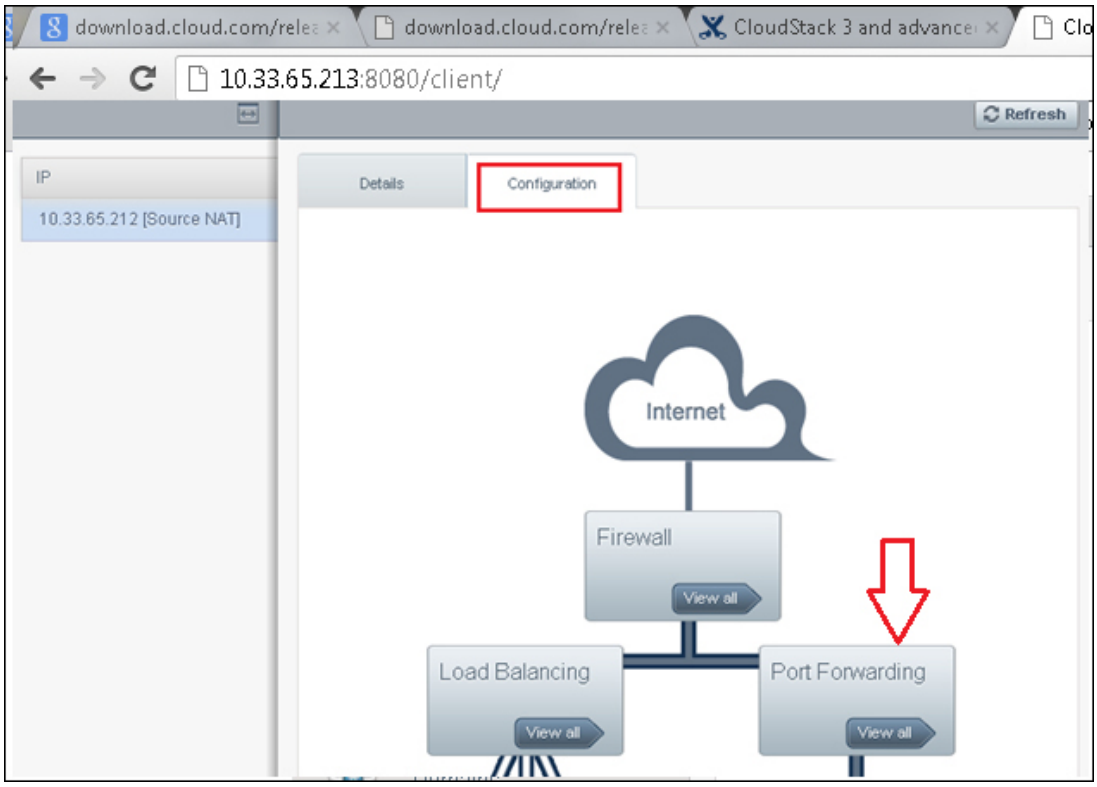
5. Click on **Add** to add this rule, as shown in the following diagram:



CloudStack also allows port forwarding feature. Using port forwarding, users can redirect the traffic to some specific port on the user's VMs. Users can open some range of ports on which the incoming traffic is allowed on the public IP address, then set port forwarding rules to redirect traffic on each of these individual ports to some specific port on the guests' private IP. The port forwarding service allows defining a set of rules that define the port forwarding policy. After you have created such service, it can be applied to one or more guest machines and there are multiple port forwarding services applied to a guest VM. In case the guest is one more than one network then the port forwarding rule must be applied to the default network otherwise it won't work.

In order to set up port forwarding, follow these steps:

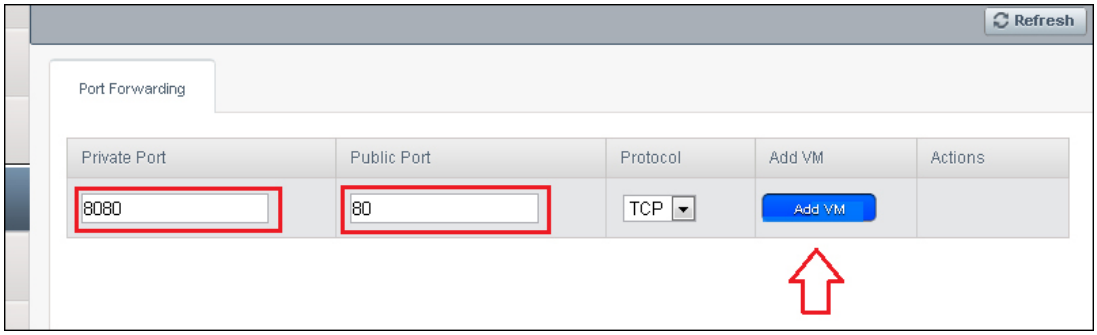
1. After you have added a Public IP address range to a zone in CloudStack, and have provisioned one or more instances in that zone.
2. Click on the **Network** tab in the left pane.
3. Click on the **Guest Network** that the guest VMs are running on.
4. Click on **View IP Addresses**.
5. Click on the IP address from the list, you can either chose an existing one or acquire a new one.
6. In the port forwarding node of the diagram, click on **View All**, the following screen appears.



7. Fill in the fields as per the information given as follows:

    - **Public Port**: This is the port on which if the Public traffic receives the traffic should forward to the private port.
    - **Private Port**: This is the port of the instance where the traffic on the public IP should be forwarded to.
    - **Protocol**: The protocol to be used between the two public IP and the private IP.

8. After entering the details, click on **Add VM** and chose the name of the guests to which this rule should be applied.



Virtual Private Network

A VPN is yet another feature that can be added to a network service offering and provided to users as a service. Using VPN, users can create virtual private networks (VPNs) to access their guest instances if it belong to a network created from a network offering having the VPN feature enabled. CloudStack provides a Layer 2 Tunneling Protocol-over-IPSec-based VPN service using the virtual router, thus enabling users to access their guest machines over the internet while also maintaining a high level of security.

As we have seen, each network has its own virtual router so the VPN service cannot be shared across networks. CloudStack allows clients such as Windows, Mac OS X, and iOS to connect to the guest instances. CloudStack uses a separate database table instead of using its own accounts' database and allows the creation and management of users for the VPN access. The account owner is responsible for providing VPN access to the users.

There can be two scenarios when a VPN is configured, they can be either of the following:

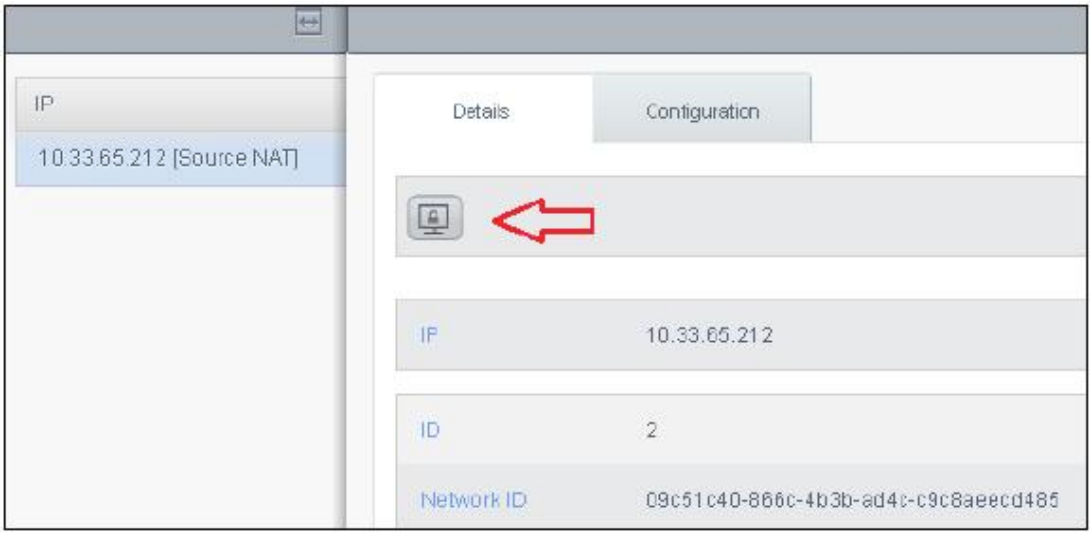
- Road warrior/remote access: This type of VPN allows connecting users from a home or office network securely to a private network in cloud. In this case the IP address of the client that is connecting is dynamic, and is changing at some time so it cannot be preconfigured on the VPN server.
- Site-to-Site VPN connection: This type of VPN is used when two private subnets are connected to each other securely over the public Internet using a VPN tunnel. Two gateways are used in this scenario. One is at the cloud user's subnet which is connected to the network in the cloud and other is at the user's site. The IP address of the gateway is preconfigured on the VPN server in the cloud.

  CloudStack allows you to configure VPN but before that you must set some configuration parameter in the global settings like:

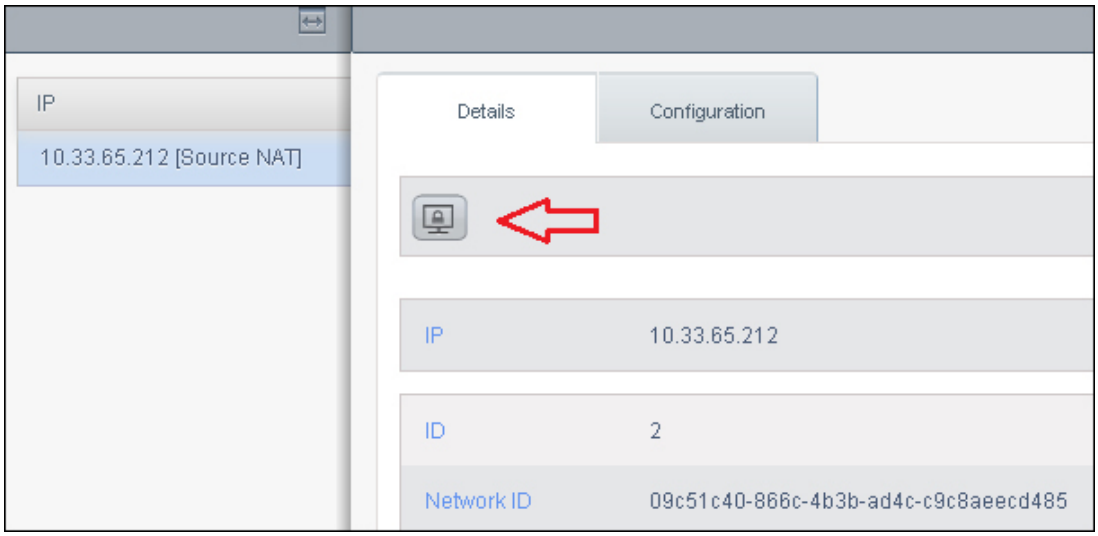
- `remote.access.vpn.client.ip.range`: Enter the range of the IP addresses that are to be allocated to remote access VPN clients. The first IP address in this range will be used by the VPN server.
- `remote.access.vpn.psk.length`: This defines the length of the IPSec key.
- `remote.access.vpn.user.limit`: This parameter defines the number of VPN users per account.

Now, to enable VPN for a particular network in CloudStack, the network must be created using a network offering with VPN feature. Follow these steps:

1. Go to **Network** tab in the left pane.
2. Click on the name of the network that you want to enable VPN access on.
3. Click on **View IP Addresses**.
4. Click on one of the IP address name which will be allowed for VPN access.
5. Click **Enable VPN** button.



6. This action will enable the VPN and return you the IPSec key in a popup window. Take a note of this key and the IP address. It will also list one or more users' credentials, if there are no users then you must add at least one new user.



CloudStack VPN access settings vary depending on the OS that you are using on the client side. Before they configure the settings, the users must make sure that the VPN is not the default route. The users connect to the VPN using the IP address and the pre-shared IPSec key. You will need user's credential to connect to the VPN through any client.

CloudStack provides a modular set of interfaces with a configuration management system that is dynamic where one can always modify the basic set of configurations to create and manage guest networks. Administrators can configure networks using different network offerings with various sets of network services as well as create private guest networks. These networks can vary in their scope—they can be made private to an account or a domain or can be made public making it visible to the complete zone.

CloudStack networking components

Now that we have seen all the services that can be provided to users using all the features, let's talk about the components that make the CloudStack networking work. CloudStack's APIs are designed in modules and the configuration management system of CloudStack is also very dynamic which can be molded by users to provide their own kind of management instead of using CloudStack's traditional way to guest network management. The components discussed below are basically interfaces which can be implemented by users to extend the CloudStack network services.

NetworkGuru

Whenever a new guest network is first created, it exists as a database entry prior to its first initialization. After the first initialization, the port group entry is instantiated onto the hosts. This component comes into play when the guest network is defined. The NetworkGuru considers all the parameters like CIDR, gateway or VLAN that are provided by the users for the new guest network. When the first instance is created in this network, the guest network is actually created with the help of NetworkGuru. The NetworkGuru calls the `implement()` method which acquires the resources from the physical network and allocates them to the guest network which were configured during the creation.

Every time a virtual machine starts in a guest network, it calls the `reserve()` method to let the NetworkGuru know the resources it needs to start and in the expunge phase, the `release()` method is called to inform the NetworkGuru for the resources that are being released by the virtual machine and when all the virtual machines in a guest network are stopped, the garbage collector is ran to inform the NetworkGuru using the `shutdown()` method to release all the

resources that have been allocated to this guest network—the network is set to virtual state without any physical resources being allocated to it. The `trash()` method is used to inform the NetworkGuru when the guest network is deleted. The NetworkGuru is basically responsible for designing the virtual networks and the management of IP addresses.

All these Gurus that are declared in `components.xml` and are loaded when the CloudStack starts, except the exclusive ones. The Gurus themselves take care that they do not clash with other NetworkGurus. So it is the duty of the CloudStack administrator to configure `components.xml` to avoid any type of collisions while defining the extra plugins or modules. Like the NetworkGuru for providing VLAN based guest networking cannot work parallel with the NetworkGuru managing the L2-in-L3 networks.

Network element

The network elements represent the basic components that make up the CloudStack network. These components are the most basic components that provide all kinds of networking services and also provide the infrastructure for building the virtual networks. The network elements include CloudStack virtual routers, external load balancers that can be any third-party device such as NetScaler or SRX or external DHCP servers or any Open vSwitch. The network element is implemented as an interface which has the methods for different operations. For example, the `implement()` method configures a specified network on a network element. An instance of the network element is verified to be ready by the `ready()` method. The `prepare()` method is used for preparing the network element to add a new network interface to the network which provides basic connectivity. The service provider which is associated with the current instance of network element is provided by the `getProvider()` method.

Network managers

All resources that are managed by the network elements are managed by network managers. They are loaded at the CloudStack's startup using the cloud configuration manager.

Resources

CloudStack defines many resource classes which abstract the physical or virtual resources that are being managed by the CloudStack. These resource classes can be hypervisors, load balancers, or storage devices.

CloudStack networking flows

The main components of CloudStack networks have been described above; let's see how these network flows occur when we create a new network and when we create a new guest instance in that network. There are several processes that are invoked at various steps during the flow. A brief overview of the steps followed is as follows:

1. When a new network is created, first the network offering from which the network has been created is validated.
2. The account that is creating the new network has the administrative rights for the shared networks.
3. The IP address and the CIDR information provided for the new network is validated by checking the user's CIDR limit and the overlapping of any other CIDR.
4. A new network model object is created with all the properties of CIDR, gateway, broadcast domain type and broadcast URI.
5. Call all the NetworkGurus and the design method is invoked.

   The new network created is in virtual state until a guest instance is created in that and when a new VM is created in this new network, the following series of events takes place.

6. The `prepare()` method is invoked when the VM starts up.

7. The NICs are retrieved for the instance. For each of the network interface, a retrieve data model object is retrieved.
8. Configure the network on the virtual or physical networking resources such as configuring CIDR and Gateway.
9. The list of providers that are included in the network offering is listed and for each of the network elements in the list of providers, the `implement()` method is invoked on the instance if it is not already implemented in some other network.
10. Create the network interface profile and broadcast the Network URI.
11. The `reserve()` method is invoked on the NetworkGuru to reserve the resources.
12. IP addresses, MAC addresses. Broadcast, netmask, and gateway address is set for the network interfaces.
13. The network elements are asked for the new NIC, using the `prepare()` method on each of them.
14. The connectivity of the new NIC created is tested and make sure that it is properly connected across all the network elements in the current network offering.
15. The entries of the NIC are added to the DHCP service provider and the User Data service providers.
16. The new NIC is added to the VM profile.

Summary

This chapter covered the networking features available in CloudStack. The chapter also describes the ways in which the CloudStack administrators can use the different components (the interfaces and the classes) to create a network service other than the standard network offering of CloudStack.

We will continue to explore CloudStack and move on to the storage configuration and support in CloudStack in the next chapter.

Chapter 5. Apache CloudStack Storage

By now you should be aware that CloudStack has two major categories of storage, that is primary and secondary storage. Primary storage is used for providing disk volumes for guest VMs and is associated with a cluster of the CloudStack deployment. Secondary storage is associated with a zone, common to all the pods in the zone and is used for storing templates, ISOs and the disk volumes' snapshots.

We will now discuss the storage aspect of CloudStack, which will include the following:

- Architecture of storage in CloudStack
- Creating, modifying, and deleting various types of storage by both users and administrators
- Scaling storage as required

Primary storage in a cluster depends on the hypervisor being used in the zone. Primary storage can vary from NAS to SAN to even DAS depending upon the capabilities and requirements of the hypervisors. Primary storage can be added to CloudStack using direct attached storage, local storage, iSCSI, or NFS. CloudStack 4.0 supports using an RBD storage pool as primary storage, whereas secondary storage is accessed only by using NFS. CloudStack also supports OpenStack object storage (Swift) as secondary storage.

Primary storage is used for providing storage space in the form of volumes to Cloud VMs whereas secondary storage is a file-based storage used to store files such as snapshots, templates, and ISO images. Lets' take a walk through adding and using both types of storages in CloudStack.

Primary storage

Primary storage in CloudStack is common to a cluster and is used to provide disk volumes to the guest VMs. It means all the virtual machines on all the hosts in a cluster are provided with their own root volumes as well as other data volumes from the primary storage associated with that cluster. There can be multiple primary storages added to a cluster with different storage tags and the guest VMs are allocated disk volumes from these primary storages by CloudStack. The storage tags (discussed later) are to be used while provisioning the guest. The storage tag is matched with the tags of the primary storage added to CloudStack.

System requirements and configuration

These primary storages can be iSCSI or NFS servers or a DAS which are added to CloudStack. For storage servers to be added as primary storages to CloudStack, the server must meet some basic requirements, listed as follows:

- All the industry standards-compliant storage servers including iSCSI and NFS that are supported by the underlying hypervisor are supported by CloudStack. Some of the storage servers that are tested to work with CloudStack 4.0.x are as follows:
    - Dell EqualLogis for iSCSI
    - Network Appliances filers for NFS and iSCSI
    - Scale Computing for NFS
- CloudStack 4.0 also supports adding Ceph **Rados Block Device** (**RBD**) to be used as primary storage. This can be done by adding the RBD pool as primary storage. Using RBD as a primary storage requires the Qemu on hypervisor to be

compiled with RBD enabled and the libvrt version should be newer than 0.10 with RBD enabled. Snapshot functionality is not currently supported if you are using RBD as primary storage.

- The server is recommended to have a sufficient amount of hard disks for primary storage to support the organization's need depending upon the number of virtual machines required, and it should ideally be managed by a hardware RAID controller.
- A primary storage can only be added to a cluster after at least one host is added to it.
- The storage server to be used as primary storage should be provisioned as shared storage or else the global parameter `system.vm.local.storage.required` should be set to `true`, otherwise the guest VM will not be provisioned.
- It is recommended to use disks with higher IOPS for primary storage.
- CloudStack also supports the use of local storage for primary storage. To use this feature, the parameter `system.vm.use.local.storage` in the global settings must be set to `true` and CloudStack automatically creates a local disk storage pool on each of the host when this option is enabled.

As already discussed, the primary storage provides disk volumes to the guest VMs which includes their root volumes. When the virtual machine is created, the root volumes are automatically created from the primary storage and they are deleted when the virtual machine is expunged after termination. The extra volumes as requested by users are also provided by the primary storage. One can easily demand more volume and perform operations such as attach/detach on them dynamically. These additional volumes which are added to the guest dynamically are not deleted when the virtual machine is destroyed.

Note

If the underlying hypervisor is an Oracle VM, the guest must be in the stopped state when the volume is being attached/detached to it.

In addition, some versions of Linux do not support the dynamic addition of volumes, they require a restart after the volume has been attached so that it appears in the `/dev` drive to be mounted.

CloudStack provisions storage to the users as required, but the administrator must continuously check the status of the primary storage and must add more when it is low.

The provisioning of storage by CloudStack depends on the type of underlying hypervisor and the storage server. Due to this CloudStack can support thin and thick provisioning. In the case of thin provisioning, the size of the allocated volume is less than the requested size and it can be expanded up to the requested size as and when the demand increases, whereas, in the case of thick provisioning, the volume of the requested size is allocated at the request time.

The following are the cases when we would use the different types of hypervisors in CloudStack:

- When using XenServer, a Clustered LVM is used to store VM images on iSCSI and Fiber channel volumes and it does not support over provisioning, but, CloudStack can support thin provisioning if the storage server itself supports it.
- KVM supports Shared mountpoint storage. This means that, a local file system path (for example, `/mnt/primary1/`) is shared by each server, which is assumed to be a cluster filesystem such as OCSF2. The filesystem path must be identical across all the hosts in the cluster. It is the responsibility of the administrator to ensure that the storage is available and it is not mounted or un-mounted by CloudStack as it does in case of NFS.

- In the case of Oracle VMs, the hypervisor supports both iSCSI and NFS storage but the CloudStack administrator is responsible for setting up the iSCSI on the host. When the host is down, the mounted storage is also lost, so it must be re-mounted by the administrator whenever the host recovers.

  > **Note**
  >
  > Oracle VM may not be fully supported in CloudStack depending upon the version.

In all the other cases, the CloudStack is responsible for mounting and un-mounting the iSCSI target on the host when it discovers that there has been a connection/disconnection.

In the case of thin provisioning, there may come a time when the actual total requirement of storage of these machines exceeds the total storage available. This is termed as over provisioning. We can control the amount of over provisioning in CloudStack in case of NFS.

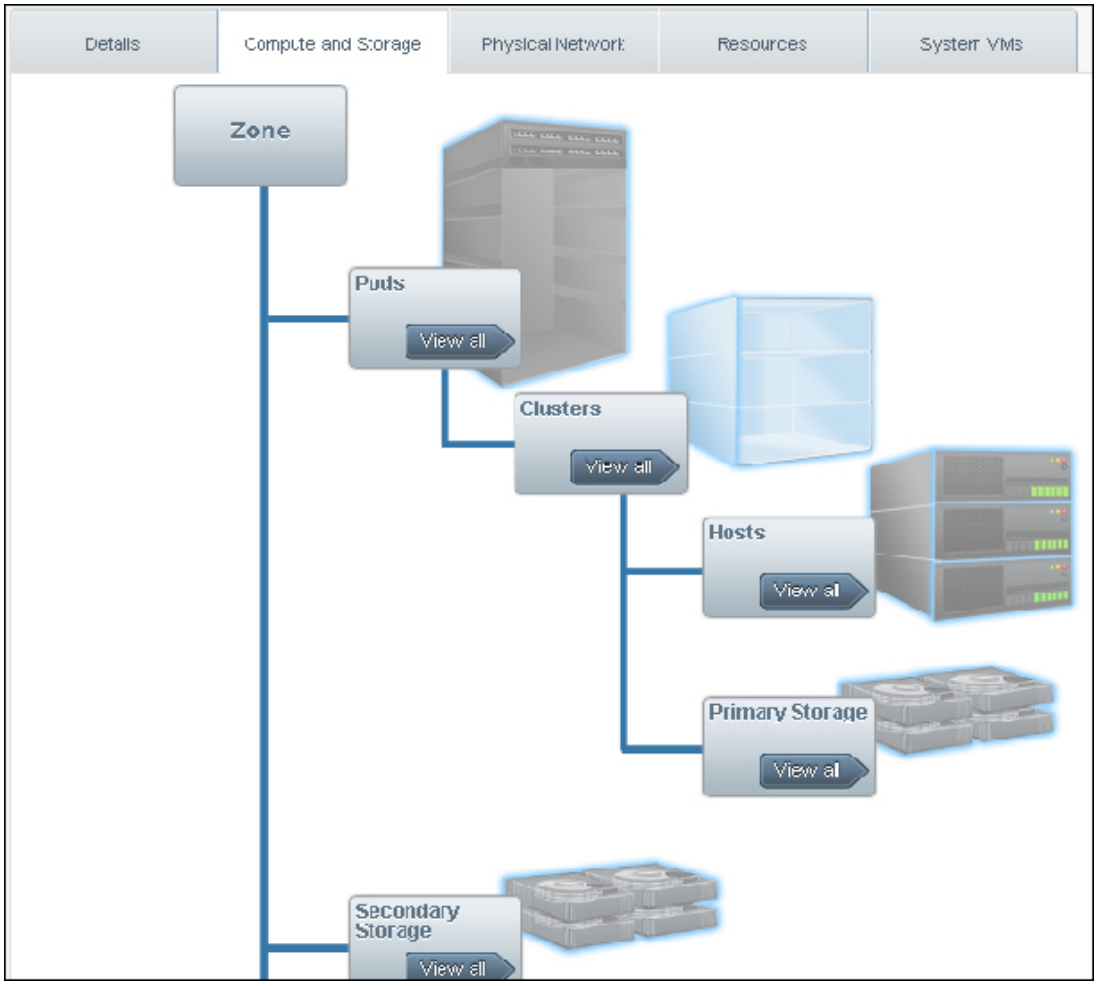
When NFS share is used for primary storage, CloudStack manages the over provisioning independent of the hypervisor. If you want to use this feature the parameter `storage.overprovisioning.factor` in the global setting must be set, this parameter controls the degree of over provisioning that is allowed.

CloudStack supports multiple storages to be added as primary storage and enables creating multiple storage pools. The administrators can add multiple NFS shares as primary storage or can add some NFS shares with iSCSI LUNs and add more when the first has reached its capacity.

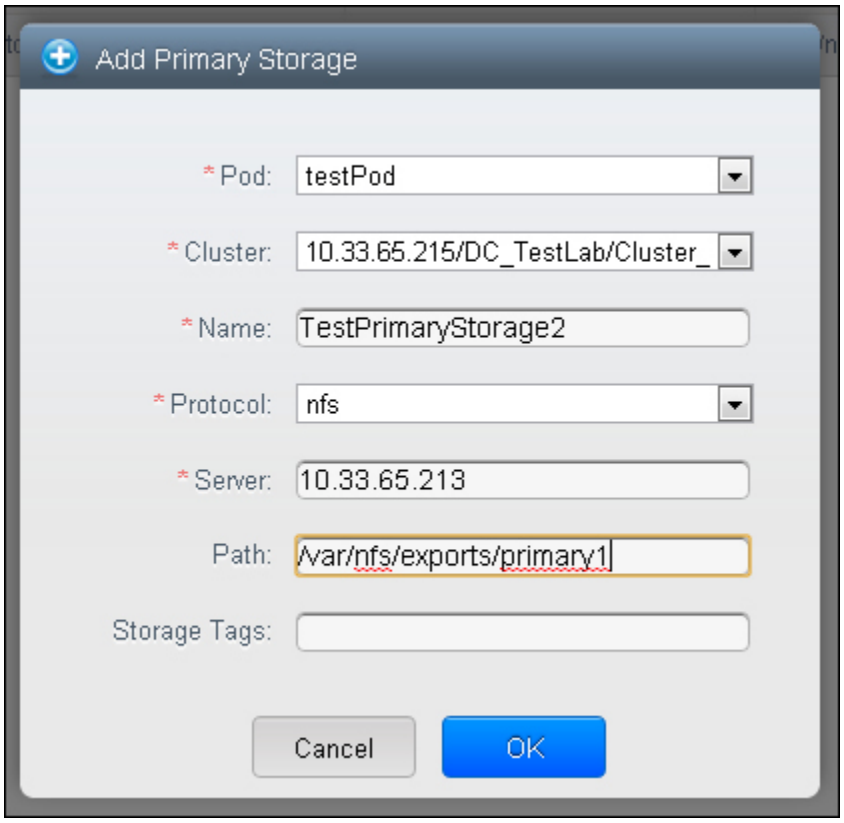
Adding a primary storage

CloudStack allows adding the first primary storage when creating a zone and you can add more after the zone has been created. To add more primary storage, use the following steps:

1. Go to the **Infrastructure** tab on the left panel.
2. Click on the zone you want to add the primary storage to.
3. Go to the **Compute and Storage** tab of the zone and click on the **Primary Storage** icon associated to the cluster in which this primary storage should be added. The screen is displayed as follows:



4. After clicking on the **Primary Storage** button, click on the **Add Primary Storage** button. A pop up is displayed, allowing you to add the details of the new storage.



The fields to be entered are as follows:

- **POD**: In this field you can select the required POD.
- **Cluster**: This is the cluster to which the Primary storage is added within that POD.
- **Name**: This is the name you want to give this primary storage.
- **Protocol**: Here you can select the protocol for the primary storage from the drop-down list of iSCSI or NFS. This can also be shared in case of KVM hypervisors.
- **Server**: This provides the primary storage server.
- **Path**: This provides the path of the storage.
- **Storage Tags**: This is the tag that you want to associate with this particular storage.

When the primary storage has been added to the CloudStack, the description page appears on clicking the primary storage name, there are some actions that you can perform on them. One can place

the primary storage in maintenance mode. Placing a primary storage in maintenance mode is recommended if you need to perform maintenance on that storage, for example replacing hardware in it.

When a primary storage is placed in maintenance mode, it first stops all the guest VMs which are provisioned on that device and then it stops the guests which have any extra volume provisioned from this primary storage. In case, the host and VMs are marked as Highly Available, so the VMs residing on the host will be migrated to another host which is not marked in maintenance mode ensuring that they remain available. The other host where the VM is migrated must be in the same cluster in case of VMware hypervisor, or the same pool in case of XenServer.

After completion of maintenance the administrator can bring the storage back online by cancelling the maintenance mode. CloudStack brings back all the guests that were switched off at the time the primary storage went into maintenance mode.

Storage tags

Storage tags can be considered to be similar to the VMware concept of profiles.

When we add storage to CloudStack, we can also specify tags for the different storage servers. These tags are simple names given by administrators to the storage, which can be used to provide extra information about the storage.

These tags are useful for categorizing storage. These tags are matched against the tags defined in service offerings and disk offerings. All the tags that are defined in service and disk offerings must be present in the primary storage before any root or data volumes are provisioned by CloudStack.

When a user requests a volume based on a disk offering, the volume is provisioned from the primary storage which has the same tag as defined in the offering. It is recommended that you use the same set

of tags in all the clusters of a POD to avoid confusion. For example, you can tag "high-disk-volumes" to be used as a root device in a service offering and add a suitable primary storage with the same tag.

Secondary storage

Apart from the primary storage, CloudStack has another category of storage known as secondary storage. This storage is associated with a zone, that is, all the hosts in all the clusters in all the pods of a zone have access to the secondary storage of that zone. The secondary storage is used to store templates, ISO images, and snapshots of volumes. The resources stored in the secondary storage are available to all the hosts in the zone.

CloudStack allows NFS server or OpenStack object storage (Swift) to be added as secondary storage. For storage to be added as a secondary storage, it is important that the storage server meets the following requirements:

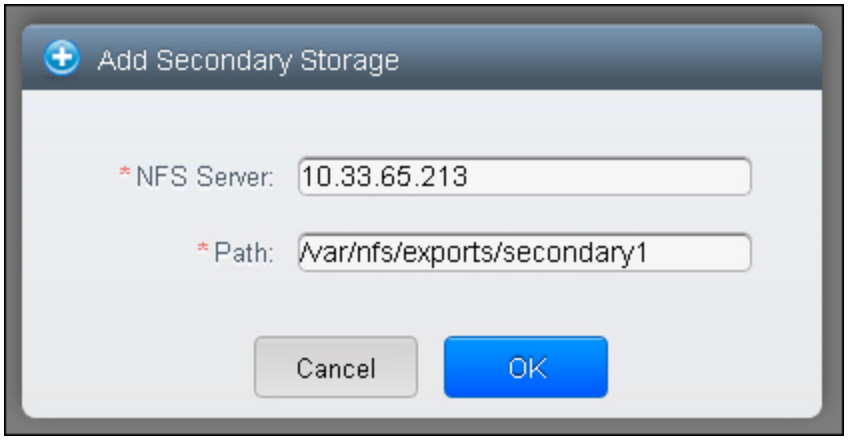
- It must be an NFS storage appliance or a Linux NFS server.
- Optionally OpenStack object storage (Swift).
- The required capacity depends on your organization's needs.
- This secondary storage device must be located in the same zone as the guest VM it serves and must be available to all the guests in that zone. There can be multiple secondary storages in a zone.
- It is highly recommended to have storage with larger drives and lower IOPs than the primary storage because it has a high read-write ratio.

Adding a secondary storage

When a new zone of any type is created, CloudStack allows adding the first secondary storage and more secondary storage can be added after the zone has been added. CloudStack takes some time to initialize the device depending upon the network speed.

Once the system is initialized, the secondary storage is ready to use. To add a secondary storage in CloudStack, follow these steps:

1. Click on the **Infrastructure** tab in the left panel.
2. Click on the **View All Zones** option and click on the zone you want to add the secondary storage to.
3. Click on the **Compute and Storage** tab of the zone.
4. Click on the **Secondary Storage** button for the secondary storage list.
5. Click on the **Add Secondary Storage** button on the left-top corner. A window will appear, as shown in the following screenshot:



In this window, fill in the following fields:

- **NFS Server**: The NFS server which is to be added as secondary storage
- **Path**: The path to the storage on this server

When a secondary storage is added to a zone, CloudStack creates a secondary storage VM based on the system VM template to manage this secondary storage. This secondary storage VM is created on a host and the VM carries out variety of tasks in the background such as downloading a new template to a zone, copying templates between zones, and performing snapshot backups. In fact, all the submissions to the secondary storage go through the secondary storage VM. CloudStack uses a separate network interface for all these types of traffic such as templates and snapshots. It is recommended to use a high bandwidth network for the storage traffic, this allows fast downloading and snapshot copying.

Changing secondary storage IP address

After the secondary storage has been added to the CloudStack, the administrator can also change the secondary storage or its IP address without any loss of data, but the proper procedure must be followed for this. After changing the IP address of the secondary storage, the administrator must also change it in the database to ensure the proper functioning of CloudStack.

To change the database entry of the secondary storage, log in to the database server and execute the following commands:

```
mysql –u root –p password
mysql> Use cloud;
mysql> SELECT ID FROM HOST WHERE
TYPE='SecondaryStorage';
mysql> UPDATE HOST_DETAILS SET VALUE='<NEW_VALUE>'
WHERE HOST_ID=HOSTID AND NAME='orig.url';
mysql> UPDATE HOST SET NAME = '<NEW_VALUE>' where
type = 'SecondaryStorage';
mysql> UPDATE HOST SET URL='<NEW_IP_ADDRESS>'WHERE
TYPE = 'SecondaryStorage';
mysql> UPDATE HOST SET GUID='NEW_VALUE' WHERE TYPE
='' 'SecondaryStorage';
```

This is done to update the new IP address of the secondary storage in the CloudStack database. The database "cloud" is the database that is used by CloudStack for storing the value of the resources. The steps are also depicted in the following screenshot:



Changing the secondary storage

After making these changes, the administrator must log in to the UI and stop and then start (not reboot) the secondary storage VM. The administrator can also change the secondary storage when he wants to by using the following steps:

1. First, stop all the running management servers and wait for some time, say around 30 minutes, so that all the writes to the secondary storage are complete.
2. After waiting for a sufficient amount of time, copy all the files to the new secondary storage from the old one.
3. Next, use the earlier steps to change the secondary storage IP address if required.
4. After completing these steps, restart the management server.

Using OpenStack object storage (Swift)

CloudStack also allows OpenStack object storage (Swift) to be added as a secondary storage. In cases where Swift is being used for secondary storage, it is enabled for the entire CloudStack, and then after Swift is configured, NFS secondary storage is set up for each zone. All traffic; such as templates and other secondary storage data passes through this NFS storage before being forwarded to Swift. In a way the NFS server acts as a staging area for all secondary storage traffic.

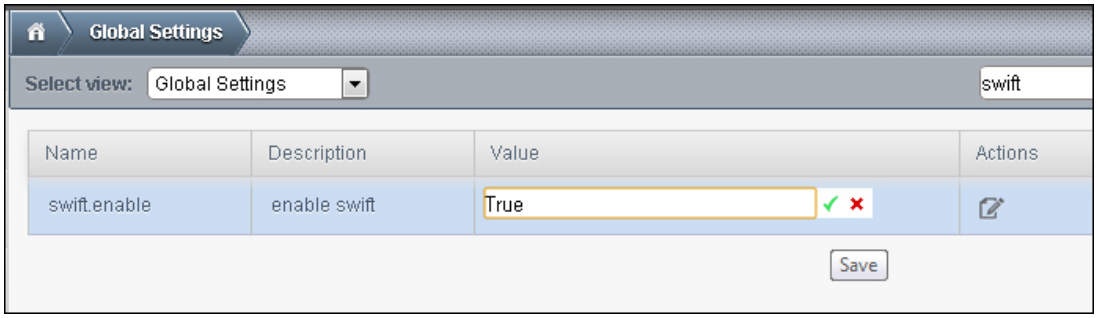
This Swift storage is an object storage which spans across the cloud providing data to all zones. Each of the objects is stored in the Swift container which can be pulled by any secondary storage from anywhere in the cloud.

When an NFS share is used as the secondary storage, you might need to copy templates and snapshots from one zone to another.

Using Swift avoids this situation because all the templates and snapshots are stored in Swift, which is accessible from all over the cloud.

Before adding any zone with NFS share as secondary storage, Swift must be configured in the cloud. To do this, perform the following steps:

1. Change the parameter **swift.enable** to true in the global settings page and then restart the management server for the changes to take effect. Next, log in again.



2. Go to the **Infrastructure** tab on the left and click on **Add a zone.**
3. To use Swift, click on **enable swift** and provide the following details as per the screen.
4. Add the details of the secondary storage.

    - **URL**: This is the address of the Swift storage
    - **Account**: This is the Swift account
    - **Key**: This is the Swift key

Swift is deployed as a part of the storage foundation in CloudStack and apart from secondary storage it is the underlying core engine of the storage infrastructure in CloudStack. CloudStack allows Swift to span multiple zones which can be in different geographical locations or in a different network. Swift allows an object of a maximum 5 GB in size to be uploaded and downloaded. It can also support unlimited sizes of the object using segmentation where the object is split into segments using the Swift tool so that parallel uploads can be done.

Volumes

All the drives (root or extra data disk) connected to guest VMs are provided by the primary storage as volumes. Volumes can be requested by users from the interfaces and can be attached to any guest dynamically (except when you are using Oracle VM).

The volumes are attached to any guest with the same type of hypervisor and may not be attached to any other guest of any other hypervisor type in any other cluster because different hypervisors have different disk image formats. These volumes are the units of storage in CloudStack that are available to guests. These volumes can be used as root drives by guests as "/" in the filesystem, which is the boot device or it can be optionally mounted as an additional disk volume such as "D:" or "/opt".

The users can request additional volumes based on the disk offerings created by the administrators and attach or detach them dynamically. Users can also create templates from these volumes. All these volumes are provided by the primary storage in the cluster.

When a user requests a new guest, the root volume for the guest is automatically created from the primary storages based on the service offering storage tag, or randomly if no tags are associated with the storage.

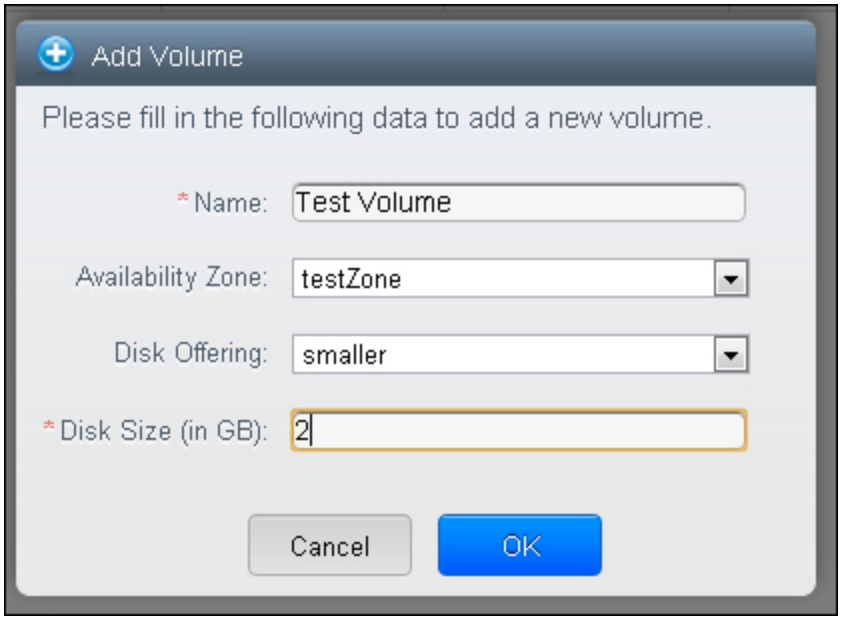
The users can perform variety of operations on these volumes, which are discussed in the following sections.

Creating a new volume

In order to add more volumes to a guest, the user must create volume based on the disk offerings defined by the administrator. New volumes can be requested and attached to any guest by both administrators and users.

When a new volume is created in CloudStack, it is added as an entity only and no physical storage is allocated to it until it is attached to any guest. This helps in optimizing the storage usage. In order to create a new volume, use the following steps:

1. Click on **Storage** in the left navigation pane and select volumes in the drop-down menu.
2. Click on **Add Volume** on the top-right corner. A pop-up window is opened, like the one shown in the following screenshot:



The data to be filled in the fields of this window are as follows:

- **Name**: The name you want to give to the volume.
- **Availability Zone**: This drop down lets you select the zone where you want the volume to reside. Generally it should be near the guest that is going to use it.
- **Disk Offering**: This drop down lets you select the disk offering. The offering defines the characteristic of the volume being created.
- **Disk Size**: The size you want for the volume in GB.

After you have entered these values, a new volume appears in the volume screen with the state set to **Allocated**. This volume is just an entity in CloudStack and no physical storage is allocated to it, physical storage will be allocated when you attach this volume to a guest.

A volume is created based on the disk offering selected by the user which defines the size of the volume, but it doesn't allow changing the size of the volume once it is created. However, it is possible to change the size of the disk using tool known as **VHD Resizer**, this tool allows you to import the VHD and resize the volume. The VHD Resizer can be downloaded from http://vmtoolkit.com/files/folders/converters/entry87.aspx. After you have changed the size of the VHD using the tool, follow these steps:

1. Upload the resized VHD.
2. Create a new VM instance.
3. On a Linux guest, the filesystem must be extended and the user needs to manually resize the partitions using the utilities provided by the OS.

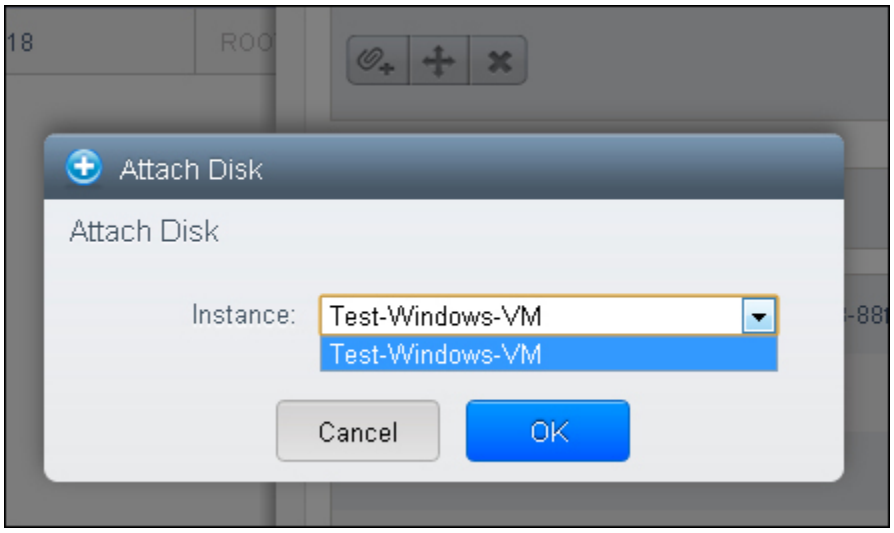
Attaching a volume to Guest VM

After a new volume is created, this volume can be attached to suitable guests, the users can also move storage volume from one guest to another by detaching the volume from the first and attaching it to another. A volume can be attached to a guest VM to provide extra storage to it. Any volumes which are not attached to any guests are free to attach to the guest VMs, it can be the newly created volume or when a volume is migrated from one storage pool to another. To attach a volume to a guest, follow these steps:

1. Navigate to the storage by clicking on **Storage** on the left pane.
2. Select **Volumes** from the drop-down menu.
3. Click on the volume name you want to attach and then click on the **Attach Disk** button.



4. A pop up is opened with the list of instances to choose from as shown in the following screenshot. Users can only see those instances which were created by them and an administrator will have more choices. The list of instances is dependent upon the cluster in which the volume is present. Choose the instance you want to attach the volume to.



Note

If the hypervisor is Oracle VM, the instance must to which we are attaching the volume must be in the stopped state.

Detaching a volume from an instance

CloudStack provides a feature to detach a volume from one guest instance and attach it to another instance in the same cluster. A volume cannot be attached to any instance until and unless the instance is of the same hypervisor type as the volume, because different hypervisors support different disk formats. If you are attaching a volume to an instance in a different cluster, it may take several minutes. In order to detach a volume from an instance, follow these steps:

1. Click on the **Storage** tab in the left panel, and choose **Volumes** from the drop-down menu.
2. Click on the name of the volume that you want to detach and click on the **Detach Disk** button.



The volume becomes available now and can be attached to any other instance of the same hypervisor type. Both the administrator and the users can perform this operation.

Deleting a volume

When a guest is destroyed, the root volumes are automatically deleted with it but the data disk volumes mounted on it are not

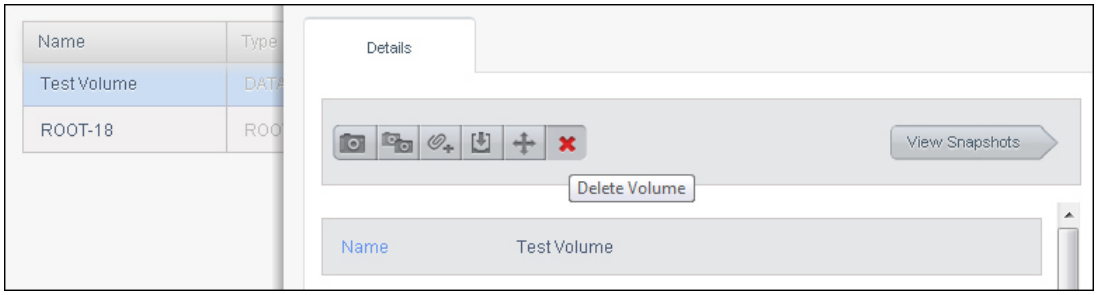
deleted. When a volume is deleted, the storage space is allocated back to the primary storage. CloudStack has a garbage collection process which permanently destroys the volumes (expunged) when they are deleted. The users can control the expunge time of the volume by setting some configuration parameters in the global settings page. These parameters are as follows:

- `expunge.delay`: This parameter determines the age of the volume in seconds before it is irreversibly expunged, that is, permanently destroyed.
- `expunge.interval`: This parameters determines the time interval between the garbage collection process runs to check for expunging the volumes. CloudStack checks the volumes that are already destroyed and then expunges them if the interval has expired.

After deletion the volume can still be recovered, but after the volume is expunged it is destroyed permanently and the entry in the database including the meta data is removed.

To delete the volume use the following steps:

1. Click on the **Storage** tab and migrate to the volumes page by selecting **Volumes** from the drop-down menu.
2. Click on the volume you want to delete. After confirmation the volume is deleted.



Snapshots

The data and state of a volume or an instance can be saved by creating a snapshot. Using snapshots can help in replicating a volume, backing up a volume, creating a template from an existing volume, and saving the state of the volume before making any other changes.

The snapshots created for each volume can be incremental, which means the new snapshot created for the volume will have only the data which has changed since the last snapshot time. The snapshots form a tree structure, where each snapshot can have one or zero parent snapshots.

CloudStack supports two types of snapshots, they are:

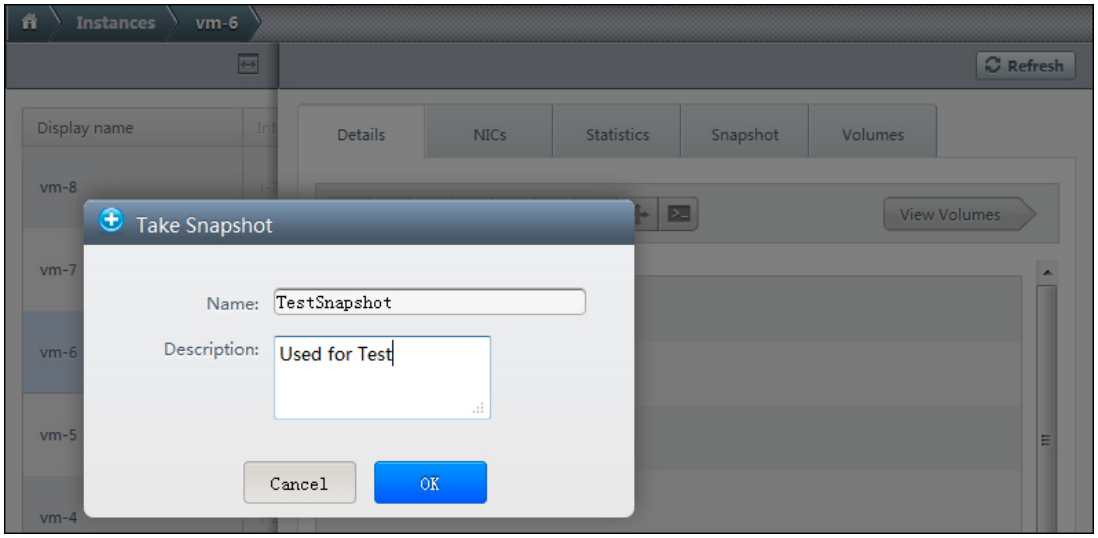
- **Disk snapshot**: In this type of snapshot, only the data on the disk is backed up. This type of snapshot is used for volume snapshots in which the volume is backed up and stored in the secondary storage. It can be created when the VM is in the running or stopped state.
- **Disk and memory snapshot**: In this type of snapshot, data on the disk and CPU/memory is backed up. This type of snapshot is for the VM snapshot in which the complete snapshot of the VM along with its data disks, CPU and memory state, is backed up and stored in the secondary storage. It can be created when the VM is in the running state.

In the case of KVM, the disk snapshot cannot be taken when the VM is running. The disk and memory snapshot is not supported in the case of XenServer community edition. If the VM is running, the choice can be provided to quiesced the filesystem and if the memory state is also needed.

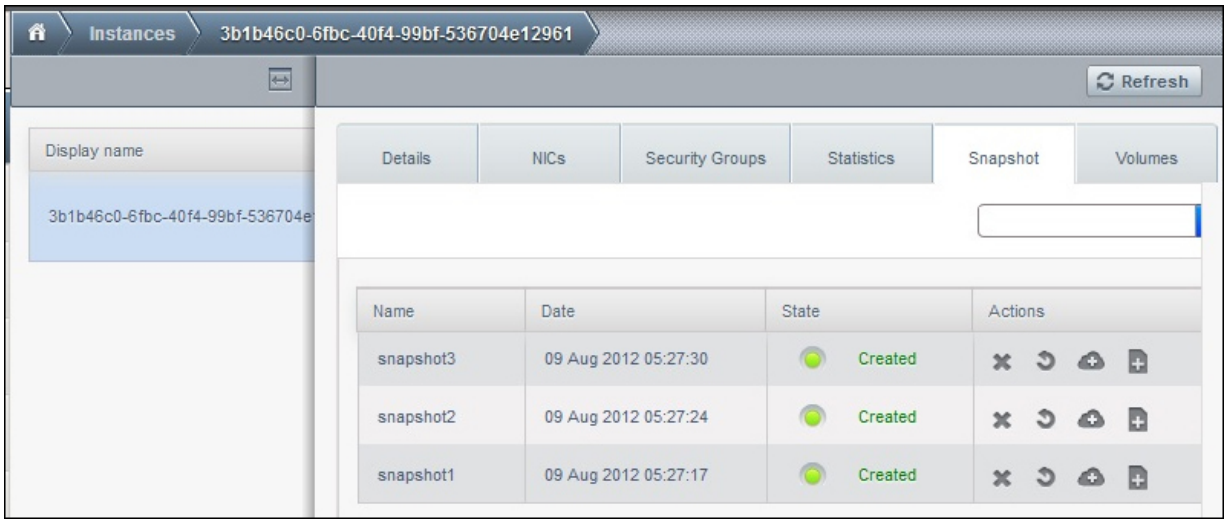
Creating a snapshot

In order to create a snapshot, the instance to which the volume is attached should be in the stopped state to avoid inconsistency of data and in the case of KVM hypervisors, the snapshot cannot be created if the VM is in the running state. Follow these steps to create a snapshot from a volume:

1. Go to the **Instances** tab in the CloudStack console.
2. Click on the VM whose snapshot you want to create.
3. Click on the **Take Snapshot** button.
4. The following screen appears, provide the information requested in the fields.



The snapshot will be created and you can view it in the **Snapshot** tab of the **Instance** tab for the instances as shown in the following screenshot:

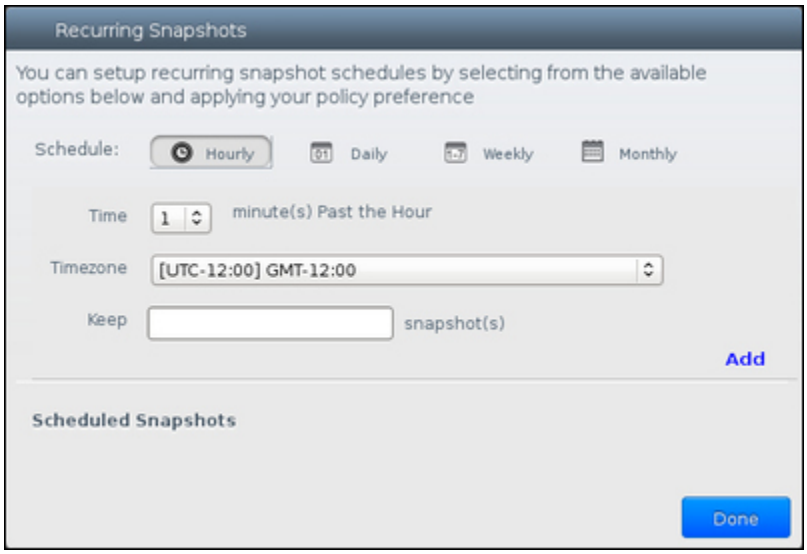


The snapshot can also be created for volumes using the same set of steps from the **Volumes** tab. These snapshots can be used to create another instance, template, or volume. The snapshot can be created from the **Instances** tab as well as from the **Storage** tab by selecting the volume for which you want to take the snapshot.

Creating recurring snapshots

CloudStack also provides the ability to program the automatic creation of a snapshot, that is, the users can program the creation of a snapshot which creates recurring snapshots of volumes as per the configuration. In order to create recurring snapshots of a volume, use the following steps:

1. Go to the **Storage** tab and the select the volume you want to program the recurring snapshot creation.
2. Click on the **Create Recurring Snapshot** button.
3. The following screen appears, which allows you to program the snapshot creation:



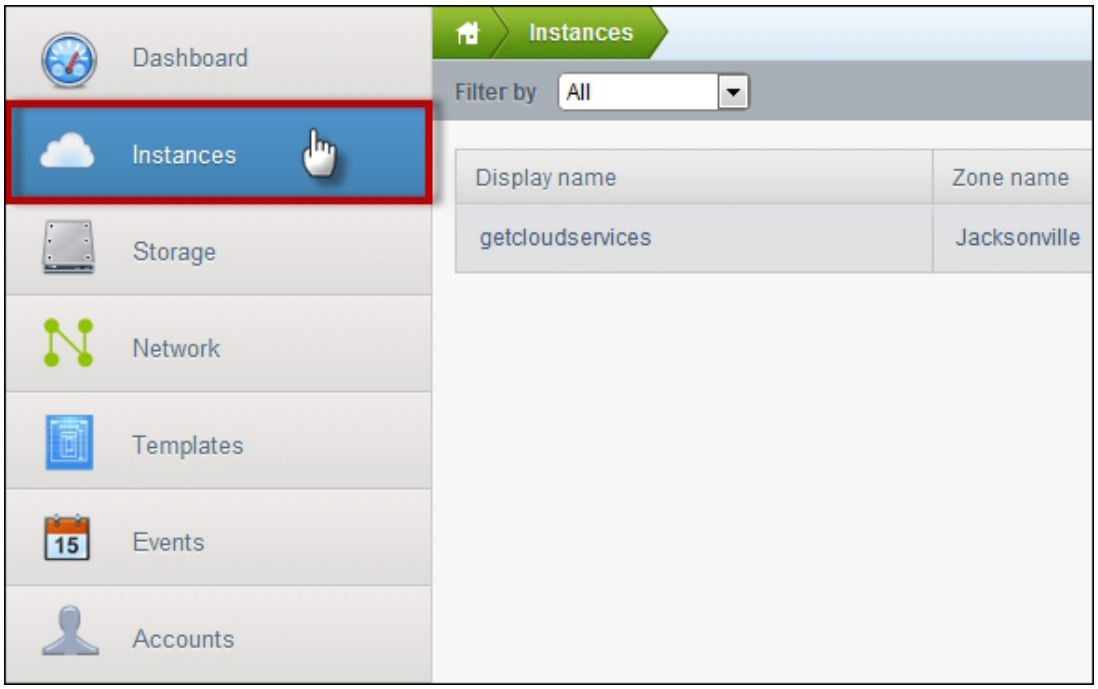
4. Schedule the snapshot creation by providing the time frame for the snapshot's creation.

In case of recurring snapshots, the users can specify the number of snapshots that should be kept, which is known as a retention period, by specifying a value in the **Keep snapshot(s)** field in the preceding window. The snapshots that are older than the retention period are deleted.

Creating a volume from a snapshot

Follow the steps listed below to create a volume from a snapshot:

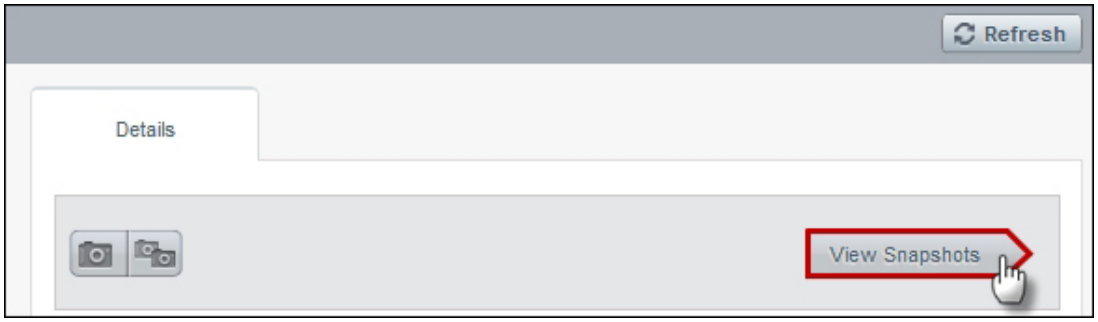
1. Go to the **Instances** tab.



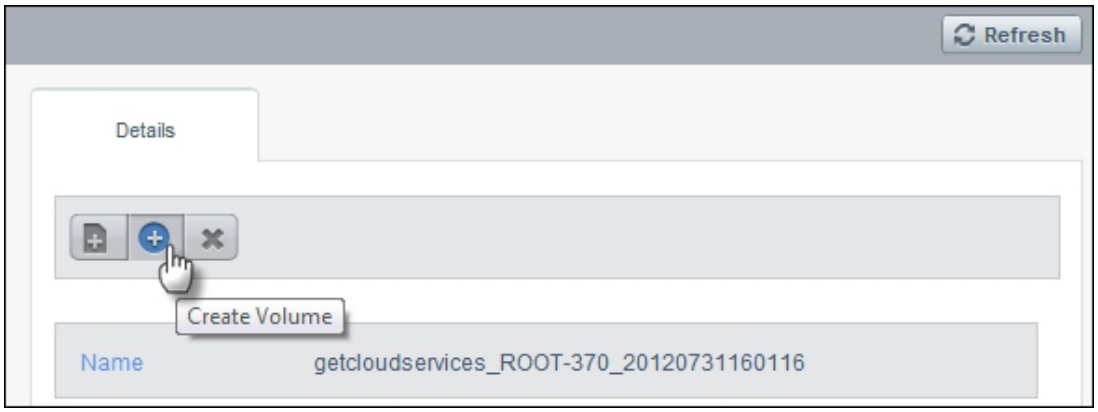
2. Click on the instance for which the snapshot is a part of and click on **View Volumes** to see all the volumes.



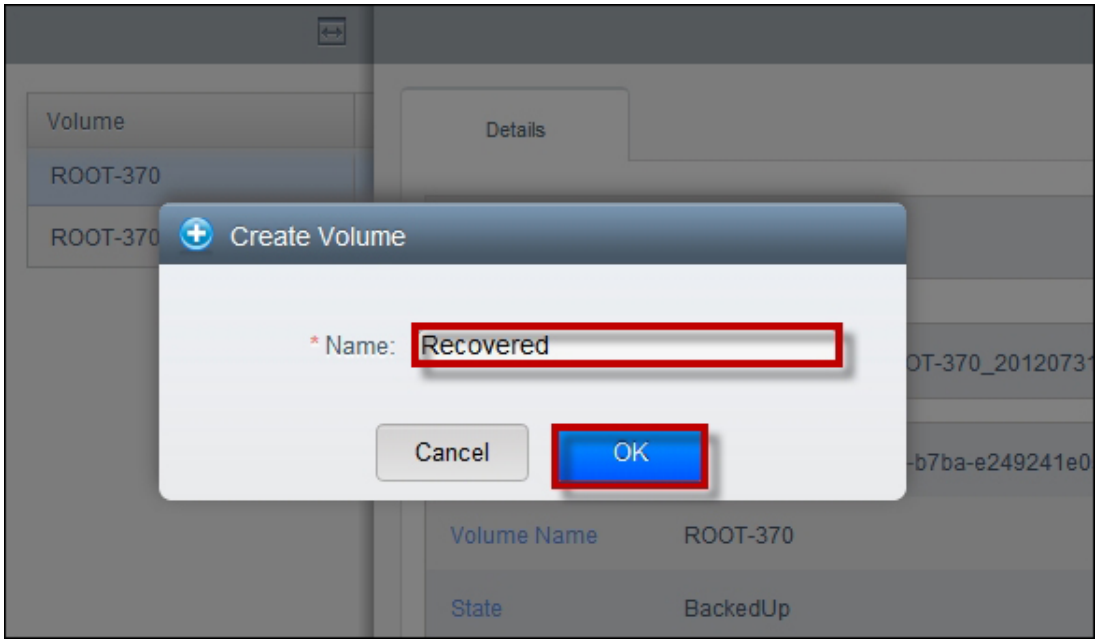
3. Click on the volume to select it, and click on **View Snapshots** to view the snapshots:



4. Select the snapshot you want to use to creating the volume and click on the **Create Volume** button, as shown in the following screenshot:



5. The following screen appears, allowing to enter the **Name** for the new volume to be created:

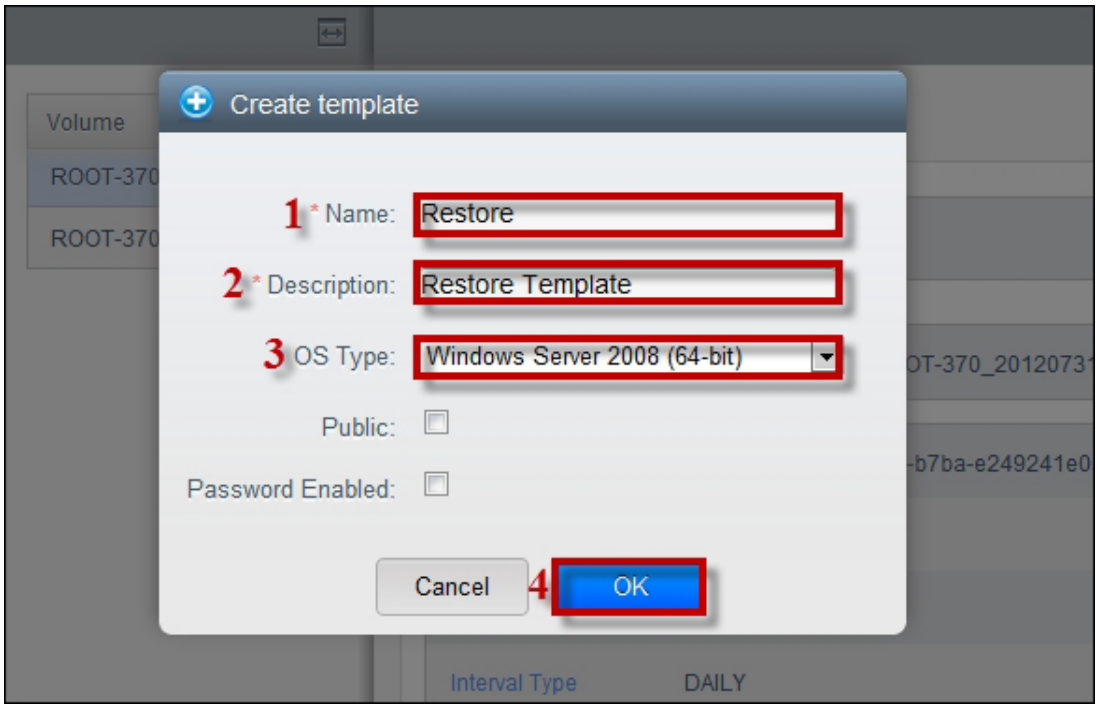


The new volume will be created from the snapshot and will contain all the data and configuration information that is in the snapshot.

Creating a template from a snapshot

To create a template from a snapshot use the following steps:

1. Go to the snapshot page as described in the earlier section for the volume of the instance that you want to create a template from and click on the **Create Template** button as shown:



2. Enter the following information:

    - **Name** and **Description**: The name and description for the new template
    - **OS Type**: This option specifies the type of operating system
    - **Public**: This parameter specifies whether this template should be publicly accessible
    - **Password Enabled**: This option specifies whether the password is enabled by CloudStack for the VM

The template created from the snapshot can be used to create new instances.

CloudStack allows the creation of temporary passwords and allows you to change the password for admin or root users in guest VMs

from the CloudStack UI. The password management feature of the template can be enabled which includes downloading and installing a script on the template. When new instances boot up from such templates, the script in these instances tries to contact the virtual router via HTTP and the virtual router sets the password for the new instance. If the script is unable to contact the virtual router, the instances boot up without setting any password.

CloudStack allows the administrator to control the snapshot feature. They can be used to control the creation and use of snapshot in CloudStack. The configuration parameters are as follows:

- `backup.snapshot.wait`: This parameter is used to define the timeout for the `BackupSnapshotCommand` command which is used to create snapshots
- `create.private.template.from.snapshot.wait`: This parameter defines the timeout for the creation of template from snapshots using the API command `CreatePrivateTemplateFromSnapshotCommand`
- `create.volume.from.snapshot.wait`: This parameter defines the timeout for creating volumes from snapshot, for the API command
- `max.account.snapshots`: This parameter defines the maximum limit for snapshots that can be created in a CloudStack account
- `max.project.snapshots`: This parameter defines the value for the maximum number of snapshots that can be in a project in CloudStack
- `snapshot.delta.max`: This parameter defines the maximum delta or difference between two snapshots creation in case of recurring snapshot creation
- `snapshot.max.daily`: This parameter defines the value for the number of snapshots that can be created daily for one volume
- `snapshot.max.hourly`: This parameter defines the value for the maximum number of snapshots that can be created in one hour for a volume

- `snapshot.max.monthly`: This parameter defines the value for the maximum number of snapshots that can be created in a month for a volume
- `snapshot.max.weekly`: This parameter defines the value for the maximum number of snapshots that can be created in a week for a volume
- `snapshot.poll.interval`: This parameter defines the value for the time interval when the CloudStack Management server polls for the snapshot to be scheduled

VM storage migration

CloudStack allows migrating a volume from one storage pool to another, but the pools must be in the same zone. The volumes can be root drives or any other data disk drives. This feature is supported only in XenServer, KVM, and VMware. This feature helps with balancing the load over different storage pools and increases the reliability of virtual machines by moving them into a different storage pool when the original pool is experiencing issues. In order to migrate a volume from one storage pool to another, use the following steps:

1. Detach the volume as described in the *Detaching a volume from an instance* section.
2. Using the CloudStack API command `migratevolume`, users can migrate the volume by passing the volume ID and the ID of the storage pool in the zone.
3. The volume status changes to `Migrating` and then `Ready` when migrated.
4. Once the volume has been migrated, it can be attached to any other instance in the same cluster.

CloudStack also allows the migration the root volume of an instance from one storage pool to another. To do this, follow these steps:

1. Detach the volume from the instance as described in the *Detaching a volume from an instance* section.
2. Stop the associated instance.

3. Use the CloudStack API command `migrateVirtualMachine` with the ID of the VM to migrate along with the destination host and destination storage pool ID. They must be in the same zone as the source.
4. The volume status goes to `Migrating` and then to `Ready`.
5. Once the status of the volume is `Ready`, restart the machine.

Summary

In this chapter we covered the use and importance of storage in CloudStack. You should now be familiar with the core concepts and the importance of storage in the cloud, the different types of storage in CloudStack, and how CloudStack provides features to support the scalability of storage in the cloud.

We can now proceed with the next chapter, which will give you more insight into the service offerings that are supported by CloudStack and how various types of users can utilize them.

Chapter 6. Service Offerings and Virtual Machines

In the previous chapters, we explained the utilization of storage in the cloud and various types of storage devices supported by CloudStack and role of these storages. In this chapter we will introduce you to the following:

- Offerings and their purpose in the cloud
- Various types of offerings supported by CloudStack
- How users can use these offerings to request resources from the cloud
- The virtual machine lifecycle

Introducing service offerings and virtual machines

Service offerings are an integral part of any cloud service. The service offerings are used by the cloud administrators to define the services that the users can request from the cloud. Based on the service offering configuration, the users are granted resources.

As an example the cloud provider can configure and define services in the areas of compute, network, storage, and common services such as DHCP, DNS, and so on. These offerings can be made available to the users and then be subscribed by the users. In CloudStack the service definition includes the compute offerings, disk offerings, network offerings and system offerings. End users request resources by selecting any of the offerings defined by the cloud provider. For example, if a user has to request a virtual machine, then he can select the template or ISO file, compute offering and disk offerings based on which the virtual machine will be created. In this chapter we are going to discuss the basic role of

service offerings, service offering configurations and how the users use it to request resources from the cloud.

The different types of service offerings in CloudStack are:

Compute offering

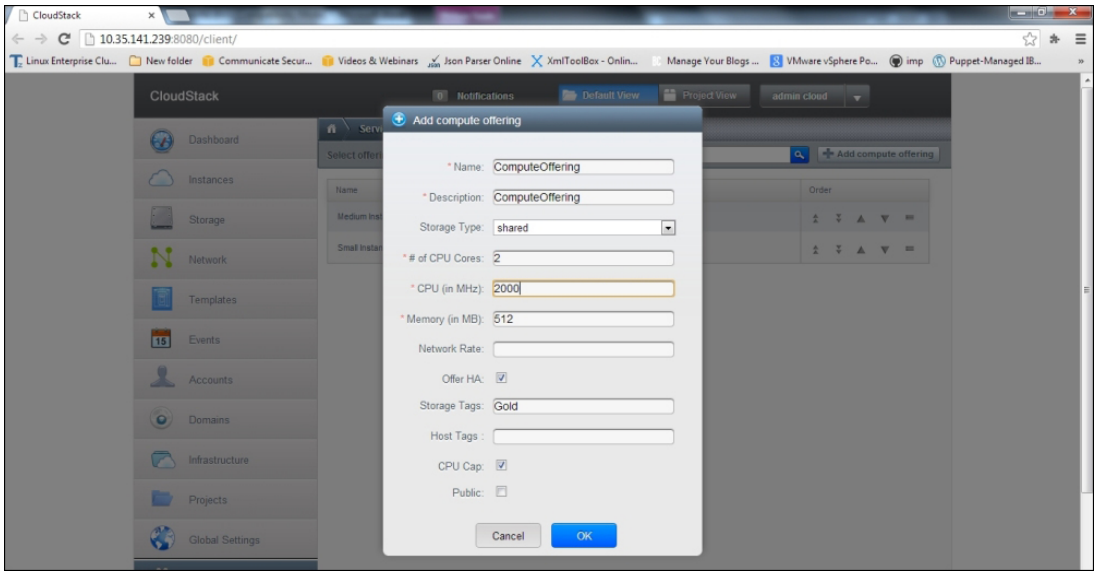
This service offering is defined for the compute resources for the guest VMs in the cloud.

In this offering, the administrators define the amount of CPU in terms of cores as well as the processing speed of a CPU in Mega-Hertz, the amount of memory, the network rate, and other resources.

When a user requests a unit from this offering, the resources mapped with the service offering are provided to him. CloudStack tries to match the logical value of MHz with the value provided in the compute offering.

The administrators can create a compute offering from the Service offering page, using the following steps:

1. Go to **Service offering**; select the **compute offerings** option from the drop-down menu.
2. Click on **Add compute offering**, you should get the following screen:



3. Fill in the fields described as follows:

- **Name**: This field gives the name of the compute offering.
- **Description**: This field gives the description of the compute offering.
- **Storage Type**: This field lets you select the storage type which is to be used in this offering. It can be either shared or local storage. When local storage is selected, the storage from the local host is selected to provide the storage for the guest created from this compute service offering.
- **# of CPU Cores**: This field specifies the number of CPU cores that should be allocated to the guest created from this offering.
- **CPU in MHz**: This field specifies the speed of CPU cores that should be provided to the guest instance. It is specified in MHz.
- **Memory (in MB)**: This field specifies the amount of memory to be allocated to the guest.
- **Network Rate**: This field specifies the network rate in Mbps.
- **Offer HA**: This field allows monitoring the guest and making it as highly available as possible. This field places the guest host on a HA-enabled host.

- **Storage Tags**: This field specifies the storage tags that are to be used while provisioning the guest. The storage tag is matched with the tags of the primary storage added to CloudStack. When a user requests a virtual machine using a service offering, the primary storage matching to the service offering is used to allocate storage to the guest.
- **Host Tags**: This field specifies the host tag which is to be associated with this service offering. This tag must match with the tag that has been used to organize the host in the infrastructure. The host with the matching tag is associated with this service offering.
- **CPU Cap**: This field is to limit the amount of CPU available to the user to an assigned value even if there is spare capacity available.
- **Public**: This field is provided for the visibility of the service offering. Public means that this offering will be available to all the domains in the Cloud. If you don't want this offering to be public, CloudStack will let you configure the scope of visibility of the offering. We can define the scope of the visibility of the offering to a subdomain. CloudStack will prompt to select one of the subdomains from the drop-down list.

The guest VMs can be tagged as **Highly Available** (**HA**). This is enabled by selecting the **Offer HA** option. CloudStack provides a global configuration parameter, `ha.tag`, to set the hosts which can be used for HA enabled guests only. If the guest is selected to **Offer HA**, it will be provisioned on the hosts dedicated for HA enabled VMs.

In case of VMware hypervisor, CloudStack uses the native HA feature provided by VMware.

In case of KVM/XenServer, CloudStack uses the storage heartbeat to detect if the HA is to be performed or not for the HA enabled guest VMs. The agent resides on the XenServer/KVM continuously writes

the time stamp on the shared storage, and in case the storage ping times out the HA job will be started for the guest VMs that are HA enabled.

The HA service is installed on the host, but the cluster is the basic unit of availability in CloudStack, so all hosts in the cluster must be enabled for cluster uniformity.

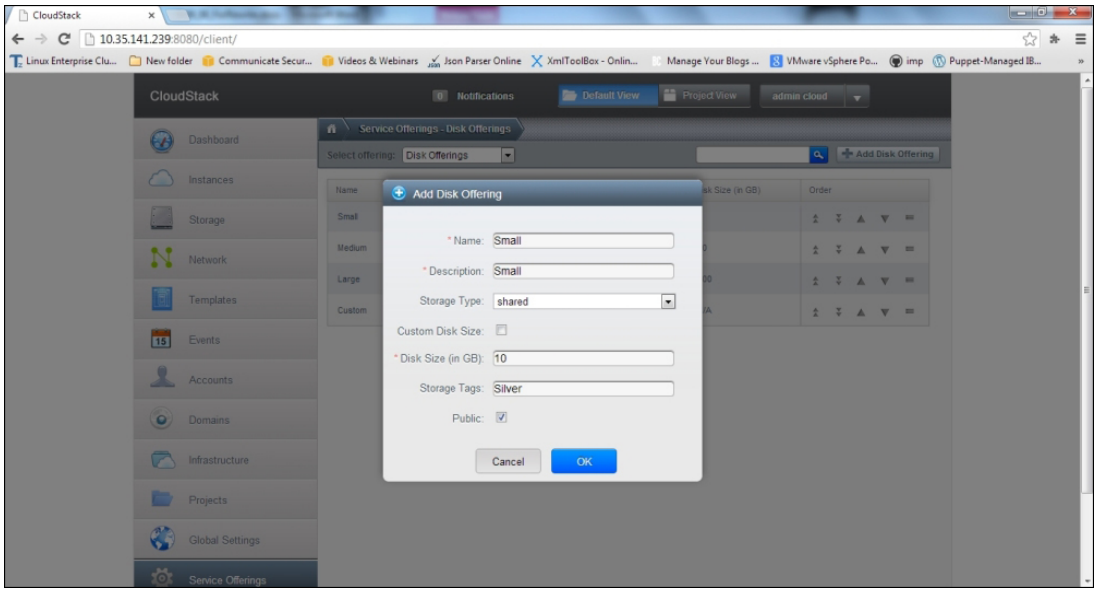
Disk offering

The other type of service offering is related to the disk or the storage resource. The disk offering is used by users to request storage from the cloud. The administrator defines different types of disk offerings based on the size and type of storage that the users can request from the cloud.

When a user requests storage using one of the disk offerings, the resources mapped to that offering are allocated for consumption.

The steps for defining a disk offering are discussed as follows:

1. Go to **Service offering** and select **disk offering** in the drop-down menu.
2. Click on **Add Disk Offering** and you should see the following screen:



3. Fill in the following fields:

- **Name** and **Description**: This field specifies the name and description of the disk offering that you are creating.
- **Custom Disk Size**: This field specifies that the users can define a custom disk size of their own choice, or else the administrators define a fixed size of disk with this service offering.
- **Disk Size (in GB)**: This field specifies the size of the disk to be associated with this service offering.
- **Storage Tags**: This field specifies the storage tags which are to be associated with this service offering. These storage tags are used by CloudStack to match with the storage tag associated with the storage added to CloudStack. When a user requests a volume using this disk offering, the tags are matched with the primary storage in the CloudStack and a disk with the configuration in the offering is provided to the user.
- **Public**: This option is for the permission associated with this disk offering.

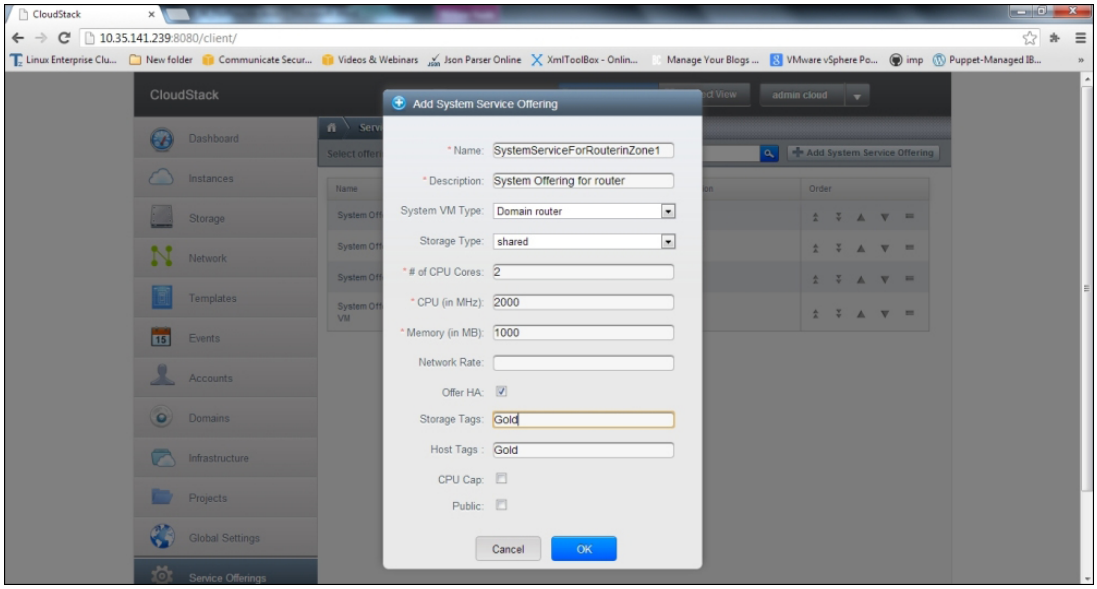
System service offering

The system service offerings are used for creating the CloudStack system VMs such as virtual router, console proxies, and other system virtual machines. The system service offering is used to create virtual machines that are used by CloudStack to manage and provide services to the cloud. These system service offerings are used while creating a new network offering (to be discussed next). When the administrator creates a new network offering that has services such as DHCP, DNS, and security group, these services are provided by the system VMs of type Domain router in the CloudStack which can be created using one of the system service offerings.

While defining the system service offering, we provide the information about System VM Type. CloudStack provides some default system service offerings. System service offering provides a way to create custom system offerings and use them for creating system VMs.

In order to create a new system service offering, follow the given steps:

1. Go to the service offering page and select system offering from the drop down menu.
2. Click on **Add System Service Offering**, and you should get the following screen:



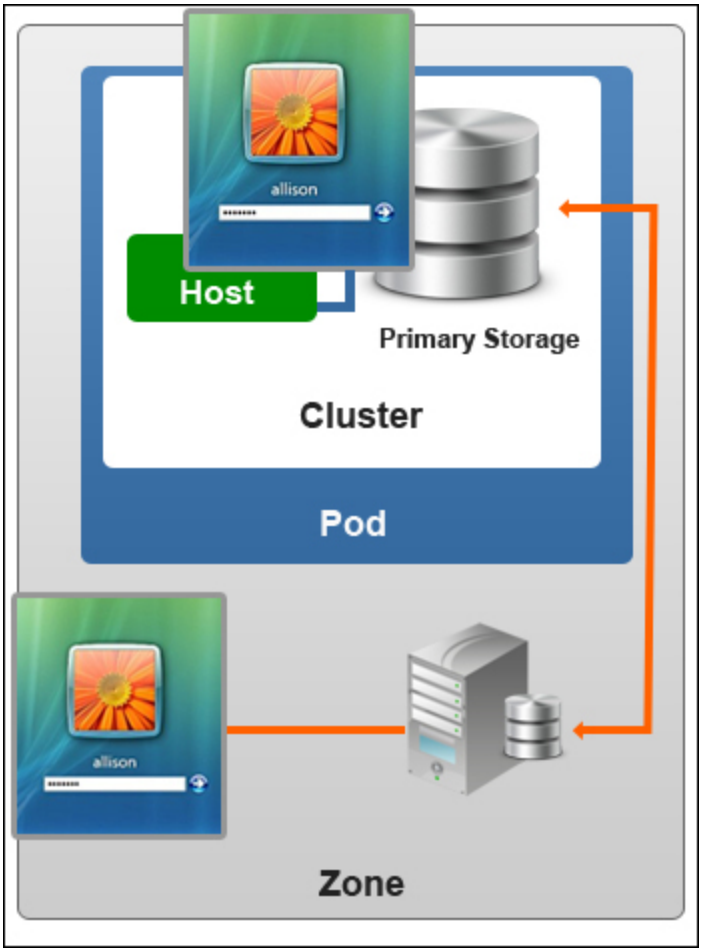
The preceding screen has the same options and fields as creating a new compute offering. For more details please see Chapter 4, *Apache CloudStack Networking*.

The other type of service offering is the network offering. The network offering is used to create a network service. Whenever an administrator creates a new network in any zone, a network offering is selected based on which the new network will have the services and scope. The network offering defines the kind of network services which are included with it. A more detailed insight of creating a network offering was discussed in the previous chapter.

The CloudStack system VMs are based on some system service offerings. For example, a security group service in a network is provided using the CloudStack virtual router, whereas a Firewall service in a network can be provided using the CloudStack virtual router or a hardware device. The networks are created using these network offerings.

The complete process

The various types of service offerings are used to create guest VMs by the user and system VMs by the CloudStack. We will see the steps followed by CloudStack when the user requests a VM. A user can add an instance from the instances page. When a user requests a new VM instance, the steps shown in the following screenshot take place:

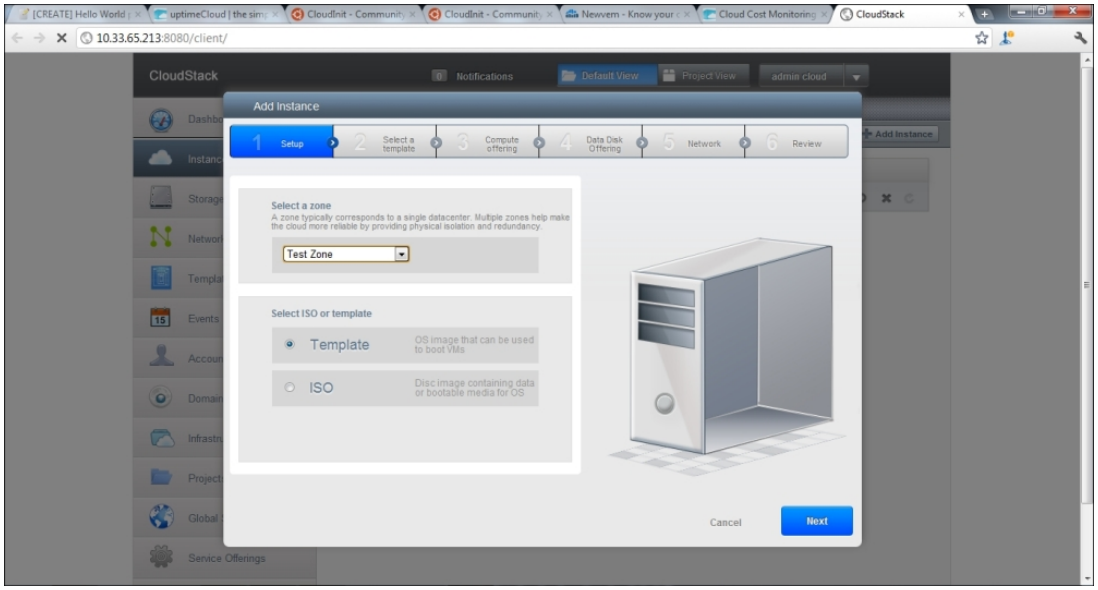


Let's take look at these steps:

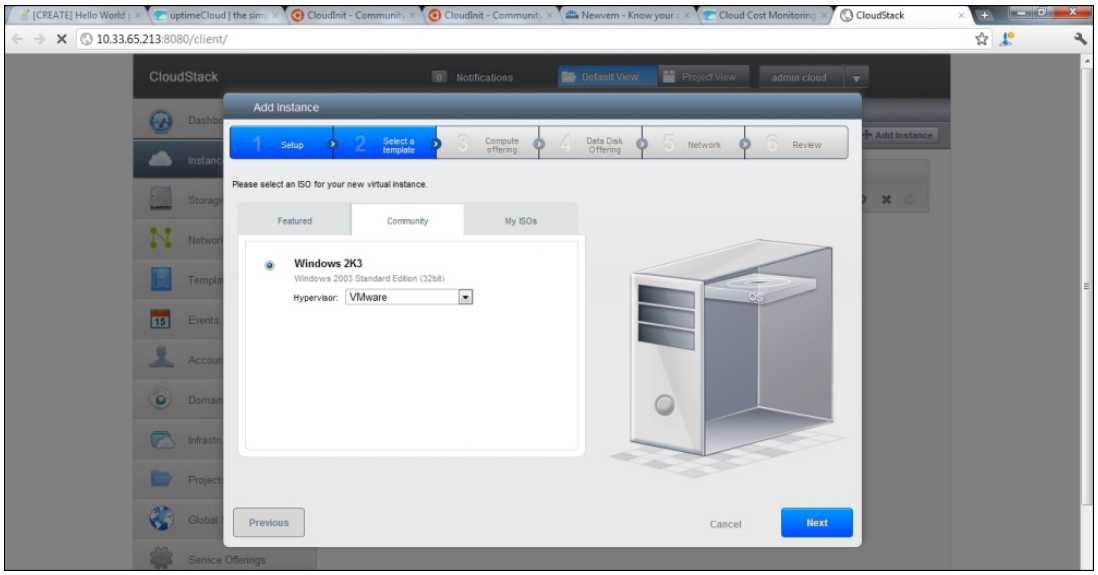
1. The user selects the zone in which he wants the VM instance to be created. The network configuration or optional network

services such as DHCP, routing, and load balancing, are already provisioned for the zone while the zone was created.

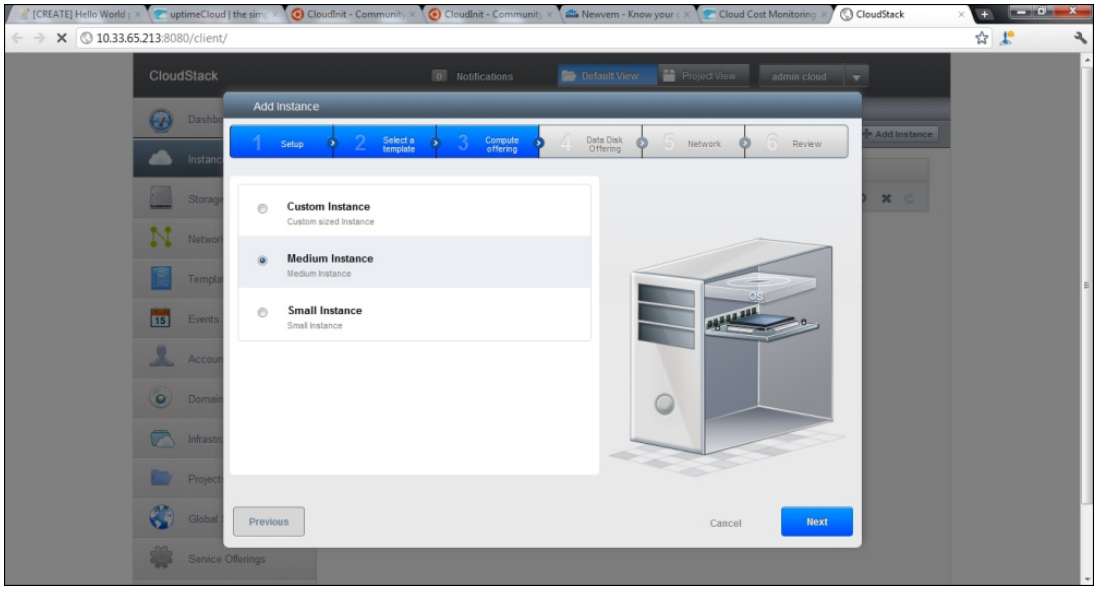
2. If the user has privileges over these services in the zone, the services are provisioned for the virtual machine being requested.



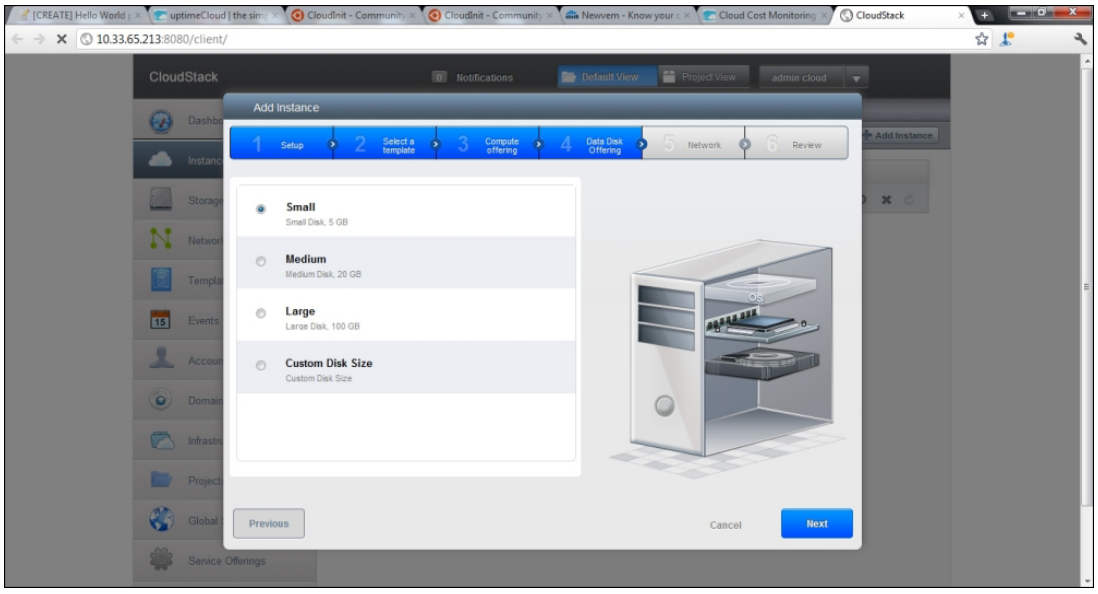
3. The user then selects the source of the VM instance, that is a template or an ISO image. The VM instance will inherit all the features as per the selection of the user. If a user selects a template, then all the resources and configuration in the template is provided in the VM instance. And when the user selects the ISO image, the VM instance is created and the ISO is loaded into the CD ROM drive of the guest. The instance boots into the virtual machine BIOS and the user can install the operating system from the ISO image by opening the console of the guest virtual machine from the CloudStack Web UI.
4. Based on the selection of source of the VM instance, the user selects the ISO image or template for the VM, from the list shown in the next screen:



5. The user then selects the Compute and Disk offering which is to be used to provide the root volume of the VM instance. The data volume of the VM instance is created using the disk offering selected by the user and is provided from the primary storage of the cluster.
6. If the user selects any extra data volume, then it is provided from the primary storage of the cluster or the local storage based on the offering selected. The local storage option is available for XenServer, KVM, and VMware Hypervisors.



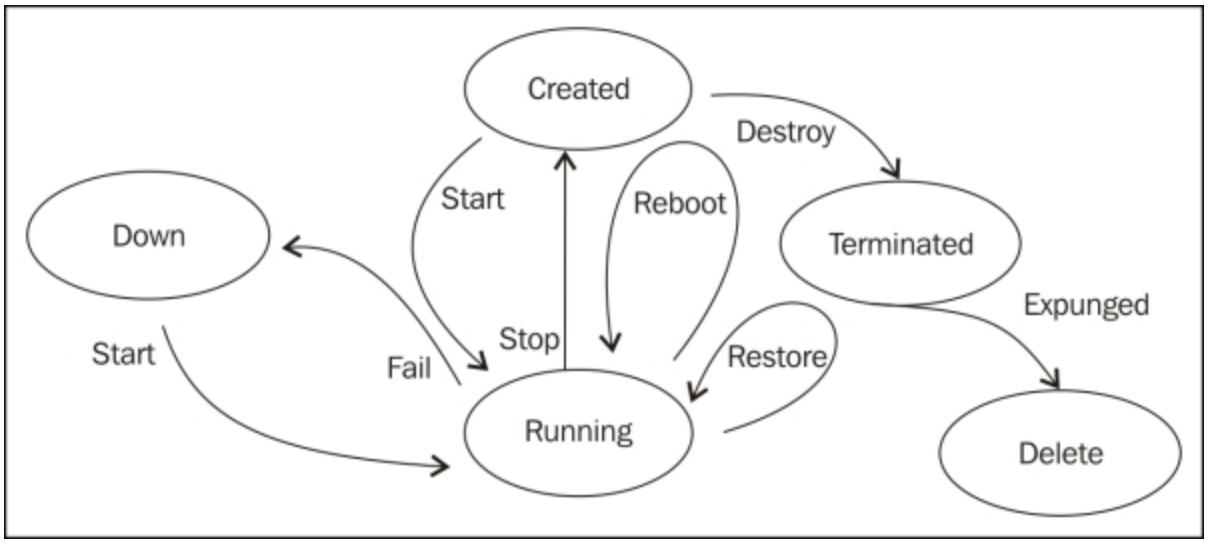
7. After choosing the required Compute offering, the user must select the Data Disk Offering for the guest instance, as shown in the following screen:



8. After the user has provided the basic information for the VM instance to be created, CloudStack initiates the copy of the template or the ISO from the secondary storage of the zone to the primary storage of the cluster. CloudStack also tries to localize the services for the accounts to as few clusters as possible to ensure security and optimize the performance of the provisioned services.
9. After the copy of the template or the ISO is done, CloudStack instructs the host to create the VM instance using the template or ISO on the primary storage and then start the instance.

After the VM instance is provisioned, the user can perform certain operations on it such as stopping, starting, and terminating. The following figure shows the state diagram of the VM lifecycle. Once the instance has been terminated, it can still be recovered. After the guest VM is terminated, the users have the option to restore the images. The next state of the guest VM is the Expunged state, which destroys the guest VM permanently, meaning that the instance cannot be recovered. By default, the Expunged state interval is set to

be 86400 seconds. The administrator can change the Expunged state interval value by changing the configuration parameter from the global settings page. The global setting parameter which you can use to change the expunged state interval is `expunge.interval`.

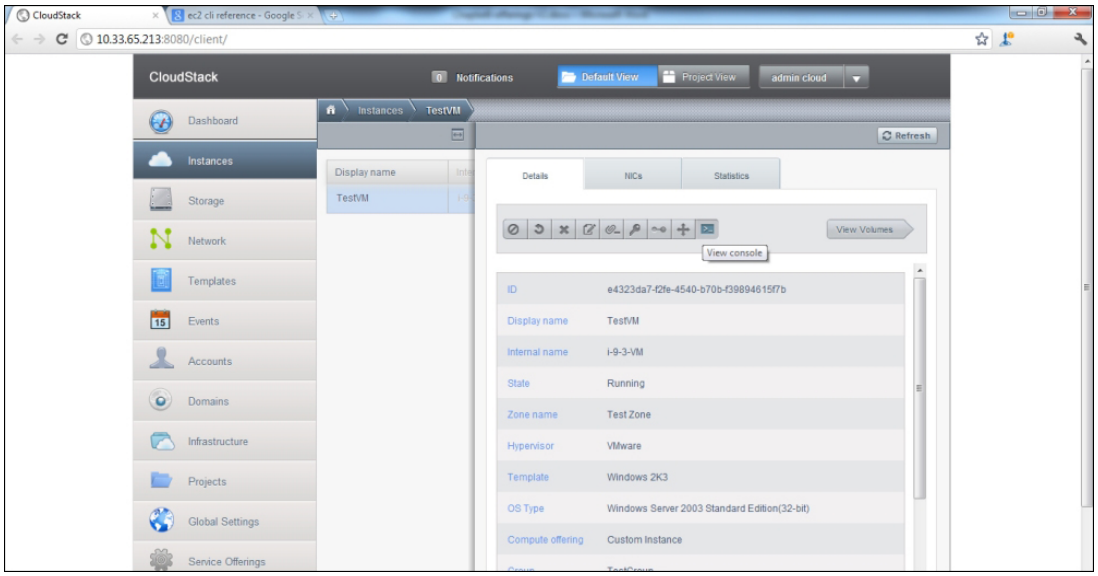


Once the VM is created, the user can perform the operations on it as described in the following sections.

Accessing the VM

After the VM instance is created, the user will need to access it and perform operations on it such as deploying applications, and testing them. The user can access his/her instance by using the following steps:

1. Go to the instance page by clicking on **Instances** in the left panel.
2. Select the instance that you want to access.
3. Click on the **View Console** button.



The user can also access the VM instance directly, but before this the user must ensure that the SSH port is open on the Firewall, otherwise the user needs to disable the firewall to access the guest VM instance.

In cases where SSH is not enabled in the instance, the user will not be able to remotely access the VM via SSH. This depends upon the template or ISO from which the instance was created. In order to enable the access via SSH, the user must access the VM from the console and enable SSH before accessing it directly.

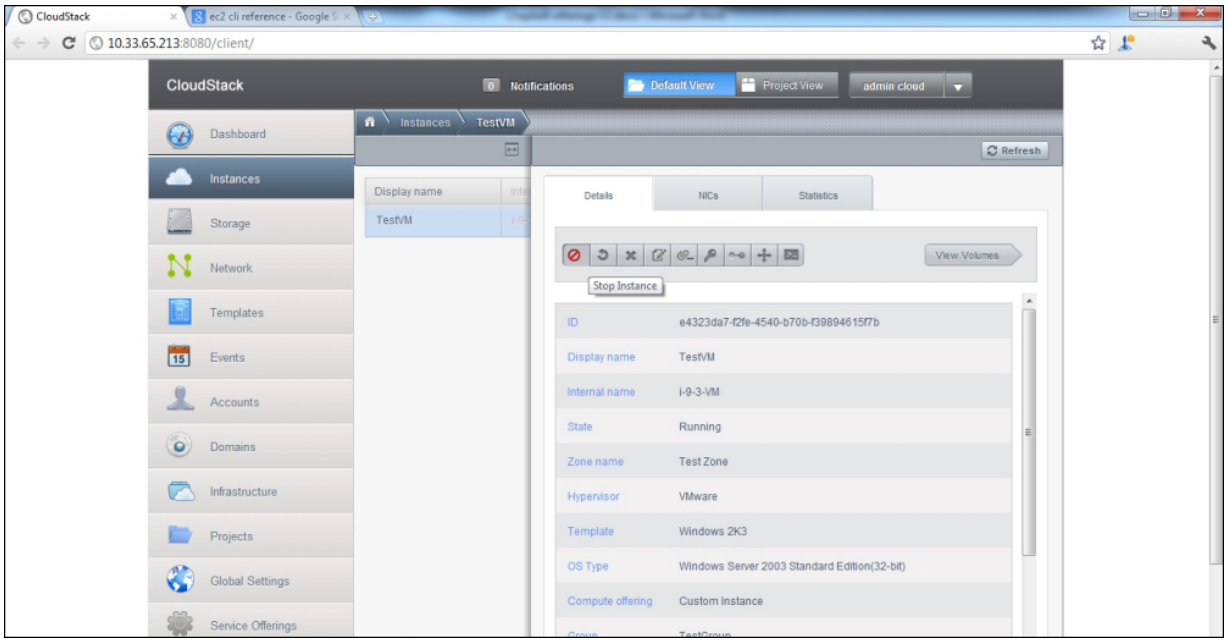
In case of a basic network zone, if the user wants to perform operations which require other ports on the instance, he must provide a rule in the security group which should allow the communication on that port from a desired source.

If there is some other external firewall protecting the VM instance, the user will have to add rules to allow the communication for accessing the VM directly.

Starting, stopping, rebooting, and destroying the VM instance

After the VM instance is created, the user can perform these operations directly from the console by right-clicking on the VM instance. The user can stop the instances which will shut down the operating system by shutting down all the applications. The user can also reboot the instance or destroy the instance.

Once the guest is destroyed, the VM can still be recovered until it is not expunged. After the instance is destroyed, the guest is expunged after the expunge interval. Once the VM is expunged, it cannot be recovered. When the instance is destroyed all the resources that are being used by it are reclaimed and given back to the resource provider. The volumes associated with the VM are deleted and recollected by the primary storage. We can see these icons in the following screenshot:



Live migration of VMs between hosts

CloudStack allows live migration of the VM instance, both automatically and manually. CloudStack live migrates a guest VM instance if the High Availability feature is enabled in the service offering from which the guest has been created. CloudStack also live migrates the VMs from the host which is marked for maintenance to another host in the cluster in order to prevent any disruption of

service. The manual live migration operation can only be performed by the root administrator of CloudStack who belong to, root domain's admin account.

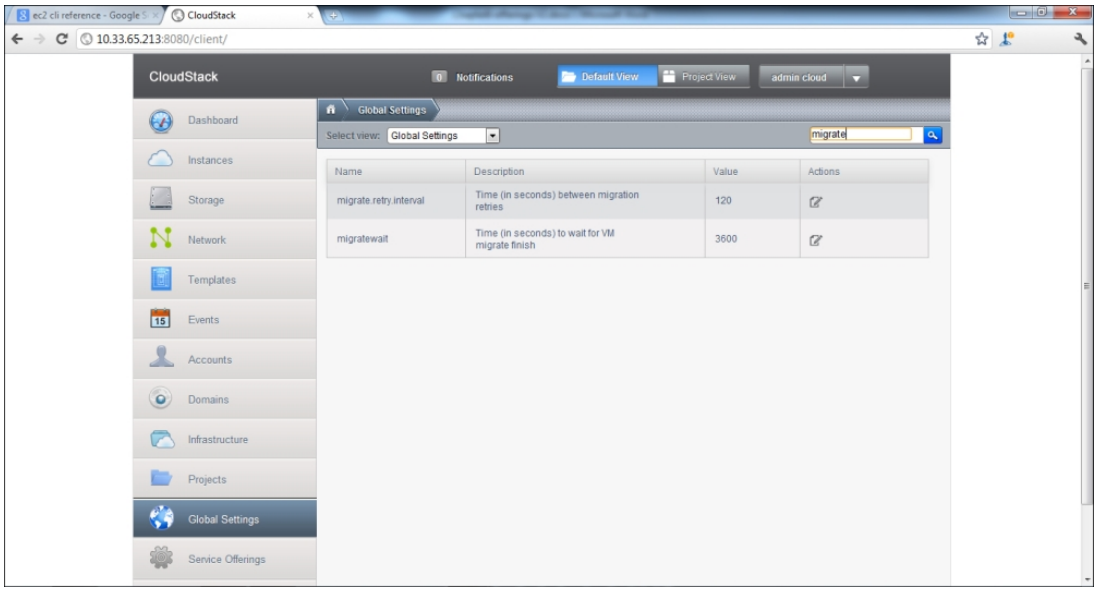
The root administrator can move a VM instance from one host to another host without any interruption to the services to the users or without going into maintenance mode. The VM instance must be in the running state for this operation and there should be resources on the destination host so that it can accommodate the VM. If the destination host doesn't have enough resources for the VM, the VM will remain in the migration state until it gets all the resources on the destination host. The VM that is to be migrated must not have any local disk associated with it.

Note

In case the VM is running on OVM, the VM must not have any ISO attached to itself because the instance with attached ISO cannot be live migrated on OVM.

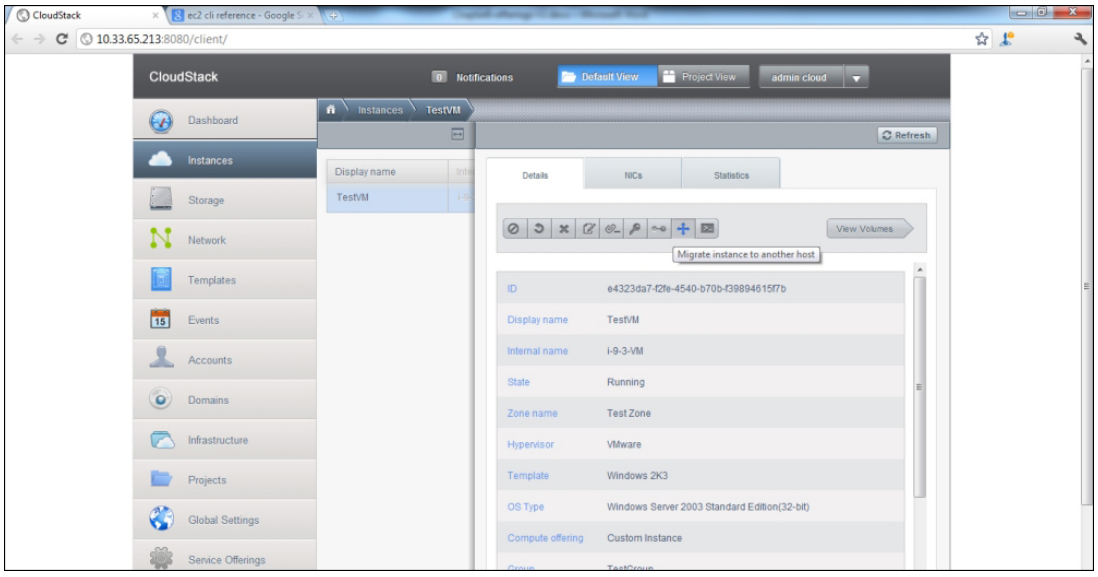
The administrator can also set some parameters such as the retry time interval for migration when it fails, and the wait time before the migration finishes. These parameters are given as follows and are shown in the following screenshot:

- **migrate.retry.interval**: This parameter defines the time interval between the retries of VM migration
- **migratewait**: This parameter defines the time in seconds before VM migration is complete



The administrator can also migrate the instance manually. To do this, use the following steps:

1. Go to the instance page by clicking on the **Instances** tab on the left panel.
2. Click on the instance that needs to be migrated.
3. Click on the **Migrate instance to another host** button.
4. This will open the option to select the host to which the VM is to be migrated.
5. Select the destination host and click on **OK**.



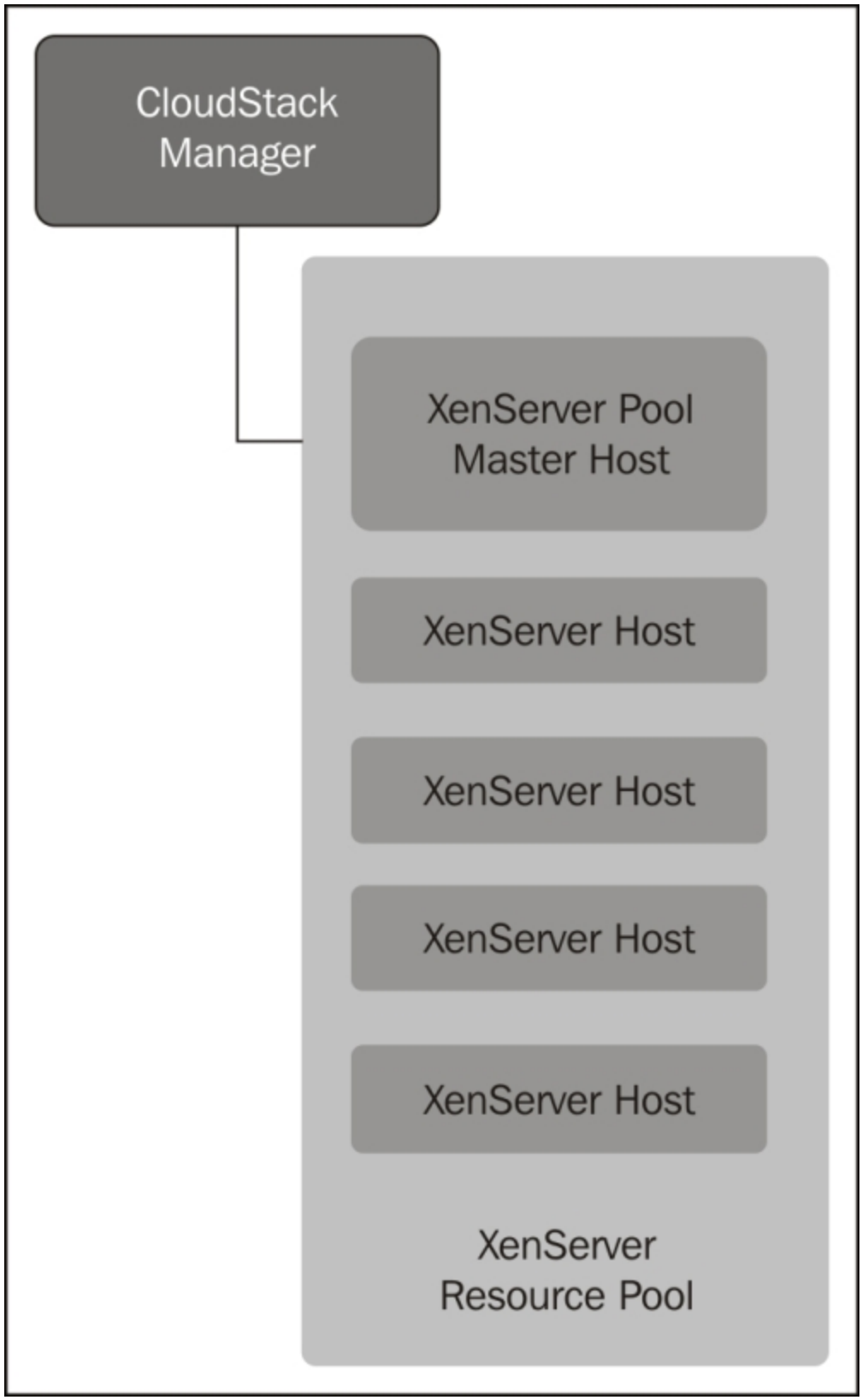
The administrator can also perform various other operations from the console after the VM has been provisioned such as detaching the ISO, editing the service offering that was used to create the VM (the VM must be stopped before doing this), resetting your password, and editing the details of the instance. The CloudStack Web UI also provides the console for viewing the details such as statistics of the VM instance and the extra network interfaces that are attached to the VM.

CloudStack with different hypervisor

CloudStack follows different steps depending upon the hypervisor on which the VM instance is to be created; this is because CloudStack integrates with different hypervisors differently. We are going to discuss how CloudStack integrates with different hypervisors.

Citrix XenServer

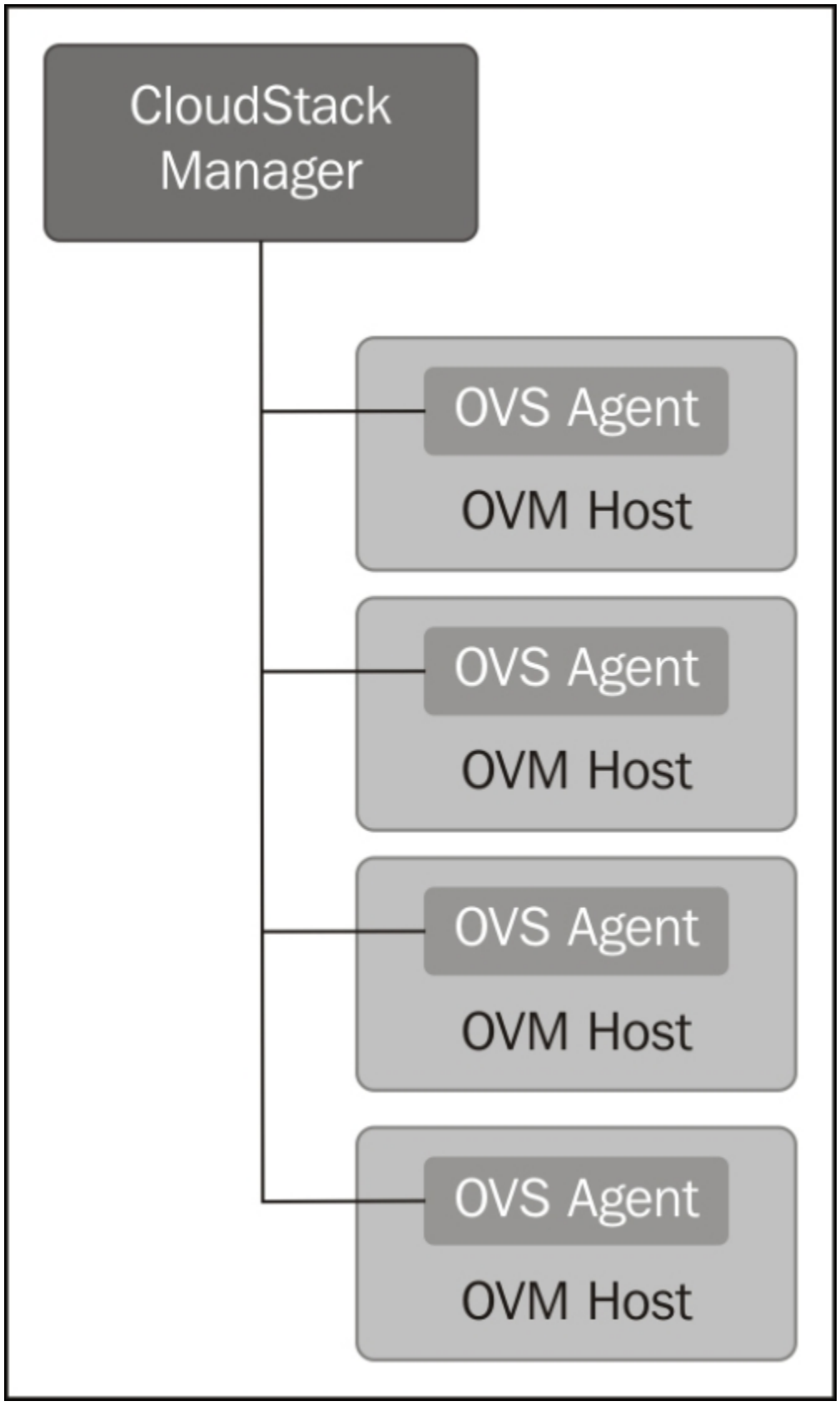
In case of Citrix XenServer, a XenServer poll master is created using the XenCenter client which contains various hosts. While doing this CloudStack adds all the slave hosts in the XenServer pool master. The system VM control channel and the network management both work at the host level. Refer to the following screenshot:


CloudStack
Manager
XenServer Pool
Master Host
XenServer Host
XenServer Host
XenServer Host
XenServer Host
XenServer
Resource Pool

Oracle VM

In case of Oracle VM, CloudStack integrates with the OVS-agent deployed on the hosts and doesn't use the OVM manager. CloudStack itself configures the ocfs2 nodes. There are various types of templates provided by Oracle such as the DB frameworks application built-in which helps the customers to quickly deploy the Oracle services. In this case, CloudStack requires a helper cluster to support the creation of the OVM instance. The domain router is automatically created in the helper cluster when the first OVM instance is created. The OVS agent helps in storing data in the local database on the host. Obviously, only supported OS types are provisioned on the Oracle host. All the Linux/Solaris templates must be obtained from the Oracle site whereas Windows can be installed from an ISO on Oracle VM hosts. Oracle recommends the usage of iSCSI devices for storage while using Oracle VM. It is the user's responsibility to set up the iSCSI device on every host.

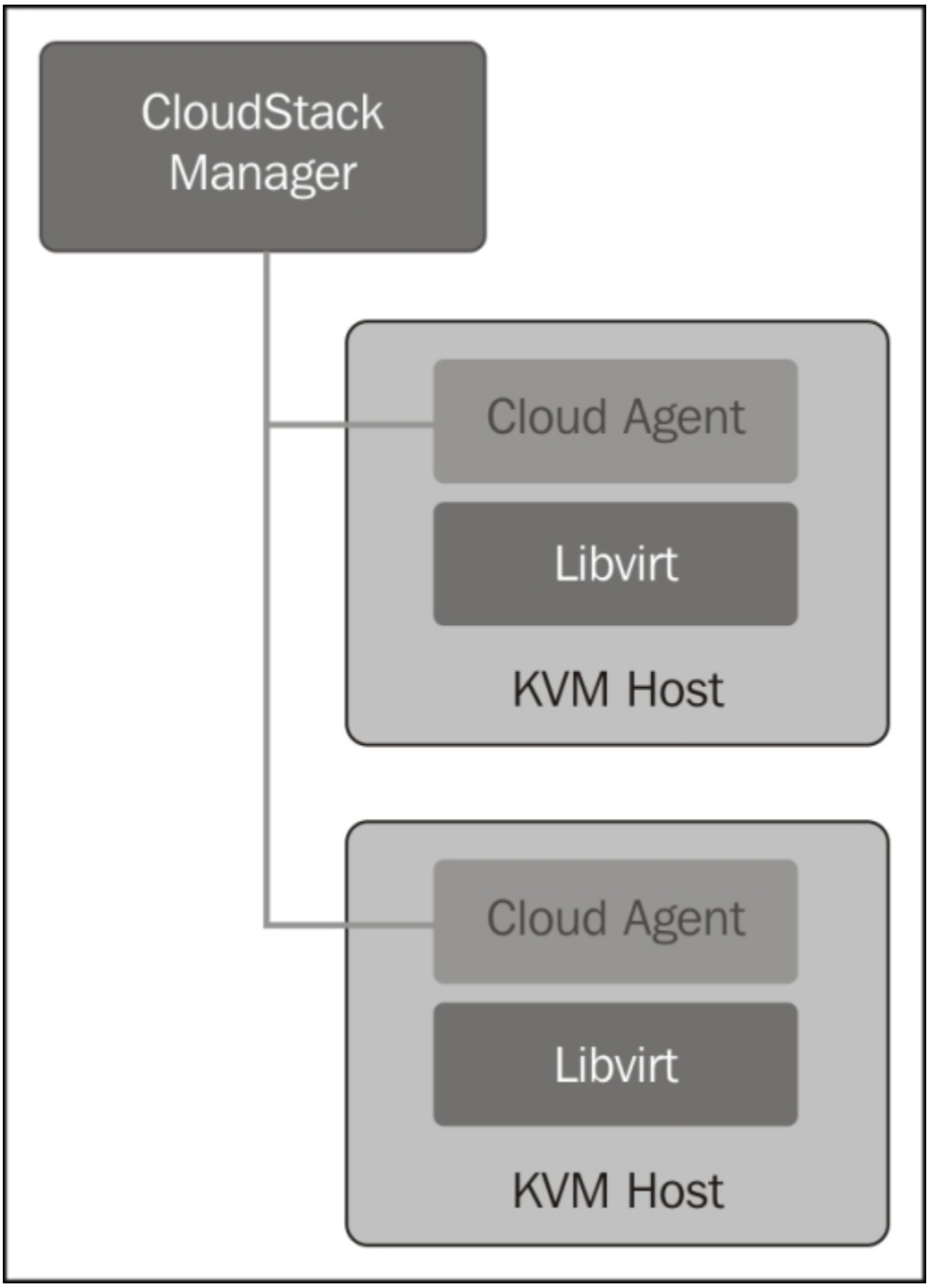
While using OVM, there are certain limitation like CloudStack needs a helper cluster such as XenServer or KVM or VMware to support OVM hosts and the system VM cannot be deployed on these hosts because OVM doesn't support Debian guests.


CloudStack Manager
OVS Agent
OVM Host
OVS Agent
OVM Host
OVS Agent
OVM Host
OVS Agent
OVM Host

RedHat Enterprise Linux (KVM)

In case of RHEL KVM, CloudStack integrates directly with libvirt and Qemu using the cloud agent and manages the snapshots, system VM control channel, and networks at the host level. CloudStack supports RHEL 6+ and Ubuntu 12.04-based KVMs. In fact, CloudStack depends on the libvrt and Qemu versions, so if you are using any other operating system, make sure the following versions are available:

- Libvrt v0.9.4 or higher
- Qem or KVM, v1.0 or higher



VMware vSphere

In case of VMware vSphere, CloudStack integrates using the VCenter. CloudStack supports system VM control channels using the CloudStack private network and deploys a system VM on the hosts to manage the snapshot and the volume. The networking, in case of

the VMware cluster, is managed using the vSphere vSwitch. While adding a VMware based zone, it is recommended to create a cluster of hosts on the vCenter and then add the cluster to the CloudStack. The vCenter cluster is mapped directly to a CloudStack cluster under the Pod.

The vCenter cluster integrated with CloudStack can only belong to one vCenter datacenter because the scope of the vCenter datastore used by the vCenter cluster is limited to one vCenter datacenter and the scope of the vCenter vSwitch that is being used by the vCenter cluster is limited to one vCenter datacenter. When a vCenter datacenter resource is shared outside of CloudStack, it may create problems.

When a new VMware cluster is added to the CloudStack, the CloudStack management server bootstraps a system VM and a secondary storage VM to manage the secondary storage which helps in template processing and volume, snapshot or template operations.

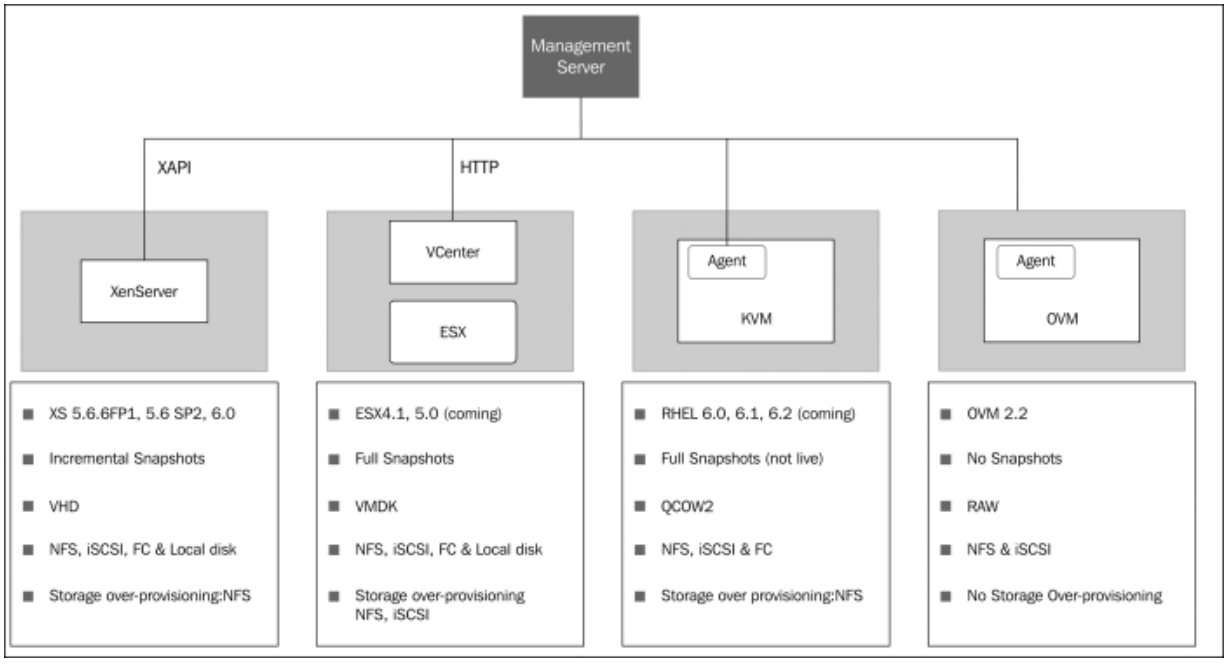
CloudStack manages network in VMware using the vCenter vSwitch. The setup of vSwitch and NIC-bonding is done through the vCenter. CloudStack creates networks by creating portgroups dynamically and propagates the network across the cluster which helps in independent VM live migration both in CloudStack and vCenter. The vCenter default vSwitch ports usually need to be extended to support the networks in CloudStack.

CloudStack takes snapshots at the volume level whereas a snapshot in vCenter is taken at the VM level. Use the following steps to take a snapshot:

1. The user creates a VM snapshot, or if it is for a detached volume in CloudStack, a new worker VM is created.
2. The VM snapshot meta-data is processed and CloudStack builds the disk chain information at the volume basis.
3. A new intermediate VM is created on top of the selected disk chain.

4. The full back up of the VM is exported to the secondary storage and CloudStack proceeds with the cleanup process to remove the resources created while this process.

The summary of CloudStack integration with different types of hypervisors is depicted in the following diagram:



Summary

In this chapter we covered the various types of offerings and the configuration of the same. We also discussed the various types of offerings that are supported by different hypervisors in CloudStack. In the next chapter, we will introduce to you to the concept of domains, accounts, projects, and users in the CloudStack, and their use to isolate different types of users in the cloud.

Chapter 7. Domains, Accounts, Projects, and Users

Till Chapter 6, *Service Offerings and Virtual Machines requesting as well as working with Templates*, we have covered the basics of setting up CloudStack, configuring it, and other important aspects of cloud. Further, in this chapter we are going to cover the following points:

- Role-based access mechanisms for managing users in the CloudStack
- How CloudStack allows us to segregate and group users for efficient usage of resources
- Various terminologies used in CloudStack such as domains, accounts, projects, and so on
- The working of domains, accounts, projects, and users

Domains, projects, accounts, and users are the various terms used in CloudStack to manage and group users in the cloud, which helps in segregating, isolating users as well as enabling collaboration between different users to work as a team.

RBAC is an essential mechanism to provide different access permissions to different users. Users are granted different permissions in terms of resources so that they can create and manage the resources that they are granted permissions for. As of CloudStack Version 4.0, there is no RBAC mechanism as such, except there are only two levels of users, they are admin and users. CloudStack administrators can create domains within which they can create projects, accounts, and users within them.

CloudStack administrators are of the following types:

- **Root administrator**: The root administrator is the administrator of the root domain. The root administrator role

has all privileges to administer the CloudStack environment. The root administrator can create subdomains of the root domain, accounts, projects and so on. The root administrator can create domain administrators/project administrators to further delegate the administrative activities for specific sub-domains/projects.

- **Domain administrator/project administrator**: The domain administrator is the administrator of a domain and all the sub domains under that domain. The domain administrator also works as the project administrator. The domain administrator can manage the resources under the domains and projects. There can be only one project administrator and the role of project administrator can be transferred to any domain administrator.

Domain

Domains in CloudStack can be thought of similar to the domains in LDAP. They are basically a group of accounts that are connected to each other by some logical relationship. Domains are created when we need to create multiple hierarchies of customers. Domains are means to hosting multiple tenants for a single cloud provider so that these tenants can be isolated from each other.

A domain usually has one or more domain administrators; the administrator is a user with more privileges than other users. These administrators can be provided with authority over the domain or its subdomains. There can be multiple domain administrators in a single domain. An administrator can create multiple accounts or add users in the domain and has administrative access over the virtual resources of the cloud belonging to this domain.

The domain administrator cannot access the physical resources of the cloud but has access to all the subdomains and accounts in that domain, the domain administrator can create/delete any user in that domain or change properties such as passwords. The domains do not own any resources; they are just containers that contain objects

that can own resources. They can include projects and accounts but cannot include Cloud resources such as volumes, instances, networks, and templates. In a basic zone configuration, the domain can have resource like a CloudStack virtual router which can be used by all the subdomains and the accounts under it, so that it can serve multiple networks and users.

Root domain

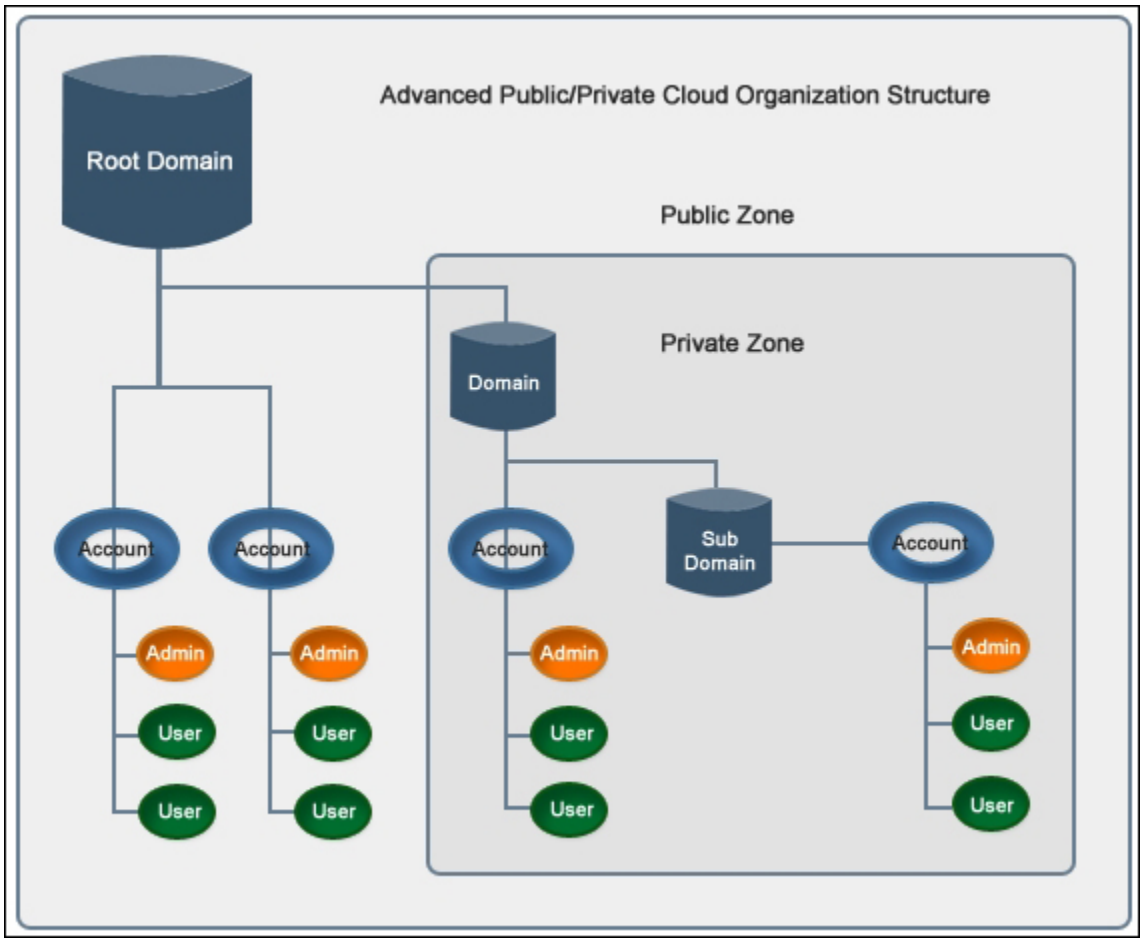
CloudStack by default has a root domain within which all the domains are created. This root domain is a special domain. The administrators can change the name of this using the API call. All the other domains are created under this domain. These domains can contain multiple accounts of user and admin types (discussed later in the chapter).

Domains contain accounts and an account can be of two types—Admin or User.

All users created under an Admin type account, under root domain are cloud administrators and have privileges over the CloudStack environment however the users are tenants and can consume the resources provided under the domain or project.

Domain and zones

The domain and zones share a many-to-many relationship. A single domain can host multiple zones or a zone can also belong to a particular domain. In case of configuring a public cloud type architecture, the Root domain can contain a zone which will be accessible by all the sub-domains. It is recommended to have sub domains under the root domain and the zones must be created for these subdomains.



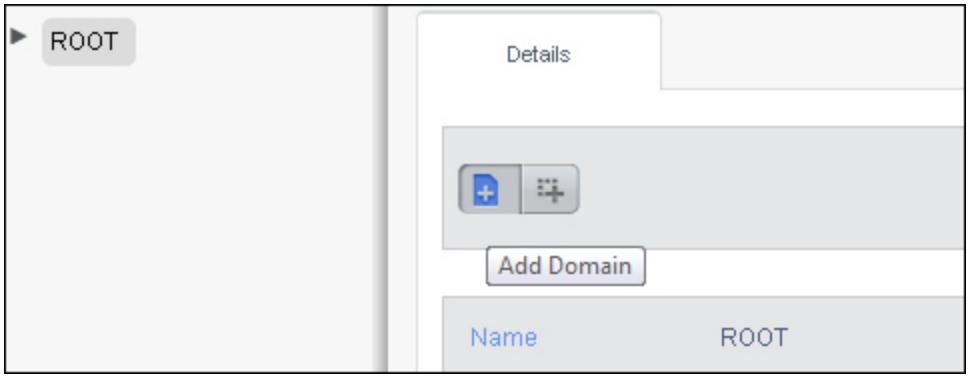
The domains serve as the resource boundary and help in providing security as well. The domains provide a limit on the amount of the resources, as well as the scope of resource usability; for example, who can use those resources. The domains provide a user structure as well as the resources these users can have access to.

Creating a domain

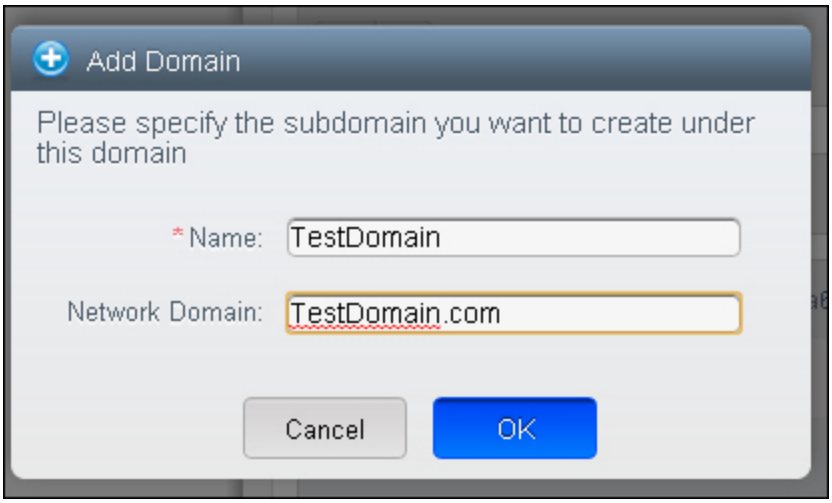
As explained above, all domains are created within the root domain. To create a new domain, follow these steps:

1. Log in to the CloudStack UI as CloudStack administrator.
2. Navigate to the domain page by clicking on the domain on the left panel.
3. Click on the **ROOT** domain. This will open the details page of the root domain. You can create a domain within any domain;

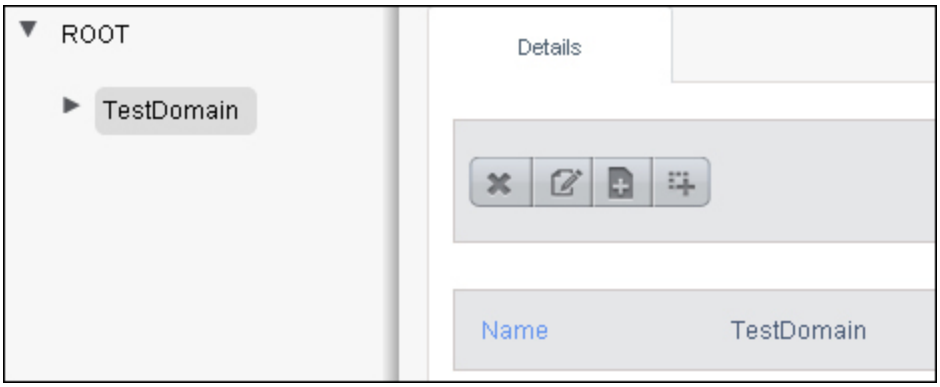
in this case it is created within the root.



4. Click on **Add Domain**. This will open a pop up to enter details for the new domain.



5. After entering the **Name** and optional **Network Domain** that defines the network domain for the domain and clicking on **OK**, a new domain is created.



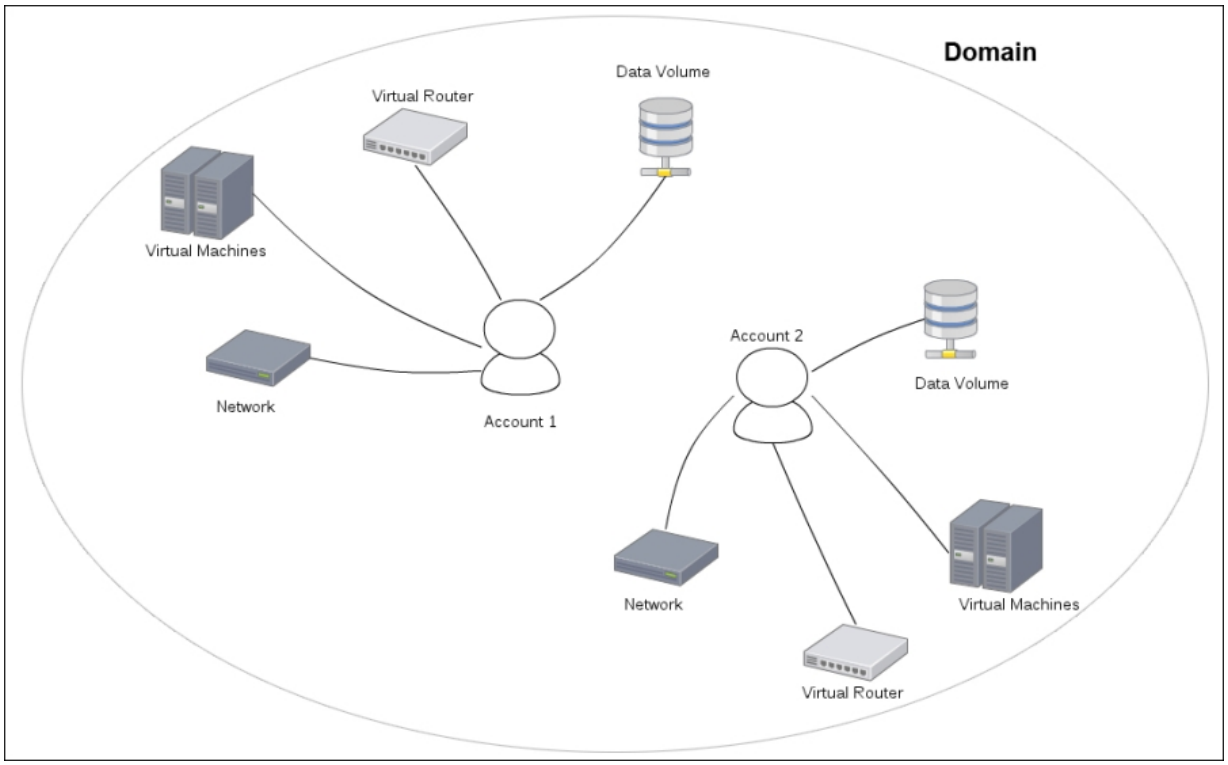
A new domain is created under the root account and now you can perform certain operations on it such as editing, deleting the domain, creating a new subdomain, or updating the resource count for that domain.

For deleting a domain, click on the cross button and it will ask whether to force delete the domain or simply delete the domain. A domain has accounts and projects within it, and if you select the **Force delete** option, the operation will delete all the resources within it like the accounts in that domain, projects, and the users in that domain.

Accounts

CloudStack allows creating accounts to represent a customer of a service provider or a department in an organization. An account represents an independent entity created under a domain and is isolated from other accounts in the cloud. Accounts in a single domain may have a set of administrators who have privileges over the domain and their subdomains.

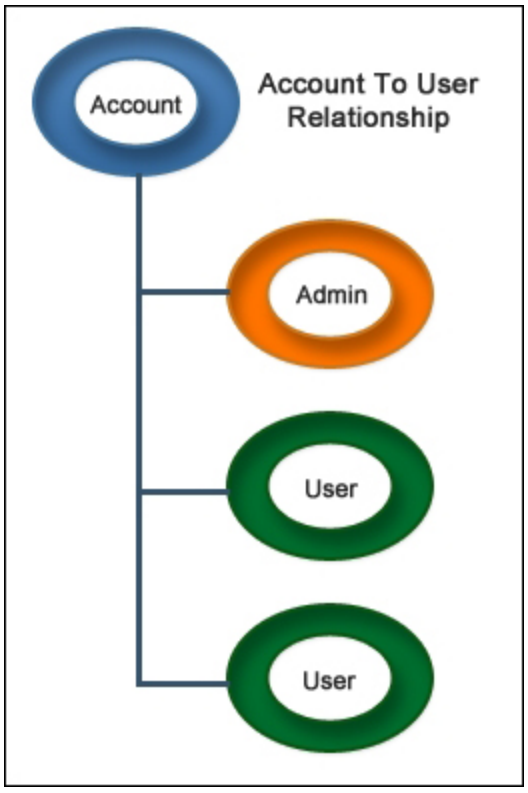
Accounts in a particular domain are related to each other logically, thus there can be accounts for different customers in a domain of a reseller or there can be multiple accounts representing multiple departments of an organization. A normal user account has normal users who have only restricted permissions. The resources in an account are isolated from other accounts and they can be accessed by the users in that account only.



Accounts can own resources; these resources are limited by the Root domain administrator.

As mentioned earlier there can be only two types of users in CloudStack—the admin and the user. The accounts can be of admin or user type, as shown in the following screenshot.

- **Admin accounts**: The admin accounts have users who have administrative privileges to the domain and all the subdomains under the domain in which the account is present. The admin can create additional accounts in the domain and can also generate API keys for the users. The admin is limited to the number of resources as defined by the Root domain administrator(s).
- **User accounts**: The user accounts have permissions to create and manage resources with limited privileges for administrating them.

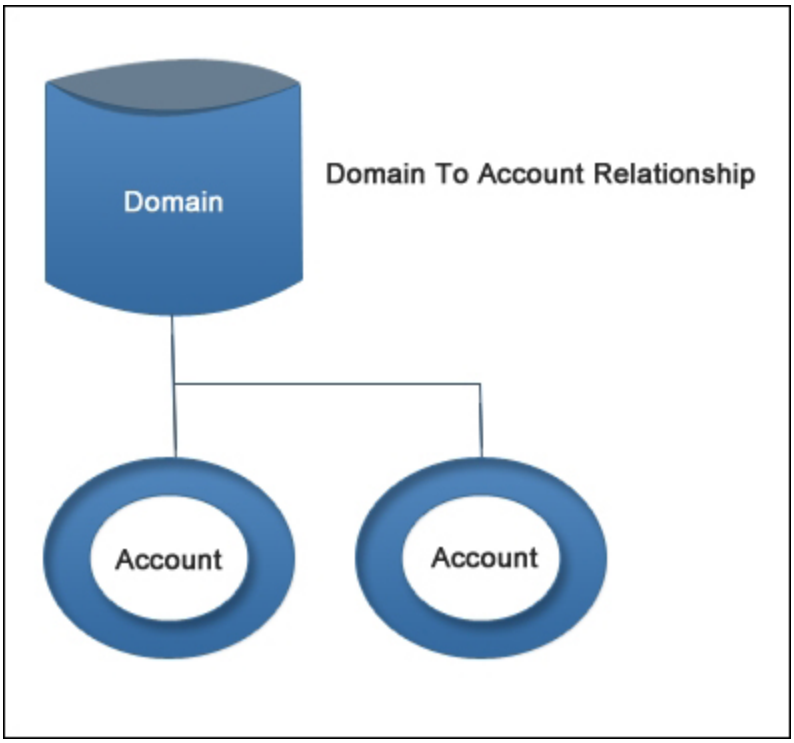


The accounts also own the usernames and passwords or the API keys which are used to log in to the CloudStack Web User Interface and access the APIs respectively.

The users don't own any resources, they just use the resources provided. The users can consume the resources within the limits of the Account they are member of. Thus the limits are set at Account level and Project level and not at individual user level.

The users within the same domain must have unique names. Users within the same account cannot have different sets of permissions, the permissions given to a user are inherited by the account they are a part of.

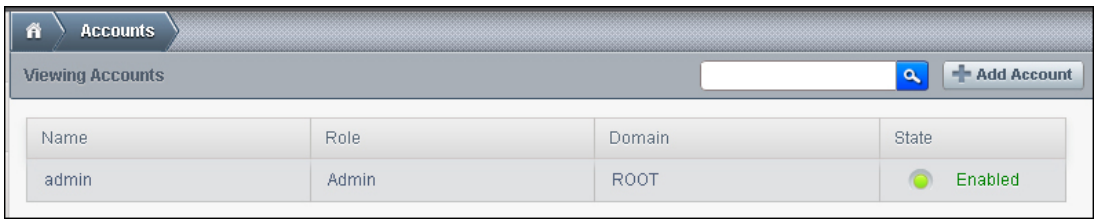
There are multiple users in an account who can have shared resources among themselves but they are isolated from users in other accounts in the cloud. By default, CloudStack provides one user while creating an account. While creating an account, the administrator specifies whether it is an administrator or a normal user. These domain administrators don't have access to the physical servers or other domains. The administrator can add any user to the account. A user can belong to only one account at a time.



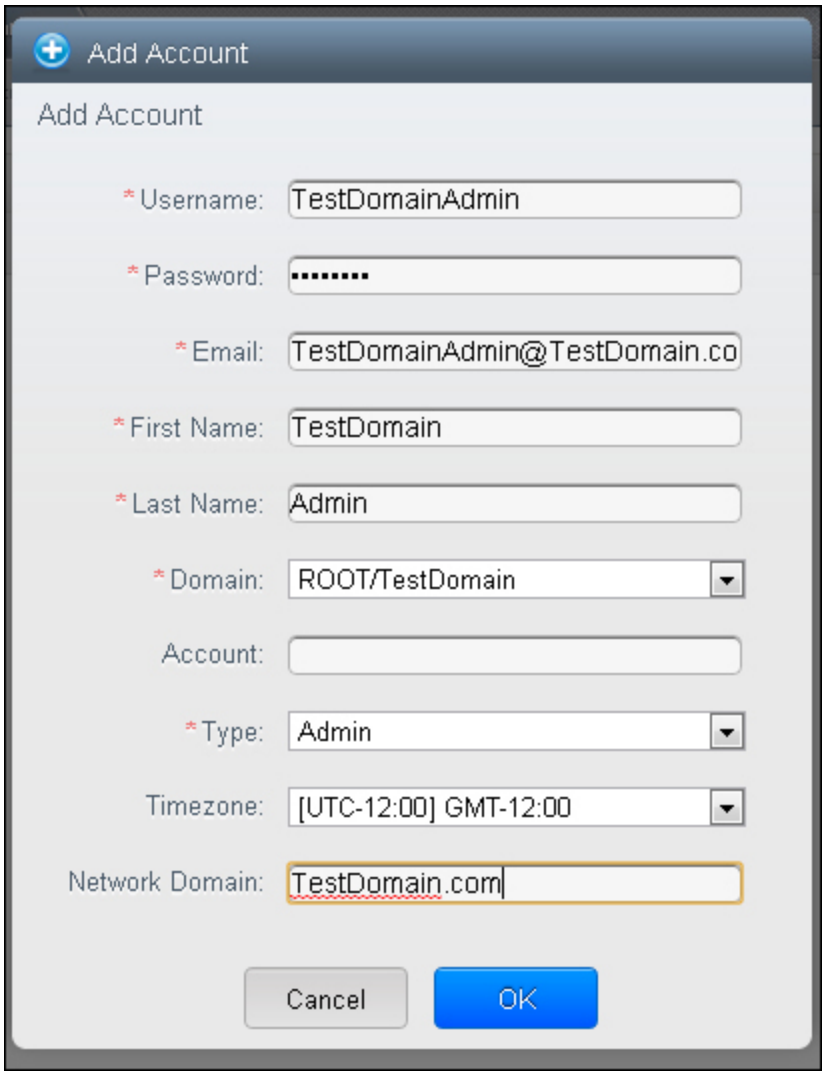
Creating an account

An account can be created within a domain. For creating a new account, the domain administrator or the CloudStack administrator can follow these steps:

1. You can either migrate to the domain you want to create the account in and click on **View Accounts** button. This will take you to the account page displaying all the accounts within that domain.
2. Or migrate to the accounts page by clicking on the **Accounts** tab on the left-hand side. This will display all the accounts the user is associated with.
3. Click on the **Add account** button.



4. A pop up will open to enter details for the account, as shown in the following screenshot:



Fill in the fields, which are explained as follows:

- **Username**: Name of the account and when a new account is created with this name and this is used for logging in.
- **Password**: Password of the user which is created in this account.
- **Email**: The e-mail ID of this user.
- **First Name & Last Name**: The first and last name of the user.
- **Domain**: Select the domain in which the account is to be created; it can be either a root domain or a subdomain within it.

- **Type**: When creating an account, you must specify the type of account; it can be either Admin or User.
- **Time zone**: This is optional. Select the time zone for the account.
- **Network Domain**: This optional field is used to specify the network domain of the account.

The network domain can be assigned at the account level, domain level, or the zone, and they override the one defined in the above hierarchy. And if it is not specified in any of them, it is taken from the global configuration.

After entering all the details and clicking on **OK**, we create a new account as per the details entered. Once the account has been created, there can be various operations that can be performed on that account such as viewing all the users, creating new users, deleting, and so on.

There can be multiple accounts of the two types (admin/user) within a domain. The accounts are useful in organizations that want multiple administrators for different departments and have shared resources among these departments.

The accounts own the resources. Whenever an account is deleted all the resources associated with the account are also removed.

Projects

CloudStack also provides a feature for creation of projects so that cloud users can divide their workforce into logical groups which can share virtual resources in the cloud. The projects can be created by any users and that user becomes the project administrator. It helps the group of users in an organization to share the virtual resources among themselves so that they can work collaboratively on them.

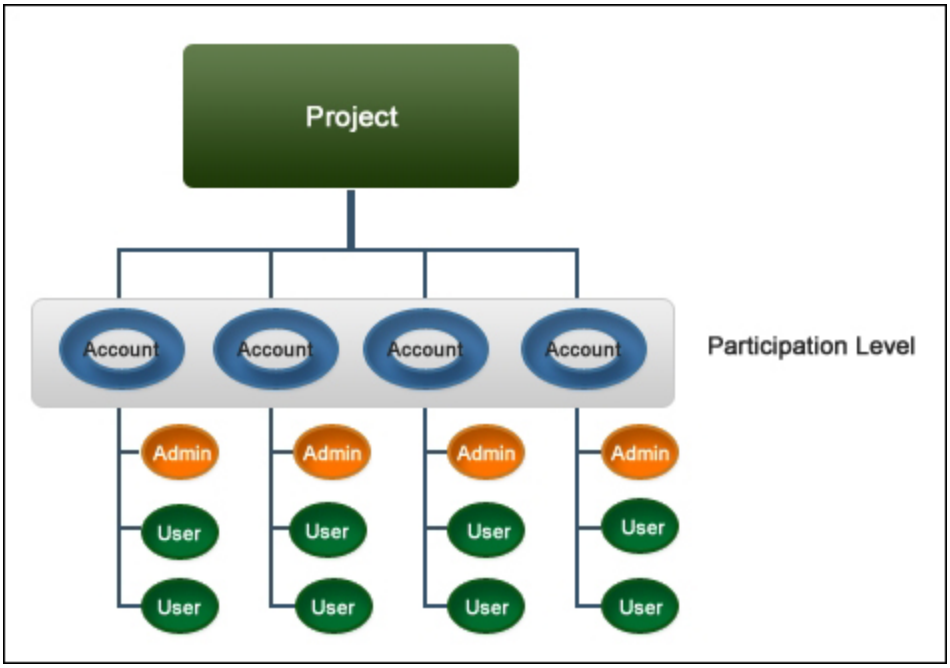
CloudStack projects help in bridging the gap between the multiple users by grouping them so that they can be managed commonly and the different resources can be grouped into various projects. For example, a team of developers might create a project "development" in CloudStack and create resources in it so that they all can work on them and manage the resources. CloudStack provides any user with the capability to create projects and that user becomes the administrator of that project.

CloudStack can be configured to allow any user to create a new project or this ability can be restricted to just the CloudStack administrators. There can be only one owner of a project at a time and he has special permissions on the resources and other users. The administrator can set the limit on various resources such as VM instances, snapshots, templates, and volumes. A project can have any number of users and a user can be a member of any number of projects at a given time.

Projects and accounts

Projects allow the users to share the control of resources between multiple accounts. The resources can be owned by the project and they can be accessed and controlled by multiple accounts within the same domain. This can be useful when different account members want to work on shared resources, then a project can be created to include the members from different accounts. There is only one project administrator in a project who can invite and revoke to add

and remove accounts within the same domain as the project. The project administrator doesn't have any permissions on the account level resources. The project administrator can be changed and may belong to any account within the project. A brief understanding of project, as explained above, can be shown in the following diagram:

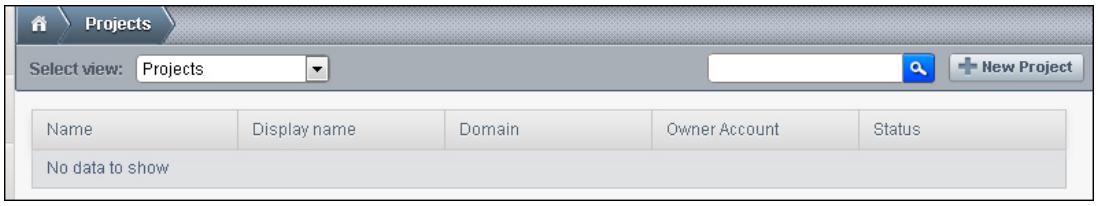


Creating a project

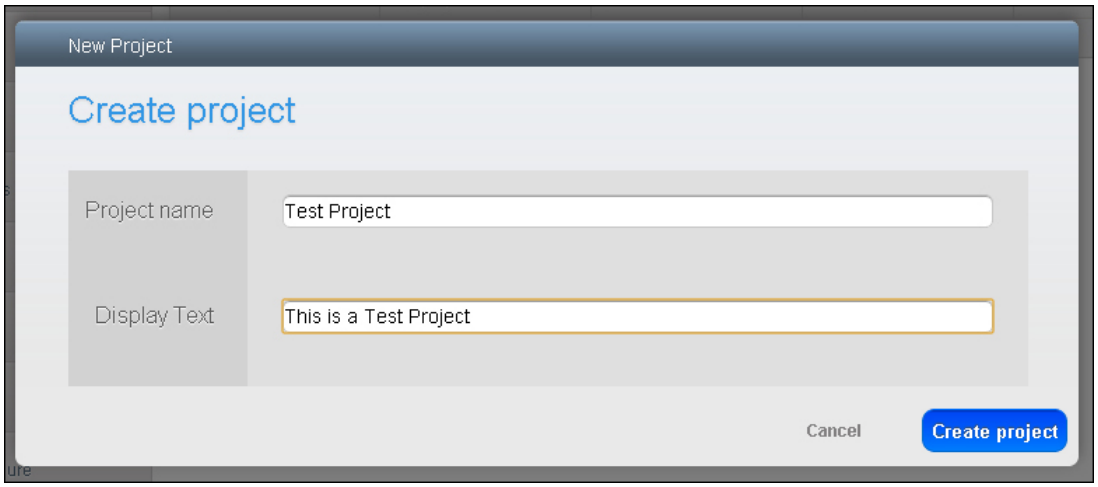
CloudStack is configurable to allow end users to create projects by setting the global parameter `allow.user.create.projects` to be `true`. This will allow end users to create project otherwise only CloudStack administrators or domain administrators will be allowed to add new projects. After changing any global parameter, it is necessary to restart the management server to make the changes take place.

For adding a new project, follow the following steps:

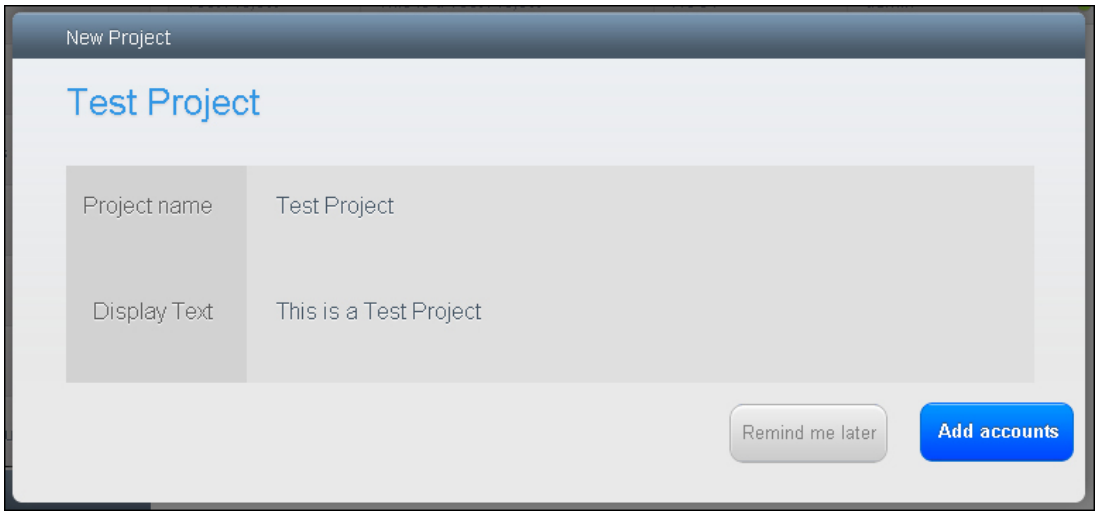
1. Log in to CloudStack as the cloud admin or the domain administrator in which you want to add the project.
2. Go to the **Projects** tab by clicking on **Project** on the left panel. This will display all the projects in the domain.



3. Click on the **Add New** project. This will open the new project pop up as given in the following screenshot:

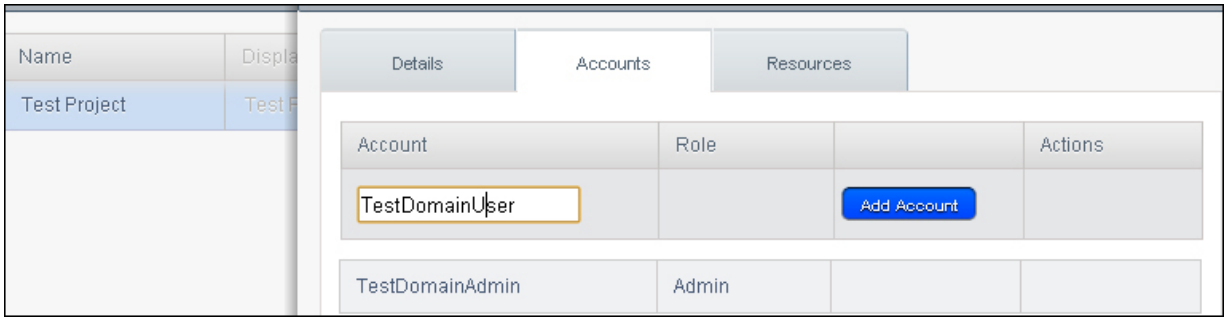


4. After entering the name and the display text, click on **Create project**. This will create a new project.
5. After the project has been created, the administrator can add accounts to it immediately or do it later.



Adding members to the project

The project owner can add accounts in the same domain as the project. Click on the **Add accounts** button to add accounts to the project. This can also be done by navigating to **Project** in the **Project** page and adding a new account on the **Accounts** tab, as shown in the following screenshot:



Enter the name of the new account you want to add and then click on **Add account**.

The **Projects** page has various operations that the project owner can perform on the project such as updating the resource counts and editing the details.

Resource management in Projects

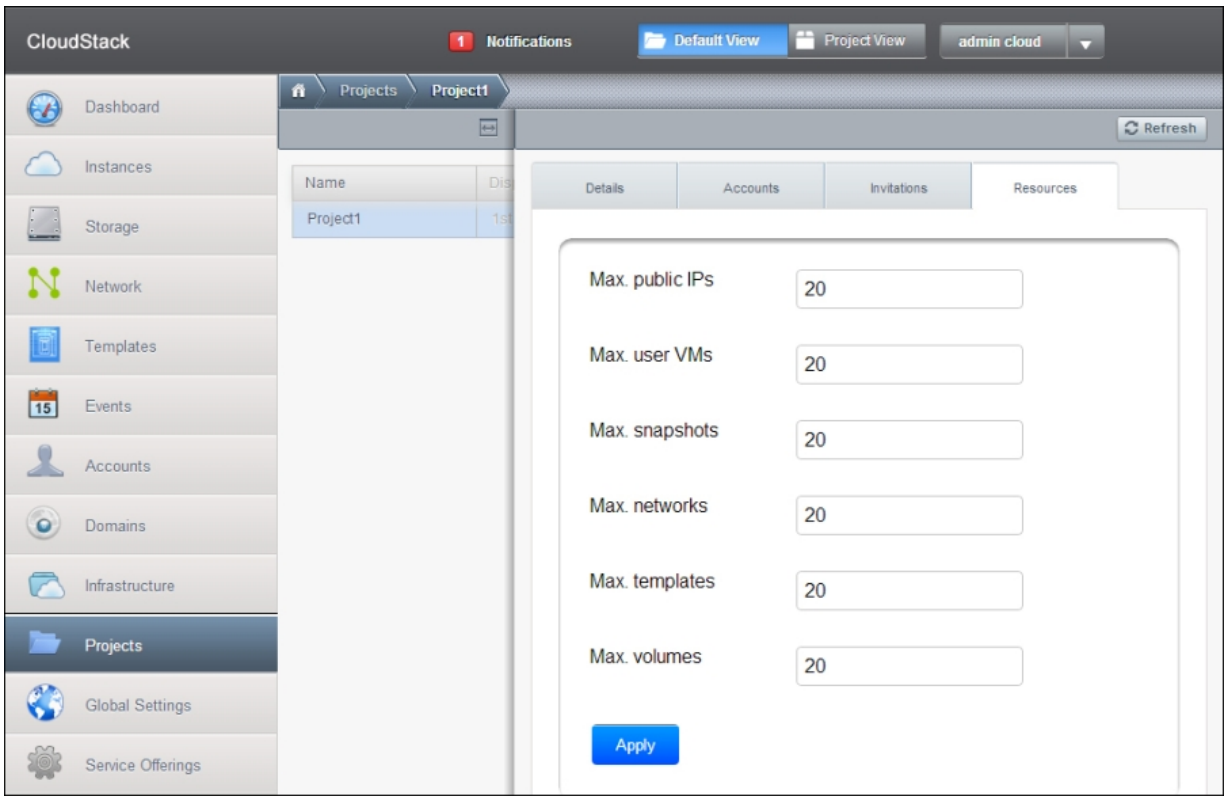
The **Resource** tab of the project is used to provide limits to resources in the project such as:

- Maximum number of VMs
- Maximum number of public IPs
- Maximum number of volumes
- Maximum number of templates
- Maximum number of snapshots

The project owner can edit this information up to the maximum default limit set by the global administrator of the CloudStack.

When the project owner decreases the limits, and there are already more resources than the new limit, then the resources whose count

is above the limit are not affected, though new resources can only be added when the resource count goes below the limit given.



The default limits that are provided to the resources when any project is created can be changed by editing some global configuration parameters by navigating to the **Global Settings** page by clicking on the **Global Setting** tab on the left panel. This action can be performed only by the CloudStack administrator. These parameters are:

- `max.project.public.ips`: The maximum number of public IPs to be allowed in any project in the cloud.
- `max.project.snapshots`: The maximum number of snapshots that can be allowed in any project in the cloud
- `max.project.templates`: The maximum number of templates to be allowed in any project in the cloud.
- `max.project.volumes` The maximum number of volumes to be allowed in any project in the cloud

- `max.project.uservms`: The maximum number of VM instances to be allowed in any project in the cloud.

After setting these parameters, the management server must be restarted in order to make the changes take effect.

Invitation setup

This feature allows the administrator to send invitations to users to join a project.

Users can be added to project directly or by sending invitation to them. If we want to use the invitation feature, we have to set the global parameter `project.invite.required` to `true`. There are other parameters which are also related to the invitations. These are as follows:.

- `project.email.sender`: This parameter is used to define the email address that will appear in the from field on the invitation email.
- `project.invite.timeout`: This parameter is used to configure the time after which the invitation will become invalid.
- `project.smtp.host`: The address of the host which will act as the SMTP host for sending the invitations mail.
- `project.smtp.port`: The port on which the SMTP server is listening.
- If the SMTP server requires authentication to send e-mails then you must set the global parameter `project.smtp.useAuth` to `true`. To use the SMTP server, CloudStack must be configured with the SMTP server username and password. This can be done by the configuration parameter, `project.smtp.password` that holds the password required by the SMTP server, `project.smtp.username` holds the username of the SMTP server.

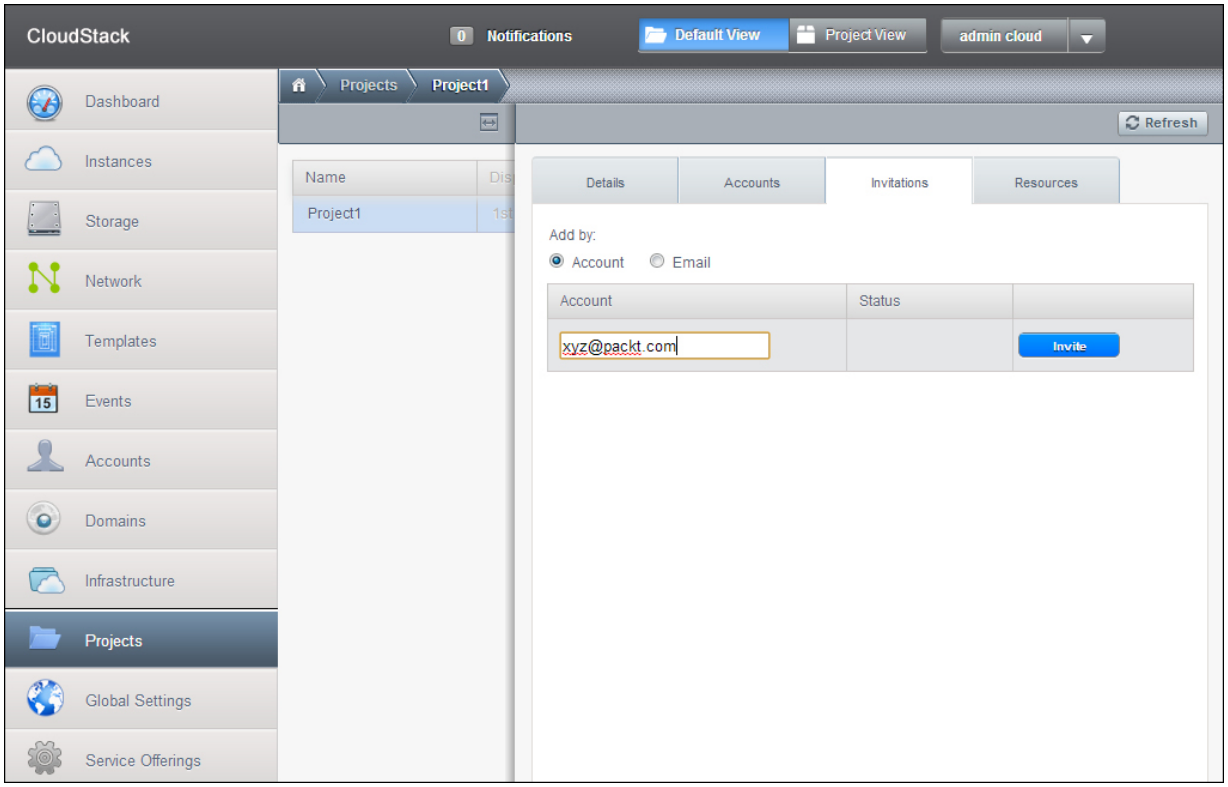
After the configuration of invitation to be used to add members to the project, the project owner can invite members to the project by sending them invitation by following these steps:

1. Log in as the **Project owner** or the CloudStack administrator.
2. Navigate to the **Projects** tab by clicking on the **Project** button on the left panel.
3. Select the project you want to work on.
4. Click on the **Invitations** tab and then click on **Add** with one of the following options selected:

    - **Account**: This will send the invitation to the user's invitation tab in his user interface.
    - **E-mail**: This method will send an e-mail to the user at the e-mail address we provide. The e-mail message will send with a unique code that will identify the user when he accepts the invitation and responds to the CloudStack.

5. If you have chosen e-mail, then enter the e-mail address of the member whom you want to add and enter username or the e-mail address of the CloudStack user if you want to add by account.

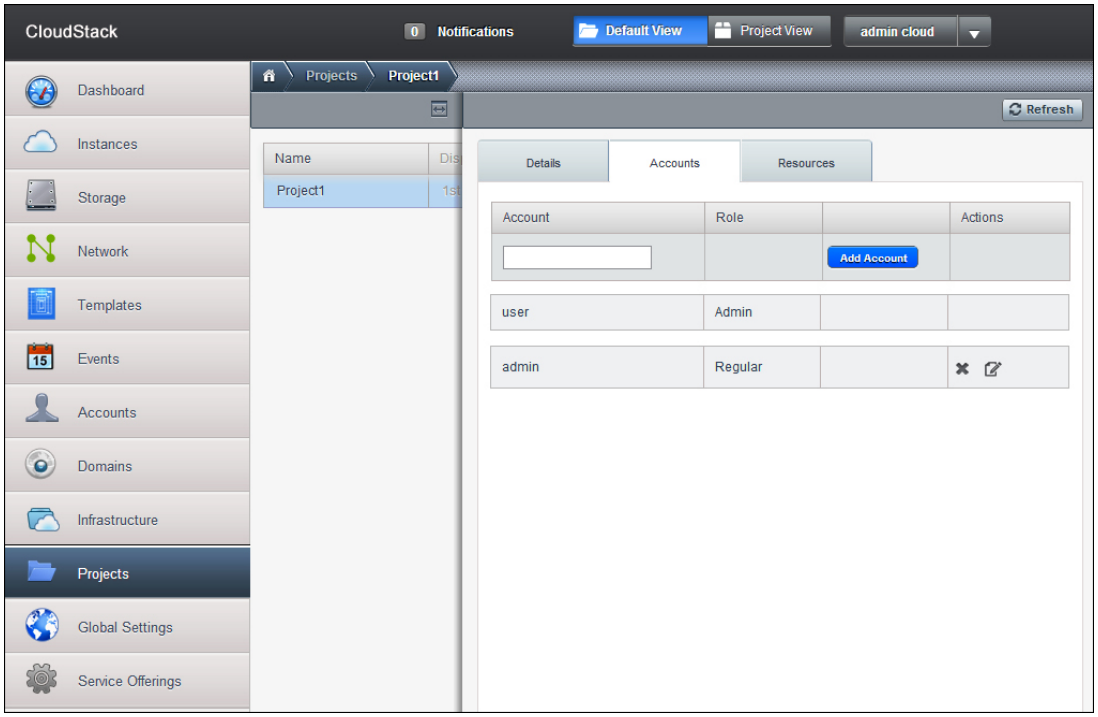
This will send the invitation to the user. When the member accepts the invitation before the timeout, he will be added to the project. The project owner can manage invitations by going to the **Invitation** tabs for the project.



Removing a member from a project

The project owner, the domain administrators (to which the project belongs), and the CloudStack administrator have the permissions to remove any member from the project. When a user is removed from the project, the resources that belong to the user are still in the project which can be used by other members. For removing a member from the project, follow these steps:

1. Log in to the CloudStack UI as project owner, domain administrator of the domain to which the project belongs, or the CloudStack administrator.
2. Navigate to the project page by clicking on the project tab on the left panel.
3. Click on the project you want to work on.
4. Navigate to the **Accounts** button and click on the name of the member you want to delete.
5. Click on the **Delete** button to delete the member.

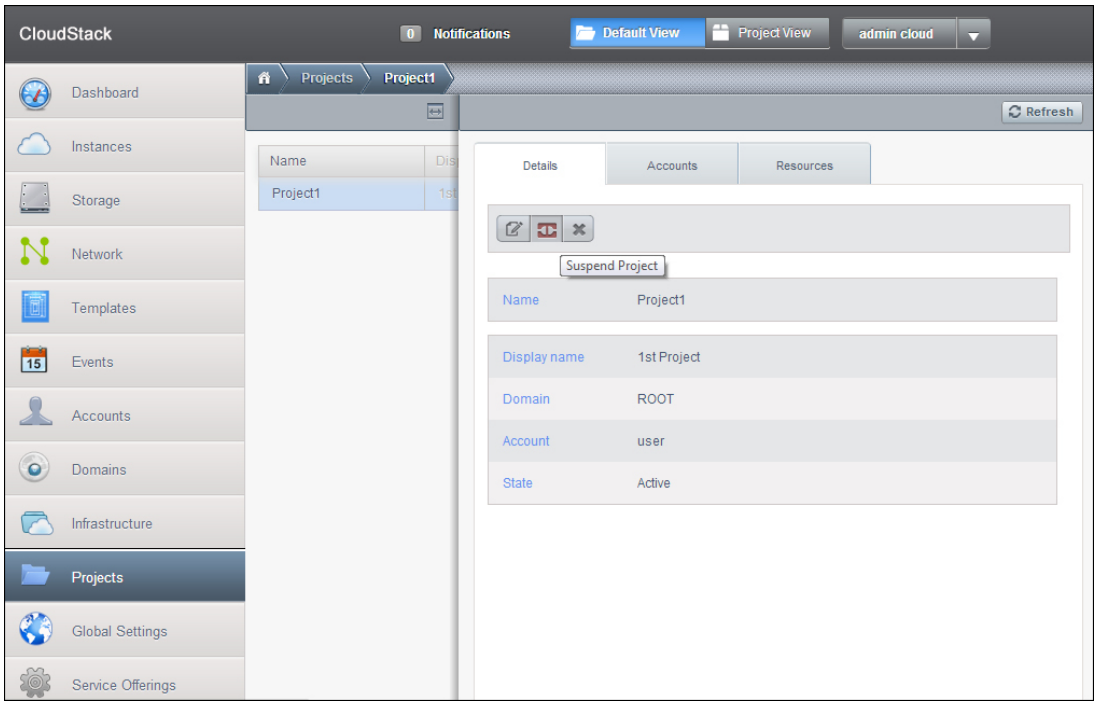


CloudStack projects can also be suspended or deleted as per the need. When a project is suspended, the resources owned by it are still present, but they cannot be used by any user, neither can any user create new resources in that project.

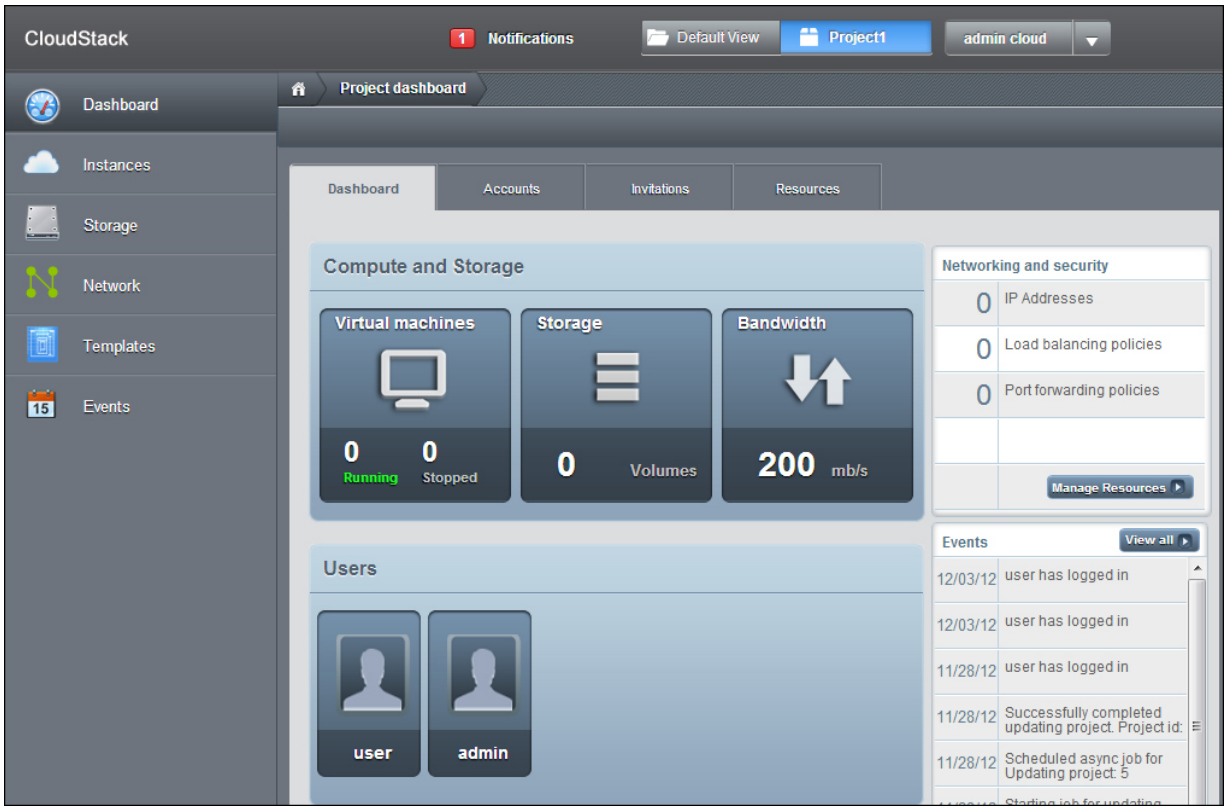
When a project is deleted, all the resources in that project are destroyed and all the members are removed from that project. The status of the project changes to disabled, pending deletion and then after all the resources are removed, the project is deleted.

For suspending or deleting a project follow the steps below:

1. Log in to the CloudStack user interface as a project owner or the domain administrator of the domain to which the project belongs, or as the CloudStack administrator.
2. Navigate to the project tab by clicking on the project in the left panel.
3. Click on the project you want to work on.
4. Click the action (suspend or delete) that you want to perform.



CloudStack provides two different types of views on the user interface for users who are associated with any projects. These users have **Default** and **Project** view in the console. They can switch between any of these two views and see the resources associated with each. While switching to the **Project** view, the users can select which project they want to view. When a user switches to the **Project** view, by selecting a particular project, they can only view information related to that project ; for example, only the VMs provisioned in that project, and so on.



There can be only one project administrator at a time and the project administrator can pass the role of administration to other user.

The users in the project can still create resources outside the project and these resources can be created under the account to which the user belongs. The resources created in the project by any user are owned by the project and any user in the project has the permissions to work on them. The resources that the user creates in a project are still within the project even if the administrator removes the user from the project.

CloudStack also allows creating project-level network configurations in the cloud so that the project traffic is isolated. These network configurations may include any services such as port-forwarding, load balancing, and VPN.

CloudStack projects can also make use of shared resources available outside the project available within the domain. The project can use any offerings defined in its domain.

Summary

In this chapter we introduced you to the various levels of users in CloudStack and how they can be isolated from each other or grouped to make the best use of the cloud in your organization. In the next chapter, we will introduce you to:

- The high availability architecture for CloudStack
- How users can ensure their resources are highly available in the cloud
- How to scale resources in the cloud and the support of CloudStack

Chapter 8. High Availability and Scaling

In the previous chapters, we have discussed the different components and configuration of CloudStack. In this chapter we are going to discuss the following:

- Highly available deployment of CloudStack
- High availability manager in CloudStack
- Deploying applications in the CloudStack environment in a high availability configuration
- Scaling applications up and down in CloudStack

Ensuring high availability in CloudStack

High availability is an important aspect in a cloud computing environment. It ensures minimum downtime of the cloud service and the applications utilizing the cloud environment. High availability can be ensured by providing multiple instances of a system so that if one of the systems is down, the others continue to provide service and thus ensuring minimum downtime of users' applications.

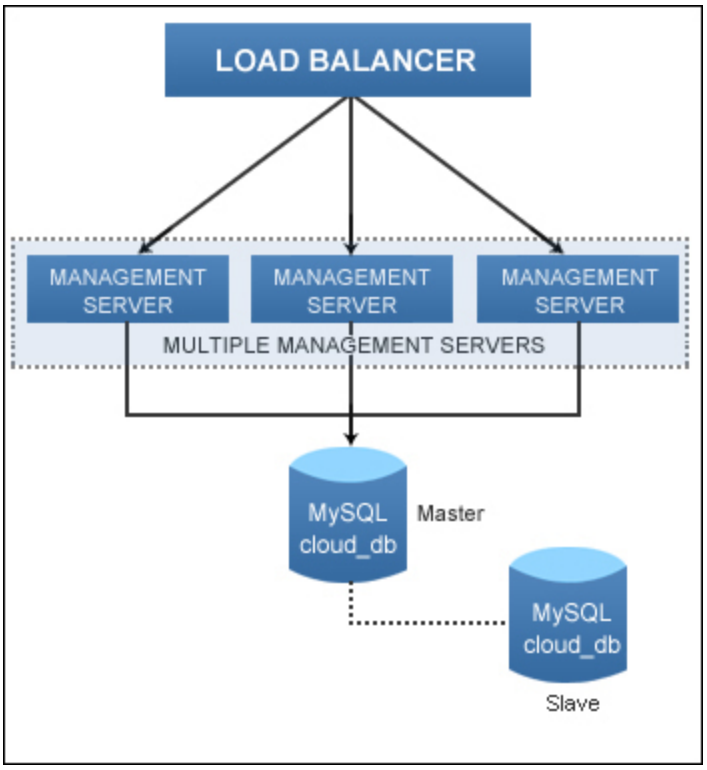
CloudStack provides users with a number of features to ensure high availability of both the CloudStack system as well as the resources deployed using CloudStack. First we will look at how CloudStack ensures high availability of the CloudStack infrastructure and then we will describe the basic components of CloudStack which can be leveraged to ensure the high availability of the VMs deployed in CloudStack.

CloudStack infrastructure high availability

CloudStack allows users to deploy the management server (its core component) in high availability mode so that if one of them is down, the other continues to service and there is no interruption in the service. These multiple management servers can be load balanced as well, so that the requests can be distributed among them.

All the management servers use a MySQL database in the backend that can also be deployed in high availability mode by using MySQL Replication.The MySQL cluster has a master among them and other servers replicate the data from the master, so that if the master is down, one of the slave nodes can be promoted to start functioning as the master. The MySQL database provides manual failover in the event of data loss.

We have already seen the Multi node installation of CloudStack in the installation chapter. In a multi-node installation of CloudStack, the database and the management server are deployed on two different systems. The management server can be installed on multiple systems to provide high availability of the management server. These multiple management servers are stateless and are behind a load balancer. Since the management servers are stateless any request to any of them would yield the same result. They are in active-active mode, which means all the management servers are continuously working at any point in time. The load balancer helps in distributing the request load between the multiple management servers. All the management servers interact with the same MySQL database in the backend.



The preceding diagram describes the architecture of a multi-node highly available CloudStack deployment. It is clear that if any of the management servers are down for any reason, the other management servers can continue to function. While deploying CloudStack, one can add more features to it by deploying different management servers on different physical hosts so that the chances of failure become less. However it should be noted that since all management servers connect to the same MySQL server, the database server is the single point of failure. There will be some time lag between the master database server failing and the slave server being promoted to master.

CloudStack redundant virtual router

As mentioned earlier, CloudStack uses a virtual router to provide several network services. These virtual routers are created for each network which is created from a network offering that has a virtual router to provide several network services. This virtual router is very much critical to the cloud, because it provides the network services to the guest virtual machines. CloudStack provides a facility to

enable a redundant virtual router in the guest network. This redundant virtual router provides the guest network the ability to recover and continue operation if one virtual router is down due to any reason. If the user enables this feature, there will be two virtual routers in every network. One of them is known as the **master** that functions actively and accepts/responds to the user VM's request, and other is called the **backup** and it is present in the passive mode. In case the master router is down, the backup router takes its place. The backup router is activated in a few seconds of time and ensures minimum downtime. The backup router after activating becomes the master router and another router is launched as a new backup.

To enable this feature of redundant virtual router in CloudStack, follow these steps:

1. Create a new network offering as described in Chapter 4, *Apache CloudStack Networking*.
2. When the offering contains a virtual router as a service provider for source NAT, the option for **Redundant router capability** should be visible. Enable it.
3. Create a new network with the network offering as described in Chapter 4, *Apache CloudStack Networking*.
4. For all the existing networks, click on the **Update** button to update the network to include the redundant router capability.

For recovering the network if the router is down, in a network with redundant router capability enabled, follow these steps:

1. First the administrator must be sure that the router VM is lost and cannot be recovered forever. Only if it is true, go to next step.
2. Destroy the router VM.
3. Recreate a router, and then create a working pair of redundant router.

In case the router appears to be running, but the host is already disconnected, follow these steps:

1. Stop the router using the `stopRouter` function with the force option.
2. Ensure that the other router is running and then destroy this router.
3. Restart the network with `cleanup` option set to `false`.

CloudStack storage high availability

In this section we shall discuss storage high availability for both primary and secondary storage.

Primary storage failure

The Primary storage used by CloudStack is used to store its root and data disks. In case of a failure of the Primary storage the instances having storage disks on the failed storage device can crash.

In case of primary storage failure, the hypervisor immediately stops all VMs stored on that storage device. Guests that are marked for HA will be restarted when the primary storage comes back on line. CloudStack supports multiple primary storages to one availability zone for high availability, however automatic failover is not available in case of failure of primary storage.

The administrators can use other approaches, such as a Distributed Replication Block Device or other storage level replication techniques to provide high availability in case of primary storage failure.

In case iSCSI disks are used for providing primary storage, the administrator can use disk mirroring and RAID to provide high availability at the storage tier.

Secondary storage failure

A secondary storage failure in CloudStack will not impact the running guests as the guests root devices and data storage is not on the

secondary storage but primary storage. However, since the snapshots are stored on secondary storage, the users will not be able to use these features and thus no snapshots can be taken or restored till the time the secondary storage is unavailable. Templates are also placed on the secondary storage. Thus a secondary storage failure will impact the templates, ISO images and snapshots since these items are stored in secondary storage. Secondary storage should be backed up using storage replication and other such techniques to prevent loss of templates, ISO images and snapshots.

In case there are multiple secondary storages configured in a zone, the templates marked as public will be replicated to other secondary storages for redundancy and thus will remain available in case of failure of one of the secondary storages.

CloudStack and high availability

CloudStack provides in-built features which helps in deploying and maintaining a highly available system. CloudStack supports Network Interface Card bonding and the use of separate networks for storage as well as iSCSI multipath which helps in ensuring high availability in various situations. Let's see more about how CloudStack can be leveraged to provide high availability to VM instances.

CloudStack HighAvailabilityManager

CloudStack has a component called **HighAvailabilityManager** which provides the process for implementing high availability for the infrastructure. CloudStack HighAvailabilityManager provides a well-defined process and works with many other components of CloudStack to provide high availability features in CloudStack. The process of HighAvailabilityManager can be described in three simple steps; they are:

- **Investigation**: Determining the state of the VM, like finding out whether the VM is up or down or it is impossible to determine its status.
- **Fencing**: This fences the VM and restricts the usage of network or storage. This is done to prevent any corruption of the VM.
- **Start**: Turns on the VM instances.

HighAvailabilityManager performs the investigation and fencing with the use of adapters. There can be multiple adapters for each of these steps. VM instances can be deployed in several ways in the cloud; finding out the status of these instances can vary, so there can be multiple adapters for a process like this.

The adapters can be added or removed from the `components.xml` file. The first two steps bring the VMs status to halt/stop if not already, so the start operation is performed in order to bring the VM instance up again. For starting the instance, HighAvailabilityManager uses the normal way to start operation.

1. HighAvailabilityManager leverages the investigator to perform the first step for knowing the status of VM instance. CloudStack supports multiple types of hypervisors. The investigator will perform the operation and can return three

states; running, stopped or unknown. The investigator sends the unknown state when it is incapable of determining the state of the VM or the environment the VM is in. An investigator which is written and is integrated with VCenter using its APIs should report unknown as the status of a VMrunning on XenServer. The investigator must be implemented properly so that it returns the correct state for any VM instance.

2. HighAvailabilityManager uses Fencer to fence the VM instance so that it is fenced from corrupting its own disk or networks. The fencer returns true if it has performed the task successfully and false if the task was not completed due to any reason.

The process followed by HighAvailabilityManager:

HighAvailabilityManager has its own queue and the process is carried out in various steps. It keeps the VM HA process updated as the steps are performed in the database. A VM instance can be scheduled for HA in the following conditions:

- The host on which the VM was running is found to be down.
- The host might be disconnected for more than a configurable amount of time (30 minutes, by default).
- The instance which is stuck at the start process for more than one hour.
- A management server which determines that some another management server has gone down, and is unwinding any VM instance which was been started by the peer.
- A management server on the restart, unwinding any VM instance which was being started by it.

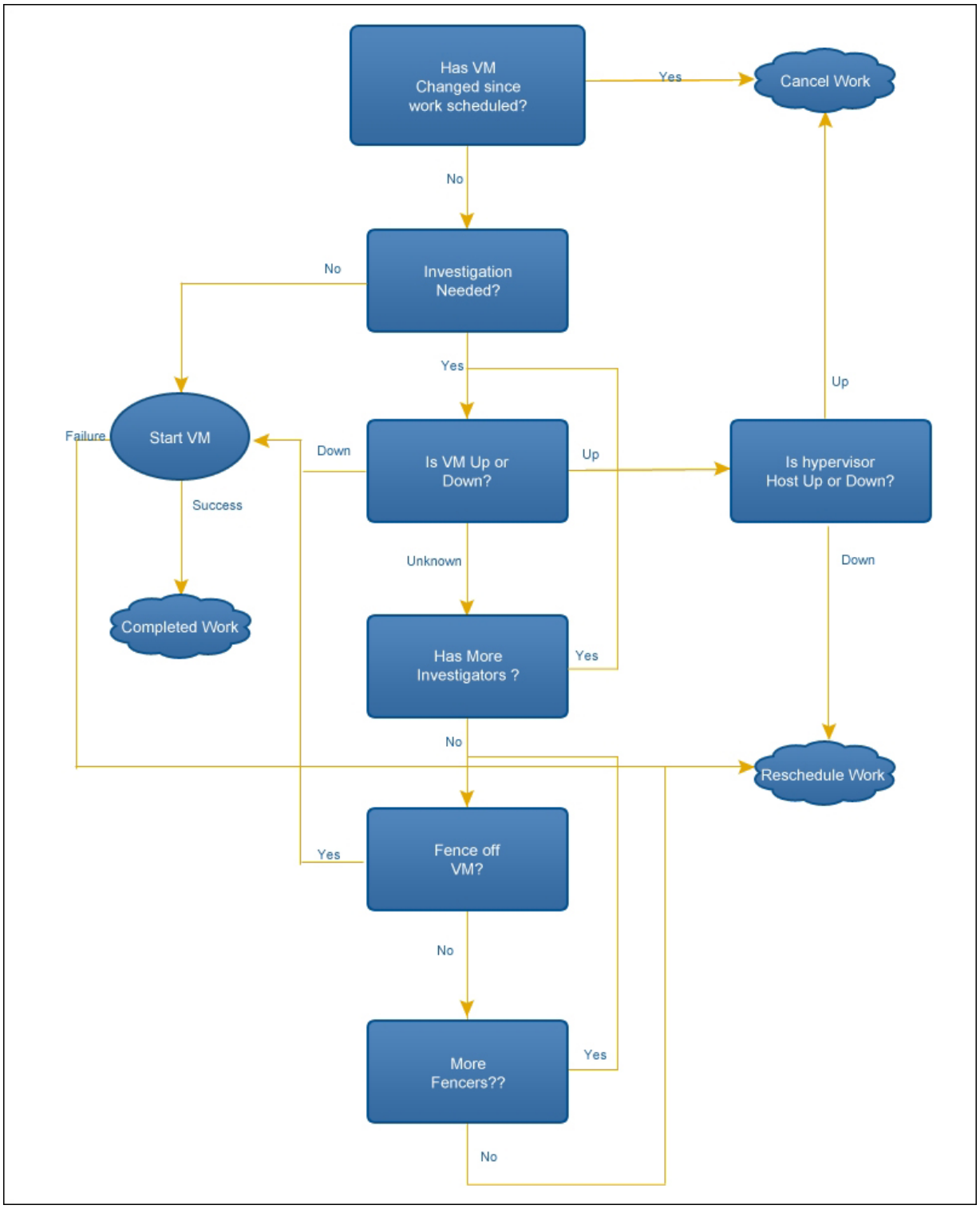
HighAvailabilityManager – the Queue

CloudStack HighAvailabiiltyManager uses the Queue component to run the following actions:

- If any of the above conditions are met, the HighAvailabilityManager starts the worker threads which will then start the process in the queue and update the database accordingly.
- During each stage of this process, HighAvailabilityManager first determines the change in state of the VM instance and takes action based on it.
- If the state has changed, the HA process might be terminated and if the state has not changed, it checks whether there is an investigation required again for the VM instance. If there is a requirement for investigation, HighAvailabilityManager asks each Investigator to investigate the VM until any one of them reports that the VM instance is stopped or running.
- If the VM instance is found to be running in between any of the steps in the queue, then HighAvailabilityManager checks the status of the host on which the VM is running and if that host is also up, then the queue is terminated and if the host is found to be down, HighAvailabilityManager reschedules the work item.
- If none of the investigators are able to find the state of the VM, HighAvailabilityManager initiates the fencing of the VM. If any of the fencers return the fenced state for the VM, it restarts the VM instance. If none of the fencers can complete the task, the work item is rescheduled. Once the VM is restarted successfully, the process is marked as complete.
- If the VM start fails with an error that HighAvailabilityManager recognizes, the work is rescheduled, else the task is cancelled and reported as failed.

As mentioned, the HighAvailabilityManager updates the state of the process in the database. The state can be checked in the `op_ha_work` table. CloudStack also has a cleanup process which deletes the cancelled or completed tasks.

The whole process can be depicted in the following flowchart.



HighAvailabilityManager can be configured as well for some of the parameters like the ones given below. To edit these configuration parameters, follow these steps:

1. Log in to the CloudStack interface as root administrator.
2. Go to the **Global settings** page and search for the following:

    - `ha.retry.wait`: This parameter defines the time to wait before retrying the work.
    - `stop.retry.wait`: This parameter defines the time to wait before retrying stop.
    - `time.between.cleanup`: This parameter defines the time to wait before the cleanup thread runs.
    - `max.retries`: This parameter defines the number of times to retry the start operation.
    - `time.to.sleep`: This parameter defines the time to sleep in case there are no work items.
    - `workers`: This parameter defines the number of worker threads to spin off for the processing.

High availability of applications running on Cloudstack

CloudStack allows the users to deploy the application in a highly available fashion by providing access to basic building blocks. Users can ensure that their application is highly available by choosing the deployment of their applications. The physical resources in CloudStack are divided into various logical groups and the users have the ability to customize the deployment of the application into these various logical groups.

The users, in order to design a highly available application, can chose between these logical groups linking to the physical resources to create a highly available application.

For example, the users can choose to deploy multiple copies of application servers on to different hosts in a cluster in a single zone with a load balancer to forward requests to these servers. This scenario prevents the downtime of the application even if one of the hosts in that cluster is down. In case any host is down due to any reason in the above deployment, the application will be running on the other hosts with the load balancer forwarding requests to the servers on other hosts.

While the users chose to deploy their application in a highly available manner, they can also create clones of their applications in different zones in a passive state to recovery from some unwanted disaster. By this kind of deployment, the users can recover from disaster by starting the application servers on different zones or pods. For example, you can create two copies of your application servers in two different zone or different pods with one in active state and one is passive state. The data in the active servers are continuously backed up and synchronized with the other zone. As these zones and pods are linked to different physical environment, when the active state applications are down, the copies of the application

servers and the data in the other physical region can be started with a minimum amount of downtime.

The users can take care of the following while deploying the applications:

- Create different components of applications on different physical resources.
- Create multiple copies of the application components and deploy them on different physical resources.
- Use a load balancer to balance the load among different servers on different hosts.
- Create a backup of resources in the other zones or pods for disaster recovery.
- Deploy different components of applications in different networks with proper security configuration.

For example, we will discuss the deployment of a three tier application in a highly available manner in CloudStack in following example:

- We can deploy multiple instances of the web servers on different hosts in the same cluster with a load balancer configured to distribute load among them.
- These web servers can have incoming ports open from the Internet and ssh ports open only from the users' own IP addresses.
- The middle tier can be deployed in the same fashion on different hosts with security ports open only from the IP address of the web servers.
- The databases can also be deployed on different hosts with a shared storage available to multiple instances and some read replicas. All the read operations take place through the replicas of the master database and write operations are done on the master database.
- The data is continuously synchronized from the master database to the slaves and in case of failure of the master, one of the slaves is promoted as the master.

CloudStack storage migration

CloudStack also supports storage migration between hosts which allows users to migrate VM instances from one host to another in the same cluster or different clusters in the same pod or different pods in the same zone.

For migrating the VM instance, the destination can be flexible which allows the VM instance to migrate to a specific host, or just cluster/pod with some id or storage pool by providing the ID.

CloudStack allows storage migration for all the hypervisors such as XenServer, KVM, and VMware.

A VM instance may be migrated from one host to another in the following scenarios:

- In cases when the load on the storage pool is to be distributed. If any storage pool gets heavily loaded, then some of its VM instances can be migrated to another storage pool so that the usage is balanced.
- In order to simplify resource management, the VM instance can be migrated to any host where resources are available.
- In order to enhance the availability of the VM, in case the host, or the storage pool or cluster to which the VM instance belongs is in some kind of trouble, the VM can be migrated to a safer place.

The storage migration in CloudStack is carried out using an API **MigrateVMCmd** that requires parameters to migrate the VM, such as `pod-id`, `cluster-id`, `host-id` and `VM-id`. Among these parameters, the `VM-id` is required and the other parameters are optional. The VM migration is dependent on the parameters provided for example, if the cluster-id is present and the host-id is not given, the VM will be migrated to the any of the hosts in the cluster with the required space.

The process is as follows:

- Whenever the user passes the API call, all the parameters are validated and capability of the destination is also checked to ensure the hosting of the VM instance.
- CloudStack creates a migrate worker **Value Object** (**vo**) in the database.
- A runner thread is also created with the worker thread and submits it to the thread pool.
- The runner then stops the VM, initiates the copy of the volume from source to destination, and starts the VM again.
- If the process is successful, the job is marked completed else it is marked as failed.

The storage migration involves the copying of the data volume of the VM instance from the source to the destination. So, this may consume a lot of bandwidth. CloudStack supports the use of separate networks for storage as well as it supports iSCSI multi path.

The usage of separate paths for the storage ensures that the network's bandwidth doesn't get choked while performing tasks like storage migration.

Scaling in CloudStack

CloudStack provides a feature of auto scaling of applications horizontally, that is, by increasing the number of instances of the application. This is an important feature as the users might need to auto scale instances based on the number of requests. Auto scaling features allow scaling up as well as scaling down the servers running users' application or services based on various conditions. These conditions can be defined by users. For example, if the number of requests increases for a web application running on a server is more than the amount one server of a particular configuration can handle, the server must be scaled out in order to ensure smooth functioning of the application. The users can define conditions, such as CPU utilization, based on which the servers can scale up. For example, users can define that if the CPU utilization of a server goes above 60 percent for more than 10 minutes continuously, CloudStack should provision a replica of the server to handle the requests. Now, these two servers are behind the load balancer which distributes the requests to the servers at the backend, thus ensuring optimum use of resources.

These servers which can auto scale up and down are grouped together in a group known as an auto scale group and this auto scale group is registered with the load balancer. Thus, the load balancer distributes the requests to the auto scale group servers. The scaling can be defined based on alarms or conditions which require continuous monitoring of the servers in the auto scale group.

The conditions based on which the scaling is to happen may vary from simple conditions such as CPU utilization to complex conditions such as measuring the responsiveness of the group of servers. NetScaler load balancer can be used for load balancing the requests to the group of servers. The load balancers monitor the health of the servers in the group and initiate the scaling up and down of resources.

There are some standard terms associated with this feature, which are used to define an auto scaling group. Let's look at them one by one.

Counters

Counters are related to the performance of the VM instances and are used for monitoring the health of the backend services and the servers. These counters can be SNMP-based counters or can be other counters that are supported by the load balancers.

Conditions

Conditions are the criteria based on which the action is taken on the auto scaling group. It utilizes the counters defined above. Conditions are defined based on the counters and time frame. For example, a condition defined as *x* percent of CPU Utilization for *y* minutes can be used to trigger an auto scaling action. These conditions can be used to trigger an auto scale action.

Auto scale policy

Every auto scaling group has an auto scaling policy which defines the auto scaling action based on the conditions defined in the group. The auto scaling action can be used to scale up or scale down the number of servers.

Auto scale VM profile

Auto scale VM profile defines the profile of the VM that is to be used in the autoscaling group. It may be defined based on the some templates or service offerings, etc. Whenever any machine is provisioned based on the scaling group, the VM instance will be created using the profile defined here.

Auto scale VM group

The auto scale VM group is used to associate the policies with the load balancing rules. We can define the size of the group in the auto scale VM group. The size of the group defines the maximum number of instances that can be present in the group.

The modules described above are defined in CloudStack. It provides various APIs for performing actions whereas the modules described below are a part of the NetScaler system.

Collector/Monitor

The collector or the monitor collects information from the guest VM instances periodically. The timers are defined for the collector to collect data from the instances based on some given metrics. The timer can be configured by users. The monitoring framework in NetScaler supports packet inspection.

Aggregator

The aggregator collects the metric values based on the algorithm defined such as average, maximum, and minimum. The aggregator collects metric data from the collector and computes the sum or average, as configured for a given time duration.

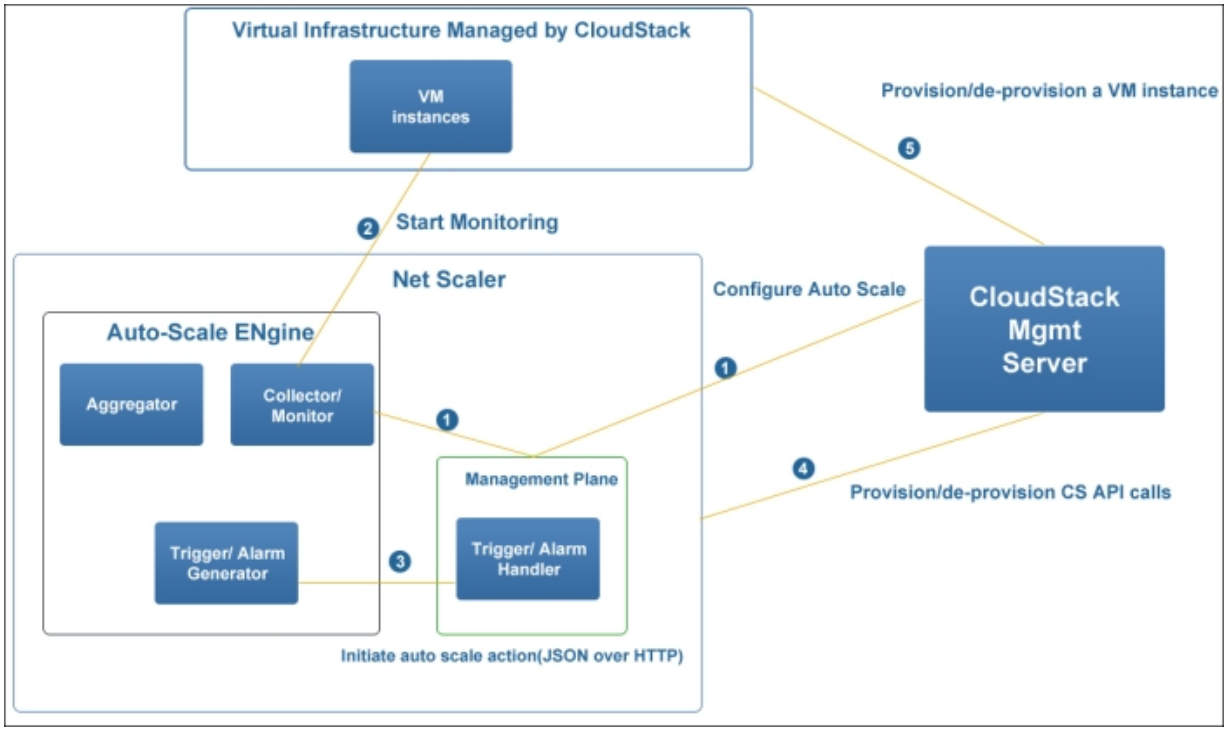
Trigger/alarm generator

The triggers are alarms which monitor the output of the aggregators, and then evaluate the auto scale conditions based on the all these parameters, and if the conditions are satisfied, generates a trigger or an alarm. The trigger generator runs its job on a scheduled basis.

Trigger/alarm handler

The trigger/alarm generator passes the trigger/alarm to the trigger/alarm handler over HTTP using a JSON payload. The handler in turn evaluates the alarm and initiates the action of scale up/down in the autoscaling group.

The following diagram defines the autoscale group workflow and the placement on each of the components defined above.



Steps executed while autoscaling:

1. CloudStack management server is used to configure the autoscaling group and based on the configuration provided, it registers a NetScaler load balancer with the autoscaling group.
2. The NetScaler management plane defines the trigger/alarm handler, trigger generator, collector, and aggregator.
3. The collector then starts the monitoring of the VM instances in the group.
4. Based on the trigger generated, the trigger handler triggers the CloudStack management server to

provision/decommission a VM instance in the autoscale group.
5. Based on the trigger the CloudStack Management Server creates the virtual machine instances.

Autoscaling can be configured in CloudStack using the APIs provided and any other device like NetScaler.

Autoscaling and high availability features play an important part in any cloud environment. These features increase the application's availability and performance by providing a minimum amount of downtime and scaling up and down the number of resources based on the number of requests to optimize the resources used for an application and thus minimizing cost and resource utilization. For example, High Availability ensures there are replicas of applications running in case one of them is down and scaling ensures the optimum number of servers as per the needs of the application at any given point in time.

Summary

In this chapter, we have covered how we can use CloudStack to deploy a highly available application, how CloudStack helps in providing high availability and scaling of applications, how the components of CloudStack such as primary, secondary storage is provided in highly available mode, and the recovery of CloudStack components on failure and how to configure redundancy.

In the next chapter we will introduce to you the various ways in which we can extend CloudStack to meet the needs of your organization. We will cover customizing the User Interface, extending reporting, and extending the networking capabilities of CloudStack.

Chapter 9. Extending Apache CloudStack and Performance Tuning

In the previous chapters we have discussed the different aspects of the CloudStack architecture and have gone through the installation and configuration of CloudStack. In this chapter we are going to discuss the following:

- Extending CloudStack to fulfill our organizational requirements
- The basic components of CloudStack to be used for extending and integrating with other tools
- Using CloudStack Usage Server for monitoring the usage
- Enhancing CloudStack to cater to the organization's need

Extending CloudStack

Being open source, CloudStack provides users with the ability to modify and use it as per their requirements. Users can customize CloudStack in several ways such as extending its network features, storage features, authorization features, and modifying its dashboard. CloudStack exposes application programming interfaces for users to use and integrate with other tools.

Extending CloudStack networking

As described in Chapter 4, *Apache CloudStack Networking*, CloudStack implements its networking features using many different components which users can take advantage of to create a customized service of their own.

CloudStack provides a module-based implementation of classes interfaces, and dynamic configuration management system using which the users can modify the management of different types of networks in CloudStack.

Using these classes interfaces, the users have the ability to replace the traditional way in which CloudStack manages the guest networks. The basic components of the CloudStack networking module are mentioned as follows, the detailed functioning of each is given in Chapter 4, *Apache CloudStack Networking*.

- **Network Guru**
    - Takes care of the design and implementation of the virtual networks
    - IP address management
    - Exists at `com.cloud.network.guru.NetworkGuru`
- **Network Element**
    - Represents the components present in the CloudStack network
    - Helps the components provide any kind of network service
    - Supports the virtual networking infrastructure and its interfaces
    - Is implemented in `com.cloud.network.element.NetworkElement`
- **Network Managers**
    - Handles the resources managed by the network elements
    - Extends the `com.cloud.utils.component.Manager` interface
    - Loaded at startup through the CloudStack's configuration manager

    - For example, the L2-L3 networks that are set up using Open vSwitch is managed by `com.cloud.network.OvsTunnelManagerImpl`
    - Another example is the virtual router lifecycle that is managed by `com.cloud.network.router.VirtualApplianceManagerImpl`
- **Resources**
    - Abstracts the physical as well as virtual resources that are being managed by CloudStack.
    - Different resource classes are defined such as hypervisors, load balancers, and storage devices
    - They are also the manager classes which are an implementation of `com.cloud.resource.ServerResource` that extends the `Manager` class

CloudStack allows us to add new network components which can be managed using the resources. The components can be managers, elements, resources, or network gurus which can be added to the `component.xml` file. Whenever a new component is added to CloudStack, it is instantiated using the **component locator**. The users can define the components using the following steps:

1. Edit the `components.xml` file located at `client/tomcatconf/components.xml`.
2. For adding new network gurus, find the adapters key for Network Guru using the following:

```
<adapters key
="com.cloud.network.guru.NetworkGuru">
```

3. Add the entries for the new network guru component as:

```
<adapter name="AnotherGuestNetworkGuru"
class="com.cloud.network.guru.AnotherGuestN
etworkGuru">
```

4. For adding the network element(s), find the adapter for it and make the new entries:

```
<adapters
key="com.cloud.network.NetworkElement">
<adapter name="Another"
class="com.cloud.network.element.AnotherEle
ment"/>
```

Just like defining the network gurus and network element(s), we can define new components for network managers and resources, as per our requirements.

The users can either edit the `components.xml` file or extend the component library by specifying the library in `components.xml` as follows:

```
<management-server
class="com.cloud.server.ManagementServerExtImpl"
library="com.cloud.configuration.AnotherComponentL
ibrary">
```

After the components are defined in `components.xml` or in the extended library, they can be referenced using the `@Inject` annotation. This annotation loads the instance for the component loaded from the component locator.

The users might need to add a global configuration parameter for enabling the network managers. It must be ensured while coding that the code must also check the value for the parameters in order to ensure that gurus/elements/managers are not triggered for the wrong kind of network.

Users can declare the new network component and create new components so that they can create custom virtual guest network services and manage them via CloudStack. Using their knowledge of the different network components, users can plug new network solutions into CloudStack and provide it as a service by creating network offerings using the new network elements.

Integrating NetScaler with CloudStack

Features such as the load balancing service for the guest VMs are provided using the CloudStack's virtual router's HA proxy as well as external network elements, which can also provide capabilities such as firewalls and load balancing. CloudStack supports third-party devices such as Juniper SRX for providing external firewall and load balancing feature and F5 BigIP as an external load balancer device.

CloudStack also supports integration with Citrix's NetScaler to provide network services for the guest VMs in the cloud. Citrix's NetScaler is available with different options providing capabilities such as application firewalls and load balancing.

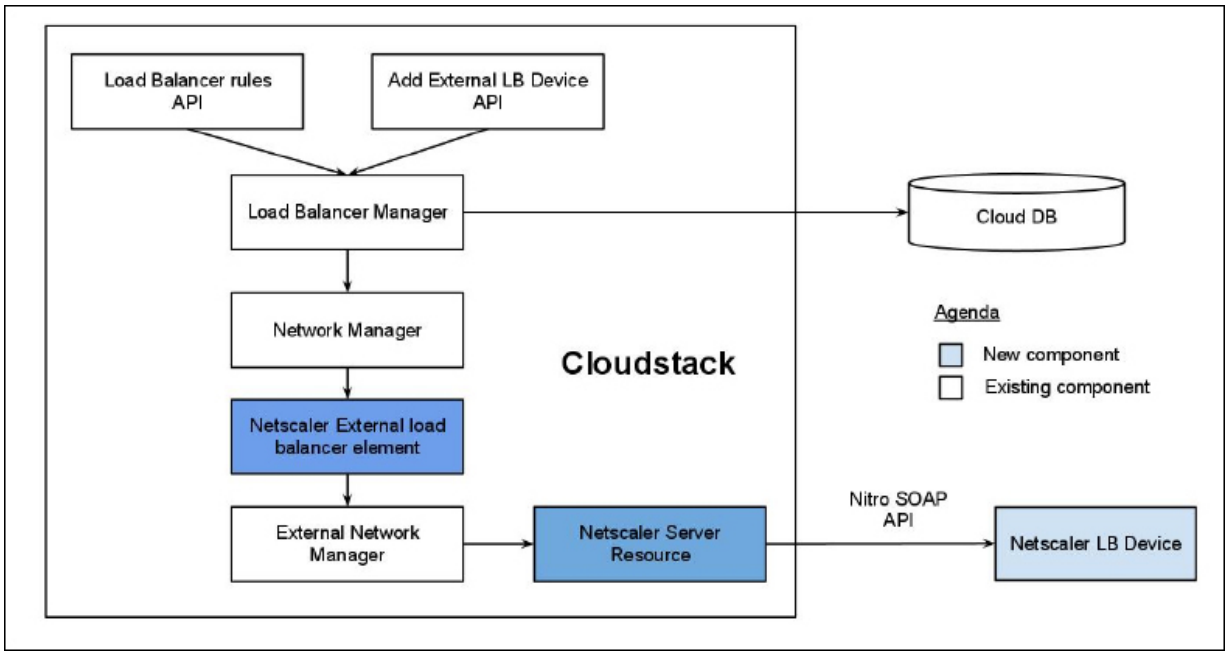
Citrix NetScaler is available as a network device or it can be used as a virtual appliance. It has rich capabilities and is used as a leading load balancer and application firewall.

NetScaler is made available in three different forms and all these forms can be integrated with Apache CloudStack to provide their services to the users guest VMs in the cloud.

The three different forms of NetScaler offerings are:

- NetScaler MPX, which is basically a hardware offering
- NetScaler VPX is a software-based virtual appliance and can run on VMware ESXi, Citrix XenServer, and Microsoft Hyper-V
- NetScaler SDX, which is a fully multi-tenant solution and can be used to run multiple instances NetScaler VPX on a single appliance

The following diagram shows the context diagram for the integration of NetScaler devices with CloudStack to function as an external load balancer in a zone with virtual networking configured:



For this integration, some of the existing network element framework like the existing agent framework will be used as well as a new network element for NetScaler load balancer will be added to the CloudStack.

The NetScaler load balancer will be used and managed as a server resource.

Integrating NetScaler in CloudStack has some functional requirements which must be met in order to use the functionalities of NetScaler, they are described as follows:

- The integration of NetScaler as load balancers in CloudStack will be supported as an external load balancer device only in the zones that are configured for virtual networking, and only one external load balancer will be supported in one zone.
- This NetScaler load balancer, once configured, will be used by the virtual guest networks for balancing load among them and the virtual router should not serve as the load balancer.
- The NetScaler load balancer should be configured with two network interfaces; one for the public network and one for the private network.

- It cannot be added into a zone which already has LB rules deployed over the guest VMs in the zone and it cannot be removed from a zone until all the LB rules have been deleted for the guest VMs.
- Once the NetScaler load balancer is added to the CloudStack zone, it should deny creating load balanced rules using the default public IP address that is associated with the account.
- The NetScaler load balancer must support round robin technique s and least connection load balancing methods. In addition, it should be able to track load balancing stats such as usage, bytes sent, and bytes received.

There are various other functional requirements (explained in the next section) which must be fulfilled in order to add a NetScaler load balancer in a zone to balance the load of the guest virtual machines. Using this integration we can provide NetScaler load balancer as a network service in cloud.

Functional requirements

In order to integrate NetScaler with CloudStack, there are some functional requirements which should be met; they are as follows:

- In the case of an advanced zone, NetScaler should be supported as the load balancing service provider for isolated networks.
- In the case of a basic zone, NetScaler should be supported as EIP and ELB service provider.
- NetScaler integration should support the load balancing methodologies such as round robin and least connection.
- The persistence mechanisms that should be supported are LB cookie, App cookie, and source-based persistence methods.
- Each physical network in a zone should be able to provision multiple NetScaler devices in CloudStack.
- Dedicated and shared provisioning of NetScaler should be supported in CloudStack per tenant. Shared provisioning of

NetScaler allows multiple tenants to use a single NetScaler load balancer and in case of a dedicated NetScaler load balancer, one load balancer is dedicatedly allocated to a tenant.

- The CloudStack administrator should be able to provision a dedicated and shared NetScaler load balancer.
- The administrator should also be able to specify the capacity of the shared NetScaler load balancer and it should not exceed the number of tenants that are allowed to use the load balancer.
- The type of consumption of the NetScaler (dedicated or shared) should be configured in the network offerings.
- CloudStack should be responsible for provisioning a dedicated NetScaler load balancer in case the network offering has a dedicated NetScaler load balancer feature.
- In case the network offering provides a shared load balancer, CloudStack should be responsible for provisioning a shared load balancer from the pool of dedicated load balancers that are provisioned in a zone.
- NetScaler VPX provisioning using a NetScaler SDX should also be supported by CloudStack. CloudStack should be able to provision VPX on NetScaler SDX and also add the provisioned VPX to the pool of the NetScaler load balancers in the zone.
- The NetScaler VPX provisioned using the NetScaler SDX load balancer must be usable as a shared or dedicated load balancer in CloudStack.
- The NetScaler load balancers provisioned in the cloud should be able to track the usage statistics such as bytes sent and received per load balancer rule.
- The NetScaler device that has been provisioned in the cloud should be removable only when there is no tenant in the zone using the appliance.

The NetScaler load balancer can be added to a zone to serve as an external load balancer. The administrator must provide the login credentials to the NetScaler otherwise the addition will fail.

Along with the credentials, the public and private interface of the NetScaler external load balancer should be provided.

The load balancer feature in the NetScaler load balancer must be enabled in case CloudStack fails to enable it.

Guest network with NetScaler load balancer

When a new guest network is created using a network offering with NetScaler load balancer feature enabled, it is created using the following series of steps:

1. Depending upon the type of NetScaler load balancer provided in the network offering, CloudStack picks up a NetScaler load balancer pool in the zone. If it is a dedicated load balancer, then CloudStack selects a free load balancer from the pool.
2. In case NetScaler SDX load balancer in configured in the network offering, CloudStack attempts to deploy a NetScaler VPX load balancer over SDX and allocates the provisioned load balancer to the network.
3. In case the network offering has a shared load balancer, CloudStack tries to allocate a load balancer with free capacity which can accommodate the network and if there are no free NetScaler load balancers available in the pool, CloudStack will try to provision a NetScaler VPX load balancer on SDX load balancer.
4. CloudStack maintains `network_external_lb_device_map` in which it maintains the mapping of the NetScaler device ID and network ID once the network is provisioned.
5. CloudStack also configures the private network interface ID for the allocated NetScaler device so that it becomes a part of the virtual guest network. It does so by binding the private network interface of the load balancer to the VLAN and the subnet allocated for the virtual guest network. The private

interface that is allocated is self-IP; that is, the second IP from the guest subnet.

LB rule with public IP

CloudStack also creates a load balancer virtual server on the NetScaler LB device which is assigned for the network using the public IP and port whenever the LB rule is added on the public IP.

Assigning a VM to the load balancer rule

Whenever a guest VM is assigned to a load balancer rule, a new server and a service instance is created using the guest VM's IP, and port using the guest's IP and port number on the NetScaler load balancer device.

This service is bound to the load balancer virtual server corresponding to the load balancer rule.

Unassigning a VM from a load balancer rule

Whenever a guest VM in the CloudStack is removed from the load balancer rule, the service and server corresponding to the VM's IP and port on the NetScaler device are unbound and then deleted.

Deleting a load balancer from a zone

A load balancer can be deleted from a zone only when there is no load balancer rule deployed in the zone.

When the NetScaler load balancer is deleted from the CloudStack zone, the zone is re-configured to use the virtual router for the load balancer services.

Load balancer with EIP in a basic zone

In a basic zone with an EIP service enabled, CloudStack allocates a private IP and a public IP address to the guest VM which is provisioned in the zone.

CloudStack also ensures that there is static NAT established between the public IP address and the private IP address.

When the VM is created

CloudStack configures an **Inbound NAT** (**INAT**) on the NetScaler device which creates a mapping between the public IP and the private IP addresses.

It also creates a **Routable NAT** (**RNAT**) rule on the NetScaler device which is used for reverse mapping between the public IP and private IP addresses.

When the VM is destroyed

When the guest VM is deleted, the INAT and the RNAT rules created on the NetScaler device are destroyed.

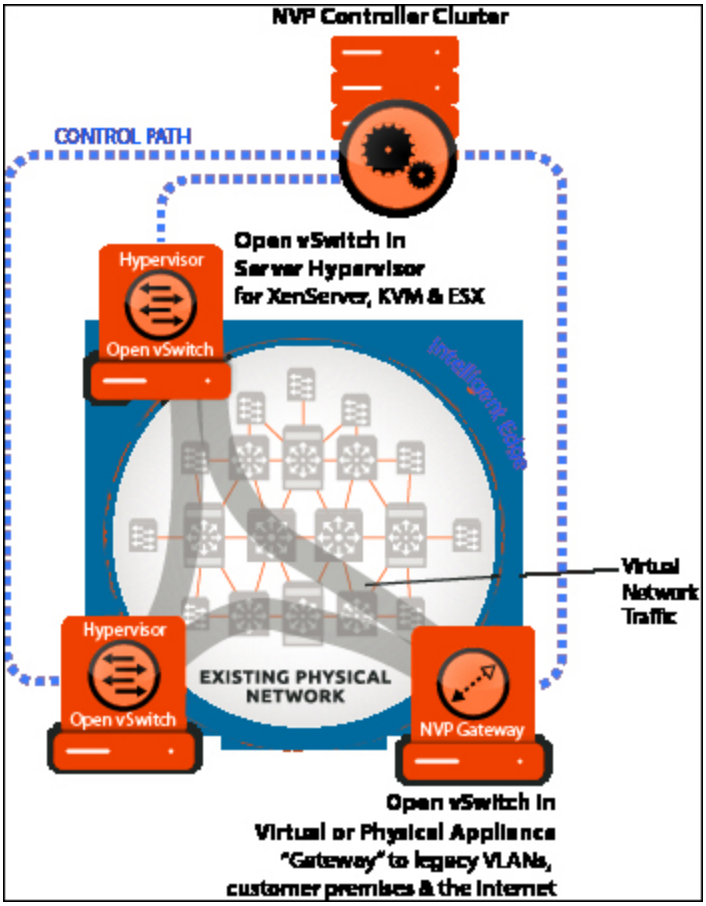
Note

As per CloudStack Version 4.0.x, a default public IP cannot be used for load balancing purpose when we are using an external load balancer in CloudStack. Thus, whenever we configure a zone with an external load balancer, a new public IP address is fetched to apply the load balancer rules.

CloudStack with Nicira NVP

Nicira NVP is used to provide network virtualization in CloudStack. Network virtualization helps in decoupling the virtual networks from the underlying network hardware. After the virtualization is enabled, the physical network is used only for packet forwarding and is treated as an IP backplane.

The network virtualization allows us to create a virtual network programmatically and it operates completely decoupled from the underlying hardware while also offering the same features as the physical network.



Nicira NVP is a software solution which creates an intelligent abstraction layer between the hosts and the existing physical

network.

It allows the creation of a pool of network capacity and also enables programmatic creation of tens of thousands of isolated virtual networks with the resources connected in the cloud.

Nicira NVP is managed by a distributed controller system which is an intelligent edge and helps in transforming the physical network into a pool of networks.

Open vSwtich (**OVS**) is basically a software switch which helps in creating an abstraction layer between the servers and the physical network.

The controller cluster is a highly available distributed system that is used to manage all virtualized network components and connections. The controller cluster defines the virtual networks and is capable of controlling and managing thousands of OVS edge devices

Note

CloudStack 4.0.0 allows integration of Nicira NVP for XenServer hypervisors using the Nicira plugin.

The Nicira plugin has been tested with Nicira NVP Version 2.1.0, 2.2.0, and 2.2.1.

CloudStack allows using Nicira NVP as a network provider to create a virtualized network for CloudStack networks and services. This is done in CloudStack using the Nicira NVP plugin.

The Nicira NVP plugin is used to create Layer-2 networks to support the networks created by the guests. So, whenever a tenant wants to create a new network instead of a traditional VLAN, Nicira NVP can

help to create a logical network. This is done by API calls to the Nicira NVP controller.

> **Note**
>
> As per the CloudStack Version 4.0.0, all these operations are not provided in the web console, so they are to be performed using the API or command line tool.

Integrating Nicira NVP to CloudStack

Before moving onto the integration of Nicira NVP to CloudStack, let us review some of the prerequisites to do so:

- CloudStack must have at least one physical network whose isolation method is set to STT and it should also be enabled for the guest traffic
- The guest traffic label and the name of the integration bridge on XenServer should be same

To continue with the integration of CloudStack, we should have the following information:

- The NVP controller's IP address
- The username to access the API
- The password to access the API
- The transport zone's UUID that contains the hypervisors in the zone
- The physical network's UUID that will used for the guest networks

Every guest network created has its broadcast type set to a logical switch which is created for this network on the NVP controller.

If the state of the network is `Implemented`, the broadcast URI has the UUID of the logical switch.

The NICs connected to the logical switches will have their logical switch UUID listed in the database table `nicira_nvp_nic_map`.

Enabling the network service provider in CloudStack

In order to use the Nicira NVP plugin in CloudStack, the network service provider has to be enabled on the physical network which can be done using the following API calls:

```
addNetworkServiceProvider
name = "NiciraNVP"
physicalnetworkid = <the uuid of the physical
network>

updateNetworkServiceProvider
id = <the provider uuid returned by the previous
call>
state = "Enabled"
```

When we configure a new zone, we need to create a new physical network for the guest traffic and select STT for the isolation type as given in the following screen:



We also need to set the Xen guest traffic label to the same as the label of the integration bridge.

To add a new Nicira NVP device, we can use the following API commands:

```
addNiciraNvpDevice
physicalnetworkid=<see step 1>,
hostname=<hostname or IP of the controller>
username=<admin username>
password=<admin password>
transportzoneuuid=<transport zone uuid>
```

Using Nicira NVP

When we create a new guest network, we must create it in the physical network with its isolation type set to STT. The `NiciraNvpNetworkGuru` will configure a logical switch on the NVP controller when the first virtual machine is launched in the guest network.

When this virtual machine is starting up in the guest network, the `NiciraNvpElement` will create logical ports for the NICs of the virtual machine in the guest networks and then attach these NICs to the existing logical switch.

As per CloudStack Version 4.0.x, all the Nicira NVP setup is considered as a device which can be added and removed from a physical network. This device is added to the physical network using the API call `addNiciraNVPDevice` which enables the plugin on the physical network and any guest networks created on that network will be provisioned using the Nicira NVP controller.

To delete this device, we can use:

```
deleteNiciraNVPDevice
nvpdeviceid: the UUID of the device.
```

And to list such devices we can use:

```
listNiciraNVPDevices
```

Integrating with OpenStack object storage (Swift)

CloudStack uses secondary storage to store ISOs, snapshots, templates, and so on. As discussed in Chapter 5, *Apache Cloud Storage*, we can provide multiple storage devices to work as the secondary storage in CloudStack.

We can also use OpenStack object storage, commonly known as **Swift** to be used as the secondary storage in the CloudStack environment. Secondary storage in CloudStack is an object storage which can be accessed by all the hosts in all the clusters in all the pods and in all the zones.

Instead of using any other device, CloudStack also supports using Swift as a part of the storage infrastructure in cloud for increasing the performance charts. Swift acts as an underlying storage engine for the secondary storage. Swift cluster is scalable and replaces the secondary storage for storage only. For example, when creating a VM from a template, the template is first copied from Swift to secondary storage and then it is downloaded to the primary storage while VM creation.

The Swift storage act as the cloud-wide storage and all the items that were previously stored in secondary storage is now stored in Swift. The secondary storage just acts as the staging area for the resources. Any resource being stored in Swift will pass through the secondary storage and any resource being retrieved from Swift will be passed down to the secondary storage. Swift is a flat storage system where every object is stored as an object inside a container.

CloudStack provides direct support for adding a Swift object store while adding infrastructure resources to the cloud as discussed in Chapter 5, *Apache Cloud Storage*.

Customizing the CloudStack user interface

Though the default user interface that CloudStack provides is easy to use, the users might want to change the user interface. CloudStack provides various APIs to do so.

CloudStack UI is built using the technologies such as HTML/JSP, CSS, and JavaScript, and it uses jQuery 1.4 as the JavaScript library for all the AJAX calls, event handling, and animations. The current UI shows the resources dashboard, infrastructure configuration page, and many other features, but users may use the APIs to modify and implement some more functions via UI.

All the UI-related files are present on the management server at `/usr/share/cloud/management/webapps/client/`. You may find various files in this directory which implement the UI for CloudStack. The users can change the dashboard style by editing the CSS file which implements the dashboard style configuration.

The users can also change various parameters, as we will see in the following sections.

Changing the API path

The default path for the API is configured as `client/api`, if you want to change the default path, use the following steps:

1. Open the file at `/usr/share/cloud/management/webapps/client/WEB-INF/web.xml`.
2. You will find the xml tags like:

```
<servlet-mapping>
<servlet-name>apiServlet</servlet-name>
<url-pattern>/api/*</url-pattern>
</servlet-mapping>
```

3. Change the value in the `<url-pattern>` tag to the new path that you desire.
4. After changing the API path, you will have to modify the `cloud.core.init.js` file.
5. Open the file at `/usr/share/cloud/management/webapps/client/scripts/cloud.core.init.js`.
6. There is a query in this file as follows:

```
$.ajaxSetup({
url: "/client/api",
dataType: "json",
cache: false,
error: function(XMLHttpResponse) {
handleError(XMLHttpResponse);
},
beforeSend: function(XMLHttpRequest) {
if (g_mySession == $.cookie("JSESSIONID"))
{
return true;
} else {
$("#dialog_session_expired").dialog("open")
;
return false;
}
}
});
```

7. Change `url /client/api` to the new API path.
8. Once this is done, all the AJAX calls will be made to the new location specified.

Changing the session timeout

The users can also change the default session timeout for the UI from 30 minutes.

1. Open the file at `/usr/share/cloud/management/webapps/client/WEB-INF/web.xml`.
2. You will need to add or modify the following XML tag:

```
<session-config>
<session-timeout>60</session-timeout>
</ session-config>
```

The value defines the session timeout period in minutes.

3. Restart the management server as follows:

```
#service cloud-management restart.
```

Single sign on integration

The CloudStack UI is implemented using the session-based CloudStack APIs. After logging into the CloudStack UI, all the requests that the users make are accompanied by the `JSESSIONID` cookie. This cookie is used until the session is terminated. The `cloud.core.callback.js` file has a method called `onLogoutCallback()`, which can be modified to redirect the user to the portal you desire after the session has timed out. This file also includes the sample AJAX login API call to the management server.

If the management server and the portal are in different network domains, then you must make the login API from the CloudStack domain otherwise all the request will be rejected for security reasons. Single sign on can be integrated into CloudStack via multiple ways, including:

- The traditional method
    - The users can use the traditional way for integrating the existing portal with CloudStack to execute the API call for login on behalf of the user.
    - The users will have to construct the login command along with the parameters such as username, account, domain, and so on to log into the CloudStack. After the login is successful, ensure that the global variable `g_loginresponse` is set to the JSON response of the login API call.

        - The user portal must have a link to the CloudStack UI containing the information to be used for proper login API call.
        - The modified file `cloud.core.callback.js` takes the request, intercepts the link and constructs the login call and executes it against the `/client/api` URL.
        - After successful login, the browser will automatically set the `JSESSIONID` cookie and the global variable `g_loginResponse` must be set to the JSON response.
- The shared key method: This method is also very similar to the traditional way. There is an additional security feature which is implemented by hashing the URL with a shared secret key while making the login API command.

        - The users need to pass in the API key and some parameters such as domain ID, username, timestamp, and signature.
        - The sample API login request make look like the following:

          ```
          https://<server>:8080/client/api?
          command=login&username=XXX&domainid=NNN&timestamp=
          YYY&signature=<secure-hash>
          ```

        - The secret key from the CloudStack database existing under the configuration table for the key `security.singlesignon.key` must be retrieved and copied to the application which is to be integrated with CloudStack.
        - The timestamp parameter is the current system time in milliseconds.
        - There is also a parameter for tolerance `security.singlesignon.tolerance.millis` which is also available in the configuration table and can be modified as per the user. This parameter specifies the time in milliseconds which is allowed between making the SSO request and receiving the SSO request.

- The timestamp value that is passed in the CloudStack login request must be within the sum of management server time and the tolerance time.

Integrating with LDAP for user authentication

CloudStack also allows the use of external LDAP servers such as Microsoft Active Directory or ApacheDS for end user authentication. The accounts in CloudStack are mapped to the external LDAP account using query filters.

The integration also includes the search base which directs the searches to the LDAP searches. CloudStack doesn't allow user provisioning in LDAP, so it is done outside of CloudStack.

CloudStack provides an API for the integration with LDAP, the API is `ldapConfig` and the following parameters are passed when making the API call.

- **Hostname**: This is the hostname of the LDAP server.
- **Port**: This is the port at which the LDAP server listens. The default port is 389.
- **SSL**: This parameter must be set to true to enable or else false.
- **Search base**: It tells the CloudStack the part of the external directory tree which is to be used for searching. The tree contains all the users in the LDAP.
- **QueryFilter**: It is used to find an external user mapped to the LDAP server. It must uniquely map each user in the LDAP to CloudStack so that the identity integrity is maintained.
- **Bind**: This represents the user in LDAP who is permitted to perform searches on the LDAP directory. This user will be permitted to search the entire CloudStack directory. Its role includes searching the LDAP directory using the QueryFilter for authenticating the cloud users. After the search result is returned, the distinguished name and passed password is used to authenticate the cloud user using the LDAP bind.
- **Bindpass**: The password for the user to bind.

LDAP user provisioning

Users can be provisioned in the integrated LDAP by the usage of the CloudStack account creation API by passing the username, e-mail, first name, and last name of the user to be provisioned, along with the domain, time zone, and the account name. The username and e-mail must be unique in the whole LDAP server as one of these will be used in QueryFilter.

The usage server

CloudStack has an optional feature—the usage server—which helps in providing the usage records of all the resources in the cloud. The usage records can be used to bill the users in the cloud as per their usage of resources. This is a separately installed part of the CloudStack.

The usage server works by pulling data from the event's log and creates the summary of the usage records and also facilitates the usage of API call `listUsageRecords` to access the usage records.

The usage records consist of the usage of resources in Cloud such as VM run time and template storage space. The run time of the usage server can be configured and it defaults to at least once per day.

The usage server(s) can be installed after the installation of the management server and it must be installed on the same server on which the management server is installed. If there are multiple management servers, the usage servers can be installed on any number of them.

For the installation of the usage server, follow these steps:

1. Ensure that the management server service is running.
2. Run the `install.sh` script.
3. Choose "S" to install the usage server when prompted.
4. Start the usage server after the installation completes using the following command:

```
#service cloud-usage start
```

Configure the usage server post-installation using the following steps:

1. Log into the CloudStack UI as the root administrator.

2. Go to the **Global Settings** page.
3. Find the configuration parameters that you want to edit (the list is provided further) and click on the action to change the value.
4. After successful editing the configuration parameters, restart the management server and the usage server:

```
#service cloud-management restart
#service cloud-usage restart
```

The configuration parameters that you can edit for the usage server are listed as follows:

- `enable.usage.server`: Specifies whether the usage server is active or not
- `usage.aggregation.timezone`: Specifies the time zone of the usage records
- `usage.execution.timezone`: Specifies the time for `usage.stats.job.exec.time`, this will execute and collect the data. It defaults to the time zone of the management server
- `usage.sanity.check.interval`: Specifies the time interval between the sanity checks. It helps us to validate the data
- `usage.stats.job.aggregation.range`: Specifies the time period between the processing jobs of the usage server for processing the records in minutes
- `usage.stats.job.exec.time`: Specifies the time for running the usage server processing

Performance tuning

CloudStack has many components, so the performance tuning of CloudStack involves ensuring that each of its components is working as desired. Different components have different requirements in order to function properly. As a cloud administrator, we can ensure some properties which will help in tuning the performance of CloudStack.

Some of those parameters are discussed as follows.

Increasing the management server maximum memory

We can increase the memory allocated to the management server which will help the management server to handle the workload. The management server is the most important and critical component of CloudStack. All the resources are managed by the management server. We can increase the memory of the management server. To do so, use the following steps:

1. Edit the configuration file at /`etc/cloud/management/tomcat6.conf`.
2. Change the parameter `-XmxNNNm` to a higher value of `N`. This will denote the amount of memory allotted to the management server in MB. For example, Xmx1024m provides 1024 MB of memory to the management server.
3. Restart the management server:

```
#service cloud-management restart
```

Database buffer pool size

The database performance in CloudStack can be fine tuned to ensure better performance. For this we can increase the database buffer pool size. To do this, use the following steps:

1. Edit the MySQL configuration file at `/etc/my.cnf`.
2. Insert the following line in the `[mysqld]` section below the `datadir` line:

```
Innodb_buffer_pool-size=700M
```

3. If the line is already there, then increase the value to a much higher value to ensure smooth function of MySQL database. In case the database and management server are on the same server, it is recommended to set 40 percent of the RAM and if the database is on a separate server, set the value to 70 percent of the RAM.
4. Restart the CloudStack management server.

Setting and monitoring the hosts' capacity

The physical servers/hosts that are added to the CloudStack have some maximum limit for hosting VMs. We should continuously monitor the hosts so that they are not overloaded. In order ensure that overloading of hosts does not occur, we can set some upper limit on the total VMs that can run on a host apart from monitoring them. Using the monitoring feature, we can continuously get the update of the capacity of the hosts and it can also generate alerts when a certain threshold is crossed.

While setting the upper limit on the maximum number of VMs on a host, we must also take care of the situation when one or more host fails; otherwise, it would lead to an overload on the other hosts. So we must set the maximum limit very carefully, considering the load and resource utilization in mind.

The maximum limit of the VMs on the host can be set by editing the configuration variable in the global settings page.

Capping the resource usage

CloudStack allows setting limits on the resource usage by providing a cap which specifies the maximum limit of usage of the resources by the users in the cloud. As we have discussed, the limits can be set on the accounts, domains, and projects, CloudStack allows the configuration of some limits as a part of the global configuration parameter which can only be edited by the root administrator while other configurations are applied at the ROOT domain and can also be overridden on a per account basis. The parameters that can be set as the global configuration parameters are as follows:

- `max.account.public.ips`: The maximum number of public IPs which can be owned by any account
- `max.account.snapshots`: The maximum number of snapshots which can exist for an account at any time
- `max.account.templates`: The maximum number of templates that can exist in an account at any time
- `max.account.user.vms`: The maximum number of guest VMs in an account at any point of time
- `max.account.volumes`: The maximum number of volumes in an account
- `max.template.iso.size`: The maximum size of any ISO
- `max.volume.size.gb`: The maximum size of any volume in an account
- `network.throttling.rate`: The default data transfer rate in MBPS allowed per user
- `snapshot.max.hourly`: The maximum number of recurring snapshots to be retained for a volume
- `snapshot.max.daily`: The maximum number of daily snapshots that should be retained for a volume
- `snapshot.max.weekly`: The maximum number of weekly snapshots that should be retained for a volume
- `snapshot.max.monthly`: The maximum number of monthly snapshots that should be retained for a volume

Setting the limits on the CloudStack resources helps in maintaining the optimum usage of resources in the cloud. Apart from the global configuration parameters, there can be other parameters using which the administrators can set limits for the resources. These parameters can be edited at the domain level or the account level.

Summary

With this chapter our journey to learning Apache CloudStack comes to an end. We have learnt to install, configure, and administer Apache CloudStack in the previous chapters and in the last chapter we have learned advanced techniques to extend the capabilities of CloudStack as per our requirements. We also learned how to fine tune and optimize the various components of CloudStack for optimum performance.

We sincerely hope that our readers find this book useful and leverage the knowledge gained to implement and use one of the best open source cloud management software packages: Apache CloudStack.

Index

A

- access control, CloudStack management server / Access control
- access control management
    - about / Networks and security
- access switches
    - about / Access switches or L2 switches
- account
    - creating / Creating an account, Creating an account
- accounts
    - about / Accounts
    - and projects / Projects and accounts
- Accounts tab
    - about / Accounts tab
- admin accounts
    - about / Root domain
- administrators
    - cloud administrator / Administrators
    - project administrator / Administrators
    - domain administrator / Administrators

- Advanced Networking option
    - about / Zones and their types
- advanced zone
    - about / Zones and their types, Advanced Zone
- advanced zone configuration
    - about / Advanced Zone configuration
    - parameters / Advanced Zone configuration
    - options, for adding zones / Advanced Zone configuration
    - template, creating / Creating a template
    - VM, importing to CloudStack / Importing a VM to CloudStack
    - hosts, preparing / Preparing the hosts
- Agent Manager
    - about / Kernel
- aggregator
    - about / Aggregator
- ApacheDS / Integrating with LDAP for user authentication
- API layer, CloudStack management server / API layer
- API path
    - modifying / Changing the API path
- apt repo
    - setting up / Setting up the apt repo
- Async Job Manager

    - about / Kernel
- automation layer / Automation
- auto scale policy
    - about / Auto scale policy
- auto scale VM group
    - about / Auto scale VM group
- auto scale VM profile
    - about / Auto scale VM profile
- autoscaling
    - steps / Trigger/alarm handler

B

- backup.snapshot.wait parameter / Creating a template from a snapshot
- BackupSnapshotCommand command / Creating a template from a snapshot
- backup virtual router / CloudStack redundant virtual router
- basic zone
    - about / Zones and their types
- basic zone configuration
    - about / Basic Zone configuration
    - parameters / Basic Zone configuration
    - network traffic / Basic Zone configuration
    - options, for adding NetScaler device / Basic Zone configuration
    - options, for setting up public traffic / Basic Zone configuration

    - options, for adding Pods / Basic Zone configuration
    - options, for storage traffic / Basic Zone configuration
    - options, for cluster details / Basic Zone configuration
    - options, for host configuration values / Basic Zone configuration
    - options, for adding primary storage / Basic Zone configuration
- business logic
    - about / CloudStack management server

C

- characteristics, cloud / Introducing cloud
- CIFS
    - about / Primary storage
- Citrix XenServer / CloudStack configuration
    - about / Citrix XenServer
- Classless Inter-Domain Range (CIDR) / L3 network configuration in CloudStack
- cloud
    - about / Introducing cloud
    - deployment models / Introducing cloud
    - characteristics / Introducing cloud
    - service models / Introducing cloud
    - versus virtualized Datacenter / Introducing cloud
    - infrastructure layer / Infrastructure layer
    - management layer / Management layer
- cloud.core.callback.js file / Single sign on integration
- cloud administrator
    - about / Administrators

- CloudBridge
    - about / API layer
- cloud computing
    - about / Introducing cloud
- CloudDB
    - about / CloudDB
- Cloudstack
    - high availability of applications, running / High availability of applications running on Cloudstack
- CloudStack
    - about / Introducing cloud, Understanding CloudStack modules
    - modules / Understanding CloudStack modules
    - deployment model / Cloud deployment model
    - pre-installation tasks / Pre-installation tasks
    - configuring / CloudStack configuration
    - IT infrastructure, adding to / Infrastructure
    - VM, importing to / Importing a VM to CloudStack
    - external devices, using / Using external devices with CloudStack
    - hypervisor / CloudStack with different hypervisor
    - high availability, ensuring / Ensuring high availability in CloudStack
    - and high availability / CloudStack and high availability
    - scaling / Scaling in CloudStack
    - extending / Extending CloudStack
    - NetScaler, integrating with / Integrating NetScaler with CloudStack

    - Nicira NVP, integrating with / Integrating Nicira NVP to CloudStack
    - network service provider, enabling / Enabling the network service provider in CloudStack
    - integrating, with Swift / Integrating with OpenStack object storage (Swift)
    - user interface, customizing / Customizing the CloudStack user interface
    - integrating, with LDAP / Integrating with LDAP for user authentication
- CloudStack configuration
    - management server console / Management server console
    - service offerings / Service offerings
- CloudStack infrastructure high availability
    - about / CloudStack infrastructure high availability
- CloudStack management server
    - about / CloudStack management server
    - functionalities / CloudStack management server
    - units / CloudStack management server
    - API layer / API layer
    - access control / Access control
- CloudStack modules
    - about / Understanding CloudStack modules
    - deployment model / Cloud deployment model
    - Storage / Storage
    - secondary storage / Secondary storage
    - management server / CloudStack management server
    - kernel / Kernel
    - CloudDB / CloudDB

    - networking architecture / CloudStack networking architecture
- CloudStack MySQL database
    - installing / Installing and configuring CloudStack MySQL database
- CloudStack networking
    - extending / Extending CloudStack networking
- CloudStack networking components
    - about / CloudStack networking components
    - NetworkGuru / NetworkGuru
    - network elements / Network element
    - network managers / Network managers
    - resources / Resources
    - networking flows / CloudStack networking flows
- CloudStack Operations
    - about / CloudStack operations
- CloudStack redundant virtual router
    - about / CloudStack redundant virtual router
    - enabling, steps / CloudStack redundant virtual router
- CloudStack storage high availability
    - about / CloudStack storage high availability
    - primary storage failure / Primary storage failure
    - secondary storage failure / Secondary storage failure
- CloudStack storage migration
    - about / CloudStack storage migration

- CloudStack user interface
    - customizing / Customizing the CloudStack user interface
    - API path, modifying / Changing the API path
    - session timeout, modifying / Changing the session timeout
    - single sign on integration / Single sign on integration
- CloudStack virtual router
    - about / CloudStack virtual router
    - networking / Networking using CloudStack virtual router
- cluster
    - about / Clusters, Preparing the hosts
- collector
    - about / Collector/Monitor
- community cloud
    - about / Introducing cloud
- component locator
    - about / Extending CloudStack networking
- components.xml file / CloudStack HighAvailabilityManager
    - about / Extending CloudStack networking
- compute offerings / Compute offerings
    - about / Compute offering

- computing resources
    - about / Computing resources
- conditions
    - about / Conditions
- configuration, CloudStack / CloudStack configuration
- configuration, primary storage / System requirements and configuration
- configuration parameters, HighAvailabilityManager
    - editing / HighAvailabilityManager – the Queue
- controllers
    - about / CloudStack management server
- counters
    - about / Counters
- create.private.template.from.snapshot.wait parameter / Creating a template from a snapshot
- create.volume.from.snapshot.wait parameter / Creating a template from a snapshot

D

- DAS
    - about / Primary storage
- Dashboard tab
    - about / Dashboard tab

- database
    - configuring / Database installation and configuration
    - installing / Database installation and configuration
- database_key / Database installation and configuration
- dbpassword / Database installation and configuration
- Deb packages
    - building / Building Deb packages
- deploy-as / Database installation and configuration
- deployment model, CloudStack
    - about / Cloud deployment model
    - zones / Zones
- deployment models, cloud
    - private / Introducing cloud
    - public / Introducing cloud
    - community / Introducing cloud
    - hybrid / Introducing cloud
- dev-cloud
    - about / Understanding CloudStack modules
- DHCP
    - about / Introducing service offerings and virtual machines
- DHPC / Network offerings
- Direct-attached Storage (DAS) / Storage
- disk and memory snapshot / Snapshots
- disk offerings / Disk offerings
    - about / Disk offering

- disk snapshot / Snapshots
- Distinguished Name (DN)
    - about / Administrators
- DNS / Network offerings
    - about / Introducing service offerings and virtual machines
- domain
    - creating / Creating a domain, Creating a domain
    - about / Domain
- domain administrator
    - about / Administrators
- Domains tab
    - about / Domains tab

E

- Egress rule / Security groups
- elastic IP address
    - about / Elastic IP address
- Elastic Load Balancer / Elastic Load Balancer
- enable.usage.server parameter / The usage server
- encryption
    - about / Keys and encryption
- encryption-type / Database installation and configuration
- Events tab

    - about / Events tab
- expunge.delay parameter / Deleting a volume
- expunge.interval parameter / Deleting a volume
- external devices
    - using, with CloudStack / Using external devices with CloudStack

F

- F5 BigIP / Integrating NetScaler with CloudStack
- F5 Load balancer / Firewall and F5 Load balancer
- features, virtual router / Virtual router
- Firefox 3.5+ / CloudStack configuration
- firewall
    - about / L3 network configuration in CloudStack
- Firewall / Network offerings
- Firewall rule
    - creating / Network Address Translation
- FQDN / CloudStack configuration
- full snapshot
    - about / Kernel
- Fully Qualified Domain Name (FQDN) / Preparing the OS
- functionalities, CloudStack management server
    - about / CloudStack management server

G

- gateway

    - about / Networks and security, L3 network configuration in CloudStack
- Generic Routing Encapsulation (GRE) / CloudStack networking architecture
- getProvider() method / Network element
- global settings page
    - about / Global settings
- guest network
    - creating / Network in cloud
- guest network, NetScaler load balancer / Guest network with NetScaler load balancer
    - about / Guest network with NetScaler load balancer
    - LB rule, with public IP / LB rule with public IP
    - VM, assigning to load balancer rule / Assigning a VM to the load balancer rule
    - VM, unassigning from load balancer rule / Unassigning a VM from a load balancer rule
    - load balancer, deleting from zone / Deleting a load balancer from a zone
    - load balancer, with EIP in basic zone / Load balancer with EIP in a basic zone
- Guest Network traffic
    - about / Physical network
- guest traffic
    - about / Basic zone, Guest traffic
- Guest VM
    - volumes, attaching to / Attaching a volume to Guest VM

H

- high availability
    - about / Ensuring high availability in CloudStack
    - ensuring, in CloudStack / Ensuring high availability in CloudStack
    - and CloudStack / CloudStack and high availability
- high availability, of applications
    - running, on Cloudstack / High availability of applications running on Cloudstack
- HighAvailabilityManager
    - about / CloudStack HighAvailabilityManager
    - investigation process / CloudStack HighAvailabilityManager
    - fencing process / CloudStack HighAvailabilityManager
    - start process / CloudStack HighAvailabilityManager
    - process / CloudStack HighAvailabilityManager
    - Queue component, using / HighAvailabilityManager – the Queue
    - configuration parameters, editing / HighAvailabilityManager – the Queue
- Highly Available (HA)
    - about / Compute offering
- hosts
    - preparing / Preparing the hosts
- HTTP / Use of switches in cloud
- HTTPS / Use of switches in cloud
- hybrid cloud

    - about / Introducing cloud
- Hyper-V
    - about / Computing resources
- hypervisor, CloudStack
    - about / CloudStack with different hypervisor
    - Citrix XenServer / Citrix XenServer
    - Oracle VM / Oracle VM
    - RedHat Enterprise Linux (KVM) / RedHat Enterprise Linux (KVM)
    - VMware vSphere / VMware vSphere
- hypervisor layer / Computing resources

- @Inject annotation / Extending CloudStack networking
- IaaS
    - about / Introducing cloud
- IE7+ / CloudStack configuration
- implement() method / NetworkGuru, Network element, CloudStack networking flows
- Inbound NAT (INAT) / When the VM is created
- incremental snapshot
    - about / Kernel
- infrastructure layer, cloud
    - about / Infrastructure layer
    - computing resources / Computing resources
    - storage / Storage
    - network / Networks and security
    - security / Networks and security

- Infrastructure tab
    - about / Infrastructure tab
- installation, management server / Installing additional management server
- installation, usage server / The usage server
- instance
    - volume, detaching from / Detaching a volume from an instance
- Instances tab
    - about / Instances tab
- interface
    - about / CloudStack management server
- invitation setup
    - for users / Invitation setup
- IP address
    - modifying, for secondary storage / Changing secondary storage IP address
- IPAM / L3 network configuration in CloudStack
- IPAM system
    - about / Networks and security
- iSCSI
    - about / Primary storage

/ System requirements and configuration

- isolated network
    - about / Isolated
- isolated networks
    - about / Isolated networks
- isolated virtual network / Virtual networks
- IT infrastructure
    - adding, to CloudStack / Infrastructure
    - basic zone configuration / Basic Zone configuration
    - advanced zone configuration / Advanced Zone configuration
- IT Infrastructure Library (ITIL) / Service Management

J

- Java Simplified Encryption (JASYPT) / Keys and encryption
- JSESSIONID cookie / Single sign on integration
- Juniper SRX device / Using external devices with CloudStack

K

- kernel
    - about / Kernel
    - CloudStack Operations / CloudStack operations
    - security check / Security check
    - virtual machine manager / The virtual machine manager
    - server resources / Server resources
- keys

    - about / Keys and encryption
    - database secret key / Keys and encryption
    - database password / Keys and encryption
    - SSH keys / Keys and encryption
    - compute node root password / Keys and encryption
    - VPN password / Keys and encryption
- KVM / CloudStack configuration
    - about / CloudStack storage migration
- KVM hypervisor / CloudStack configuration

L

- L2 switches
    - about / Access switches or L2 switches
- L3 network configuration, CloudStack
    - IPAM / L3 network configuration in CloudStack
    - gateway / L3 network configuration in CloudStack
    - remotely accessible VPN / L3 network configuration in CloudStack
    - firewall / L3 network configuration in CloudStack
    - NAT / L3 network configuration in CloudStack
    - VPN / L3 network configuration in CloudStack
- layer-3 switch
    - about / Use of switches in cloud
- layer 4-7 switches
    - about / Use of switches in cloud
- LDAP / Administrators

    - about / Domain
    - CloudStack, integrating with / Integrating with LDAP for user authentication
- LDAP user provisioning / LDAP user provisioning
- live migration
    - of VMs, between hosts / Live migration of VMs between hosts
- Load Balancer / Network offerings
- load balancer
    - deleting, from zone / Deleting a load balancer from a zone
- load balancer, with EIP
    - about / Load balancer with EIP in a basic zone
    - VM, creating / When the VM is created
    - VM, destroying / When the VM is destroyed
- load balancer rule
    - VM, assigning to / Assigning a VM to the load balancer rule
    - VM, unassigning from / Unassigning a VM from a load balancer rule
- load balancing rules / Load balancing rules
- load balancing service / Integrating NetScaler with CloudStack

M

- Management console
    - used, for creating template / Creating a template using the Management console

- management layer, cloud
    - about / Management layer
    - automation / Automation
    - orchestration / Orchestration
    - task execution / Task Execution
    - service management / Service Management
- management server
    - installing / Installing additional management server
- management server console
    - about / Management server console
    - Dashboard tab / Dashboard tab
    - Instances tab / Instances tab
    - Storage tab / Storage tab
    - Network tab / Network tab
    - Templates tab / Templates tab
    - Events tab / Events tab
    - Accounts tab / Accounts tab
    - Domains tab / Domains tab
    - Infrastructure tab / Infrastructure tab
    - Projects tab / Projects
    - global settings page / Global settings
- management server installation
    - about / Management server installation, Back to management server installation, Management server installation
    - Deb packages, building / Building Deb packages
    - apt repo, setting up / Setting up the apt repo
    - repository, adding to system / Adding the repository to the system, Adding the repository to the system
    - RPM, building / Building RPM
    - yum repo, creating / Creating yum repo

    - database installation and configuration / Database installation and configuration
    - NFS share, preparing for storage / Preparing Network File System share for storage
    - NFS server, creating / Creating a separate NFS server
    - system VM template, preparing / Preparing the system VM template
- Management traffic
    - about / Physical network
- management traffic
    - about / Basic zone, Management traffic
- management_server_key / Database installation and configuration
- master virtual router / CloudStack redundant virtual router
- max.account.public.ips parameter / Capping the resource usage
- max.account.snapshots parameter / Creating a template from a snapshot, Capping the resource usage
- max.account.templates parameter / Capping the resource usage
- max.account.user.vms parameter / Capping the resource usage
- max.account.volumes parameter / Capping the resource usage
- max.project.public.ips / Resource management in Projects
- max.project.snapshots / Resource management in Projects
- max.project.snapshots parameter / Creating a template from a snapshot
- max.project.templates / Resource management in Projects
- max.project.uservms / Resource management in Projects
- max.project.volumes / Resource management in Projects
- max.retries parameter / HighAvailabilityManager – the Queue

- max.template.iso.size parameter / Capping the resource usage
- max.volume.size.gb parameter / Capping the resource usage
- MD5 hash
    - about / Management server installation
- member
    - removing, from project / Removing a member from a project
- members
    - adding, to projects / Adding members to the project
- Microsoft Active Directory / Integrating with LDAP for user authentication
- migrate.retry.interval parameter / Live migration of VMs between hosts
- MigrateVMCmd / CloudStack storage migration
- migratewait parameter / Live migration of VMs between hosts
- multinode installation
    - about / Pre-installation tasks, Multinode installation
    - management server installation / Management server installation
    - CloudStack MySQL database, configuring / Installing and configuring CloudStack MySQL database
    - CloudStack MySQL database, installing / Installing and configuring CloudStack MySQL database
- MySQL cluster / CloudStack infrastructure high availability
- MySQL database / CloudStack infrastructure high availability
- MySQL Replication / CloudStack infrastructure high availability

N

- NAS
    - about / Primary storage
- NAT
    - about / Networks and security, L3 network configuration in CloudStack
- NetScaler
    - integrating, with CloudStack / Integrating NetScaler with CloudStack
    - offerings / Integrating NetScaler with CloudStack
- NetScaler, with CloudStack
    - functional requisites / Integrating NetScaler with CloudStack, Functional requirements
- NetScaler appliance / Elastic IP address
- NetScaler load balancer
    - guest network / Guest network with NetScaler load balancer
- network
    - about / Network in cloud
- network.throttling.rate parameter / Capping the resource usage
- Network Address Translation (NAT)
    - about / Network Address Translation
    - disabling / Network Address Translation
    - enabling / Network Address Translation
    - Virtual Private Network / Virtual Private Network

- Network Attached Storage (NAS) / Storage
- network configuration
    - of zones / Network in cloud
- network element
    - about / Network element
- network elements
    - about / Extending CloudStack networking
- Network File System (NFS)
    - about / Secondary storage
- NetworkGuru
    - about / NetworkGuru
- Network Guru / Extending CloudStack networking
- networking
    - about / Networks and security
    - with CloudStack virtual router / Networking using CloudStack virtual router
- networking architecture, CloudStack
    - about / CloudStack networking architecture
    - network service providers / Network service providers
    - network offerings / CloudStack network offerings
    - network types / Types of network in CloudStack
    - L3 network configuration / L3 network configuration in CloudStack
    - access switches / Access switches or L2 switches
    - L2 switches / Access switches or L2 switches
    - virtual router / CloudStack virtual router

    - firewall / Firewall and F5 Load balancer
    - F5 Load balancer / Firewall and F5 Load balancer
    - security groups / Security groups
- networking flows
    - about / CloudStack networking flows
- networking services
    - NAT / Networks and security
    - IPAM System/DHCP / Networks and security
    - access control management / Networks and security
- network manager
    - about / Kernel
- Network Managers / Extending CloudStack networking
- network managers
    - about / Network managers
- network offerings / Network offerings
    - about / Network offerings
    - enabled state / Network offerings
    - disabled state / Network offerings
    - inactive state / Network offerings
- network service provider
    - enabling, in CloudStack / Enabling the network service provider in CloudStack
- network service providers / Network service providers
- network services
    - about / Network services
    - public IP addresses / Public IP addresses

    - elastic IP address / Elastic IP address
- Network tab
    - about / Network tab
- network traffic
    - types / Basic Zone configuration
- network traffic, advanced zone
    - about / Advanced Zone
    - guest traffic / Guest traffic
    - isolated / Isolated
    - shared / Shared
    - management / Management traffic
    - public traffic / Public traffic
    - storage traffic / Storage traffic
- network traffic, basic zone
    - guest traffic / Basic zone
    - management traffic / Basic zone
    - storage traffic / Basic zone
- network types, CloudStack
    - physical / Physical network
    - virtual / Virtual network
    - isolated / Isolated networks
    - shared / Shared networks
- NFS
    - about / Primary storage
- NFS server
    - creating / Creating a separate NFS server

/ System requirements and configuration
- NFS shares
    - preparing, for storage / Preparing Network File System share for storage
- Nicira NVP
    - about / CloudStack with Nicira NVP
    - integrating, with CloudStack / Integrating Nicira NVP to CloudStack
    - using / Using Nicira NVP

O

- onLogoutCallback() method / Single sign on integration
- OpenStack object storage (Swift)
    - using / Using OpenStack object storage (Swift)
- Open Vswitch / Use of switches in cloud
- Open vSwtich (OVS)
    - about / CloudStack with Nicira NVP
- Oracle VM / System requirements and configuration
    - about / Oracle VM
- orchestration engine
    - about / CloudStack management server
- Orchestration layer / Orchestration

P

- PaaS
    - about / Introducing cloud
- performance tuning
    - about / Performance tuning
    - management server maximum memory, increasing / Increasing the management server maximum memory
    - database buffer pool size / Database buffer pool size
    - hosts' capacity, monitoring / Setting and monitoring the hosts' capacity
    - hosts' capacity, setting / Setting and monitoring the hosts' capacity
    - resource usage, capping / Capping the resource usage
- physical network
    - about / Physical network, Physical networks
- pod
    - about / Pods
- Port Forwarding / Network offerings
- port forwarding
    - about / Networks and security
- pre-installation tasks, CloudStack
    - about / Pre-installation tasks
    - single node installation / Pre-installation tasks, Single node installation
    - multinode installation / Pre-installation tasks
    - requisites / Requirements

- prepare() method / Network element, CloudStack networking flows
- primary storage
    - about / Primary storage, Primary storage
    - volumes / Volumes
    - system requisites / System requirements and configuration
    - configuring / System requirements and configuration
    - adding / Adding a primary storage
    - storage tags / Storage tags
- private cloud
    - about / Introducing cloud
- project
    - member, removing from / Removing a member from a project
- project.email.sender / Invitation setup
- project.invite.timeout / Invitation setup
- project.smtp.host / Invitation setup
- project.smtp.port / Invitation setup
- project administrator
    - about / Administrators
- projects
    - about / Projects
    - and accounts / Projects and accounts
    - creating / Creating a project
    - members, adding to / Adding members to the project
    - resource management / Resource management in Projects

- Projects tab
    - about / Projects
- public cloud
    - about / Introducing cloud
- public IP addresses
    - about / Public IP addresses
- public network traffic
    - about / Physical network
- public traffic
    - about / Public traffic

R

- Rados Block Device (RBD) / System requirements and configuration
- recurring snapshots
    - creating / Creating recurring snapshots
- RedHat Enterprise Linux (KVM)
    - about / RedHat Enterprise Linux (KVM)
- release() method / NetworkGuru
- remotely accessible VPN
    - about / L3 network configuration in CloudStack
- repository

    - adding, to system / Adding the repository to the system, Adding the repository to the system
- reserve() method / NetworkGuru, CloudStack networking flows
- resource classes
    - about / Resources
- resource layer
    - about / Kernel
- resource management
    - in projects / Resource management in Projects
- resources
    - about / Extending CloudStack networking
- root domain
    - about / Root domain
- Routable NAT (RNAT) / When the VM is created
- RPM
    - building / Building RPM

S

- SaaS
    - about / Introducing cloud
- Safari 4 / CloudStack configuration
- Safari 5 / CloudStack configuration
- scaling, CloudStack

    - about / Scaling in CloudStack
    - counters / Counters
    - conditions / Conditions
    - auto scale policy / Auto scale policy
    - auto scale VM profile / Auto scale VM profile
    - auto scale VM group / Auto scale VM group
    - collector / Collector/Monitor
    - aggregator / Aggregator
    - trigger/alarm generator / Trigger/alarm generator
    - trigger/alarm handler / Trigger/alarm handler
- secondary storage
    - about / Secondary storage, Secondary storage
    - requisites / Secondary storage
    - adding / Adding a secondary storage
    - IP address, modifying / Changing secondary storage IP address
    - modifying / Changing the secondary storage
- security
    - about / Networks and security
- security.singlesignon.tolerance.millis parameter / Single sign on integration
- security groups / Security groups
    - about / Security groups
    - creating, steps / Security groups
- Security Groups / Network offerings
- server resources / Server resources
- Service Management layer / Service Management
- service models, cloud
    - IaaS / Introducing cloud
    - PaaS / Introducing cloud
    - SaaS / Introducing cloud

- service offerings
    - about / Service offerings, Introducing service offerings and virtual machines
    - compute offering / Compute offerings
    - disk offering / Disk offerings, Disk offering
    - network offering / Network offerings
    - compute offerings / Compute offering
    - system service offerings / System service offering
- session timeout
    - modifying / Changing the session timeout
- SHA512
    - about / Management server installation
- shared network
    - about / Shared
- shared networks
    - about / Shared networks
- shared virtual networks / Virtual networks
- shutdown() method / NetworkGuru
- single node installation
    - about / Pre-installation tasks, Single node installation
    - OS, preparing / Preparing the OS
    - management server installation / Management server installation
- single sign on integration / Single sign on integration
- snapshot.delta.max parameter / Creating a template from a snapshot

- snapshot.max.daily parameter / Creating a template from a snapshot, Capping the resource usage
- snapshot.max.hourly parameter / Creating a template from a snapshot, Capping the resource usage
- snapshot.max.monthly parameter / Creating a template from a snapshot, Capping the resource usage
- snapshot.max.weekly parameter / Creating a template from a snapshot, Capping the resource usage
- snapshot.poll.interval parameter / Creating a template from a snapshot
- snapshot manager
    - about / Kernel
- snapshots / Projects
    - about / Snapshots
    - disk snapshot / Snapshots
    - disk and memory snapshot / Snapshots
    - creating / Creating a snapshot
    - volume, creating from / Creating a volume from a snapshot
    - template, creating from / Creating a template from a snapshot
    - VM storage migration / VM storage migration
- Source NAT / Network offerings
- Stateless transport tunneling (STT) / CloudStack networking architecture
- Static NAT / Network offerings
- stop.retry.wait parameter / HighAvailabilityManager – the Queue
- stopRouter function / CloudStack redundant virtual router
- storage
    - NFS shares, preparing for / Preparing Network File System share for storage

- Storage
    - about / Storage
    - clusters / Storage
    - primary storage / Primary storage
    - secondary storage / Secondary storage
- Storage Area Network (SAN) / Storage
- storage manager
    - about / Kernel
- Storage tab
    - about / Storage tab
- storage tags / Storage tags
- Storage traffic
    - about / Physical network
- storage traffic
    - about / Basic zone, Storage traffic
- STT / Networks and security
- Swift
    - CloudStack, integrating with / Integrating with OpenStack object storage (Swift)
    - about / Integrating with OpenStack object storage (Swift)
- switch
    - about / Use of switches in cloud
- switches, cloud environment

    - layer 3 switches / Use of switches in cloud
    - layer 4-7 switches / Use of switches in cloud
    - virtual switches / Use of switches in cloud
- system
    - repository, adding to / Adding the repository to the system, Adding the repository to the system
- system service offering
    - and virtual router / System service offerings and virtual router
    - about / System service offering
    - creating / System service offering
- system VMs
    - preparing / Preparing the system VMs
- system VM template
    - about / Preparing the system VM template
    - preparing / Preparing the system VM template

T

- Task execution layer / Task Execution
- template
    - creating, VM used / Creating a template using a running VM
    - creating, Management console used / Creating a template using the Management console
    - creating, from snapshots / Creating a template from a snapshot
- template manager

    - about / Kernel
- templates / Projects
- Templates tab
    - about / Templates tab
- time.between.cleanup parameter / HighAvailabilityManager – the Queue
- time.to.sleep parameter / HighAvailabilityManager – the Queue
- trash() method / NetworkGuru
- trigger/alarm generator
    - about / Trigger/alarm generator
- trigger/alarm handler
    - about / Trigger/alarm handler

U

- Ubuntu 12.04 LTS
    - about / Requirements
- units, CloudStack management server
    - interface / CloudStack management server
    - business logic / CloudStack management server
    - orchestration engine / CloudStack management server
    - controllers / CloudStack management server
- usage.aggregation.timezone parameter / The usage server
- usage.execution.timezone parameter / The usage server
- usage.sanity.check.interval parameter / The usage server
- usage.stats.job.aggregation.range parameter / The usage server

- usage.stats.job.exec.time parameter / The usage server
- usage server
    - about / The usage server
    - installing / The usage server
    - configuration parameter / The usage server
- user
    - VM instance, requesting / The complete process
- user accounts
    - about / Root domain, Accounts
- User Data / Network offerings

V

- Value Object (vo) / CloudStack storage migration
- VCDNI / Networks and security
- VHD Resizer / Creating a new volume
- views
    - about / CloudStack configuration
    - default view / CloudStack configuration
    - project view / CloudStack configuration
- virtualization
    - about / Introducing cloud, Computing resources
- virtualized Datacenter
    - versus cloud / Introducing cloud
- virtual machine manager

    - about / Kernel, The virtual machine manager
- virtual network
    - about / Virtual network
- virtual networks
    - about / Virtual networks
    - isolated network / Virtual networks
    - shared network / Virtual networks
- Virtual Private Network
    - about / Virtual Private Network
    - scenarios / Virtual Private Network
- virtual router
    - about / Virtual router
    - features / Virtual router
    - and system service offering / System service offerings and virtual router
- Virtual Routing and Forwarding (VRF) / Use of switches in cloud
- virtual switches
    - about / Use of switches in cloud
- VLAN / Networks and security
- VM
    - used, for creating template / Creating a template using a running VM
    - importing, to CloudStack / Importing a VM to CloudStack
    - accessing / Accessing the VM
    - migrating, between hosts / Live migration of VMs between hosts

    - assigning, to load balancer rule / Assigning a VM to the load balancer rule
    - unassigning, from load balancer rule / Unassigning a VM from a load balancer rule
- VMFS
    - about / Primary storage
- VM instance
    - starting / Starting, stopping, rebooting, and destroying the VM instance
    - stopping / Starting, stopping, rebooting, and destroying the VM instance
    - rebooting / Starting, stopping, rebooting, and destroying the VM instance
    - destroying / Starting, stopping, rebooting, and destroying the VM instance
- VM instances / Projects
- VM storage migration, snapshots / VM storage migration
- VMware
    - about / Computing resources, CloudStack storage migration
- VMware vSphere / CloudStack configuration
    - about / VMware vSphere
- VMware Vswitch / Use of switches in cloud
- volumes / Projects
    - about / Volumes, Volumes
    - creating / Creating a new volume
    - attaching, to Guest VM / Attaching a volume to Guest VM

        - detaching, from instance / Detaching a volume from an instance
        - deleting / Deleting a volume
        - creating, from snapshots / Creating a volume from a snapshot
    - VPN / Use of switches in cloud, Network offerings
        - about / L3 network configuration in CloudStack
    - VXLAN / Networks and security

W

- workers parameter / HighAvailabilityManager – the Queue

X

- Xen / Use of switches in cloud
- XenServer
    - about / Computing resources, CloudStack storage migration

  / System requirements and configuration
- XenServer hypervisor / CloudStack configuration

Y

- yum repo
    - creating / Creating yum repo

Z

- zone

    - load balancer, deleting from / Deleting a load balancer from a zone
- zones
    - about / Zones, Zones and their types, Domain and zones
    - logical units / Zones
    - pods / Pods
    - clusters / Clusters
    - basic zone / Zones and their types
    - advanced zone / Zones and their types