BASIC UNDERSTANDING ON CLOUD COMPUTING

Introduction to Cloud Computing

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models. The essential characteristics are on-demand self-service, broadcast network access, resource pooling, rapid elasticity, measured service, etc.

Service Models

Software as a Service (SaaS): The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based email), or a program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user specific application configuration settings.

Platform as a Service (PaaS): The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider.3 The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment.

Infrastructure as a Service (IaaS): The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components (e.g., host firewalls).

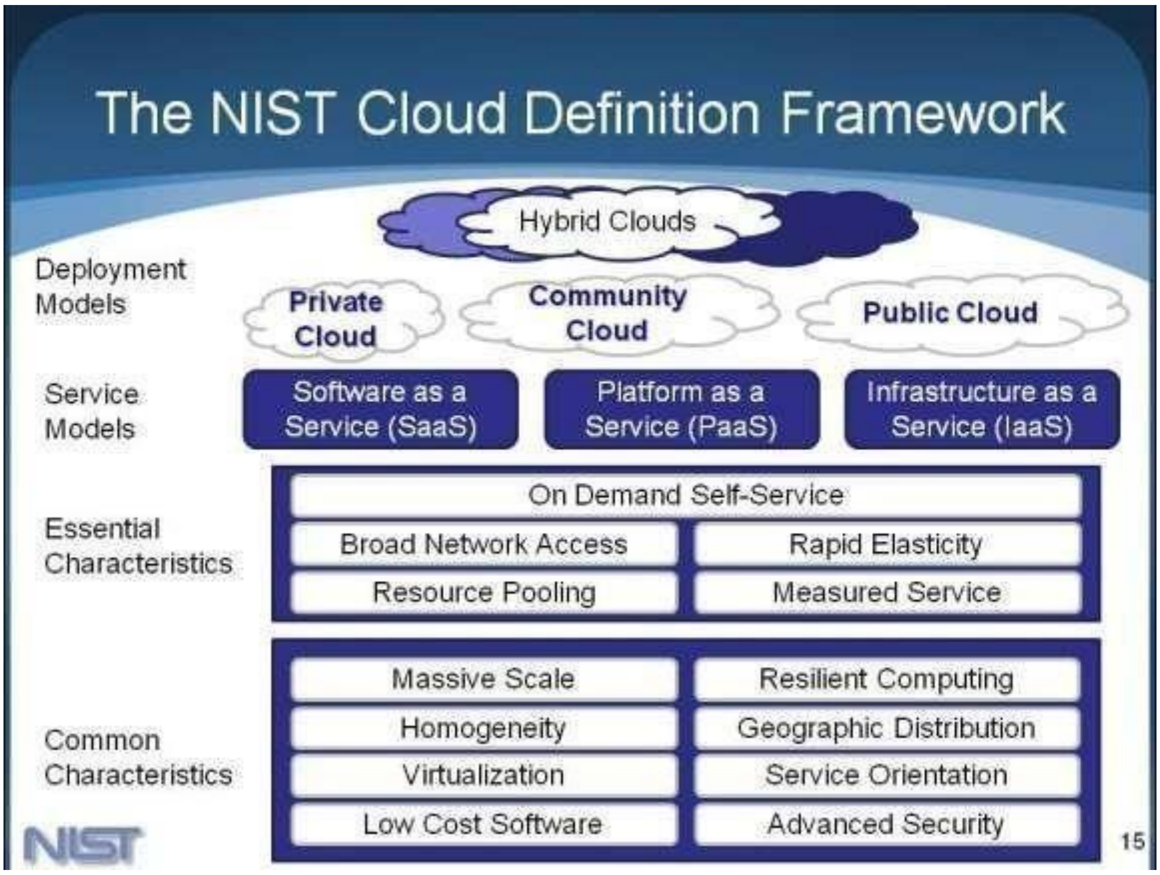
Deployment Models

Private cloud: The cloud infrastructure is provisioned for exclusive use by a single organization comprising multiple consumers (e.g., business units). It may be owned, managed, and operated by the organization, a third party, or some combination of them, and it may exist on or off premises.

Community cloud: The cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be owned, managed, and operated by one or more of the organizations in the community, a third party, or some combination of them, and it may exist on or off premises.

Public cloud: The cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud provider.

Hybrid cloud: The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds).



NIST Cloud Definition Framework

EX. NO:

DATE: **CREATION AND MANAGEMENT OF VIRTUAL MACHINE IN VIRTUALIZED ENVIRONMENT – VMWAREWORKSTATION**

AIM

To install VMWare Workstation and configure a virtual machine.

THEORY

VIRTUALIZATION

Virtualization is creation of Virtual Machines which can emulate hardware in software or in other words it is the creation of virtual version of something such as a hardware platform, operating system, storage device, or network resources (from Wikipedia). Virtualization is achieved or created with the help of software and this particular software allows you to install any number of OS on your system without using the available hardware directly. When you are running an OS over the top of another on your machine the whole environment acts like a HOST and GUEST OS. The real operating system acts as a HOST and the OS run by virtualization software acts as a GUEST OS.The entire load balancing is actually done by the HOST operating system.

Types of Virtualization: There are mainly three types of virtualization.

- Full virtualization
- OS level virtualization
- Para virtualization

Full virtualization

As the name suggests everything in a system is virtualized which includes the processor, storage, networking components etc. Virtual Box, VMware are example of "Full Virtualization" solutions.

OS Level virtualization

In this type of virtualization only applications are run inside the software. In this case the application is given a platform to work. Isolation is created and the application is made to believe that it is the only thing running on the system.

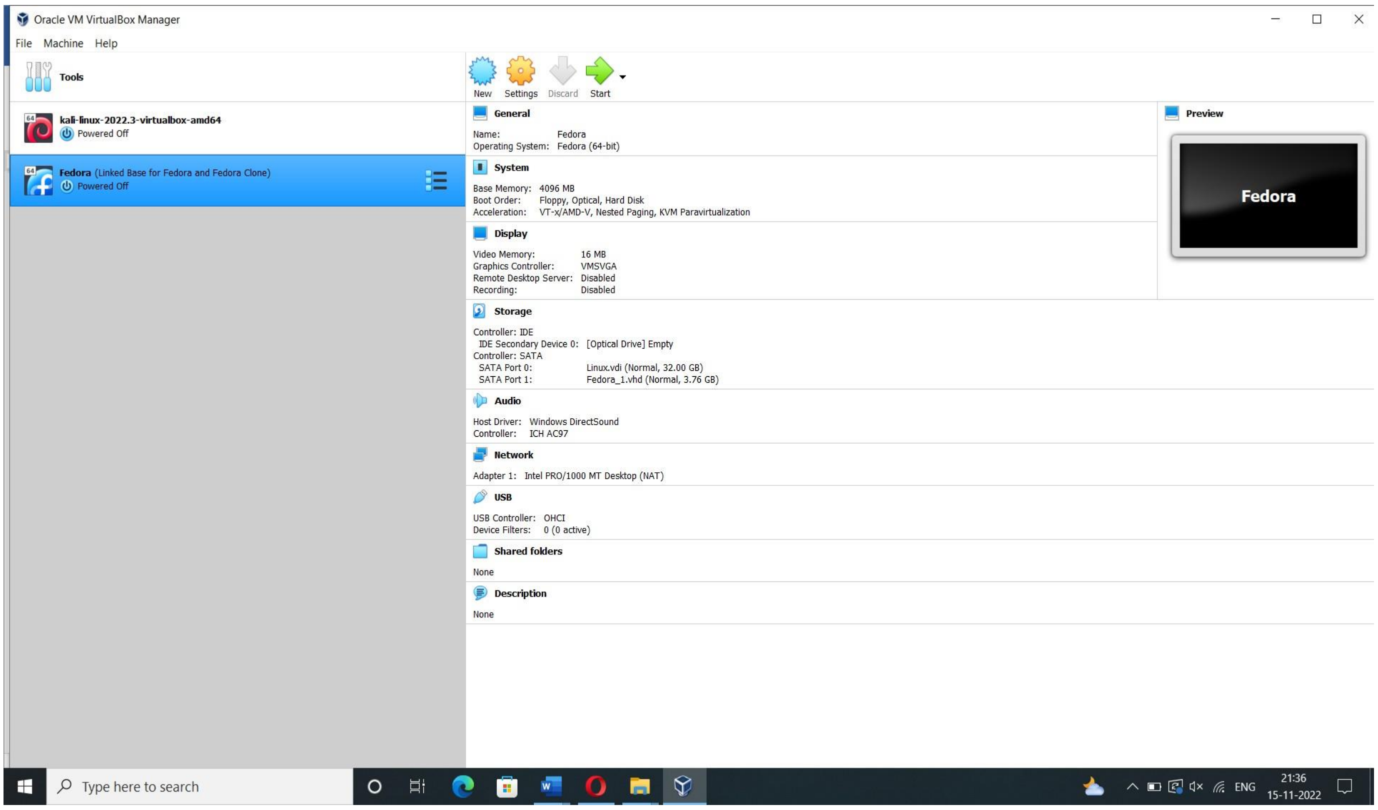
Para virtualization

It's a semi-virtualized environment created for the guest OS. A modified guest OS is created using a hypervisor. "The intent of the modified interface is to reduce the portion of the guest's execution time spent performing operations which are substantially more difficult to run in a virtual environment compared to a non-virtualized environment. The Para virtualization provides specially defined „hooks" to allow the guest(s) and host to request and acknowledge these tasks, which would otherwise be executed in the virtual domain (where execution performance is worse). A successful Para virtualized platform may allow the virtual machine monitor (VMM) to be simpler (by relocating execution of critical tasks from the virtual domain to the host domain), and/or reduce the overall performance degradation of machine-execution inside the virtual-guest.”

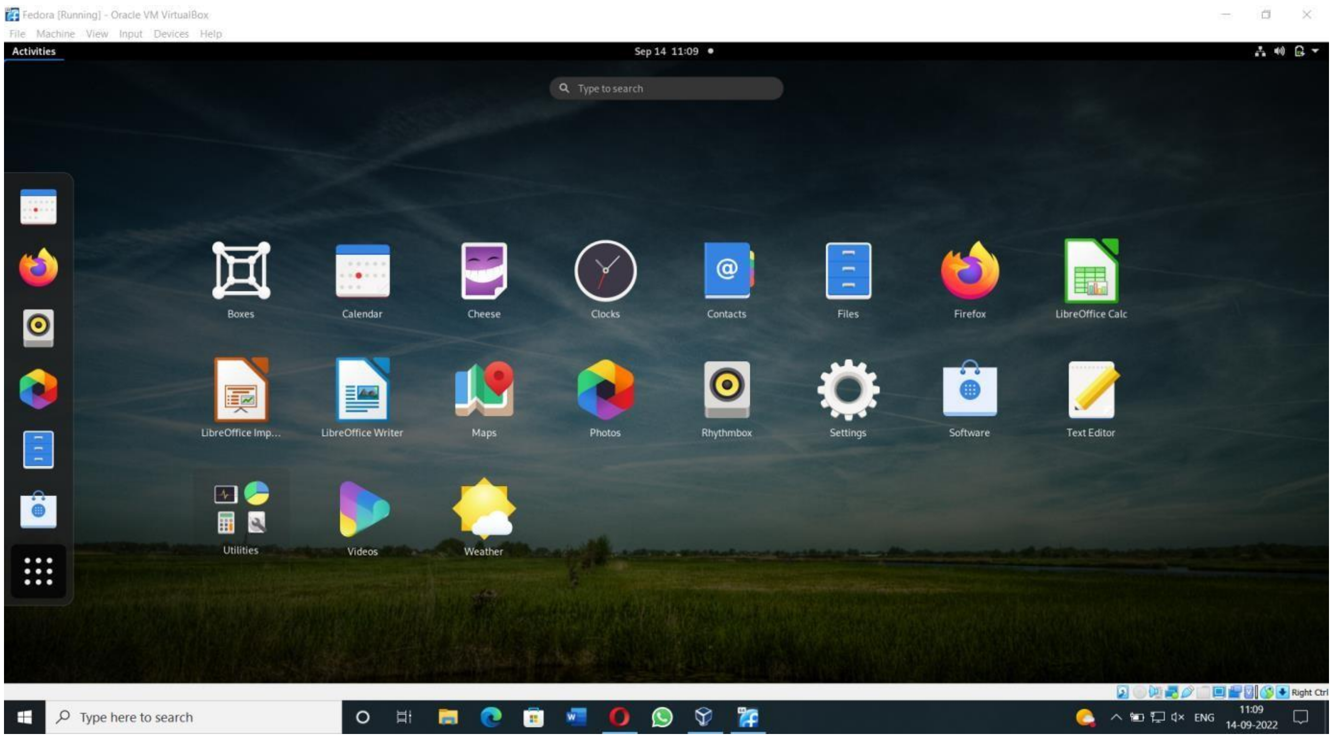
Advantages of Virtualization

- One of the biggest advantages of virtualization is scalability i.e. the ability to expand. Whenever there is excessive load on some part of application in a server you can easily create a similar virtual environment on a different server and configure the setup.
- ➢ Hardware maintenance cost is reduced because you don‟t need many servers to install different applications.
- You can save a huge amount of energy by running one physical server instead of many and less power backup is required.
- You can get faster and safer backups by taking live snapshot while server is running.
- You will get centralized monitoring of your resources as virtualization provides easy way of connecting and maintaining your virtual servers

OUTPUT:



```
General
Name:              Fedora
Operating System:  Fedora (64-bit)

System
Base Memory:   4096 MB
Boot Order:    Floppy, Optical, Hard Disk
Acceleration:  VT-x/AMD-V, Nested Paging, KVM Paravirtualization

Display
Video Memory:            16 MB
Graphics Controller:     VMSVGA
Remote Desktop Server:   Disabled
Recording:               Disabled

Storage
Controller: IDE
  IDE Secondary Device 0:  [Optical Drive] Empty
Controller: SATA
  SATA Port 0:             Linux.vdi (Normal, 32.00 GB)
  SATA Port 1:             Fedora_1.vhd (Normal, 3.76 GB)

Audio
Host Driver:   Windows DirectSound
Controller:    ICH AC97

Network
Adapter 1:  Intel PRO/1000 MT Desktop (NAT)

USB
USB Controller:  OHCI
Device Filters:  0 (0 active)

Shared folders
None

Description
None
```



RESULT

Thus the procedure to create virtual machine with instance and virtual clone is done successfully.

EX. NO:

DATE:

VIRTUALIZE A MACHINE AND CHECK HOW MANY MACHINES CAN BE UTILIZED AT A PARTICULAR TIME

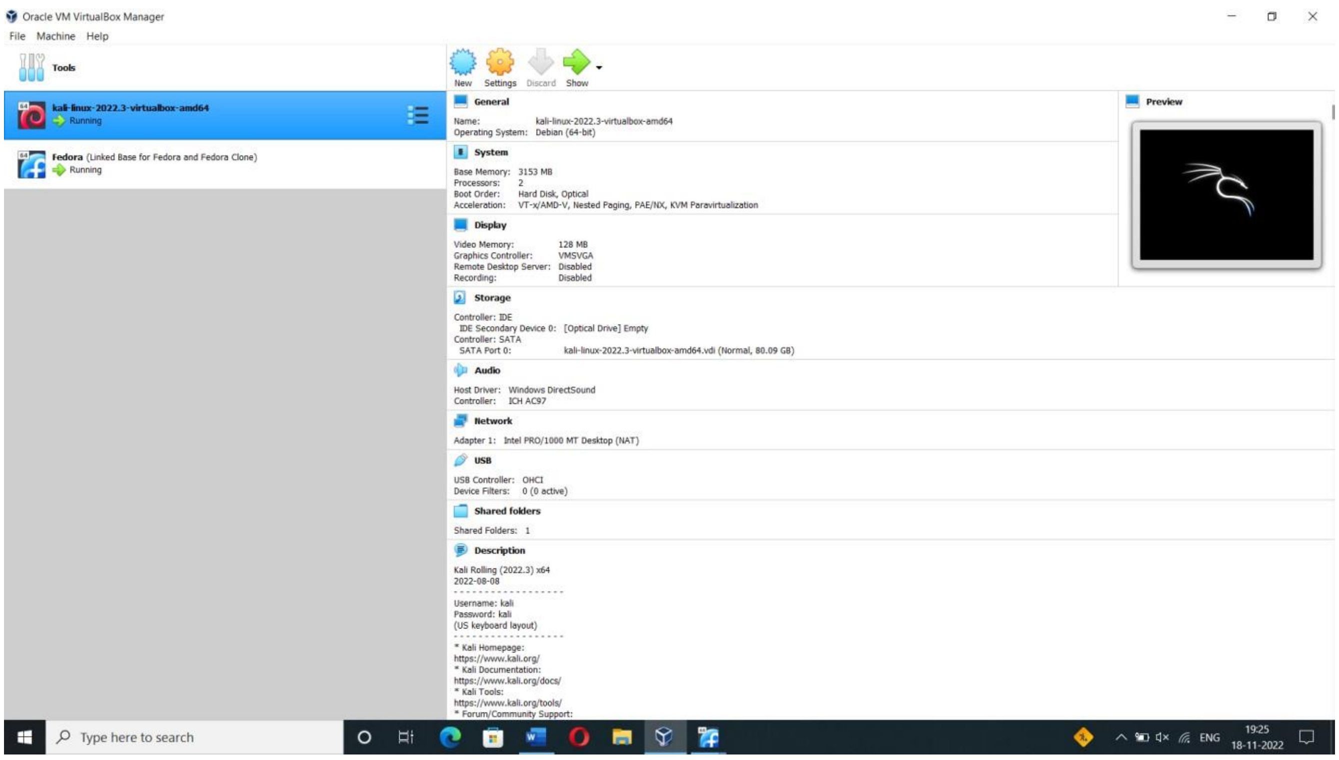
AIM

To virtualize a machine and check how many machines can be utilized at a particular time.

PROCEDURE

1. Open the VMware.
2. Run the virtual machines installed in the VMware.
3. The virtual machines run parallelly in the VMware.
4. We can observe that many virtual machines can be run at a particular time.

OUTPUT



kali-linux-2022.3-virtualbox-amd64
Running

Fedora (Linked Base for Fedora and Fedora Clone)
Running

```
General
Name:              kali-linux-2022.3-virtualbox-amd64
Operating System:  Debian (64-bit)

System
Base Memory:   3153 MB
Processors:    2
Boot Order:    Hard Disk, Optical
Acceleration:  VT-x/AMD-V, Nested Paging, PAE/NX, KVM Paravirtualization

Display
Video Memory:            128 MB
Graphics Controller:     VMSVGA
Remote Desktop Server:   Disabled
Recording:               Disabled

Storage
Controller: IDE
  IDE Secondary Device 0:  [Optical Drive] Empty
Controller: SATA
  SATA Port 0:             kali-linux-2022.3-virtualbox-amd64.vdi (Normal, 80.09 GB)

Audio
Host Driver:  Windows DirectSound
Controller:   ICH AC97

Network
Adapter 1:  Intel PRO/1000 MT Desktop (NAT)

USB
USB Controller:  OHCI
Device Filters:  0 (0 active)

Shared folders
Shared Folders:  1

Description
Kali Rolling (2022.3) x64
2022-08-08
-------------------
Username: kali
Password: kali
(US keyboard layout)
-------------------
* Kali Homepage:
https://www.kali.org/
* Kali Documentation:
https://www.kali.org/docs/
* Kali Tools:
https://www.kali.org/tools/
* Forum/Community Support:
```

RESULT

Thus virtualize a machine and checking the number of machines running particularly at a time has been done successfully.

EX. NO:

DATE:

CREATE VM CLONE AND ATTACH A VIRTUAL BLOCK

AIM

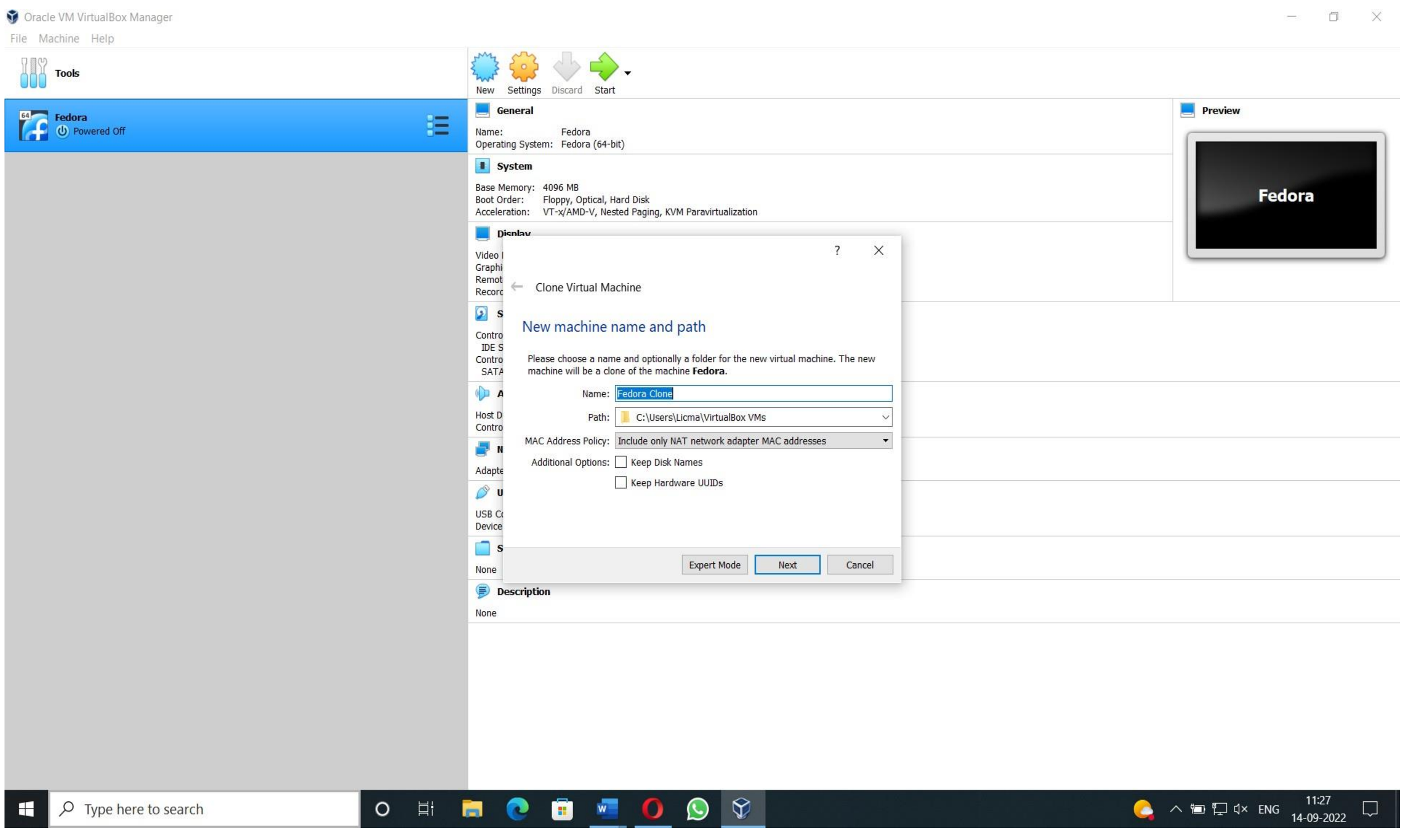
To find procedure to attach a virtual block to virtual machine and check whether it holds data even after the release of the virtual machine.

a) Create a VM clone
b) Adding an additional block to Hard-Disk in Virtual Machine.

PROCEDURE

a) Create a VM clone

Step: 1 –Click VM -> Manage -> Clone



General
Name: Fedora
Operating System: Fedora (64-bit)

System
Base Memory: 4096 MB
Boot Order: Floppy, Optical, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging, KVM Paravirtualization

Clone Virtual Machine

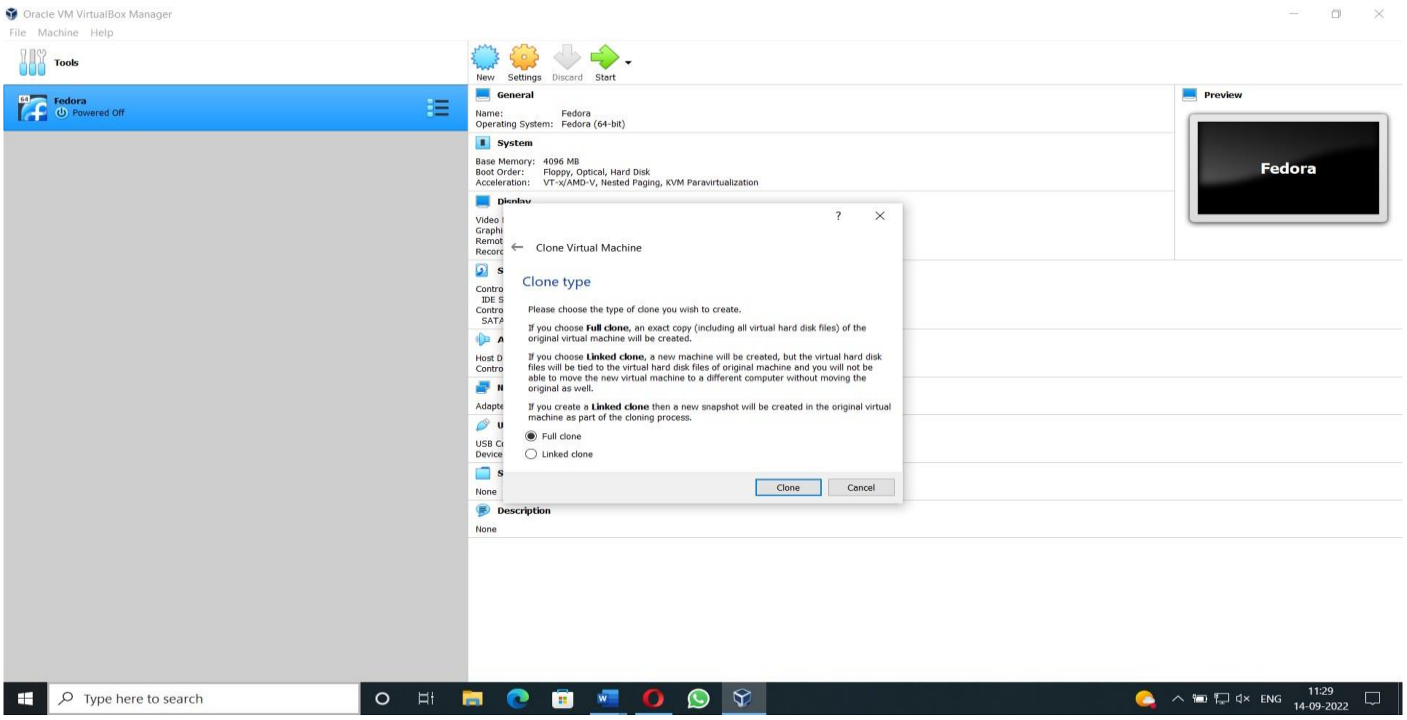
New machine name and path

Please choose a name and optionally a folder for the new virtual machine. The new machine will be a clone of the machine **Fedora**.

Name: Fedora Clone
Path: C:\Users\Licma\VirtualBox VMs
MAC Address Policy: Include only NAT network adapter MAC addresses
Additional Options: [ ] Keep Disk Names
[ ] Keep Hardware UUIDs

Expert Mode Next Cancel

Description
None



General

Name: Fedora
Operating System: Fedora (64-bit)

System

Base Memory: 4096 MB
Boot Order: Floppy, Optical, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging, KVM Paravirtualization

Clone Virtual Machine

Clone type

Please choose the type of clone you wish to create.

If you choose **Full clone**, an exact copy (including all virtual hard disk files) of the original virtual machine will be created.

If you choose **Linked clone**, a new machine will be created, but the virtual hard disk files will be tied to the virtual hard disk files of original machine and you will not be able to move the new virtual machine to a different computer without moving the original as well.

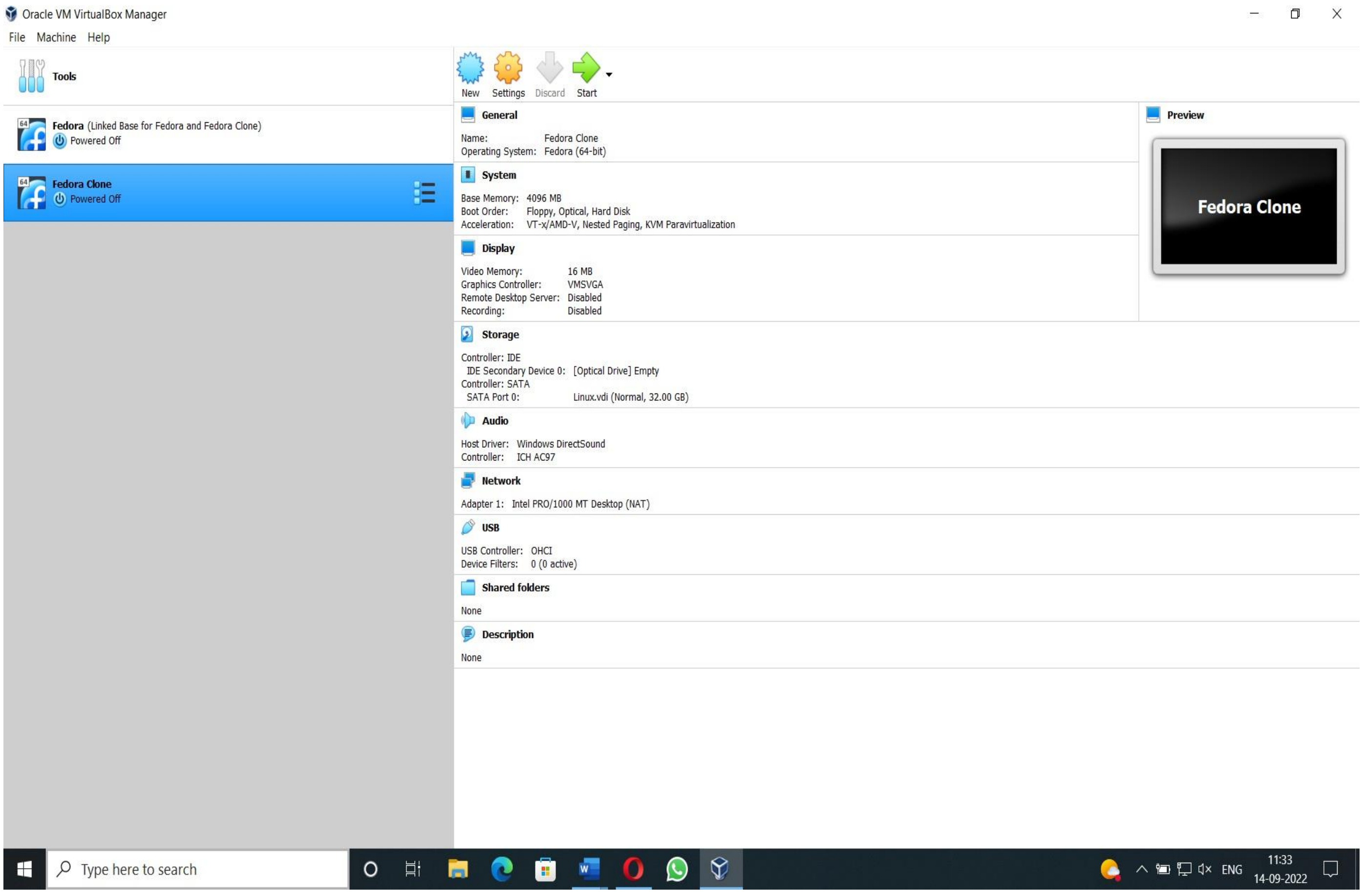
If you create a **Linked clone** then a new snapshot will be created in the original virtual machine as part of the cloning process.

[✓ Full clone] Linked clone

Clone Cancel

Description

None



General

Name: Fedora Clone
Operating System: Fedora (64-bit)

System

Base Memory: 4096 MB
Boot Order: Floppy, Optical, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging, KVM Paravirtualization

Display

Video Memory: 16 MB
Graphics Controller: VMSVGA
Remote Desktop Server: Disabled
Recording: Disabled

Storage

Controller: IDE
IDE Secondary Device 0: [Optical Drive] Empty
Controller: SATA
SATA Port 0: Linux.vdi (Normal, 32.00 GB)

Audio

Host Driver: Windows DirectSound
Controller: ICH AC97

Network

Adapter 1: Intel PRO/1000 MT Desktop (NAT)

USB

USB Controller: OHCI
Device Filters: 0 (0 active)

Shared folders

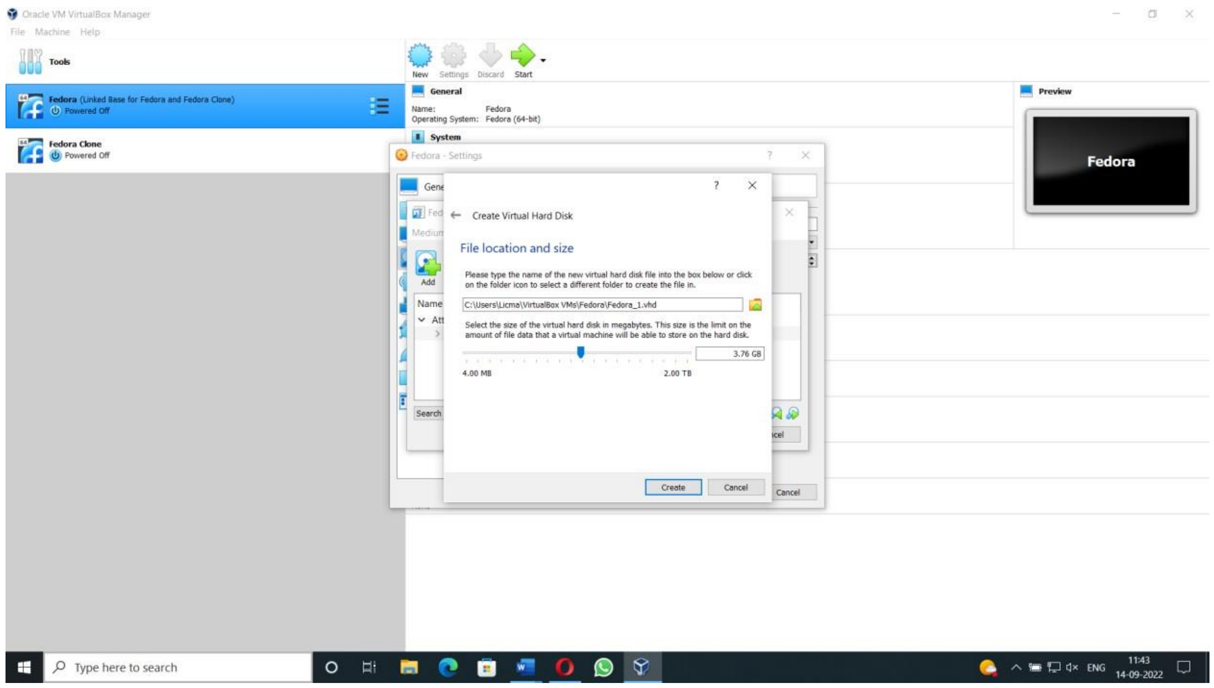
None

Description

None

b) Adding an additional block to Hard-Disk in Virtual Machine.

Step: 1 - Select hard Disk from the virtual machine settings dialog box. Select add button. In the Add Hardware wizard select the hardware type as hard disk and click next. Select the virtual disk type as IDE and click next.


Create Virtual Hard Disk

File location and size

Please type the name of the new virtual hard disk file into the box below or click on the folder icon to select a different folder to create the file in.

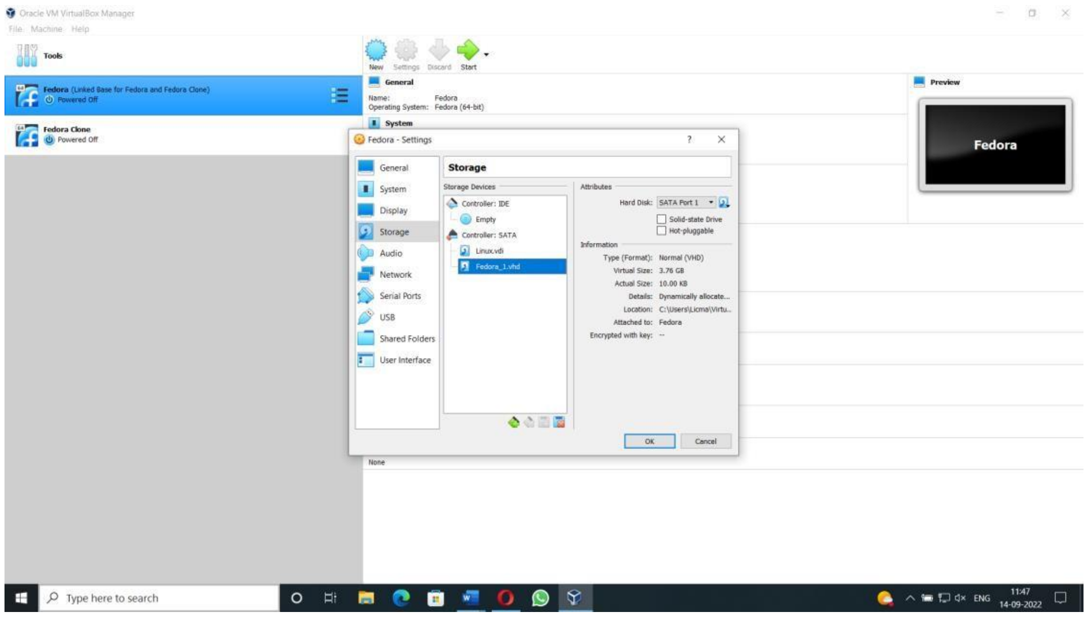
C:\Users\Licma\VirtualBox VMs\Fedora\Fedora_1.vhd

Select the size of the virtual hard disk in megabytes. This size is the limit on the amount of file data that a virtual machine will be able to store on the hard disk.

3.76 GB

4.00 MB — 2.00 TB

Step: 2 – In the next page select the Create a new virtual disk option and click next. Provide the disk capacity as 20GB and select store virtual disk as single file option and click next. Specify the disk file and click finish and virtual block is created.


Storage

Storage Devices
- Controller: IDE
  - Empty
- Controller: SATA
  - Linux.vdi
  - Fedora_1.vhd

Attributes

Hard Disk: SATA Port 1

Information

Type (Format): Normal (VHD)
Virtual Size: 3.76 GB
Actual Size: 10.00 KB
Details: Dynamically allocate...
Location: C:\Users\Licma\Virtu...
Attached to: Fedora
Encrypted with key: --

RESULT

Thus the procedure to create a VM clone and adding an additional block to Hard-Disk in virtual machine is completed successfully.

EX NO:

DATE:

SALESFORCE

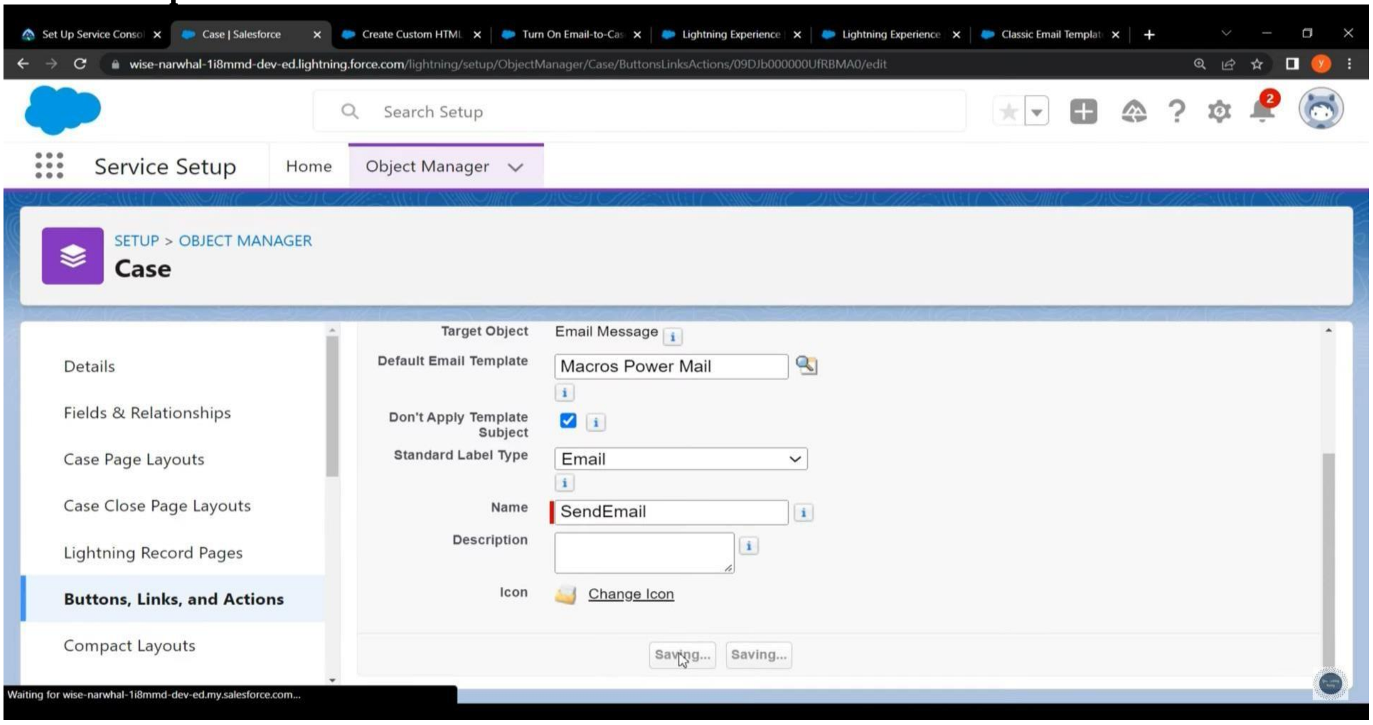
Aim:

To develop a simple E-mail automation service using salesforce.

Procedure :

1) Open Salesforce and login to the account.
2) Open the Service cloud agent productivity module
3) Do the setup service console productivity tools
   a. Give users access to quick text and macros.
   b. Customize your app for quick text, macros, and history.
   c. Set up mass quick actions.
   d. Customize an email action.
   e. Enable email notifications for case owners
4) Open the Create Macros and Quick Text to Reduce Clicks
   a. Create macros.
   b. Create quick text.
5) Open the module Use All the Service Console Productivity Tools Together
   a. Use split view.
   b. Run a macro.
   c. Use and find the keyboard shortcuts.
   d. Insert quick text.
   e. Perform mass quick actions.
   f. Use the History utility.

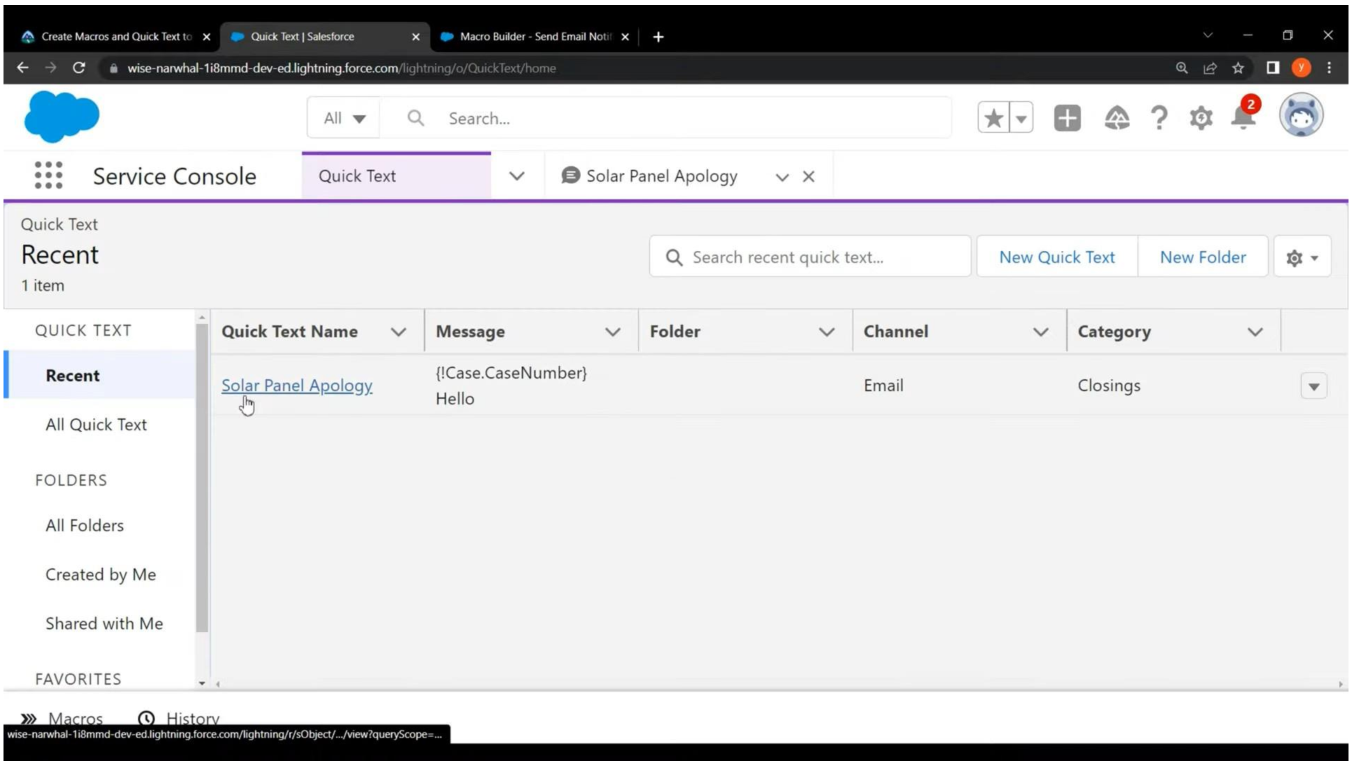
Output:


wise-narwhal-1i8mmd-dev-ed.lightning.force.com/lightning/setup/ObjectManager/Case/ButtonsLinksActions/09DJb000000UfRBMA0/edit

SETUP > OBJECT MANAGER

Case

Target Object	Email Message
Default Email Template	Macros Power Mail
Don't Apply Template Subject	✓
Standard Label Type	Email
Name	SendEmail
Description	
Icon	Change Icon

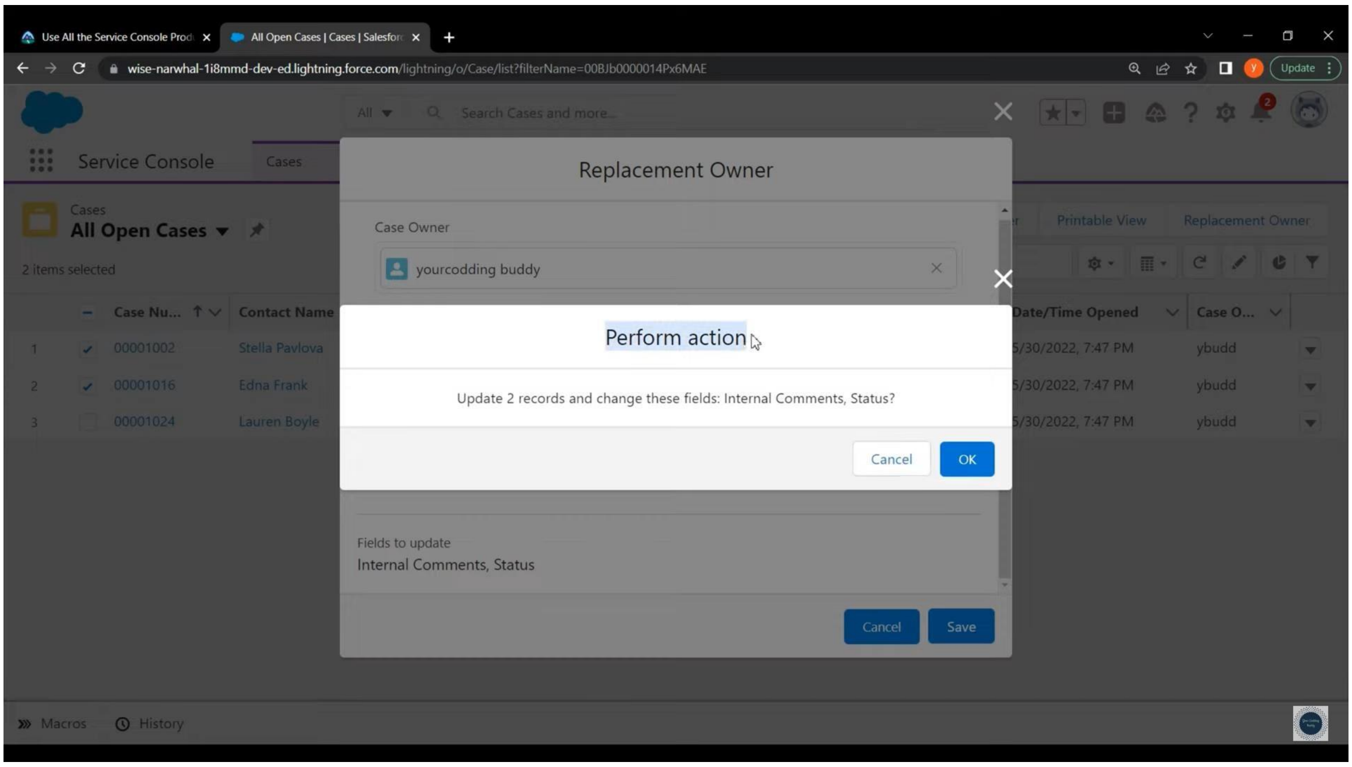


Quick Text

Recent

1 item

Quick Text Name	Message	Folder	Channel	Category
Solar Panel Apology	{!Case.CaseNumber} Hello		Email	Closings



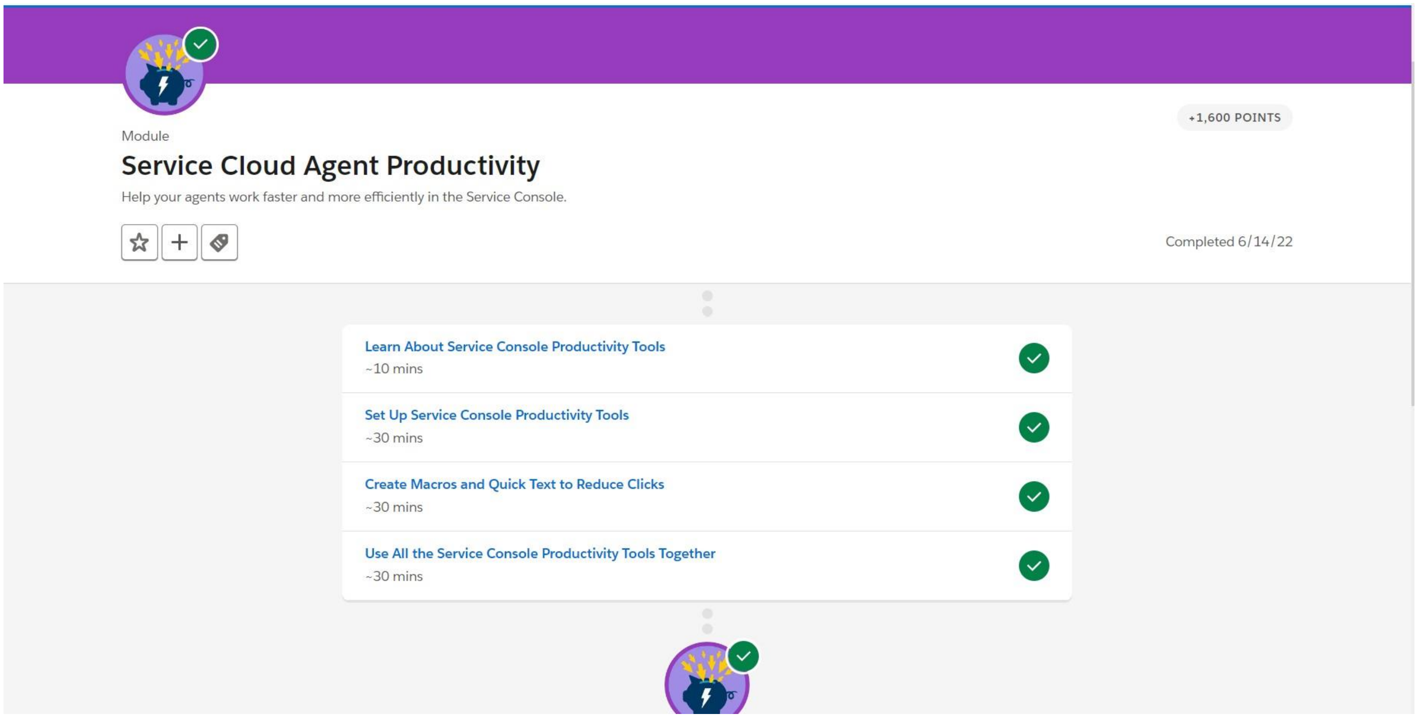
Replacement Owner

Case Owner: yourcodding buddy

Perform action

Update 2 records and change these fields: Internal Comments, Status?

Fields to update: Internal Comments, Status



Module

Service Cloud Agent Productivity

Help your agents work faster and more efficiently in the Service Console.

+1,600 POINTS

Completed 6/14/22

Learn About Service Console Productivity Tools
~10 mins

Set Up Service Console Productivity Tools
~30 mins

Create Macros and Quick Text to Reduce Clicks
~30 mins

Use All the Service Console Productivity Tools Together
~30 mins

RESULT

Thus the procedure to create an email automation service using salesforce is completed successfully.

EX NO:

DATE:

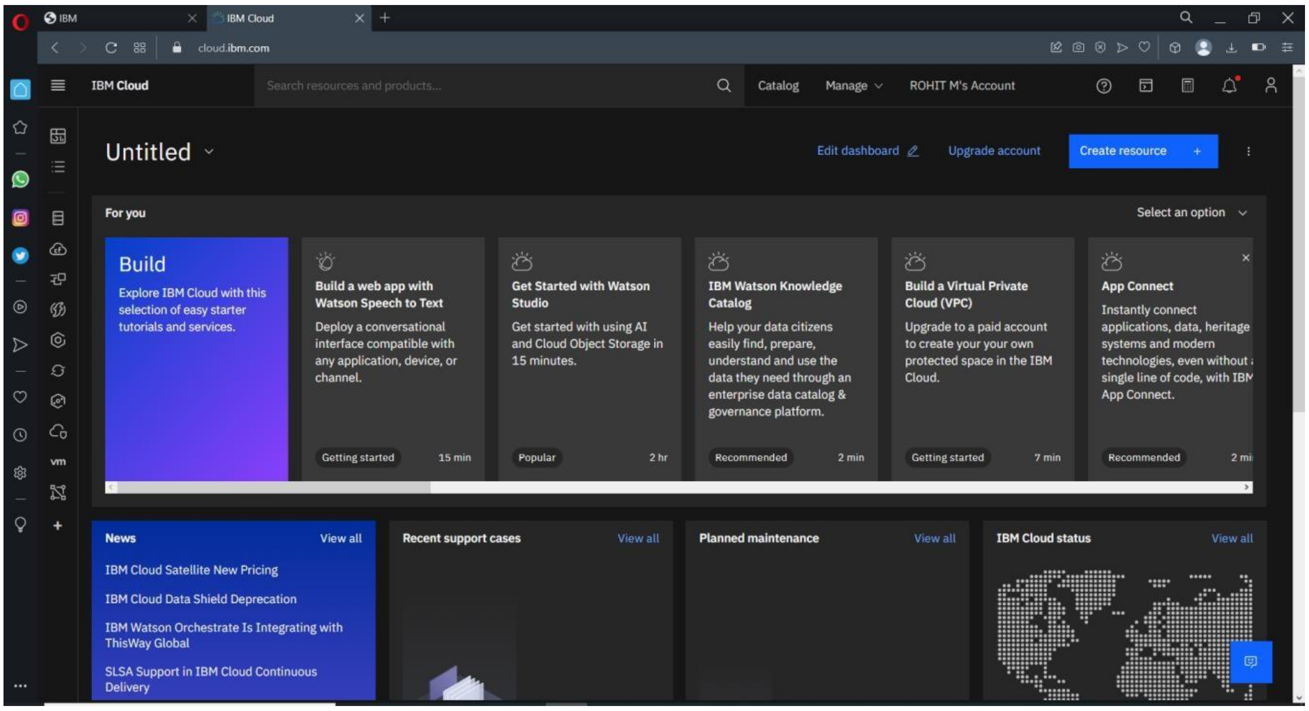
LAUNCH A CLOUD INSTANCE USING IBM CLOUD

Aim:

To launch a cloud instance using a public IaaS cloud service like the IBM cloud.

Procedure :

1. Register to IBM Cloud
2. Sign in with your credentials
3. Log in to your IBM Cloud account, and click on Catalog



Build
Explore IBM Cloud with this selection of easy starter tutorials and services.

Build a web app with Watson Speech to Text
Deploy a conversational interface compatible with any application, device, or channel.
Getting started 15 min

Get Started with Watson Studio
Get started with using AI and Cloud Object Storage in 15 minutes.
Popular 2 hr

IBM Watson Knowledge Catalog
Help your data citizens easily find, prepare, understand and use the data they need through an enterprise data catalog & governance platform.
Recommended 2 min

Build a Virtual Private Cloud (VPC)
Upgrade to a paid account to create your your own protected space in the IBM Cloud.
Getting started 7 min

App Connect
Instantly connect applications, data, heritage systems and modern technologies, even without a single line of code, with IBM App Connect.
Recommended 2 min

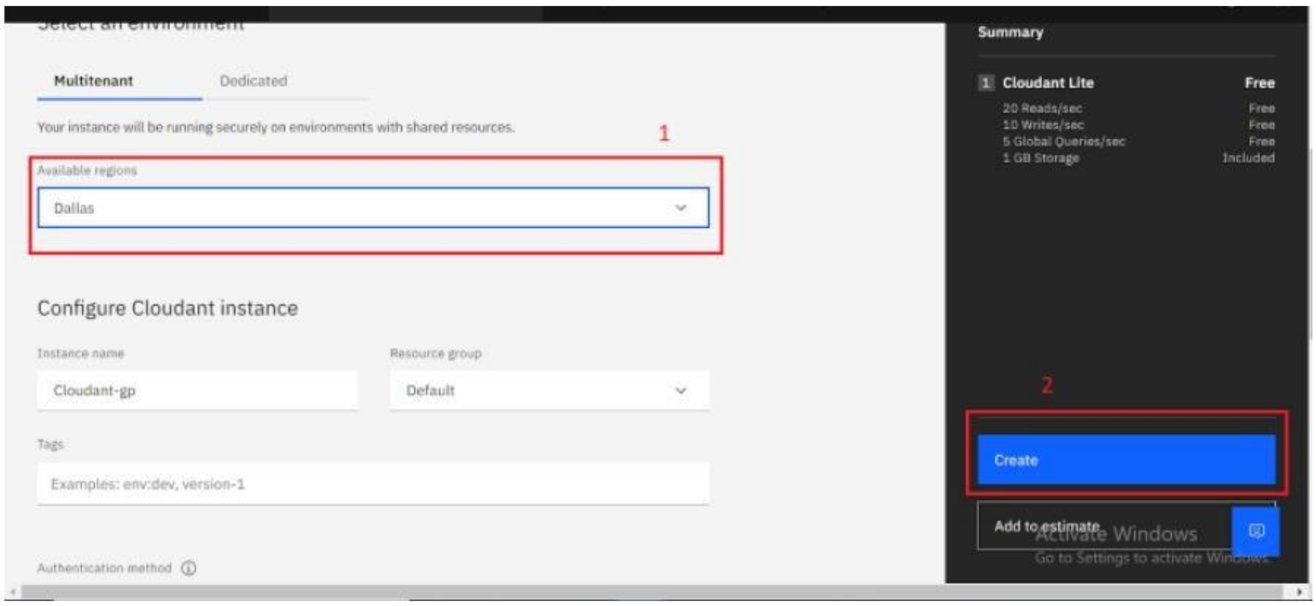
News — View all
IBM Cloud Satellite New Pricing
IBM Cloud Data Shield Deprecation
IBM Watson Orchestrate Is Integrating with ThisWay Global
SLSA Support in IBM Cloud Continuous Delivery

Recent support cases — View all

Planned maintenance — View all

IBM Cloud status — View all

4. Type Cloudant in the Search bar and click to open it.
5. Select an offering and an environment
6. Select region as Dallas & Type an instance name then click on create service.



Multitenant | Dedicated

Your instance will be running securely on environments with shared resources.

1

Available regions: Dallas

Configure Cloudant instance

Instance name: Cloudant-gp

Resource group: Default

Tags: Examples: env:dev, version-1

Authentication method

Summary

1 Cloudant Lite	Free
20 Reads/sec	Free
10 Writes/sec	Free
5 Global Queries/sec	Free
1 GB Storage	Included

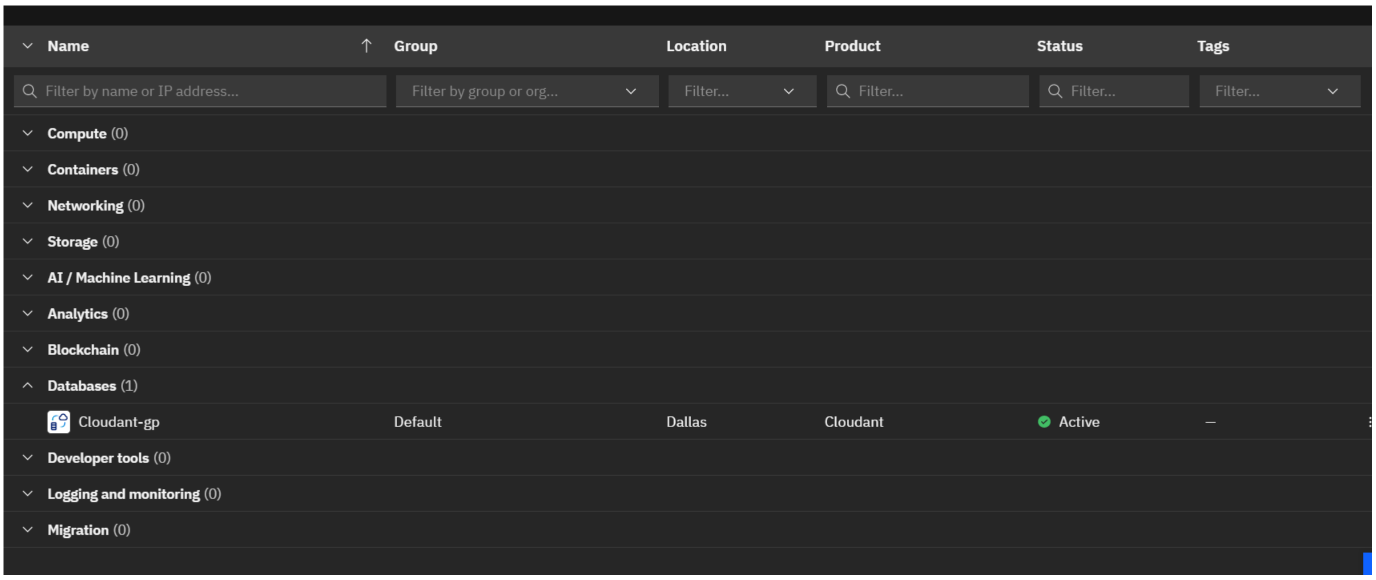
2

Create

Add to estimate

7. After you click create the system displays a message to say that the instance is being provisioned, which returns you to the Resource list. From the Resource list, you see that the status for your instance is, Provision in progress.
8. When the status changes to Active, click the instance.

OUTPUT



Name	Group	Location	Product	Status	Tags
Filter by name or IP address...	Filter by group or org...	Filter...	Filter...	Filter...	Filter...
Compute (0)					
Containers (0)					
Networking (0)					
Storage (0)					
AI / Machine Learning (0)					
Analytics (0)					
Blockchain (0)					
Databases (1)					
Cloudant-gp	Default	Dallas	Cloudant	● Active	–
Developer tools (0)					
Logging and monitoring (0)					
Migration (0)					

RESULT

Thus the launching of a cloud instance using a public IaaS cloud service like the IBM cloud has been done successfully.

EX. NO:

DATE:

PUBLIC CLOUD SERVICE

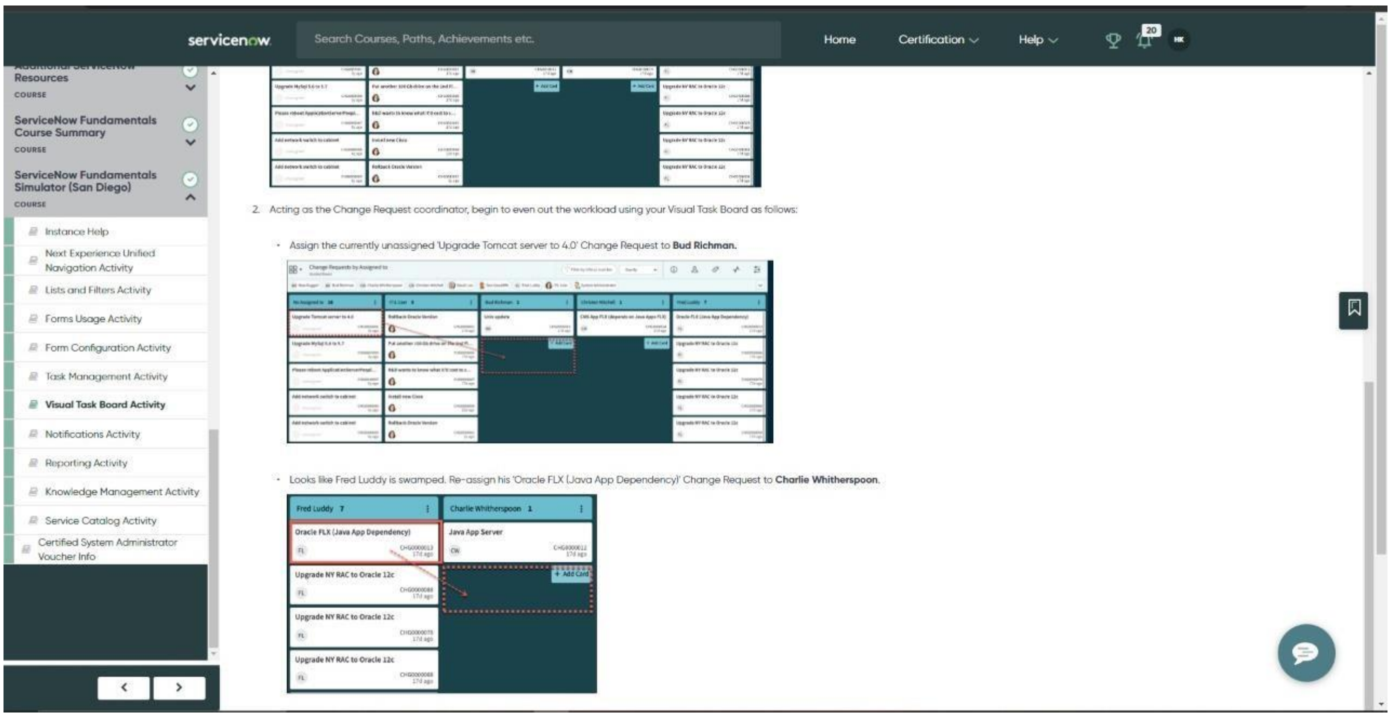
AIM:

To work with a public cloud service such as ServiceNow.

DESCRIPTION

Visual Task Boards (VTB) transform the navigation of lists and forms into an interactive graphical experience. With Visual Task Boards, you can view and update multiple task records, which appear as *cards* that can be moved between *lanes*. An *activity stream* on the board displays recent activity so you can easily track changes to tasks. You can add task cards from any table that extends Task to intuitively and easily track updates and esdit records directly from the board. Any user can use task boards, regardless of role, though access control rules (ACLs) may limit which cards each user can see. The Visual Task Board interface provides a graphic-rich environment suited for managing and collaborating on records. For example, a support manager might create a board for the team to track their assigned incidents by state in real time.

OUTPUT

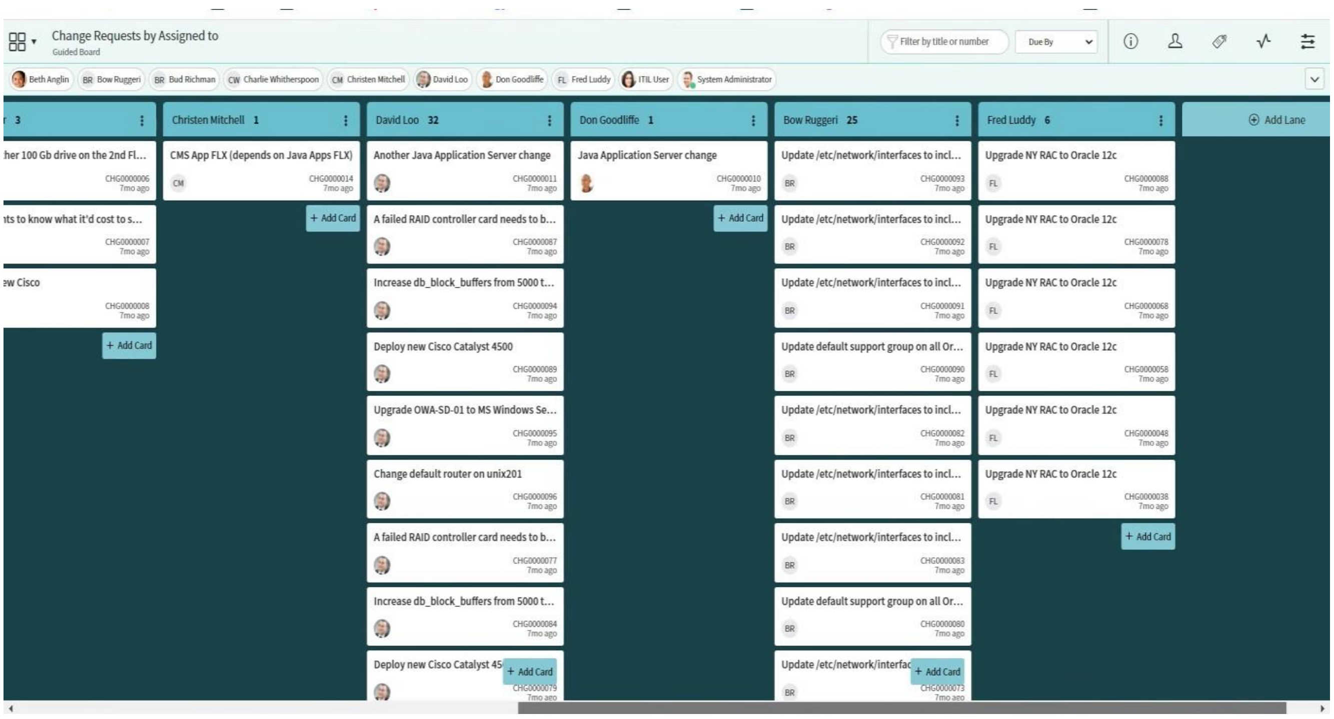

servicenow
Search Courses, Paths, Achievements etc.
Home
Certification
Help
Resources
COURSE
ServiceNow Fundamentals Course Summary
COURSE
ServiceNow Fundamentals Simulator (San Diego)
COURSE
Instance Help
Next Experience Unified Navigation Activity
Lists and Filters Activity
Forms Usage Activity
Form Configuration Activity
Task Management Activity
Visual Task Board Activity
Notifications Activity
Reporting Activity
Knowledge Management Activity
Service Catalog Activity
Certified System Administrator Voucher Info

2. Acting as the Change Request coordinator, begin to even out the workload using your Visual Task Board as follows:

- Assign the currently unassigned 'Upgrade Tomcat server to 4.0' Change Request to **Bud Richman.**

- Looks like Fred Luddy is swamped. Re-assign his 'Oracle FLX (Java App Dependency)' Change Request to **Charlie Whitherspoon.**

Fred Luddy 7
Oracle FLX (Java App Dependency)
CHG0000013
Upgrade NY RAC to Oracle 12c
Charlie Whitherspoon 1
Java App Server
CHG0000012
+ Add Card



Change Requests by Assigned to
Guided Board

Filter by title or number | Due By

Beth Anglin · BR Bow Ruggeri · BR Bud Richman · CW Charlie Whitherspoon · CM Christen Mitchell · David Loo · Don Goodliffe · FL Fred Luddy · ITIL User · System Administrator

r 3
- her 100 Gb drive on the 2nd Fl... — CHG0000006, 7mo ago
- its to know what it'd cost to s... — CHG0000007, 7mo ago
- ew Cisco — CHG0000008, 7mo ago

Christen Mitchell 1
- CMS App FLX (depends on Java Apps FLX) — CM — CHG0000014, 7mo ago

David Loo 32
- Another Java Application Server change — CHG0000011, 7mo ago
- A failed RAID controller card needs to b... — CHG0000087, 7mo ago
- Increase db_block_buffers from 5000 t... — CHG0000094, 7mo ago
- Deploy new Cisco Catalyst 4500 — CHG0000089, 7mo ago
- Upgrade OWA-SD-01 to MS Windows Se... — CHG0000095, 7mo ago
- Change default router on unix201 — CHG0000096, 7mo ago
- A failed RAID controller card needs to b... — CHG0000077, 7mo ago
- Increase db_block_buffers from 5000 t... — CHG0000084, 7mo ago
- Deploy new Cisco Catalyst 45 — CHG0000079, 7mo ago

Don Goodliffe 1
- Java Application Server change — CHG0000010, 7mo ago

Bow Ruggeri 25
- Update /etc/network/interfaces to incl... — BR — CHG0000093, 7mo ago
- Update /etc/network/interfaces to incl... — BR — CHG0000092, 7mo ago
- Update /etc/network/interfaces to incl... — BR — CHG0000091, 7mo ago
- Update default support group on all Or... — BR — CHG0000090, 7mo ago
- Update /etc/network/interfaces to incl... — BR — CHG0000082, 7mo ago
- Update /etc/network/interfaces to incl... — BR — CHG0000081, 7mo ago
- Update /etc/network/interfaces to incl... — BR — CHG0000083, 7mo ago
- Update default support group on all Or... — BR — CHG0000080, 7mo ago
- Update /etc/network/interfac — BR — CHG0000073, 7mo ago

Fred Luddy 6
- Upgrade NY RAC to Oracle 12c — FL — CHG0000088, 7mo ago
- Upgrade NY RAC to Oracle 12c — FL — CHG0000078, 7mo ago
- Upgrade NY RAC to Oracle 12c — FL — CHG0000068, 7mo ago
- Upgrade NY RAC to Oracle 12c — FL — CHG0000058, 7mo ago
- Upgrade NY RAC to Oracle 12c — FL — CHG0000048, 7mo ago
- Upgrade NY RAC to Oracle 12c — FL — CHG0000038, 7mo ago

RESULT

Thus the procedure to work with public cloud service such as servicenow has been done successfully.

EX. NO:

DATE:

MODEL A CLOUD ENVIRONMENT USING CLOUD SIMULATORS

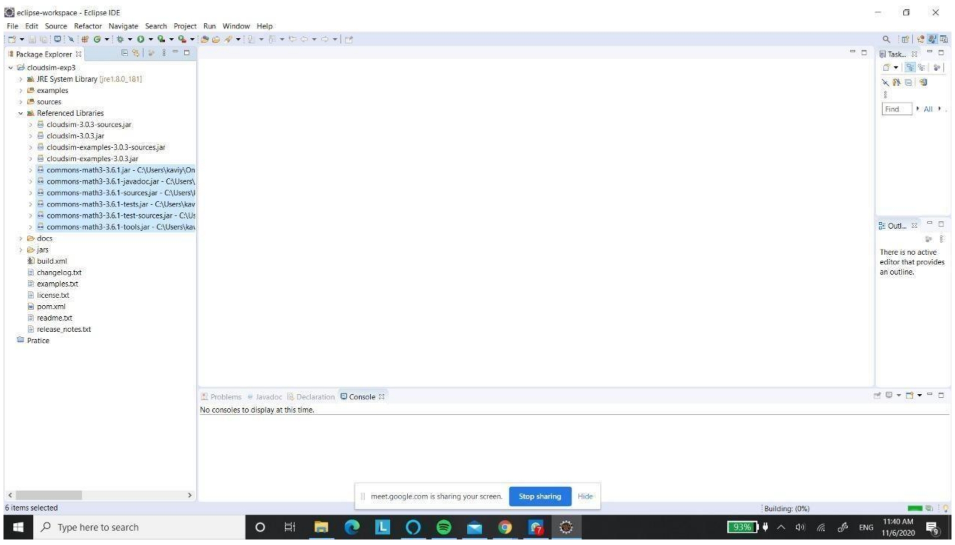
AIM:

Installation of CloudSim into Eclipse to model a cloud computing environment and analyze the VM provisioning using CloudSim.

PROCEDURE: (Installation Steps)

1) Open up Eclipse and go to Menu Section, then click File, keep on clicking New and finally select java project.
2) A JRE environment is required and hence make sure to have jdk installed and finally give a name to the project.
3) CloudSim package is required to be downloaded and extract the CloudSim zip file to set the directory for the project.
4) Common Math from apache is a jar file which is also required to run the math functions in the CloudSim file.
5) Now all the files are installed in the Eclipse environment.
6) Run a program in Cloudsim environment to check if the simulation is complete.

OUTPUT:



```
eclipse-workspace - Eclipse IDE
Package Explorer
cloudsim-exp3
  JRE System Library [jre1.8.0_181]
  examples
  sources
  Referenced Libraries
    cloudsim-3.0.3-sources.jar
    cloudsim-3.0.3.jar
    cloudsim-examples-3.0.3-sources.jar
    cloudsim-examples-3.0.3.jar
    commons-math3-3.6.1.jar - C:\Users\kaviy\On
    commons-math3-3.6.1-javadoc.jar - C:\Users\
    commons-math3-3.6.1-sources.jar - C:\Users\
    commons-math3-3.6.1-tests.jar - C:\Users\kav
    commons-math3-3.6.1-test-sources.jar - C:\Us
    commons-math3-3.6.1-tools.jar - C:\Users\kav
  docs
  jars
  build.xml
  changelog.txt
  examples.txt
  license.txt
  pom.xml
  readme.txt
  release_notes.txt
Pratice

Console
No consoles to display at this time.
```

File Edit Source Refactor Navigate Search Project Run Window Help
Problems Javadoc Declaration Console
There is no active editor that provides an outline.
meet.google.com is sharing your screen. Stop sharing Hide
6 items selected
Building: (0%)
Type here to search
11:40 AM 11/6/2020

RESULT :

Thus program for the given scenario has been compiled and executed successfully.

EX. NO:
DATE:

IMPLEMENT ROUNDROBIN TASK SCHEDULING IN BOTH TIMESHARED AND SPACESHARED CPU ASSIGNMENT

AIM:

To Model a Cloud computing environment having Data centre that had 100 hosts. The hosts are to be modelled to have a CPU core (1000 MIPS), 2 GB of RAM and 1 TB of storage. Consider the workload model for this evaluation including provisioning requests for 400 VMs, with each request demanding 1 CPU core (250 MIPS), 256 MB of RAM and 1 GB of storage. Each VM hosts a web-hosting application service, whose CPU utilization distribution was generated according to the uniform distribution. Each instance of a webhosting service required 150,000 MIPS or about 10 minutes to complete execution assuming 100% utilization.

PROCEDURE :

1) Initialize the CloudSim library and datacenters.
2) Create 2 VM's and the second VM will have twice the priority of VM1 and so will receive twice CPU time. Add the VMs to the vmList.
3) Create two Cloudlets. Where,
Cloudlet properties

```
    intid = 0;
    longlength = 40000;
    longfileSize = 300;
    longoutputSize = 300;
```

4) Add the cloudlets to a list and submit cloudlet list to the broker.
5) Bind the cloudlets to the vms. This way, the broker will submit the bound cloudlets only to the specific VM.
6) Create a datacenter. The steps needed to create a PowerDatacenter,
   (i) We need to create a list to store our machine
   (ii) Machine contains one or more PEs or CPUs/Cores.
   (iii) Create PEs and add these into a list.
   (iv) Create Hosts with its id and list of PEs and add them to the list of machines
   (v) create another machine in the Data center.
   (vi) Create a DatacenterCharacteristics object and create a PowerDatacenter object.
7) Print the Cloudlet objects and observe the result

PROGRAM FOR SPACE SHARED :-

```
package org.cloudbus.cloudsim.examples;

import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.LinkedList;
import java.util.List;
import org.cloudbus.cloudsim.Cloudlet;
import org.cloudbus.cloudsim.CloudletSchedulerSpaceShared;
import org.cloudbus.cloudsim.Datacenter;
import  org.cloudbus.cloudsim.DatacenterBroker;
import org.cloudbus.cloudsim.DatacenterCharacteristics;
import org.cloudbus.cloudsim.Host;
import org.cloudbus.cloudsim.Log;
import org.cloudbus.cloudsim.Pe;
import org.cloudbus.cloudsim.Storage;
import org.cloudbus.cloudsim.UtilizationModel;
import org.cloudbus.cloudsim.UtilizationModelFull;
import org.cloudbus.cloudsim.Vm;
import org.cloudbus.cloudsim.VmAllocationPolicySimple;
import org.cloudbus.cloudsim.VmSchedulerTimeShared;
import org.cloudbus.cloudsim.core.CloudSim;
import org.cloudbus.cloudsim.provisioners.BwProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.PeProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;

publicclass CloudSimExample3 {

/** The cloudlet list. */
 privatestatic List<Cloudlet>cloudletList;

/** The vmlist. */
 privatestatic List<Vm>vmlist;

/**
 * Creates main() to run this example
 */
 publicstaticvoid main(String[] args) {

		Log.printLine("Starting CloudSimExample3...");

		try {
			// First step: Initialize the CloudSim package. It should be called
			// before creating any entities.
			intnum_user = 1; // number of cloud users
			Calendar calendar = Calendar.getInstance();
			booleantrace_flag = false;  // mean trace events
			// Initialize the CloudSim library
			CloudSim.init(num_user, calendar, trace_flag);

			// Second step: Create Datacenters
			//Datacenters are the resource providers in CloudSim. We need at list one of
them to run a CloudSim simulation
			@SuppressWarnings("unused")
			Datacenter datacenter0 = createDatacenter("Datacenter_0");
DatacenterBroker broker = createBroker();
			intbrokerId = broker.getId();

			//Fourth step: Create one virtual machine
			vmlist = new ArrayList<Vm>();

			//VM description
			intvmid = 0;
			intmips = 250;
			longsize = 10000; //image size (MB)
			intram = 2048; //vm memory (MB)
			longbw = 1000;
			intpesNumber = 1; //number of cpus
			String vmm = "Xen"; //VMM name

			//create two VMs
			Vm vm1 = new Vm(vmid, brokerId, mips, pesNumber, ram, bw, size, vmm,
new CloudletSchedulerSpaceShared());

			//the second VM will have twice the priority of VM1 and so will receive twice
CPU time
			vmid++;
			Vm vm2 = new Vm(vmid, brokerId, mips * 2, pesNumber, ram, bw, size, vmm,
new CloudletSchedulerSpaceShared());

			//add the VMs to the vmList
			vmlist.add(vm1);
			vmlist.add(vm2);

			// add the VMs to the vmList
			broker.submitVmList(vmlist);



			//Fifth step: Create two Cloudlets
			cloudletList = new ArrayList<Cloudlet>();

			//Cloudlet properties
			intid = 0;
			longlength = 40000;
			longfileSize = 300;
			longoutputSize = 300;
			UtilizationModel utilizationModel = new UtilizationModelFull();
			Cloudlet cloudlet1 = new Cloudlet(id, length, pesNumber, fileSize, outputSize,
	utilizationModel, utilizationModel, utilizationModel);
			cloudlet1.setUserId(brokerId);

			id++;
			Cloudlet cloudlet2 = new Cloudlet(id, length, pesNumber, fileSize, outputSize,
	utilizationModel, utilizationModel, utilizationModel);
			cloudlet2.setUserId(brokerId);
			id++;
			Cloudlet cloudlet3 = new Cloudlet(id, length, pesNumber, fileSize, outputSize,
	utilizationModel, utilizationModel, utilizationModel);
			cloudlet3.setUserId(brokerId);

			id++;
			Cloudlet cloudlet4 = new Cloudlet(id, length, pesNumber, fileSize, outputSize,
	utilizationModel, utilizationModel, utilizationModel);
			cloudlet4.setUserId(brokerId);

			id++;
			Cloudlet cloudlet5 = new Cloudlet(id, length, pesNumber, fileSize, outputSize,
	utilizationModel, utilizationModel, utilizationModel);
			cloudlet5.setUserId(brokerId);

			id++;
			Cloudlet cloudlet6 = new Cloudlet(id, length, pesNumber, fileSize, outputSize,
	utilizationModel, utilizationModel, utilizationModel);
			cloudlet6.setUserId(brokerId);

			id++;
			Cloudlet cloudlet7 = new Cloudlet(id, length, pesNumber, fileSize, outputSize,
	utilizationModel, utilizationModel, utilizationModel);
			cloudlet7.setUserId(brokerId);

			id++;
			Cloudlet cloudlet8 = new Cloudlet(id, length, pesNumber, fileSize, outputSize,
	utilizationModel, utilizationModel, utilizationModel);
			cloudlet8.setUserId(brokerId);

			id++;

			//add the cloudlets to the list
			cloudletList.add(cloudlet1);
			cloudletList.add(cloudlet2);
			cloudletList.add(cloudlet3);
			cloudletList.add(cloudlet4);
			cloudletList.add(cloudlet5);
			cloudletList.add(cloudlet6);
			cloudletList.add(cloudlet7);
			cloudletList.add(cloudlet8);

			//submit cloudlet list to the broker
			broker.submitCloudletList(cloudletList);
```

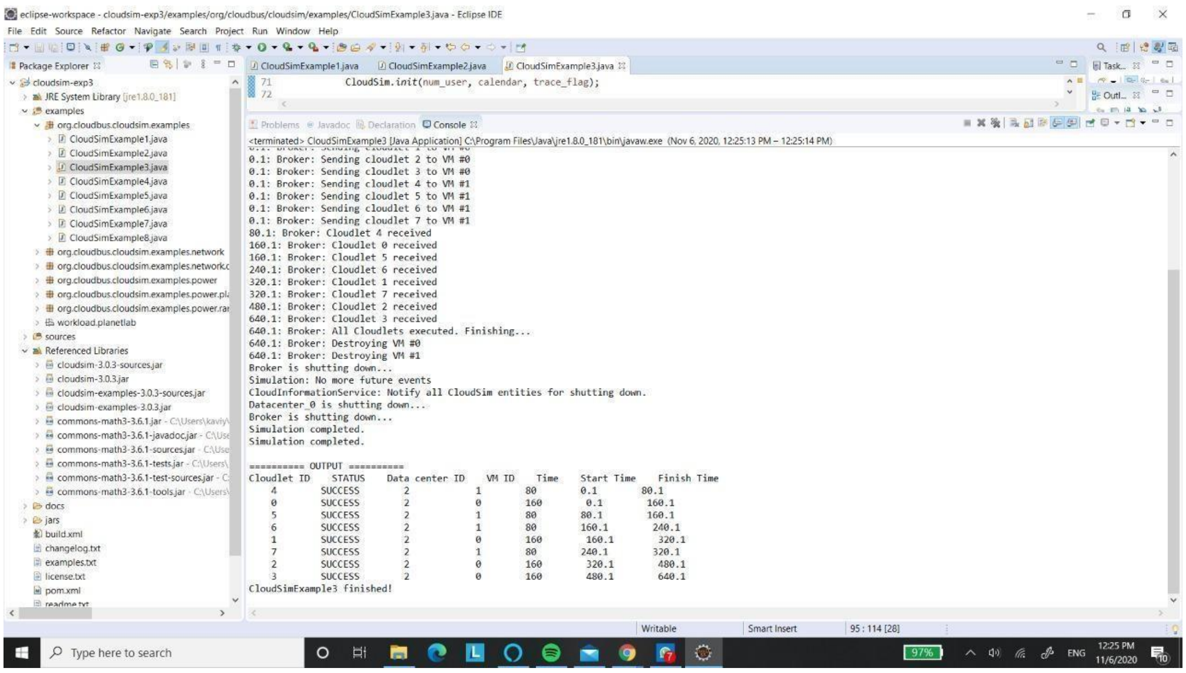
```
                    //bind the cloudlets to the vms. This way, the broker
                    // will submit the bound cloudlets only to the specific VM
                    broker.bindCloudletToVm(cloudlet1.getCloudletId(),vm1.getId());
                    broker.bindCloudletToVm(cloudlet2.getCloudletId(),vm1.getId());
                    broker.bindCloudletToVm(cloudlet3.getCloudletId(),vm1.getId());
                    broker.bindCloudletToVm(cloudlet4.getCloudletId(),vm1.getId());
                    broker.bindCloudletToVm(cloudlet5.getCloudletId(),vm2.getId());
                    broker.bindCloudletToVm(cloudlet6.getCloudletId(),vm2.getId());
                    broker.bindCloudletToVm(cloudlet7.getCloudletId(),vm2.getId());
                    broker.bindCloudletToVm(cloudlet8.getCloudletId(),vm2.getId());

                    // Sixth step: Starts the simulation
                    CloudSim.startSimulation();


                    // Final step: Print results when simulation is over
                    List<Cloudlet>newList = broker.getCloudletReceivedList();

                    CloudSim.stopSimulation();

printCloudletList(newList);

                    Log.printLine("CloudSimExample3 finished!");
          }
          catch (Exception e) {
                    e.printStackTrace();
                    Log.printLine("The simulation has been terminated due to an unexpected
error");
          }
}

privatestatic Datacenter createDatacenter(String name){

          // Here are the steps needed to create a PowerDatacenter:
          // 1. We need to create a list to store
          //    our machine
          List<Host>hostList = new ArrayList<Host>();

          // 2. A Machine contains one or more PEs or CPUs/Cores.
          // In this example, it will have only one core.
          List<Pe>peList = new ArrayList<Pe>();

          intmips = 1000;

          // 3. Create PEs and add these into a list.
          peList.add(new Pe(0, new PeProvisionerSimple(mips))); // need to store Pe id and
MIPS Rating

          //4. Create Hosts with its id and list of PEs and add them to the list of machines
          inthostId=0;
```

```
        intram = 2048; //host memory (MB)
        longstorage = 1000000; //host storage
        intbw = 10000;

        hostList.add(
                new Host(
                        hostId,
                        new RamProvisionerSimple(ram),
                        new BwProvisionerSimple(bw),
                        storage,
                        peList,
                        new VmSchedulerTimeShared(peList)
                )
        ); // This is our first machine

        //create another machine in the Data center
        List<Pe>peList2 = new ArrayList<Pe>();

        peList2.add(new Pe(0, new PeProvisionerSimple(mips)));

        hostId++;

        hostList.add(
                new Host(
                        hostId,
                        new RamProvisionerSimple(ram),
                        new BwProvisionerSimple(bw),
                        storage,
                        peList2,
                        new VmSchedulerTimeShared(peList2)
                )
        ); // This is our second machine



        // 5. Create a DatacenterCharacteristics object that stores the
        //    properties of a data center: architecture, OS, list of
        //    Machines, allocation policy: time- or space-shared, time zone
        //    and its price (G$/Pe time unit).
        String  arch  =  "x86";  //  system  architecture
        String   os   =   "Linux";  //   operating   system
        String vmm = "Xen";
        doubletime_zone = 10.0;         // time zone this resource located
        doublecost = 3.0;               // the cost of using processing in this resource
        doublecostPerMem = 0.05;                // the cost of using memory in this resource
        doublecostPerStorage = 0.001;           // the cost of using storage in this resource
        doublecostPerBw = 0.0;                          // the cost of using bw in this resource
        LinkedList<Storage>storageList = new LinkedList<Storage>();       //we are not adding
SAN devices by now

      DatacenterCharacteristics characteristics = new DatacenterCharacteristics(
arch, os, vmm, hostList, time_zone, cost, costPerMem, costPerStorage, costPerBw);
```

```
		// 6. Finally, we need to create a PowerDatacenter object.
		Datacenter datacenter = null;
		try {
			datacenter = new Datacenter(name, characteristics, new
VmAllocationPolicySimple(hostList), storageList, 0);
		} catch (Exception e) {
			e.printStackTrace();
		}

		returndatacenter;
	}

	//We strongly encourage users to develop their own broker policies, to submit vms and
cloudlets according
	//to the specific rules of the simulated scenario
	privatestatic DatacenterBroker createBroker(){

		DatacenterBroker broker = null;
		try {
			broker = new DatacenterBroker("Broker");
		} catch (Exception e) {
			e.printStackTrace();
			returnnull;
		}
		returnbroker;
	}

	/**
	 * Prints the Cloudlet objects
	 * @param list  list of Cloudlets
	 */
	privatestaticvoid printCloudletList(List<Cloudlet>list) {
		intsize = list.size();
		Cloudlet cloudlet;

		String indent = "    ";
		Log.printLine();
		Log.printLine("========== OUTPUT ==========");
		Log.printLine("Cloudlet ID" + indent + "STATUS" + indent +
				"Data center ID" + indent + "VM ID" + indent + "Time" + indent +
"Start Time" + indent + "Finish Time");

		DecimalFormat dft = new DecimalFormat("###.##");
		for (inti = 0; i<size; i++) {
			cloudlet = list.get(i);
			Log.print(indent + cloudlet.getCloudletId() + indent + indent);

			if (cloudlet.getCloudletStatus() == Cloudlet.SUCCESS){
				Log.print("SUCCESS");
```

```
                                Log.*printLine*( indent + indent + cloudlet.getResourceId() + indent +
indent + indent + cloudlet.getVmId() +
                                                    indent + indent +
dft.format(cloudlet.getActualCPUTime()) + indent + indent +
dft.format(cloudlet.getExecStartTime())+
                                                    indent + indent + dft.format(cloudlet.getFinishTime()));
                      }
            }

}
}
```

OUTPUT :-



```
<terminated> CloudSimExample3 [Java Application] C:\Program Files\Java\jre1.8.0_181\bin\javaw.exe (Nov 6, 2020, 12:25:13 PM – 12:25:14 PM)
0.1: Broker: Sending cloudlet 2 to VM #0
0.1: Broker: Sending cloudlet 3 to VM #0
0.1: Broker: Sending cloudlet 4 to VM #1
0.1: Broker: Sending cloudlet 5 to VM #1
0.1: Broker: Sending cloudlet 6 to VM #1
0.1: Broker: Sending cloudlet 7 to VM #1
80.1: Broker: Cloudlet 4 received
160.1: Broker: Cloudlet 0 received
160.1: Broker: Cloudlet 5 received
240.1: Broker: Cloudlet 6 received
320.1: Broker: Cloudlet 1 received
320.1: Broker: Cloudlet 7 received
480.1: Broker: Cloudlet 2 received
640.1: Broker: Cloudlet 3 received
640.1: Broker: All Cloudlets executed. Finishing...
640.1: Broker: Destroying VM #0
640.1: Broker: Destroying VM #1
Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
    4        SUCCESS        2             1        80       0.1          80.1
    0        SUCCESS        2             0        160       0.1          160.1
    5        SUCCESS        2             1        80       80.1          160.1
    6        SUCCESS        2             1        80       160.1          240.1
    1        SUCCESS        2             0        160       160.1          320.1
    7        SUCCESS        2             1        80       240.1          320.1
    2        SUCCESS        2             0        160       320.1          480.1
    3        SUCCESS        2             0        160       480.1          640.1
CloudSimExample3 finished!
```

PROGRAM FOR TIME SHARED :-

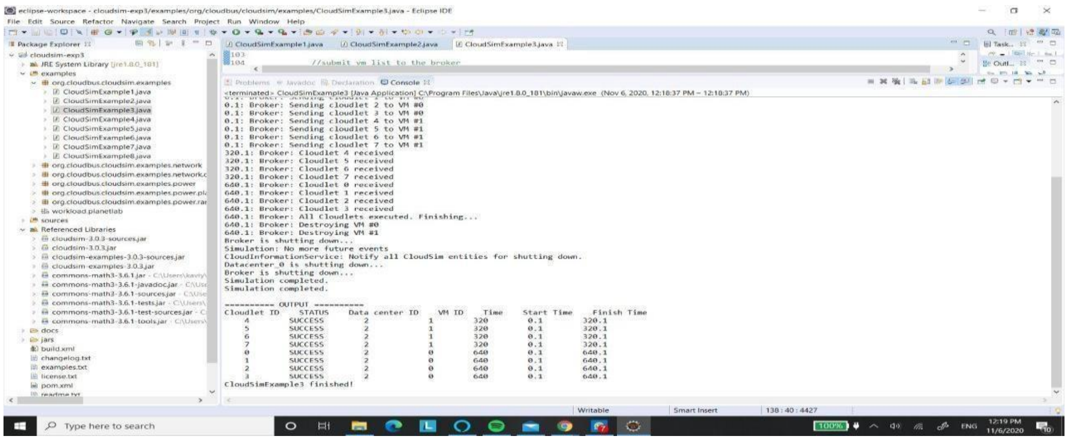
```
package org.cloudbus.cloudsim.examples;

import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.LinkedList;
import java.util.List;

import org.cloudbus.cloudsim.Cloudlet;
import org.cloudbus.cloudsim.CloudletSchedulerSpaceShared;
import org.cloudbus.cloudsim.Datacenter;
import  org.cloudbus.cloudsim.DatacenterBroker;
import org.cloudbus.cloudsim.DatacenterCharacteristics;
import org.cloudbus.cloudsim.Host;
import org.cloudbus.cloudsim.Log;
import org.cloudbus.cloudsim.Pe;
import org.cloudbus.cloudsim.Storage;
import org.cloudbus.cloudsim.UtilizationModel;
import org.cloudbus.cloudsim.UtilizationModelFull;
import org.cloudbus.cloudsim.Vm;
import org.cloudbus.cloudsim.VmAllocationPolicySimple;
import org.cloudbus.cloudsim.VmSchedulerTimeShared;
import org.cloudbus.cloudsim.core.CloudSim;
import org.cloudbus.cloudsim.provisioners.BwProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.PeProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;

publicclass CloudSimExample3 {

/** The cloudlet list. */
privatestatic List<Cloudlet>cloudletList;

/** The vmlist. */
privatestatic List<Vm>vmlist;

/**
 * Creates main() to run this example
 */
publicstaticvoid main(String[] args) {

        Log.printLine("Starting CloudSimExample3...");

        try {
                // First step: Initialize the CloudSim package. It should be called
                // before creating any entities.
                intnum_user = 1; // number of cloud users
                Calendar calendar = Calendar.getInstance();
                booleantrace_flag = false;  // mean trace events

                // Initialize the CloudSim library
                CloudSim.init(num_user, calendar, trace_flag);

                // Second step: Create Datacenters
                //Datacenters are the resource providers in CloudSim. We need at list one of
them to run a CloudSim simulation
                @SuppressWarnings("unused")
                Datacenter datacenter0 = createDatacenter("Datacenter_0");
DatacenterBroker broker = createBroker();
                intbrokerId = broker.getId();

                //Fourth step: Create one virtual machine
                vmlist = new ArrayList<Vm>();

                //VM description
                intvmid = 0;
                intmips = 250;
                longsize = 10000; //image size (MB)
                intram = 2048; //vm memory (MB)
```

```
		longbw = 1000;
		intpesNumber = 1; //number of cpus
		String vmm = "Xen"; //VMM name

		//create two VMs
		Vm vm1 = new Vm(vmid, brokerId, mips, pesNumber, ram, bw, size, vmm, 
new CloudletSchedulerSpaceShared());

		//the second VM will have twice the priority of VM1 and so will receive twice 
CPU time
		vmid++;
		Vm vm2 = new Vm(vmid, brokerId, mips * 2, pesNumber, ram, bw, size, vmm, 
new CloudletSchedulerSpaceShared());

		//add the VMs to the vmList
		vmlist.add(vm1);
		vmlist.add(vm2);

		// add the VMs to the vmList
		broker.submitVmList(vmlist);


		//Fifth step: Create two Cloudlets
		cloudletList = new ArrayList<Cloudlet>();

		//Cloudlet properties
		intid = 0;
		longlength = 40000;
		longfileSize = 300;
		longoutputSize = 300;
		UtilizationModel utilizationModel = new UtilizationModelFull();

		Cloudlet cloudlet1 = new Cloudlet(id, length, pesNumber, fileSize, outputSize, 
utilizationModel, utilizationModel, utilizationModel);
		cloudlet1.setUserId(brokerId);

		id++;
		Cloudlet cloudlet2 = new Cloudlet(id, length, pesNumber, fileSize, outputSize, 
utilizationModel, utilizationModel, utilizationModel);
		cloudlet2.setUserId(brokerId);
		id++;
		Cloudlet cloudlet3 = new Cloudlet(id, length, pesNumber, fileSize, outputSize, 
utilizationModel, utilizationModel, utilizationModel);
		cloudlet3.setUserId(brokerId);

		id++;
		Cloudlet cloudlet4 = new Cloudlet(id, length, pesNumber, fileSize, outputSize, 
utilizationModel, utilizationModel, utilizationModel);
		cloudlet4.setUserId(brokerId);

		id++;
		Cloudlet cloudlet5 = new Cloudlet(id, length, pesNumber, fileSize, outputSize,
utilizationModel, utilizationModel, utilizationModel);
		cloudlet5.setUserId(brokerId);

		id++;
		Cloudlet cloudlet6 = new Cloudlet(id, length, pesNumber, fileSize, outputSize,
utilizationModel, utilizationModel, utilizationModel);
		cloudlet6.setUserId(brokerId);

		id++;
		Cloudlet cloudlet7 = new Cloudlet(id, length, pesNumber, fileSize, outputSize,
utilizationModel, utilizationModel, utilizationModel);
		cloudlet7.setUserId(brokerId);

		id++;
		Cloudlet cloudlet8 = new Cloudlet(id, length, pesNumber, fileSize, outputSize,
utilizationModel, utilizationModel, utilizationModel);
		cloudlet8.setUserId(brokerId);

		id++;

		//add the cloudlets to the list
		cloudletList.add(cloudlet1);
		cloudletList.add(cloudlet2);
		cloudletList.add(cloudlet3);
		cloudletList.add(cloudlet4);
		cloudletList.add(cloudlet5);
		cloudletList.add(cloudlet6);
		cloudletList.add(cloudlet7);
		cloudletList.add(cloudlet8);

		//submit cloudlet list to the broker
		broker.submitCloudletList(cloudletList);


		//bind the cloudlets to the vms. This way, the broker
		// will submit the bound cloudlets only to the specific VM
		broker.bindCloudletToVm(cloudlet1.getCloudletId(),vm1.getId());
		broker.bindCloudletToVm(cloudlet2.getCloudletId(),vm1.getId());
		broker.bindCloudletToVm(cloudlet3.getCloudletId(),vm1.getId());
		broker.bindCloudletToVm(cloudlet4.getCloudletId(),vm1.getId());
		broker.bindCloudletToVm(cloudlet5.getCloudletId(),vm2.getId());
		broker.bindCloudletToVm(cloudlet6.getCloudletId(),vm2.getId());
		broker.bindCloudletToVm(cloudlet7.getCloudletId(),vm2.getId());
		broker.bindCloudletToVm(cloudlet8.getCloudletId(),vm2.getId());


		CloudSim.startSimulation();



		List<Cloudlet>newList = broker.getCloudletReceivedList();
```

```
                    CloudSim.stopSimulation();

printCloudletList(newList);

                    Log.printLine("CloudSimExample3 finished!");
          }
          catch (Exception e) {
                    e.printStackTrace();
                    Log.printLine("The simulation has been terminated due to an unexpected
error");
          }
}

privatestatic Datacenter createDatacenter(String name){


          List<Host>hostList = new ArrayList<Host>();


          List<Pe>peList = new ArrayList<Pe>();

          intmips = 1000;


          peList.add(new Pe(0, new PeProvisionerSimple(mips
          inthostId=0;
          intram = 2048; //host memory (MB)
          longstorage = 1000000; //host storage
          intbw = 10000;

          hostList.add(
                    new Host(
                              hostId,
                              new RamProvisionerSimple(ram),
                              new BwProvisionerSimple(bw),
                              storage,
                              peList,
                              new VmSchedulerTimeShared(peList)
                    )
          );
          List<Pe>peList2 = new ArrayList<Pe>();

          peList2.add(new Pe(0, new PeProvisionerSimple(mips)));

          hostId++;

          hostList.add(
                    new Host(
                              hostId,
                              new RamProvisionerSimple(ram),
                              new BwProvisionerSimple(bw),
                              storage,
                              peList2,
                              new VmSchedulerTimeShared(peList2)
                    )
          );
          String  arch  =  "x86";  //  system  architecture
          String   os   =   "Linux";   //   operating   system
          String vmm = "Xen";
          doubletime_zone = 10.0;         // time zone this resource located
          doublecost = 3.0;              // the cost of using processing in this resource
          doublecostPerMem = 0.05;              // the cost of using memory in this resource
          doublecostPerStorage = 0.001;         // the cost of using storage in this resource
          doublecostPerBw = 0.0;                          // the cost of using bw in this resource
          LinkedList<Storage>storageList = new LinkedList<Storage>();      //we are not adding
SAN devices by now

      DatacenterCharacteristics characteristics = new DatacenterCharacteristics(
arch, os, vmm, hostList, time_zone, cost, costPerMem, costPerStorage, costPerBw);

          // 6. Finally, we need to create a PowerDatacenter object.
          Datacenter datacenter = null;
          try {
                    datacenter = new Datacenter(name, characteristics, new
VmAllocationPolicySimple(hostList), storageList, 0);
          } catch (Exception e) {
                    e.printStackTrace();
          }

          returndatacenter;
}

//We strongly encourage users to develop their own broker policies, to submit vms and
cloudlets according
//to the specific rules of the simulated scenario
privatestatic DatacenterBroker createBroker(){

          DatacenterBroker broker = null;
          try {
                    broker = new DatacenterBroker("Broker");
          } catch (Exception e) {
                    e.printStackTrace();
                    returnnull;
          }
          returnbroker;
}

/**
 * Prints the Cloudlet objects
 * @param list  list of Cloudlets
 */
privatestaticvoid printCloudletList(List<Cloudlet>list) {
          intsize = list.size();
```

```
            Cloudlet cloudlet;

            String indent = "    ";
            Log.printLine();
            Log.printLine("========== OUTPUT ==========");
            Log.printLine("Cloudlet ID" + indent + "STATUS" + indent +
                                    "Data center ID" + indent + "VM ID" + indent + "Time" + indent +
"Start Time" + indent + "Finish Time");

            DecimalFormat dft = new DecimalFormat("###.##");
            for (inti = 0; i<size; i++) {
                        cloudlet = list.get(i);
                        Log.print(indent + cloudlet.getCloudletId() + indent + indent);

                        if (cloudlet.getCloudletStatus() == Cloudlet.SUCCESS){
                                    Log.print("SUCCESS");

                                    Log.printLine( indent + indent + cloudlet.getResourceId() + indent +
indent + indent + cloudlet.getVmId() +
                                                            indent + indent +
dft.format(cloudlet.getActualCPUTime()) + indent + indent +
dft.format(cloudlet.getExecStartTime())+
                                                            indent + indent + dft.format(cloudlet.getFinishTime()));
                        }
            }

}
}
```

OUTPUT :



```
<terminated> CloudSimExample3 [Java Application] C:\Program Files\Java\jre1.8.0_181\bin\javaw.exe (Nov 6, 2020, 12:18:37 PM – 12:18:37 PM)
0.1: Broker: Sending cloudlet 2 to VM #0
0.1: Broker: Sending cloudlet 3 to VM #0
0.1: Broker: Sending cloudlet 4 to VM #1
0.1: Broker: Sending cloudlet 5 to VM #1
0.1: Broker: Sending cloudlet 6 to VM #1
0.1: Broker: Sending cloudlet 7 to VM #1
320.1: Broker: Cloudlet 4 received
320.1: Broker: Cloudlet 5 received
320.1: Broker: Cloudlet 6 received
320.1: Broker: Cloudlet 7 received
640.1: Broker: Cloudlet 0 received
640.1: Broker: Cloudlet 1 received
640.1: Broker: Cloudlet 2 received
640.1: Broker: Cloudlet 3 received
640.1: Broker: All Cloudlets executed. Finishing...
640.1: Broker: Destroying VM #0
640.1: Broker: Destroying VM #1
Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
    4        SUCCESS        2              1        320       0.1         320.1
    5        SUCCESS        2              1        320       0.1         320.1
    6        SUCCESS        2              1        320       0.1         320.1
    7        SUCCESS        2              1        320       0.1         320.1
    0        SUCCESS        2              0        640       0.1         640.1
    1        SUCCESS        2              0        640       0.1         640.1
    2        SUCCESS        2              0        640       0.1         640.1
    3        SUCCESS        2              0        640       0.1         640.1
CloudSimExample3 finished!
```

RESULT:

Thus program for the given scenario has been compiled and executed successfully for space and time shared.

EX. NO:

DATE: **HADOOP INSTALLATION**

AIM

To set-up one node Hadoop cluster

- System Update
- Install Java
- Add a dedicated Hadoop user
- Install SSH and setup SSH certificates
- Check if SSH works
- Install Hadoop
- Modify Hadoop config files
- Format Hadoop filesystem
- Start Hadoop
- Check Hadoop through web UI
- Stop Hadoop

THEORY

Hadoop is an Apache open source framework written in java that allows distributed processing of large datasets across clusters of computers using simple programming models. A Hadoop frame-worked application works in an environment that provides distributed storage and computation across clusters of computers. Hadoop is designed to scale up from single server to thousands of machines, each offering local computation and storage.

HADOOP ARCHITECTURE

Hadoop framework includes following four modules:

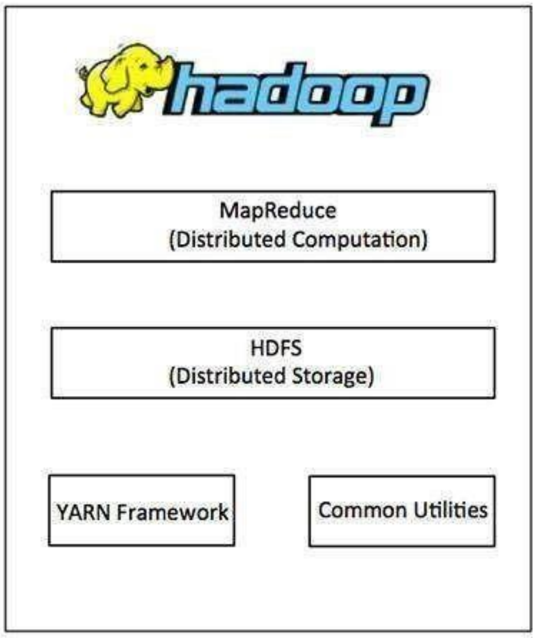
- **Hadoop Common:** These are Java libraries and utilities required by other Hadoop modules. These libraries provides filesystem and OS level abstractions and contains the necessary Java files and scripts required to start Hadoop.

- **Hadoop YARN:** This is a framework for job scheduling and cluster resource management.

- **Hadoop Distributed File System (HDFS™):** A distributed file system that provides high-throughput access to application data.

- **Hadoop MapReduce:** This is YARN-based system for parallel processing of large data sets.

We can use following diagram to depict these four components available in Hadoop framework.



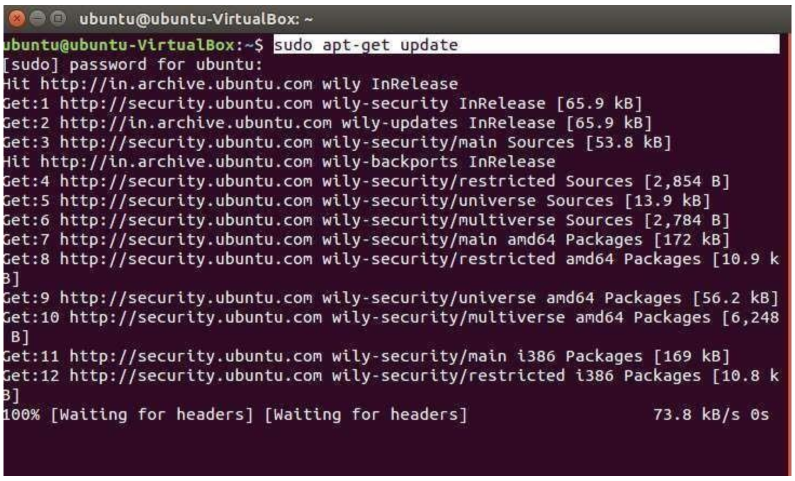
Hadoop Architecture

Since 2012, the term "Hadoop" often refers not just to the base modules mentioned above but also to the collection of additional software packages that can be installed on top of or alongside Hadoop, such as Apache Pig, Apache Hive, Apache HBase, Apache Spark etc.

PROCEDURE

Step 1 – System Update

$ sudo apt-get update



```
ubuntu@ubuntu-VirtualBox: ~
ubuntu@ubuntu-VirtualBox:~$ sudo apt-get update
[sudo] password for ubuntu:
Hit http://in.archive.ubuntu.com wily InRelease
Get:1 http://security.ubuntu.com wily-security InRelease [65.9 kB]
Get:2 http://in.archive.ubuntu.com wily-updates InRelease [65.9 kB]
Get:3 http://security.ubuntu.com wily-security/main Sources [53.8 kB]
Hit http://in.archive.ubuntu.com wily-backports InRelease
Get:4 http://security.ubuntu.com wily-security/restricted Sources [2,854 B]
Get:5 http://security.ubuntu.com wily-security/universe Sources [13.9 kB]
Get:6 http://security.ubuntu.com wily-security/multiverse Sources [2,784 B]
Get:7 http://security.ubuntu.com wily-security/main amd64 Packages [172 kB]
Get:8 http://security.ubuntu.com wily-security/restricted amd64 Packages [10.9 k
B]
Get:9 http://security.ubuntu.com wily-security/universe amd64 Packages [56.2 kB]
Get:10 http://security.ubuntu.com wily-security/multiverse amd64 Packages [6,248
 B]
Get:11 http://security.ubuntu.com wily-security/main i386 Packages [169 kB]
Get:12 http://security.ubuntu.com wily-security/restricted i386 Packages [10.8 k
B]
100% [Waiting for headers] [Waiting for headers]             73.8 kB/s 0s
```

Step 2 – Install Java and Set JAVA_HOME

//This first thing to do is to setup the webupd8 ppa on your system. Run the following command and proceed.

$ sudo apt-add-repository ppa:webupd8team/java

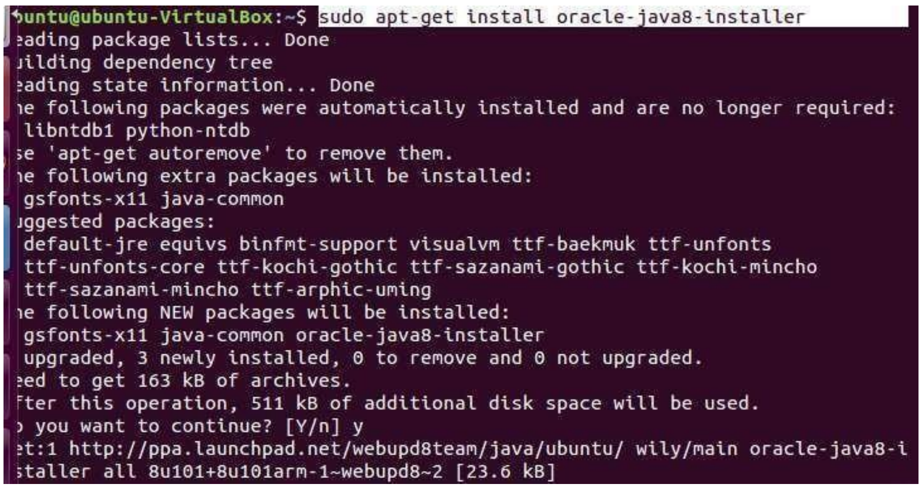
$ sudo apt-get update

//After setting up the ppa repository, update the package cache as well.

//Install the Java 8 installer

$ sudo apt-get install oracle-java8-installer

// After the installation is finished, Oracle Java is setup. Run the java command again to check the version and vendor.

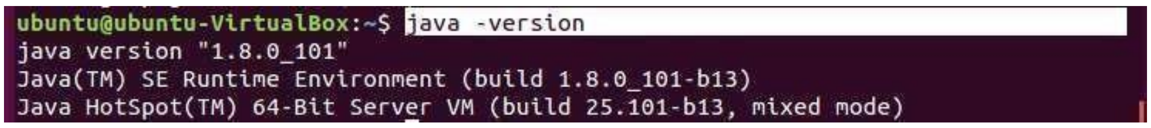


```
buntu@ubuntu-VirtualBox:~$ sudo apt-get install oracle-java8-installer
eading package lists... Done
uilding dependency tree
eading state information... Done
he following packages were automatically installed and are no longer required:
 libntdb1 python-ntdb
se 'apt-get autoremove' to remove them.
he following extra packages will be installed:
 gsfonts-x11 java-common
uggested packages:
 default-jre equivs binfmt-support visualvm ttf-baekmuk ttf-unfonts
 ttf-unfonts-core ttf-kochi-gothic ttf-sazanami-gothic ttf-kochi-mincho
 ttf-sazanami-mincho ttf-arphic-uming
he following NEW packages will be installed:
 gsfonts-x11 java-common oracle-java8-installer
 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.
eed to get 163 kB of archives.
fter this operation, 511 kB of additional disk space will be used.
 you want to continue? [Y/n] y
et:1 http://ppa.launchpad.net/webupd8team/java/ubuntu/ wily/main oracle-java8-i
staller all 8u101+8u101arm-1~webupd8~2 [23.6 kB]
```

[or]

$ sudo apt-get install default-jdk

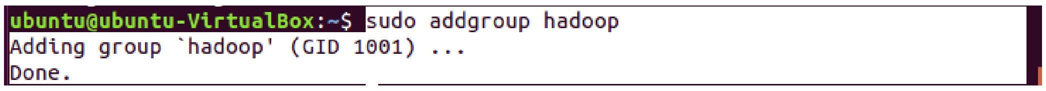
$ java -version



```
ubuntu@ubuntu-VirtualBox:~$ java -version
java version "1.8.0_101"
Java(TM) SE Runtime Environment (build 1.8.0_101-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.101-b13, mixed mode)
```

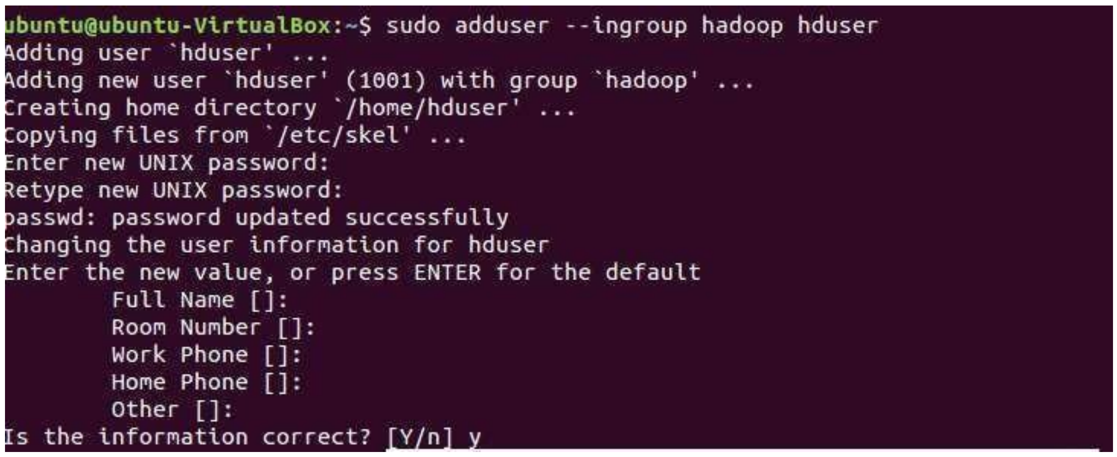
Step 3 – Add a dedicated Hadoop user

$ sudo addgroup hadoop



```
ubuntu@ubuntu-VirtualBox:~$ sudo addgroup hadoop
Adding group `hadoop' (GID 1001) ...
Done.
```

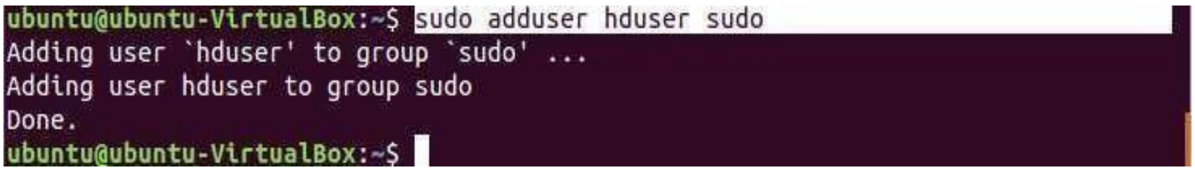
$ **sudo adduser --ingroup hadoop hduser**



```
ubuntu@ubuntu-VirtualBox:~$ sudo adduser --ingroup hadoop hduser
Adding user `hduser' ...
Adding new user `hduser' (1001) with group `hadoop' ...
Creating home directory `/home/hduser' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for hduser
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] y
```

// Add hduser to sudo user group

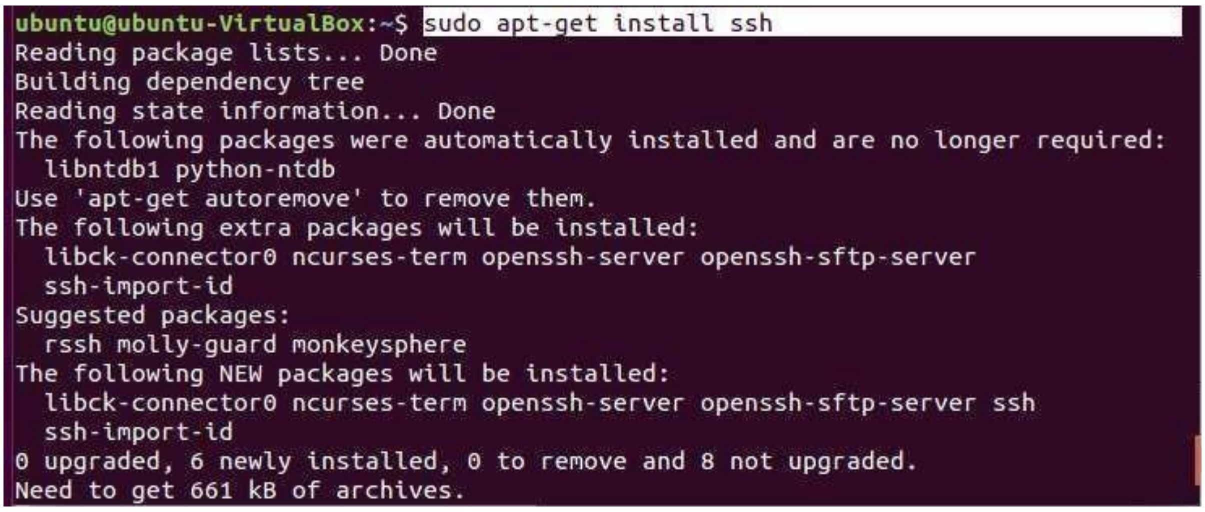
$ sudo adduser hduser sudo



```
ubuntu@ubuntu-VirtualBox:~$ sudo adduser hduser sudo
Adding user `hduser' to group `sudo' ...
Adding user hduser to group sudo
Done.
ubuntu@ubuntu-VirtualBox:~$
```

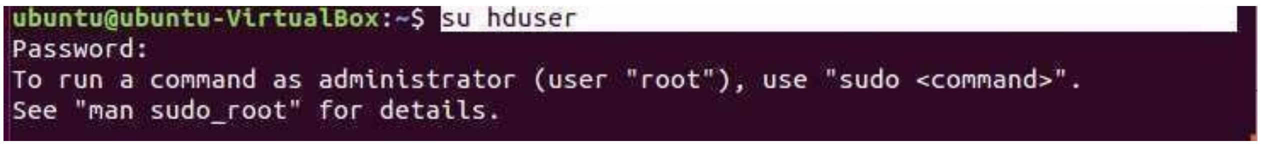
Step 4 – Install SSH and Create Certificates

$ sudo apt-get install ssh



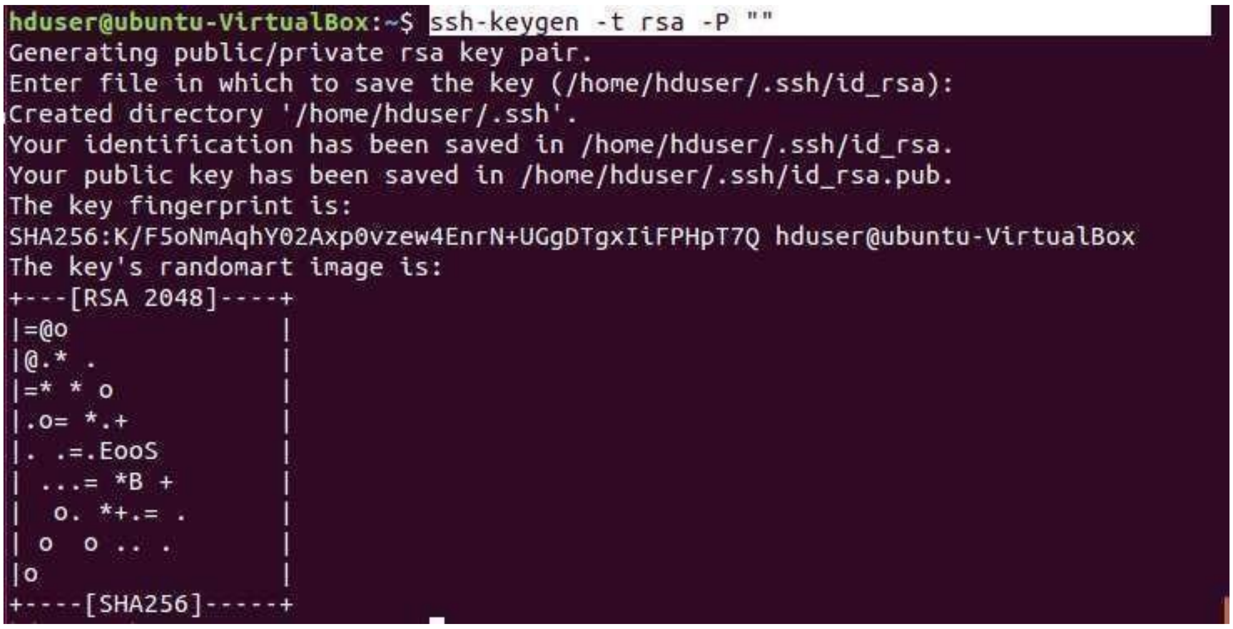
```
ubuntu@ubuntu-VirtualBox:~$ sudo apt-get install ssh
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libntdb1 python-ntdb
Use 'apt-get autoremove' to remove them.
The following extra packages will be installed:
  libck-connector0 ncurses-term openssh-server openssh-sftp-server
  ssh-import-id
Suggested packages:
  rssh molly-guard monkeysphere
The following NEW packages will be installed:
  libck-connector0 ncurses-term openssh-server openssh-sftp-server ssh
  ssh-import-id
0 upgraded, 6 newly installed, 0 to remove and 8 not upgraded.
Need to get 661 kB of archives.
```

$ su hduser



```
ubuntu@ubuntu-VirtualBox:~$ su hduser
Password:
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
```

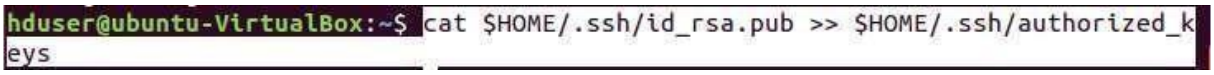
$ ssh-keygen -t rsa -P ""



```
hduser@ubuntu-VirtualBox:~$ ssh-keygen -t rsa -P ""
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hduser/.ssh/id_rsa):
Created directory '/home/hduser/.ssh'.
Your identification has been saved in /home/hduser/.ssh/id_rsa.
Your public key has been saved in /home/hduser/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:K/F5oNmAqhY02Axp0vzew4EnrN+UGgDTgxIiFPHpT7Q hduser@ubuntu-VirtualBox
The key's randomart image is:
+---[RSA 2048]----+
|=@o              |
|@.* .            |
|=* * o           |
|.o= *.+          |
|. .=.EooS        |
| ...= *B +       |
|  o. *+.= .      |
| o  o .. .       |
|o                |
+----[SHA256]-----+
```

// Set Environmental variables

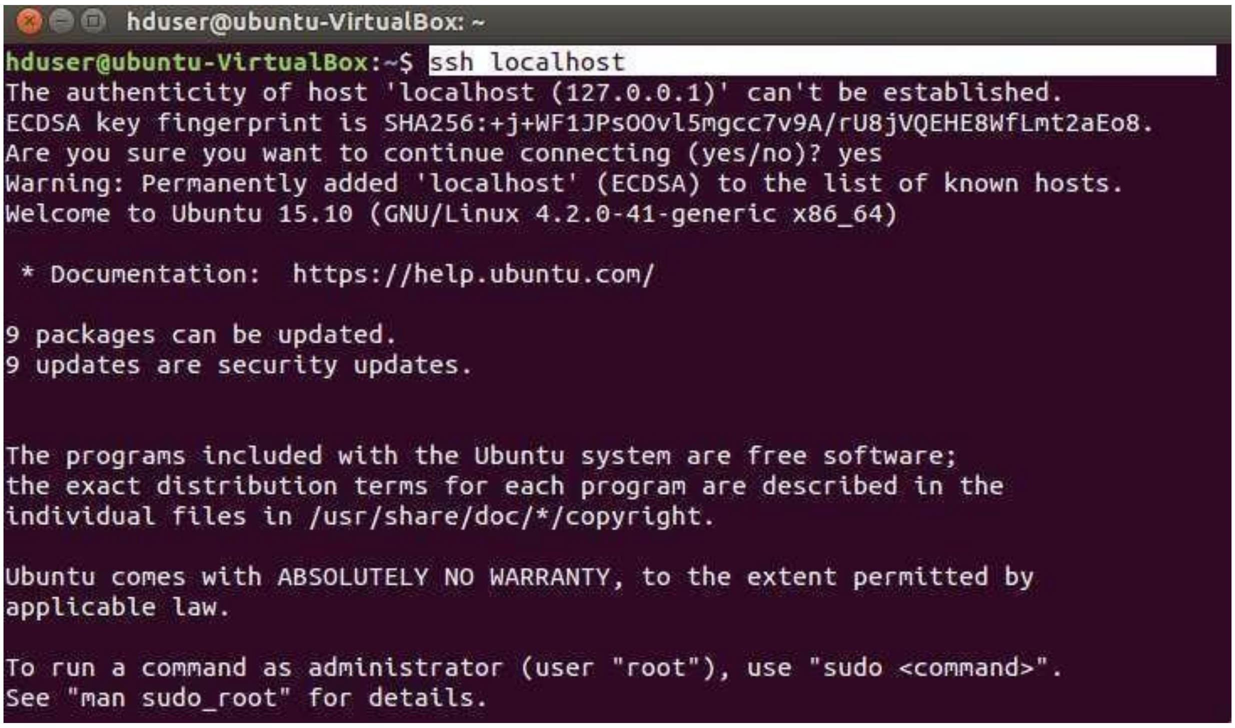
$ cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys



```
hduser@ubuntu-VirtualBox:~$ cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_k
eys
```

Step 5 – Check if SSH works

$ ssh localhost



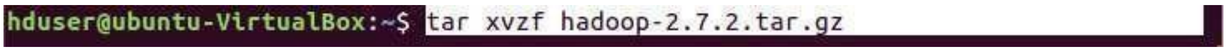
```
hduser@ubuntu-VirtualBox: ~
hduser@ubuntu-VirtualBox:~$ ssh localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ECDSA key fingerprint is SHA256:+j+WF1JPsOOvl5mgcc7v9A/rU8jVQEHE8WfLmt2aEo8.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 15.10 (GNU/Linux 4.2.0-41-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

9 packages can be updated.
9 updates are security updates.


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
```

Step 6 – Install Hadoop

$ wget https://archive.apache.org/dist/hadoop/core/hadoop-2.8.4/hadoop-2.8.4.tar.gz

// Extract Hadoop-2.8.4

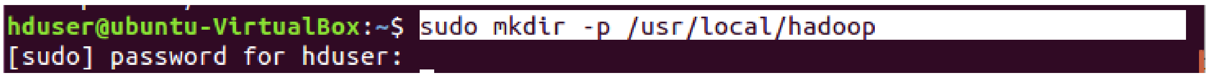
$ sudo tar xvzf hadoop-2.8.4.tar.gz



```
hduser@ubuntu-VirtualBox:~$ tar xvzf hadoop-2.7.2.tar.gz
```

// Create a folder 'hadoop' in /usr/local

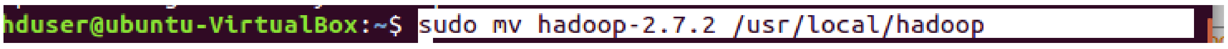
$ sudo mkdir –p /usr/local/hadoop



```
hduser@ubuntu-VirtualBox:~$ sudo mkdir -p /usr/local/hadoop
[sudo] password for hduser:
```

// Move the Hadoop folder to /usr/local/hadoop

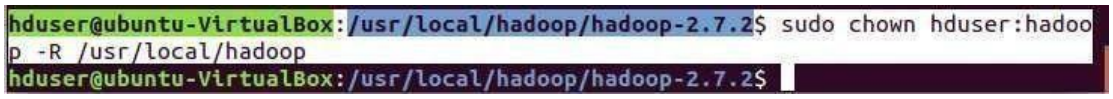
$ sudo mv hadoop-2.8.4 /usr/local/hadoop



```
hduser@ubuntu-VirtualBox:~$ sudo mv hadoop-2.7.2 /usr/local/hadoop
```

// Assigning read and write access to Hadoop folder

$ sudo chown –R hduser:hadoop /usr/local/hadoop



```
hduser@ubuntu-VirtualBox:/usr/local/hadoop/hadoop-2.7.2$ sudo chown hduser:hadoo
p -R /usr/local/hadoop
hduser@ubuntu-VirtualBox:/usr/local/hadoop/hadoop-2.7.2$
```

Step 7 - Modify Hadoop config files

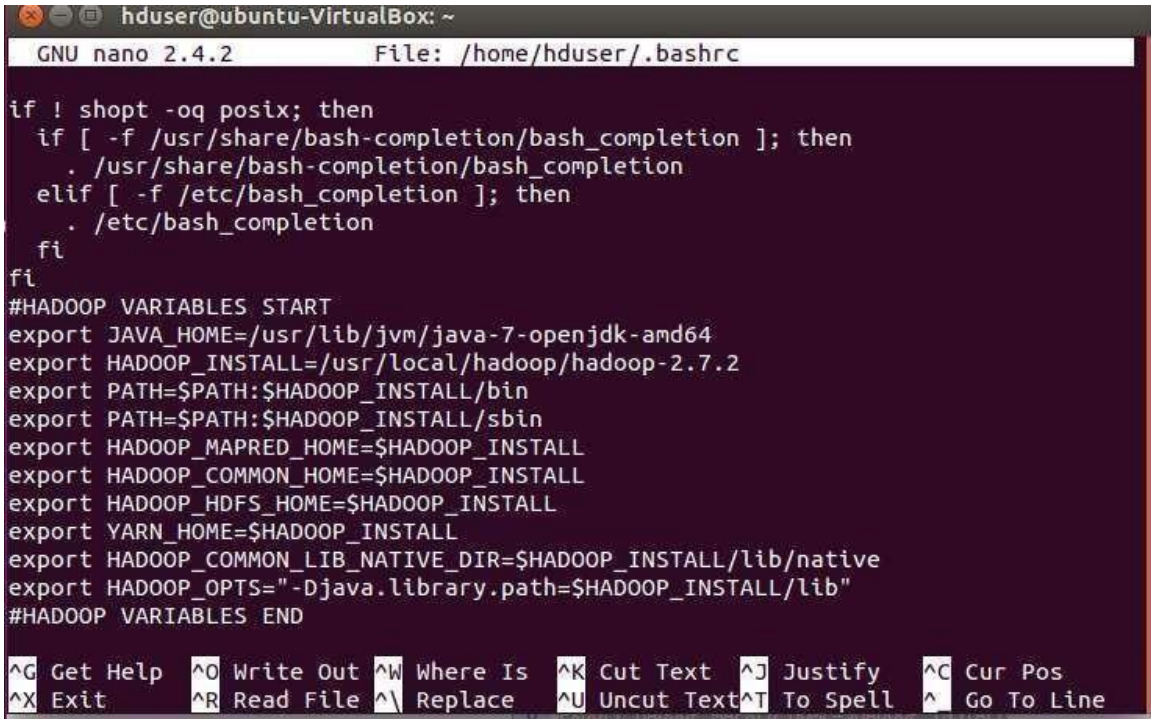
//Hadoop Environmental variable setting – The following files will be modified

1. **~/.bashrc**
2. **/usr/local/hadoop/hadoop-2.8.4/etc/hadoop/hadoop-env.sh**
3. **/usr/local/hadoop/hadoop-2.8.4/etc/hadoop/core-site.xml**
4. **/usr/local/hadoop/hadoop-2.8.4/etc/hadoop/hdfs-site.xml**
5. **/usr/local/hadoop/hadoop-2.8.4/etc/hadoop/yarn-site.xml**
6. **/usr/local/hadoop/hadoop-2.8.4/etc/hadoop/mapred-site.xml.template**

$ sudo nano ~/.bashrc

// Add the following lines at the end of the file

```
export JAVA_HOME=/usr/lib/jvm/java-8-oracle
export HADOOP_HOME=/usr/local/hadoop/hadoop-2.8.4
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-D.java.library.path=$HADOOP_HOME/lib"
export PATH=$PATH:/usr/local/hadoop/hadoop-2.8.4/bin
```

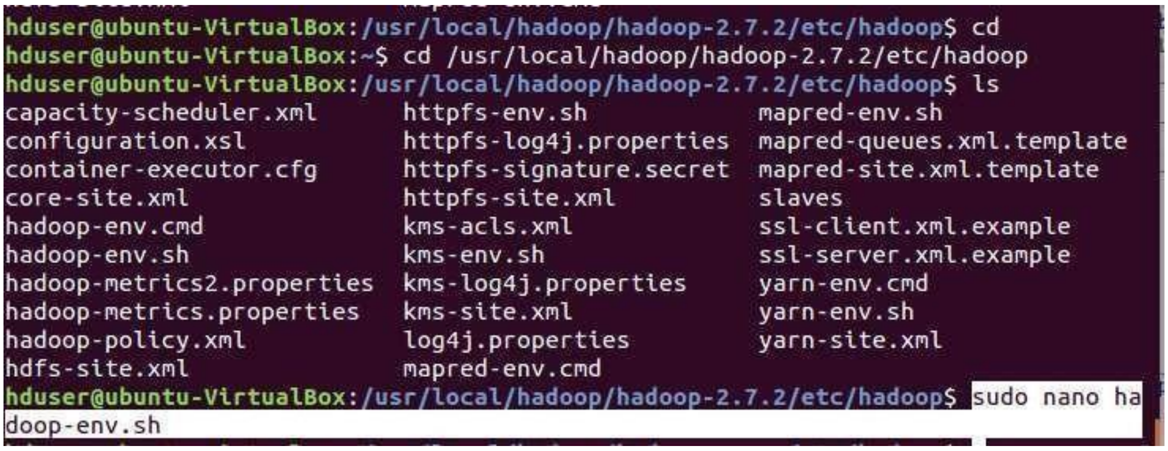


```
hduser@ubuntu-VirtualBox: ~
  GNU nano 2.4.2            File: /home/hduser/.bashrc

if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi
#HADOOP VARIABLES START
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
export HADOOP_INSTALL=/usr/local/hadoop/hadoop-2.7.2
export PATH=$PATH:$HADOOP_INSTALL/bin
export PATH=$PATH:$HADOOP_INSTALL/sbin
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_HOME=$HADOOP_INSTALL
export HADOOP_HDFS_HOME=$HADOOP_INSTALL
export YARN_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_INSTALL/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib"
#HADOOP VARIABLES END

^G Get Help    ^O Write Out  ^W Where Is    ^K Cut Text   ^J Justify     ^C Cur Pos
^X Exit        ^R Read File  ^\ Replace     ^U Uncut Text^T To Spell     ^_ Go To Line
```

// Configure Hadoop Files

$ cd /usr/local/hadoop/hadoop-2.8.4/etc/hadoop/

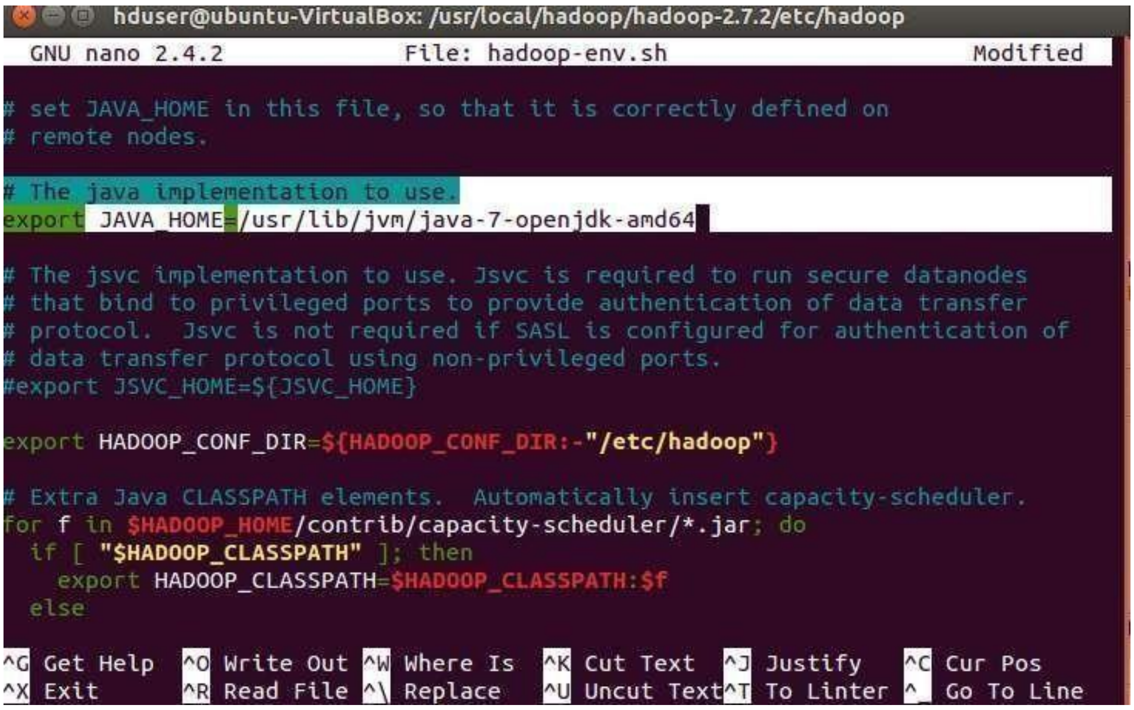
$ sudo nano hadoop-env.sh



```
hduser@ubuntu-VirtualBox:/usr/local/hadoop/hadoop-2.7.2/etc/hadoop$ cd
hduser@ubuntu-VirtualBox:~$ cd /usr/local/hadoop/hadoop-2.7.2/etc/hadoop
hduser@ubuntu-VirtualBox:/usr/local/hadoop/hadoop-2.7.2/etc/hadoop$ ls
capacity-scheduler.xml      httpfs-env.sh            mapred-env.sh
configuration.xsl           httpfs-log4j.properties  mapred-queues.xml.template
container-executor.cfg      httpfs-signature.secret  mapred-site.xml.template
core-site.xml               httpfs-site.xml          slaves
hadoop-env.cmd              kms-acls.xml             ssl-client.xml.example
hadoop-env.sh               kms-env.sh               ssl-server.xml.example
hadoop-metrics2.properties  kms-log4j.properties     yarn-env.cmd
hadoop-metrics.properties   kms-site.xml             yarn-env.sh
hadoop-policy.xml           log4j.properties         yarn-site.xml
hdfs-site.xml               mapred-env.cmd
hduser@ubuntu-VirtualBox:/usr/local/hadoop/hadoop-2.7.2/etc/hadoop$ sudo nano ha
doop-env.sh
```

// Add following line in hadoop-env.sh – Set JAVA variable in Hadoop

```
# The java implementation to use.
export JAVA_HOME=/usr/lib/jvm/java-8-oracle
```



```
hduser@ubuntu-VirtualBox: /usr/local/hadoop/hadoop-2.7.2/etc/hadoop
  GNU nano 2.4.2             File: hadoop-env.sh                    Modified

# set JAVA_HOME in this file, so that it is correctly defined on
# remote nodes.

# The java implementation to use.
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64

# The jsvc implementation to use. Jsvc is required to run secure datanodes
# that bind to privileged ports to provide authentication of data transfer
# protocol.  Jsvc is not required if SASL is configured for authentication of
# data transfer protocol using non-privileged ports.
#export JSVC_HOME=${JSVC_HOME}

export HADOOP_CONF_DIR=${HADOOP_CONF_DIR:-"/etc/hadoop"}

# Extra Java CLASSPATH elements.  Automatically insert capacity-scheduler.
for f in $HADOOP_HOME/contrib/capacity-scheduler/*.jar; do
  if [ "$HADOOP_CLASSPATH" ]; then
    export HADOOP_CLASSPATH=$HADOOP_CLASSPATH:$f
  else

^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Uncut Text^T To Linter ^_ Go To Line
```

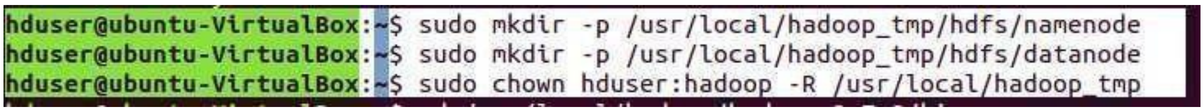
// Create datanode and namenode

$ sudo mkdir –p /usr/local/hadoop_tmp/hdfs/namenode

$ sudo mkdir –p /usr/local/hadoop_tmp/hdfs/datanode

// Changing ownership to hadoop_tmp

$ sudo chown –R hduser:hadoop /usr/local/hadoop_tmp



```
hduser@ubuntu-VirtualBox:~$ sudo mkdir -p /usr/local/hadoop_tmp/hdfs/namenode
hduser@ubuntu-VirtualBox:~$ sudo mkdir -p /usr/local/hadoop_tmp/hdfs/datanode
hduser@ubuntu-VirtualBox:~$ sudo chown hduser:hadoop -R /usr/local/hadoop_tmp
```

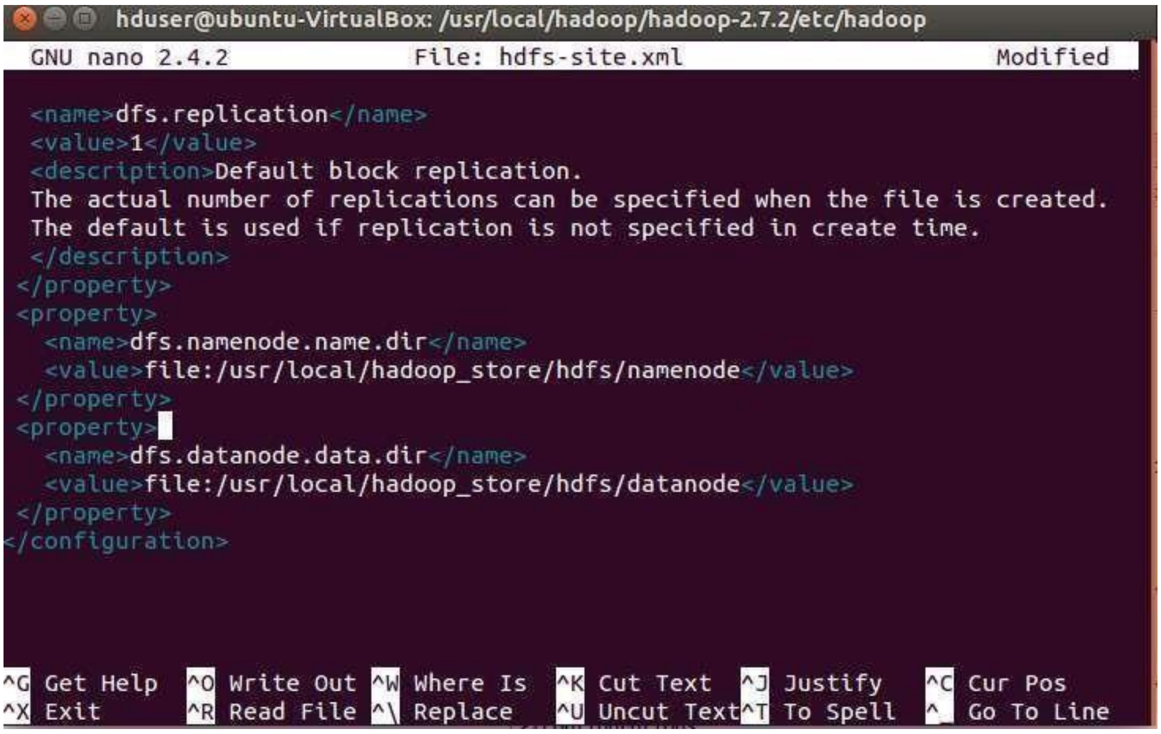
// Edit hdfs-site.xml

$ sudo nano hdfs-site.xml

// Add the following lines between <configuration> …… </configuration>

```
<configuration>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
<property>
<name>dfs.namenode.name.dir</name>
<value>file:/usr/local/hadoop_tmp/hdfs/namenode</value>
</property>
<property>
<name>dfs.datanode.data.dir</name>
```

```
<value>file:/usr/local/hadoop_tmp/hdfs/datanode</value>
</property>
</configuration>
```



```
hduser@ubuntu-VirtualBox: /usr/local/hadoop/hadoop-2.7.2/etc/hadoop
  GNU nano 2.4.2              File: hdfs-site.xml                         Modified

  <name>dfs.replication</name>
  <value>1</value>
  <description>Default block replication.
  The actual number of replications can be specified when the file is created.
  The default is used if replication is not specified in create time.
  </description>
 </property>
 <property>
   <name>dfs.namenode.name.dir</name>
   <value>file:/usr/local/hadoop_store/hdfs/namenode</value>
 </property>
 <property>
   <name>dfs.datanode.data.dir</name>
   <value>file:/usr/local/hadoop_store/hdfs/datanode</value>
 </property>
</configuration>

^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Uncut Text^T To Spell  ^_ Go To Line
```

// Edit core-site.xml

$ sudo nano core-site.xml

// Add the following lines between <configuration> …… </configuration>

```
<configuration>
<property>
<name>fs.default.name</name>
<value>hdfs://localhost:9000</value>
</property>
</configuration>
```

// Edit yarn-site.xml

$ sudo nano yarn-site.xml

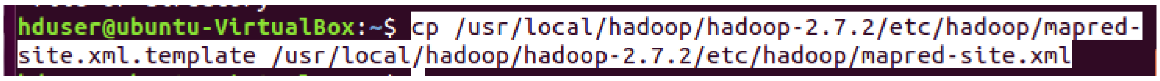
// Add the following lines between <configuration> …… </configuration>

```
<configuration>
<property>
```

```
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
<property>
<name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
<value>org.apache.hadoop.mapred.Shuffle-Handler</value>
</property>
</configuration>
```

// Edit mapred-site.xml

$ cp /usr/local/hadoop/hadoop-2.8.4/etc/hadoop/mapred-site.xml.template /usr/local/hadoop/hadoop-2.8.4/etc/hadoop/mapred-site.xml



```
hduser@ubuntu-VirtualBox:~$ cp /usr/local/hadoop/hadoop-2.7.2/etc/hadoop/mapred-
site.xml.template /usr/local/hadoop/hadoop-2.7.2/etc/hadoop/mapred-site.xml
```

$ sudo nano mapred-site.xml

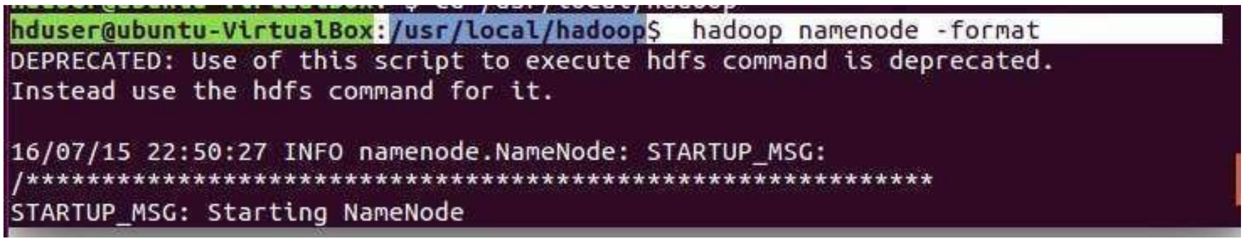
// Add the following lines between <configuration> …… </configuration>

```
<configuration>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
</configuration>
```

Step 8 – Format Hadoop File System

$ cd /usr/local/hadoop/hadoop-2.8.4/bin

$ hadoop namenode -format



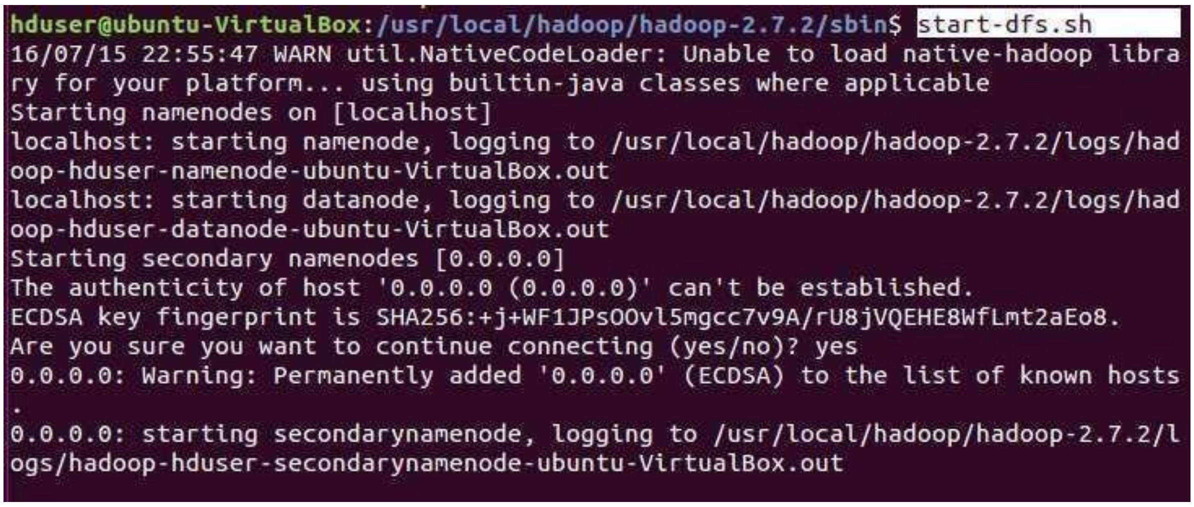
```
hduser@ubuntu-VirtualBox:/usr/local/hadoop$  hadoop namenode -format
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

16/07/15 22:50:27 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
```

Step 9 - Start Hadoop

$ cd /usr/local/hadoop/hadoop-2.8.4/sbin

// Starting dfs services

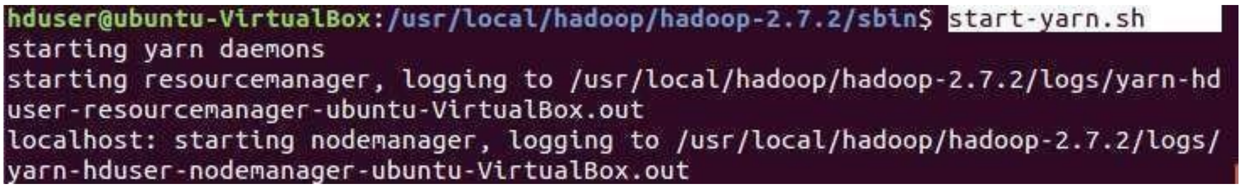
$ start-dfs.sh



```
hduser@ubuntu-VirtualBox:/usr/local/hadoop/hadoop-2.7.2/sbin$ start-dfs.sh
16/07/15 22:55:47 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
Starting namenodes on [localhost]
localhost: starting namenode, logging to /usr/local/hadoop/hadoop-2.7.2/logs/had
oop-hduser-namenode-ubuntu-VirtualBox.out
localhost: starting datanode, logging to /usr/local/hadoop/hadoop-2.7.2/logs/had
oop-hduser-datanode-ubuntu-VirtualBox.out
Starting secondary namenodes [0.0.0.0]
The authenticity of host '0.0.0.0 (0.0.0.0)' can't be established.
ECDSA key fingerprint is SHA256:+j+WF1JPsOOvl5mgcc7v9A/rU8jVQEHE8WfLmt2aEo8.
Are you sure you want to continue connecting (yes/no)? yes
0.0.0.0: Warning: Permanently added '0.0.0.0' (ECDSA) to the list of known hosts
.
0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop/hadoop-2.7.2/l
ogs/hadoop-hduser-secondarynamenode-ubuntu-VirtualBox.out
```

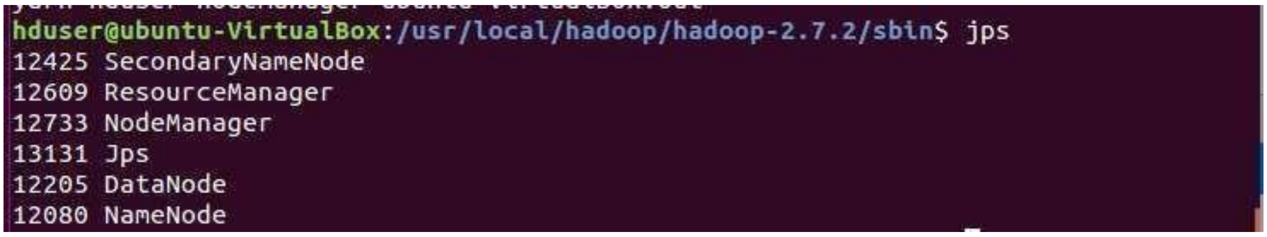
// Starting mapreduce services

$ start-yarn.sh



```
hduser@ubuntu-VirtualBox:/usr/local/hadoop/hadoop-2.7.2/sbin$ start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop/hadoop-2.7.2/logs/yarn-hd
user-resourcemanager-ubuntu-VirtualBox.out
localhost: starting nodemanager, logging to /usr/local/hadoop/hadoop-2.7.2/logs/
yarn-hduser-nodemanager-ubuntu-VirtualBox.out
```

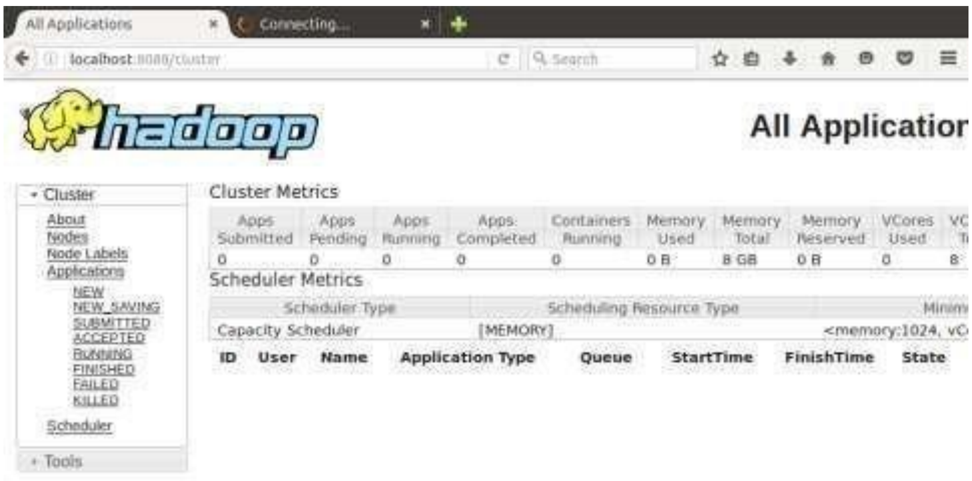
$ jps



```
hduser@ubuntu-VirtualBox:/usr/local/hadoop/hadoop-2.7.2/sbin$ jps
12425 SecondaryNameNode
12609 ResourceManager
12733 NodeManager
13131 Jps
12205 DataNode
12080 NameNode
```

Step 10 - Check Hadoop through web UI

Go to browser type **<u>http://localhost:8088</u> – All Applications Hadoop Cluster**


All Applications

Cluster Metrics

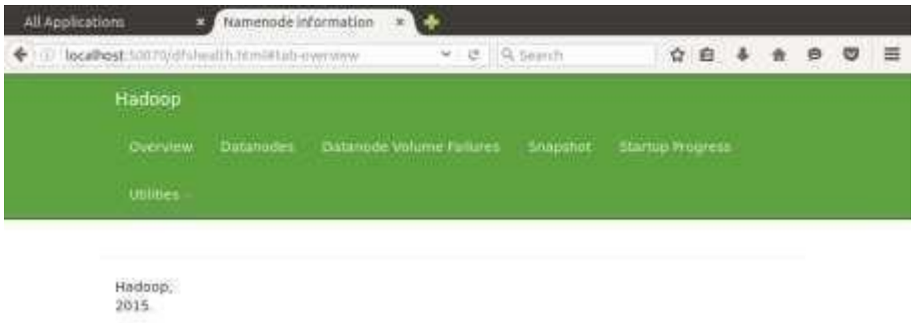
Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VC T
0	0	0	0	0	0 B	8 GB	0 B	0	8

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minim
Capacity Scheduler	[MEMORY]	<memory:1024, vC

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State

Go to browser type **http://localhost:50070 – Hadoop Namenode**


Hadoop, 2015

Step 11 - Stop Hadoop

$ stop-dfs.sh

$ stop-yarn.sh

RESULT

Thus the procedure to install single-node Hadoop is executed successfully.

EX. NO:

DATE:

DEMONSTRATION OF MAP AND REDUCE TASKS

AIM

To write a wordcount program to demonstrate the use of Map and Reduce tasks.

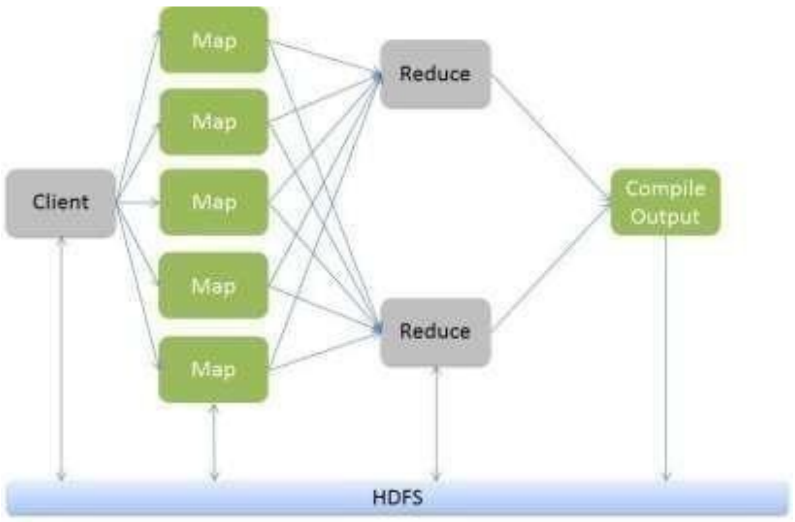
THEORY

Hadoop MapReduce is a programming paradigm at the heart of Apache Hadoop for providing massive scalability across hundreds or thousands of Hadoop clusters on commodity hardware. The MapReduce model processes large unstructured data sets with a distributed algorithm on a Hadoop cluster.

The term MapReduce represents two separate and distinct tasks Hadoop programs perform-Map Job and Reduce Job. Map job scales takes data sets as input and processes them to produce key value pairs. Reduce job takes the output of the Map job i.e. the key value pairs and aggregates them to produce desired results. The input and output of the map and reduce jobs are stored in HDFS.

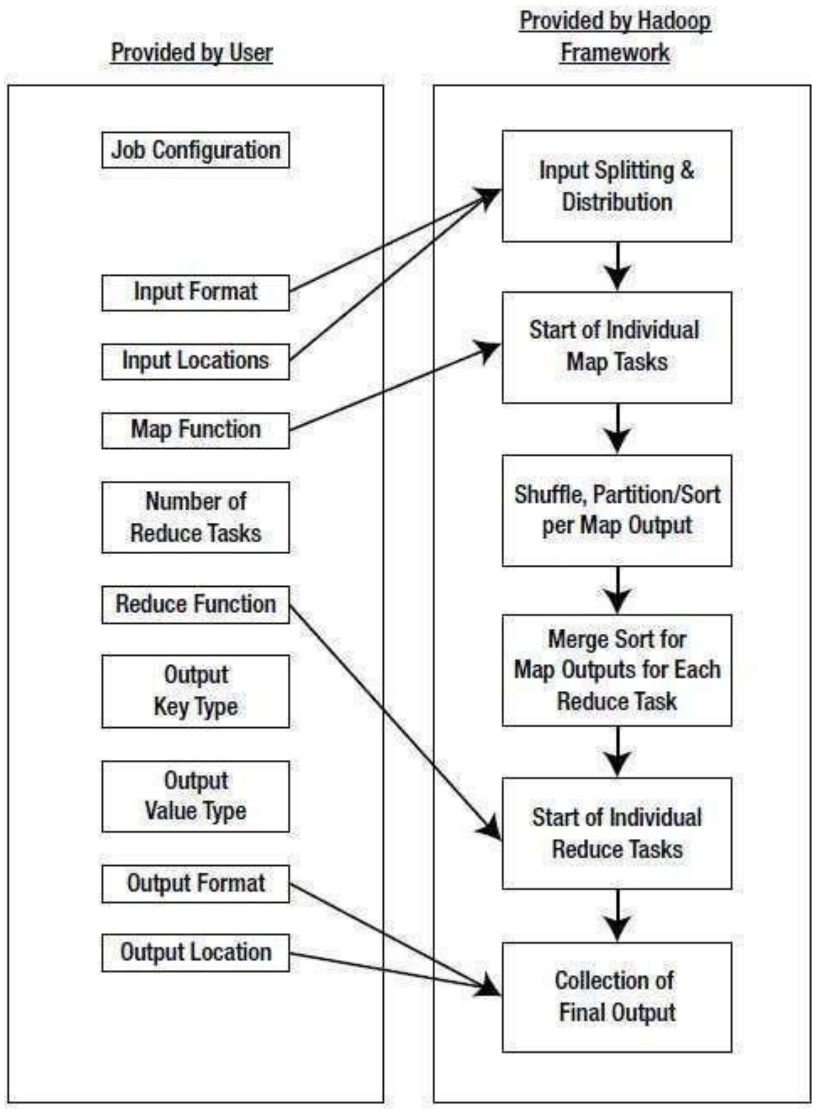
MAPREDUCE

Hadoop MapReduce (Hadoop Map/Reduce) is a software framework for distributed processing of large data sets on computing clusters. It is a sub-project of the Apache Hadoop project. Apache Hadoop is an open-source framework that allows to store and process big data in a distributed environment across clusters of computers using simple programming models. MapReduce is the core component for data processing in Hadoop framework. In layman‟s term Mapreduce helps to split the input data set into a number of parts and run a program on all data parts parallel at once. The term MapReduce refers to two separate and distinct tasks. The first is the map operation, takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). The reduce operation combines those data tuples based on the key and accordingly modifies the value of the key.



MAPREDUCE ARCHITECTURE

The figure shown below illustrates the various parameters and modules that can be configured during a MapReduce operation:



Provided by User
Provided by Hadoop Framework
Job Configuration
Input Format
Input Locations
Map Function
Number of Reduce Tasks
Reduce Function
Output Key Type
Output Value Type
Output Format
Output Location
Input Splitting & Distribution
Start of Individual Map Tasks
Shuffle, Partition/Sort per Map Output
Merge Sort for Map Outputs for Each Reduce Task
Start of Individual Reduce Tasks
Collection of Final Output

MapReduce Architecture

JobConf is the framework used to provide various parameters of a MapReduce job to the Hadoop for execution. The Hadoop platforms executes the programs based on configuration set using JobConf. The parameters being Map Function, Reduce Function, combiner , Partitioning function, Input and Output format. Partitioner controls the shuffling of the tuples when being sent from Mapper node to Reducer nodes. The total number of partitions done in the tuples is equal to the number of reduce nodes. In simple terms based on the function output the tuples are transmitted through different reduce nodes.

Input Format describes the format of the input data for a MapReduce job. Input location specifies the location of the datafile. Map Function/ Mapper convert the data into key value pair. For example let's consider daily temperature data of 100 cities for the past 10 years. In this the map function is written such a way that every temperature being mapped to the corresponding city. Reduce Function reduces the set of tuples which share a key to a single tuple with a change in the value. In this example if we have to find the highest temperature recorded in the city the reducer function is written in such a way that it return the tuple with highest value i.e: highest temperature in the city in that sample data.

The number of Map and Reduce nodes can also be defined. You can set Partitioner function which partitions and transfer the tuples which by default is based on a hash function. In other words we can set the options such that a specific set of key value pairs are transferred to a specific reduce task. For example if your key value consists of the year it was recorded, then we can set the parameters such that all the keys of specific year are transferred to a same reduce task. The Hadoop framework consists of a single master and many slaves. Each master has JobTracker and each slave has TaskTracker. Master distributes the program and data to the slaves. Task tracker, as the name suggests keep track of the task directed to it and relays the information to the JobTracker. The JobTracker monitors all the status reports and re-initiates the failed tasks if any.

Combiner classes are run on map task nodes. It takes the tuples emitted by Map nodes as input. It basically does reduce operation on the tuples emitted by the map node. It is like a pre-reduce task to save a lot of bandwidth. We can also pass global constants to all the nodes using „Counters". They can be used to keep track of all events in map and reduce tasks. For example we can pass a counter to calculate the statistics of an event beyond a certain threshold.

PROCEDURE

Step 1 - Open Terminal

```
$ su hduser
Password:
```

Step 2 - Start dfs and mapreduce services

```
$ cd /usr/local/hadoop/hadoop-2.8.4/sbin
$ start-dfs.sh
$ start-yarn.sh
$ jps
```

Step 3 - Check Hadoop through web UI

// Go to browser type http://localhost:8088 – All Applications Hadoop Cluster

// Go to browser type http://localhost:50070 – Hadoop Namenode

Step 4 – Open New Terminal

```
$ cd Desktop/
$ mkdir inputdata
$ cd inputdata/
$ echo “Hai, Hello, How are you? How is your health?” >> hello.txt

$ cat >> hello.txt
```

Step 5 – Go back to old Terminal

```
$ hadoop fs –mkdir /folder

$ hadoop fs –copyFromLocal /home/hduser/Desktop/inputdata/hello.txt /folder
```

// Check in hello.txt in Namenode using Web UI

Step 6 – Download and open eclipse by creating workspace

Create a new java project.

Step 7 – Add jar to the project

You need to remove dependencies by adding jar files in the hadoop source folder. Now Click on **Project** tab and go to Properties.Under Libraries tab, click Add External JARs and select all the jars in the folder (click on 1st jar, and Press Shift and Click on last jat to select all jars in between and click ok)

/usr/local/hadoop/hadoop-2.8.4/share/hadoop/commonand

/usr/local/hadoop/hadoop-2.8.4/share/hadoop/mapreduce folders.

Step -8 – WordCount Program

Create 3 java files named

- **WordCount.java**
- **WordCountMapper.java**
- **WordCountReducer.java**

WordCount.java

```
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;

import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;
import org.apache.hadoop.io.Text;

public class WordCount extends Configured implements Tool {

	@Override
	public int run(String[] arg0) throws Exception {
		// TODO Auto-generated method stub
		if(arg0.length<2)
```

```
                {
                        System.out.println("check the command line arguments");
                }
                JobConf conf=new JobConf(WordCount.class);
                FileInputFormat.setInputPaths(conf, new Path(arg0[0]));
                        FileOutputFormat.setOutputPath(conf, new Path(arg0[1]));
                        conf.setMapperClass(WordMapper.class);
                        conf.setReducerClass(WordReducer.class);
                        conf.setOutputKeyClass(Text.class);
                        conf.setOutputValueClass(IntWritable.class);
                        conf.setOutputKeyClass(Text.class);
                        conf.setOutputValueClass(IntWritable.class);
                        JobClient.runJob(conf);


                return 0;
        }
        public static void main(String args[]) throws Exception
        {
                int exitcode=ToolRunner.run(new WordCount(), args);
                System.exit(exitcode);

        }
}
```

WordCountMapper.java

```
import java.io.IOException;

import    org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.mapred.MapReduceBase;
import  org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.Mapper;

public class WordCountMapper extends MapReduceBase implements
Mapper<LongWritable,Text,Text,IntWritable>
{
        @Override
        public void map(LongWritable arg0, Text arg1, OutputCollector<Text, IntWritable>
arg2, Reporter arg3)
                        throws IOException {
                // TODO Auto-generated method stub

                String s=arg1.toString();
                for(String word:s.split(" "))
                {
                        arg2.collect(new Text(word),new IntWritable(1));
                }
        }
}
```

WordCountReducer.java

```
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;
import org.apache.hadoop.io.Text;

public class WordCountReducer implements Reducer<Text,IntWritable,Text,IntWritable> {
        @Override
        public void configure(JobConf arg0) {
                // TODO Auto-generated method stub
        }
        @Override
        public void close() throws IOException {
                // TODO Auto-generated method stub
        }
        @Override
        public void reduce(Text arg0, Iterator<IntWritable> arg1, OutputCollector<Text,
IntWritable> arg2, Reporter arg3)
                        throws IOException {
                // TODO Auto-generated method stub
                int count=0;
                while(arg1.hasNext())
                {
                        IntWritable i=arg1.next();
                        count+=i.get();
                }
                arg2.collect(arg0,new IntWritable(count));

        }
}
```

Step 9 - Creatr JAR file

Now Click on the Run tab and click Run-Configurations. Click on New Configuration button on the left-top side and Apply after filling the following properties.

Step 10 - Export JAR file

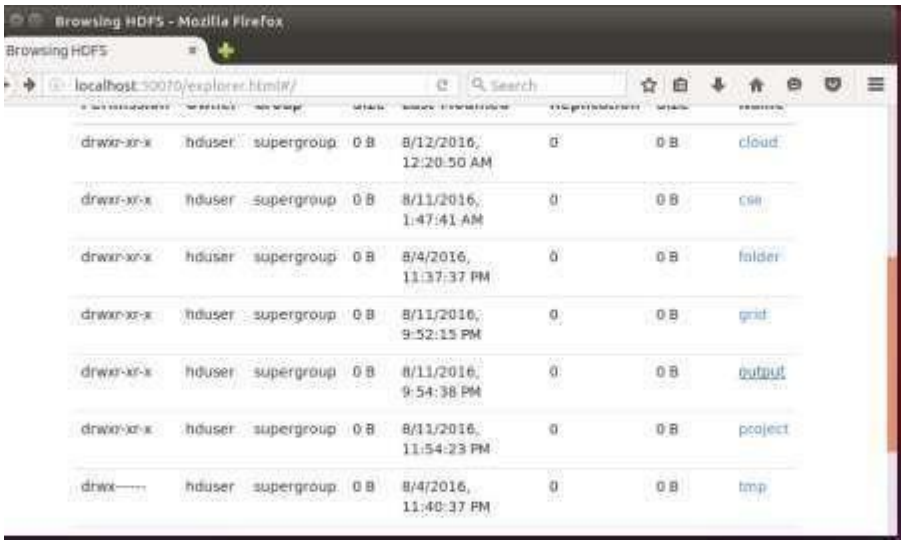
Now click on File tab and select Export. under Java, select Runnable Jar.

In Launch Config – select the config fie you created in **Step 9** (WordCountConfig).

- Select an export destination (lets say desktop.)
- Under Library handling, select Extract Required Libraries into generated JAR and click Finish.
- Right-Click the jar file, go to Properties and under **Permissions**tab, Check Allow executing file as a program. and give Read and Write access to all theusers

Step 11 – Go back to old Terminal for Execution of WordCount Program

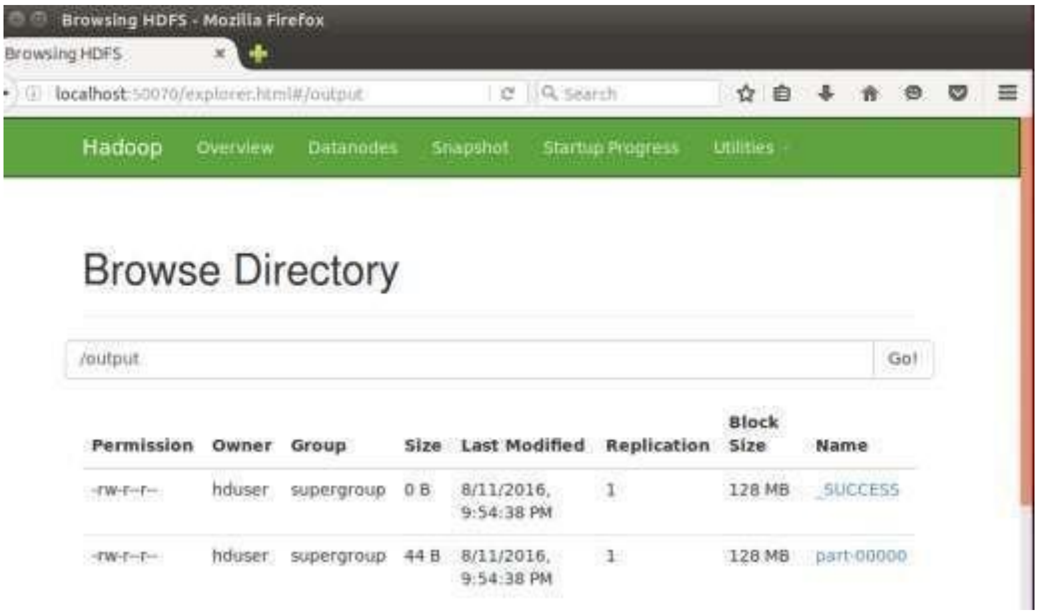
```
$hadoop jar wordcount.jar/usr/local/hadoop/input/usr/local/hadoop/output
```



drwxr-xr-x	hduser	supergroup	0 B	8/12/2016, 12:20:50 AM	0	0 B	cloud
drwxr-xr-x	hduser	supergroup	0 B	8/11/2016, 1:47:41 AM	0	0 B	cse
drwxr-xr-x	hduser	supergroup	0 B	8/4/2016, 11:37:37 PM	0	0 B	folder
drwxr-xr-x	hduser	supergroup	0 B	8/11/2016, 9:52:15 PM	0	0 B	grid
drwxr-xr-x	hduser	supergroup	0 B	8/11/2016, 9:54:38 PM	0	0 B	output
drwxr-xr-x	hduser	supergroup	0 B	8/11/2016, 11:54:23 PM	0	0 B	project
drwx------	hduser	supergroup	0 B	8/4/2016, 11:40:37 PM	0	0 B	tmp

Step 12 – To view results in old Terminal

```
$hdfs dfs -cat /usr/local/hadoop/output/part-r-00000
```


Browsing HDFS - Mozilla Firefox
localhost:50070/explorer.html#/output
Hadoop
Overview
Datanodes
Snapshot
Startup Progress
Utilities
Go!

Browse Directory

/output

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	hduser	supergroup	0 B	8/11/2016, 9:54:38 PM	1	128 MB	_SUCCESS
-rw-r--r--	hduser	supergroup	44 B	8/11/2016, 9:54:38 PM	1	128 MB	part-00000

Step 13 - To Remove folders created using hdfs

```
$ hdfs dfs -rm -R /usr/local/hadoop/output
```

RESULT

Thus the program to write a wordcount program to that demonstrates the use of Map and Reduce tasks is done and output is verified successfully.

EX. NO:

DATE:

IMPLEMENTATION OF MAX TEMPERATURE MAPREDUCE PROGRAM

AIM

To implement the Max temperature MapReduce program to identify the year wise maximum temperature from the sensor data.

Description

Sensors senses weather data in big text format containing station ID, year, date, time, temperature, quality etc. from each sensor and store it in single line. Suppose thousands of data sensors are their, then we have thousands of records with no particular order. We require only year and maximum tempertaure of particular quality in that year.

For example:
Input string from sensor:
0029029070999991902010720004+64333+023450FM-12+
000599999V0202501N027819999999N0000001N9-00331+
99999098351ADDGF102991999999999999999999

Here: 1902 is year
0033 is temperature
1 is measurement quality (Range between 0 or 1 or 4 or 5 or 9)

Here each mapper takes input key as "byte offset of line" and value as "one weather sensor read i.e one line". and parse each line and produce intermediate key is "year" and intermediate value as **"temperature of certain measurement qualities"** for that year.

The combiner will form set values of temperature. Year and set of values of temperatures is given as input **<key, value>** to reducer and Reducer will produce year and maximum temperature for that year from set of temperature values.

PROGRAM

```
*/
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;


//Mapper class
```

```
class MaxTemperatureMapper
extends Mapper<LongWritable, Text, Text, IntWritable> {

private static final int MISSING = 9999;

@Override
public void map(LongWritable key, Text value, Context context)
throws IOException, InterruptedException {

String line = value.toString();
String year = line.substring(15, 19);
int airTemperature;
if (line.charAt(87) == '+') { // parseInt doesn't like leading plus signs
airTemperature = Integer.parseInt(line.substring(88, 92));
} else {
airTemperature = Integer.parseInt(line.substring(87, 92));
}
String quality = line.substring(92, 93);
if (airTemperature != MISSING && quality.matches("[01459]")) {
context.write(new Text(year), new IntWritable(airTemperature));
}
}
}


//Reducer class
class MaxTemperatureReducer
extends Reducer<Text, IntWritable, Text, IntWritable> {

@Override
public void reduce(Text key, Iterable<IntWritable> values,
Context context)
throws IOException, InterruptedException {

int maxValue = Integer.MIN_VALUE;
for (IntWritable value : values) {
maxValue = Math.max(maxValue, value.get());
}
context.write(key, new IntWritable(maxValue));
}
}


//Driver Class

public class MaxTemperature {

public static void main(String[] args) throws Exception {
if (args.length != 2) {
System.err.println("Usage: MaxTemperature <input path=""> <output path>");
System.exit(-1);
}

Job job = Job.getInstance(new Configuration());
job.setJarByClass(MaxTemperature.class);
job.setJobName("Max temperature");

FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
job.setMapperClass(MaxTemperatureMapper.class);
job.setReducerClass(MaxTemperatureReducer.class);

job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);

job.submit();
}
}
```

OUTPUT

Input for String :
0029029070999991902010720004+64333+023450FM-12+
000599999V0202501N027819999999N0000001N9-00331+
99999098351ADDGF102991999999999999999999

Output

Output Text contain year and maximum temperature in that year as ***1902 33***

RESULT

Thus the implementation of the Max temperature MapReduce program to identify the year wise maximum temperature from the sensor data has been done successfully.