EX.No: 1	INSTALL VIRTUALBOX AND KVM WITH DIFFERENT FLAVOURS OFLINUX OR WINDOWS OS ON TOP OF OS

Aim:

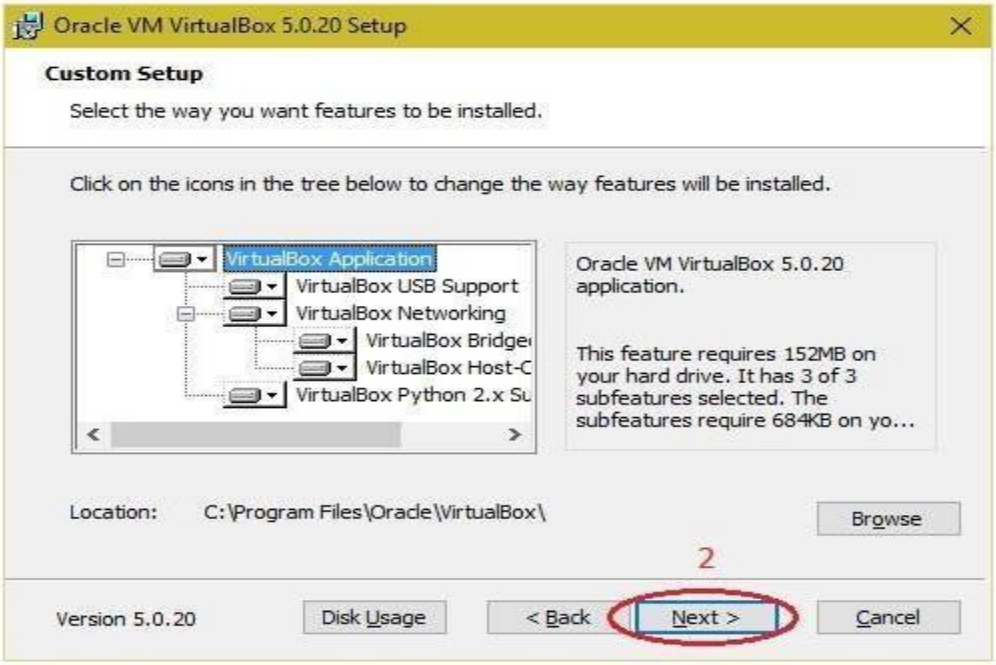
To Install Virtualbox / VMware Workstation with different flavours oflinux or windowsOS on top of windows7 or 8.

PROCEDURE:

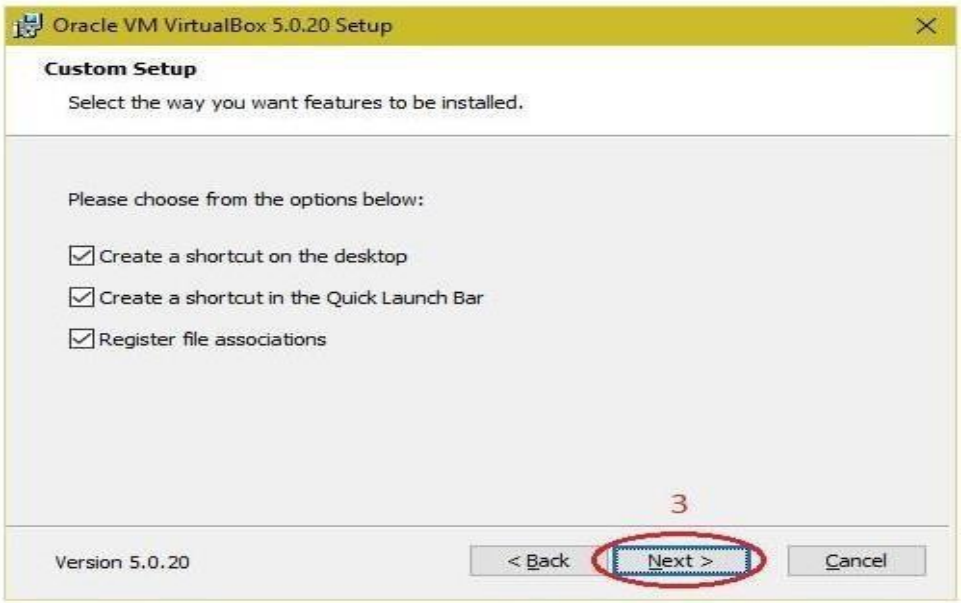
Steps to install Virtual Box:

1. Download the Virtual box exe and click the exe file…and select next button..

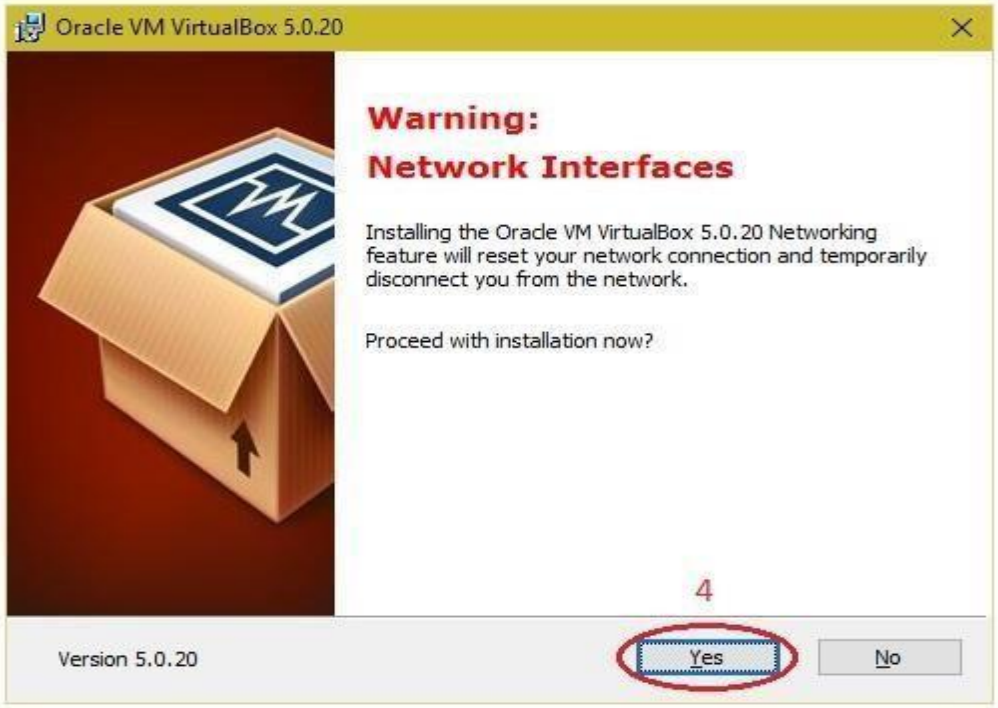
2. Click the next button..



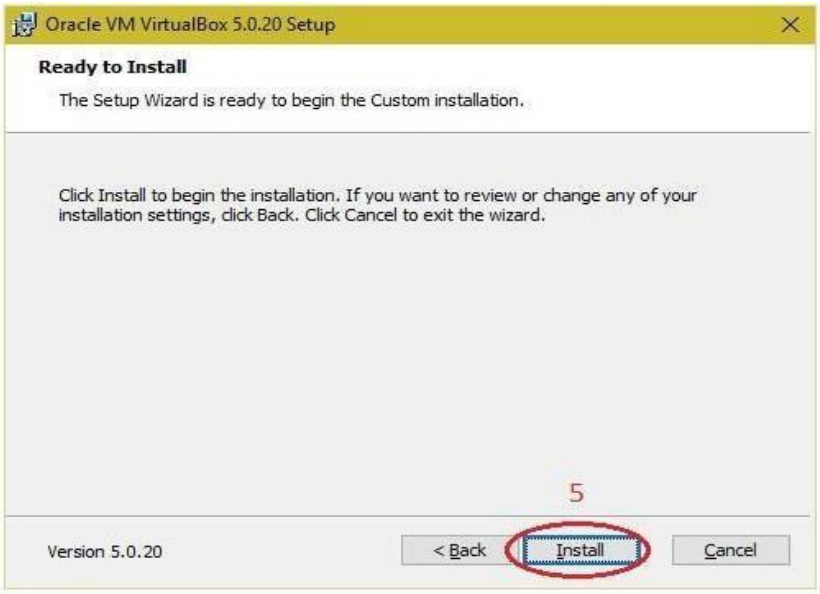
3. Click the next button



4. Click the YES button..



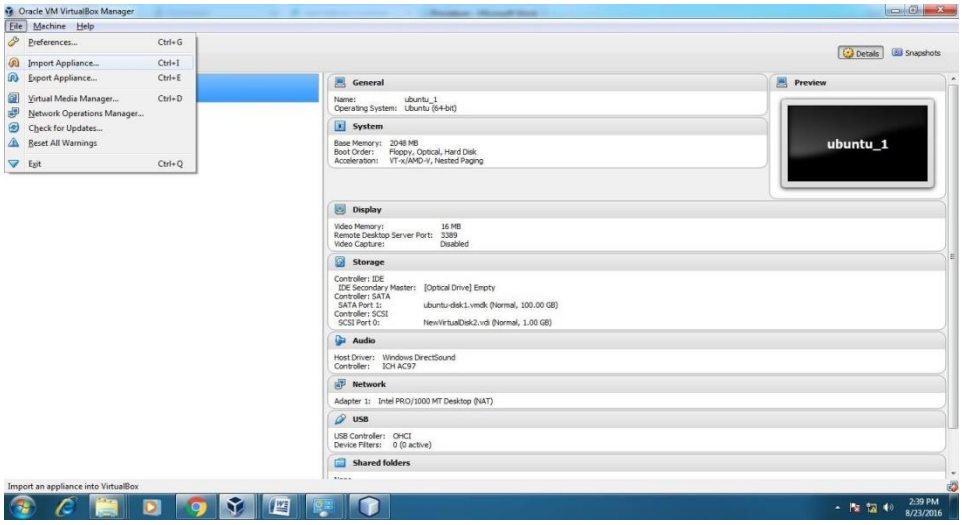
5. Click the install button…

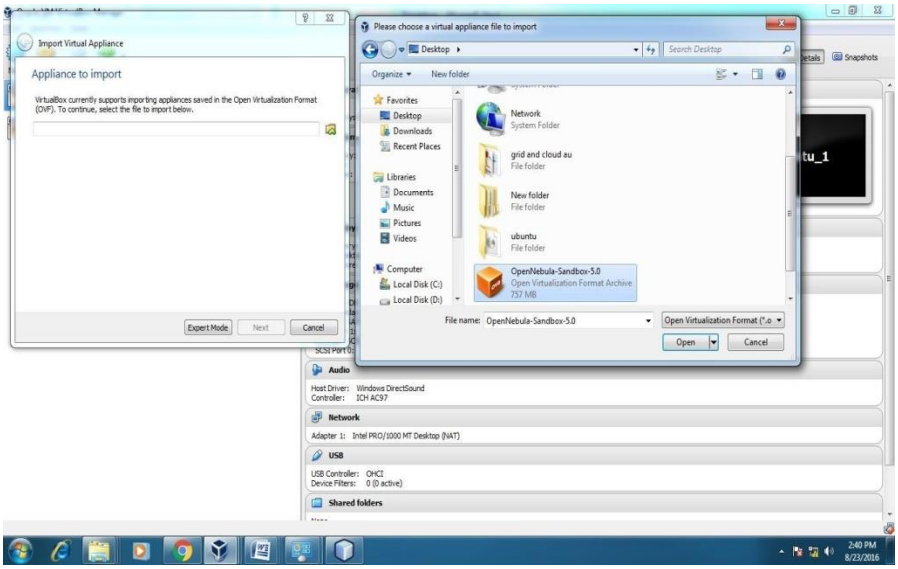


6. Then installation was completed..the show virtual box icon on desktop screen….

Steps to import Open nebula sandbox:

1. Open Virtual box
2. File àimport Appliance
3. Browse OpenNebula-Sandbox-5.0.ova file
4. Then go to setting, select Usb and choose USB 1.1
5. Then Start the Open Nebula
6. Login using username: root, password:opennebula

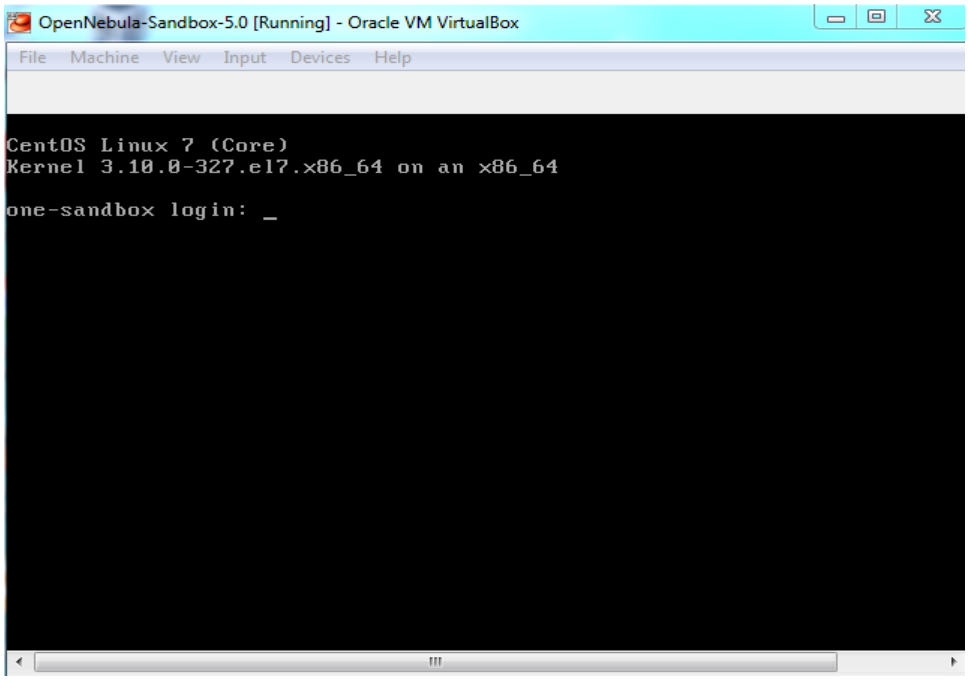


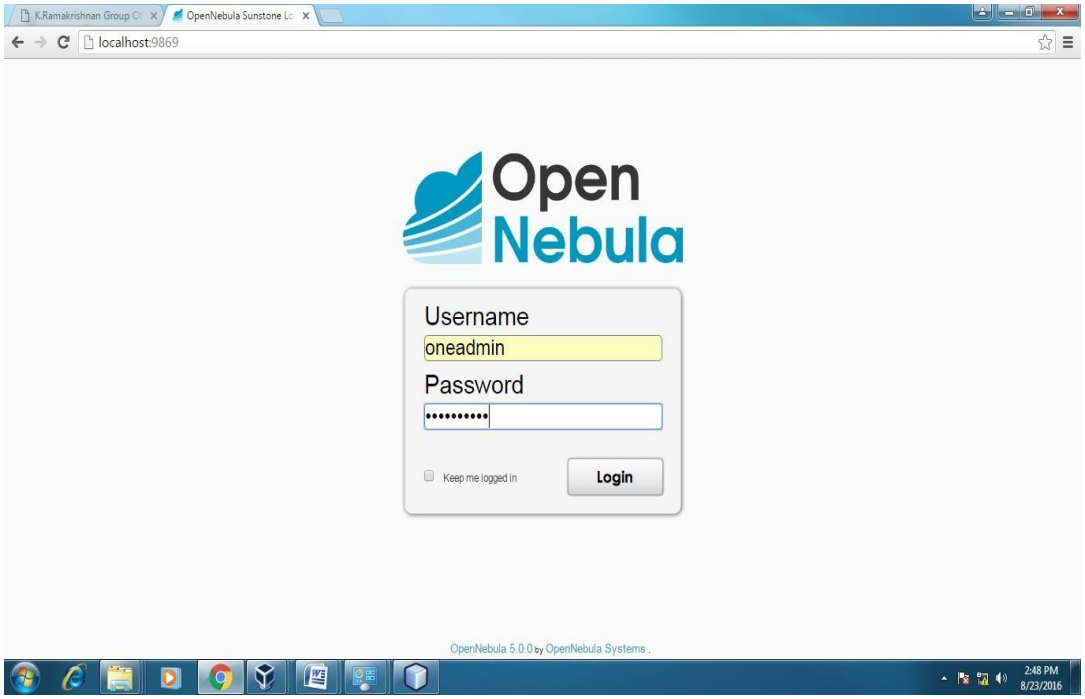


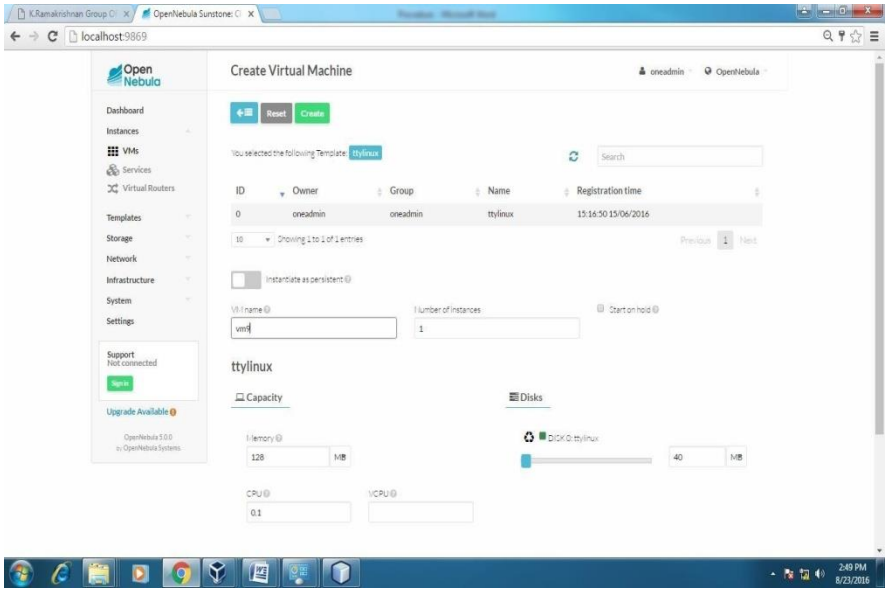
Steps to create Virtual Machine through opennebula

1. Open Browser, type localhost:9869
2. Login using username: oneadmin, password: opennebula
3. Click on instances, select VMs then follow the steps to create Virtaul machine

a. Expand the + symbol
b. Select user oneadmin
c. Then enter the VM name,no.of instance, cpu.
d. Then click on create button.
e. Repeat the steps the C,D for creating more than one VMs.


OpenNebula-Sandbox-5.0 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
CentOS Linux 7 (Core)
Kernel 3.10.0-327.el7.x86_64 on an x86_64
one-sandbox login: _


localhost:9869
Open Nebula
Username
oneadmin
Password
Keep me logged in
Login
OpenNebula 5.0.0 by OpenNebula Systems .
2:48 PM
8/23/2016



RESULT:

Thus the procedure to run the virtual machine of different configuration.

EX.No: 2	INSTALL A C COMPILER IN THE VIRTUAL MACHINE CREATEDUSING VIRTUAL BOX AND EXECUTE SIMPLE PROGRAMS

AIM:

To Install a C compiler in the virtual machine and execute a sample program.

Hardware Requirements:

1. 1 GHz processor (for example Intel Celeron) or better.
2. 1.5 GB RAM (system memory).
3. 10 GB of free hard drive space for installation.
4. Internet access is helpful (for installing updates during the installation process).

Software Requirements :

5. Ubuntu OS
6. Terminal, Geditor, GNU compiler

PROCEDURE :

1. Installing GCC on Ubuntu

First start by updating the packages list.

$ sudo apt update

Now install the build – essential package by using following command

$ sudo apt install build-essential

2. Check the GCC version for C Compiler

$ gcc -version

3. Write C program using Gedit editor

4. Write a simple Hello world program using C.

```
#include<stdio.h>
main()
{
printf(“HelloWorld\n”);return 0;
}
```

5. Save this file as “hello.c”, compile it on terminal using commamd,

```
$gcc -o hello hello.c
$./hello
```

RESULT:

Thus the C compiler was installed in virtual machine and the program was executed successfully.

EX.No: 3	INSTALL GOOGLEAPP ENGINE, CREATE HELLOWORLD APP, ANDOTHER SIMPLE WEB APPLICATIONS USING PYTHON

AIM:

To install googleapp engine, create helloworld app, and other simple web applications using python.

Hardware Requirements:

1. 1 GHz processor (for example Intel Celeron) or better.
2. 1.5 GB RAM (system memory).
3. 10 GB of free hard drive space for installation.
4. Internet access is helpful (for installing updates during the installation process).

Software Requirements :

5. Ubuntu OS
6. Terminal, GAE, Python

PROCEDURE:

A) First install the cloud SDK and then set up a cloud project for

AppEngine:Update Python Package

Step 1:- Install ppa

This PPA contains more recent Python versions packaged for Ubuntu. Install ppa by running thefollowing command.sudo add-apt-repository ppa:deadsnakes/ppa

Step 2:- Update packeges

Now, update your packages by running the following command.

$ sudo apt-get update

Step 3:- Upgrade python 2.x to python 3.x

Before install 3.7, we should have to install python 3.6 by running the following command.

$ sudo apt-get install python3.6

$ sudo apt-get install python3.7 **https://vitux.com/install-google-cloud-sdk-on-ubuntu/**
Downloadand install Cloud SDK

Pre-Requisites

Following are the pre-requisites for proceeding with the installation of Google Cloud SDK on Ubuntu 20.04:

You should have Python installed on your Linux system (Python is installed by Default on

Ubuntu)You should be able to download packages via the Internet.

Step # 1: Update the Package Cache:

sudo apt update

Step # 2: Download the Google Cloud SDK:

wget https://dl.google.com/dl/cloudsdk/channels/rapid/downloads/google-cloud-sdk-307.0.0-linux-x86_64.tar.gz

Step # 3: Extract the Google Cloud SDK File:

go to File explorer and right click on the file and click extract here cd google-cloud-sdk

Step # 4: Installation of Google Cloud SDK:

./install.sh

Do You want to Continue (Y/N) N Do You want to Continue (Y/N) Y Enter

Create a new project:

gcloud projects create example1-foo-bar-1 --name="Happy project" –

labels=type=happycommand – gcloud auth login

Verify the project was created:

gcloud projects describe example1-foo-bar-1

Initialize your App Engine app with your project and choose its region gcloud app create --project= example1-foo-bar-1

Install the following prerequisites:

Download and install Git. Debian/Ubuntu

For the latest stable version for your release of Debian/Ubuntu # sudo apt-get install git

For Ubuntu, this PPA provides the latest stable upstream Git version # sudo add-apt-repositoryppa:git-core/ppa # apt update; apt installgit

Run the following command to install the gcloud component that includes the App Engine extension for Python 3:

cd python-docs-samples/appengine/standard_python3/hello_world

gcloud components install app-engine-python Download the Hello World app

We've created a simple Hello World app for Python 3 so you can quickly get a feel for deploying anapp to the Google Cloud.

Clone the Hello World sample app repository to your local machine.

git clone https://github.com/GoogleCloudPlatform/python-docs-samples

Alternatively, you can download the sample as a zip file and extract it. Change to the directory
thatcontains the sample code.

\

RESULT:

Thus the google app engine was installed successfully and created a hello world app and executed successfully.

EX.No: 4	USE GAE LAUNCHER TO LAUNCH WEB APPLICATION

AIM:

To use a GAE launcher to launch web application.

Hardware Requirements:

1. 1 GHz processor (for example Intel Celeron) or better.
2. 1.5 GB RAM (system memory).
3. 10 GB of free hard drive space for installation.
4. Internet access is helpful (for installing updates during the installation process).

Software Requirements :

5. Ubuntu OS
6. Terminal, GAE

PROCEDURE:

Creating Your Project

To deploy your app on App Engine, you must create a Google Cloud project, which is a top level container that holds your App Engine application resources as well as other Google Cloud resources.In this task, you create a Cloud project and an App Engine application to store settings, computing resources, credentials, andmetadata for your app.If you already have a Cloud project with App Engine enabled, continue to Writing Your Web Service.

Creating a Cloud project

Use the Cloud Console to create the required resources for your app:

1. Create a new Cloud project in the

Cloud Console:Go to the console

2. Enter a name for your Cloud project and click **Create**.

Remember the project ID, shown under the **Project name** field. The project ID is important because it isused to identify your project to the Google Cloud tools.

3. Enable App Engine for your

Cloud project:Enable App Engine

4. Select a region where you want your app's computing resources located.

After you create your App Engine app, you cannot change the region. To reduce latency, choose the regionclosest to your app's intended users. For more information on the available regions, see App Engine Locations.

5. Enable billing for your project.

Key points

- You can use dependencies by listing them in your package.json file. See Specifying Dependencies for moreinformation.
- App Engine starts your application by running npm start.
- Your server must listen to the port specified by the process.env.PORT environment variable.
- You need an app.yaml file to deploy your service to App Engine.

Creating a server to listen for HTTP requests

The core of your web service is the HTTP server. The sample code in this guide uses the Express.js framework to handle HTTP requests, but you are free to use a web framework of your choice.

1. Create a new folder called my-nodejs-service for your Node.js service.
2. Navigate to the folder in your terminal, and create a package.json file by running npm init.
3. Add Express as a dependency by running:

```
npm install express
```

Confirm that Express appears in your package.json file's dependencies field. Here's an example:

```
{
 ...
 "dependenc
  ies":{
  "express":
  "^4.16.3"
 }
```

4. Add a start script to your package.json file:

```
"scripts":{
 "start":"node server.js"
}
```

```
const express =
require('express');
```

5. Create a file called server.js in the same folder and add the following code:

appengine/building-an-app/build/server.js

```
app.get('/',(req, res)=>{ res.send('Hello
  from App Engine!');
});
```

```
/ Listen to the App Engine-specified port, or
8080 otherwiseconst PORT =
process.env.PORT ||8080;
app.listen(PORT,()=>{
```

This is a very basic web server - it responds to all `GET` requests to the root path (`'/'`) with the text "Hello fromApp Engine!" Note the last four lines, where the server is set to listen to the port specified by `process.env.PORT`, an environment variable set by the App Engine runtime. If your server isn't set tolisten to this port, it will not receive requests.

Notice that if `process.env.PORT` is not set, port `8080` is used as a default. This is necessary for testing yourapp locally, because `process.env.PORT` doesn't get set during local runs - it is only set when your app is running on App Engine. You can use whichever port you prefer for testing, but this guide uses `8080`.

Running the server locally

To run the server locally:

1. Run `npm start` in your terminal. This will run your `server.js` file.
2. Point your web browser to `http://localhost:8080`.

You should see a page with the text "Hello from App Engine!"

Creating the app.yaml **file**

An `app.yaml` file specifies settings for your App Engine service's runtime environment. Your service will notdeploy without this file.

1. In your `my-nodejs-service` folder, create a file called `app.yaml`.
2. Add the following contents:

```
package.json
server.js
```

Before you begin

1. Create a Cloud project with an App Engine app.
2. Write a Node.js web server ready to deploy on App Engine.
3. Install Cloud SDK, which provides the `gcloud` command-line tool. Ensure `gcloud` is configured to use theGoogle Cloud project you want to deploy to.

Key points

- Use `gcloud app deploy` and `gcloud app browse` to deploy and view your service.

Deploying your service

In your `my-nodejs-service` folder, where the `app.yaml` file is located, run the following command in your terminal:

```
gcloud app deploy
```

Your source files are then uploaded to Google Cloud Storage. Cloud Build builds your app and deploys it toApp Engine.

For information about other ways to build and deploy your app, see Testing and deploying your app.
Important: For the best performance, your **start** script should be lightweight, because it runs whenever anew instance of your application is created.

Viewing your service

To quickly launch your browser and access your web service at https://***PROJECT_ID***.***REGION_ID***.r.appspot.com, use the following command:

```
gcloud app browse
```

Services and versions

You've just created and deployed a service on App Engine. You can specify the name of your service inthe app.yaml file. If the name is omitted, it is treated as default. The first service you deploy must be thedefault service.

You can update your service at any time by running the gcloud app deploy command again. Each time youdeploy, a new version is created and traffic is automatically routed to the latest version.

To confirm that your service has been created and a version has been deployed:

1. View your App Engine services in the Cloud Console:View services

You should see one service listed, named default. The default service is publicly accessible at the followingURL:

https://***PROJECT_ID***.***REGION_ID***.r.appspot.com

2. View your versions:

View versions

You should see one time stamped version listed, corresponding to your deployment.

Note that you can deploy multiple services other than the default service. For more information on the multi-service design pattern, see An Overview of App Engine. To learn how to send requests to specific services and versions, see How Requests are Routed.

Updating the example web service

The following sections update the example web service with a form and a handler that responds when a usersubmits the form.

Creating a form for user input

To let a user submit data to your server, use an HTML form.

1. In your my-nodejs-service folder, create a folder named views to store your HTML files.
2. In the views folder, create a file named form.html and add the following code

appengine/building-an-app/update/views/form.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>My App Engine App</title>
  </head>
  <body>
```

```
<h2>Create a new post</h2>
    <formmethod="POST"action="/submit">
      <div>
        <inputtype="text"name="name"placeholder="Name">
      </div>
      <div>
        <textareaname="message"placeholder="Message"></textarea>
      </div>
      <div>
        <buttontype="submit">Submit</button>
      </div>
    </form>
  </body>
```

This simple form allows a user to enter a name and message to send to the server. It sends the data via an HTTP `POST` request to `/submit`, as specified by the `method` and `action` attributes on the `<form>` element. At this point, you should have a file structure like the following:

```
my-nodejs-service/
  views/
    form.html
  app.yaml
  package.json
  server.js
```

Creating a handler for submitted data

When a user submits a message to the server, a `POST` request with the data is sent to `/submit`. To read the data from the body of the request, use the `body-parser` module from NPM and create a new request handler.

1. From your `my-nodejs-service` folder, add `body-parser` as a dependency by running the following command:

```
npm install body-parser
```

2. At the beginning of your `server.js` file, import `body-parser` by adding the following:

```
const bodyParser =require('body-parser');
```

3. Set your Express app to use `body-parser` by adding the line:

appengine/building-an-app/update/server.js

```
app.use(bodyParser.urlencoded({ extended:true}));
```

4. Add a `POST` handler to your `server.js` file to read the data:

appengine/building-an-app/update/server.js

```
app.post('/submit',(req, res)=>{
  console.log({
    name: req.body.name,
    message: req.body.message
  });
  res.send('Thanks for your message!');
});
```

This sample handler logs the user's name and message to the console, but you could also perform operationson the data or store it in a database.

Testing the form locally

Test your new form locally before deploying your changes.

1. Start your Node.js server:

```
npm start
```

Deploying your changes

When you deploy an update, a new version of your default service is created and traffic is automaticallyrouted to the latest version. To deploy:

1. From your `my-nodejs-service` folder, run the following command:

```
gcloud app deploy
```

This is the same command you learned in Deploying Your Web Service.

2. Confirm that a new version is listed in the Cloud Console:View versions

You should see two versions corresponding to the previous and current deployment.

After deploying, your new form is accessible at https://***PROJECT_ID***.***REGION_ID***.r.appspot.com/submit.Use it to submit a message or two!

If you do not need the previous version anymore, you can delete it from the versions page in the CloudConsole.

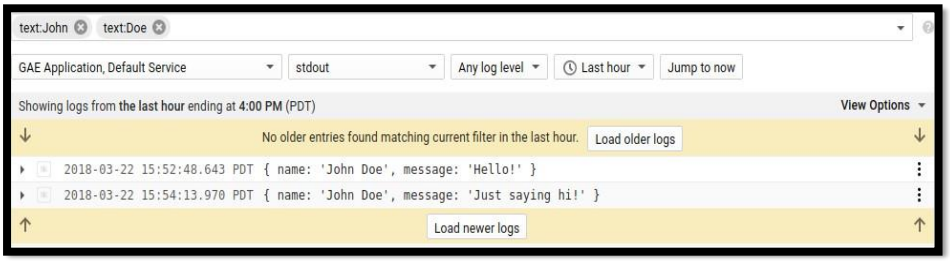
Using the Logs Viewer

In the Cloud Console, you can see the messages submitted with the form you built in Updating Your WebService. To find your service's logs:

1. Open the Logs Viewer in the Cloud Console:Go to Logs Viewer

2. In the first filter dropdown at the top of the page, ensure **GAE Application** is selected, and choose **DefaultService**.

3. Use the text field above the dropdowns to search for the name you used when you submitted your form.You should see the logs corresponding to your submissions, for example:



RESULT:

Thus the GAE launcher was launched and created a web application successfully.

EX.No: 5	TO SIMULATE A CLOUD SCENARIO USING CLOUDSIM AND RUN ASCHEDULING ALGORITHM

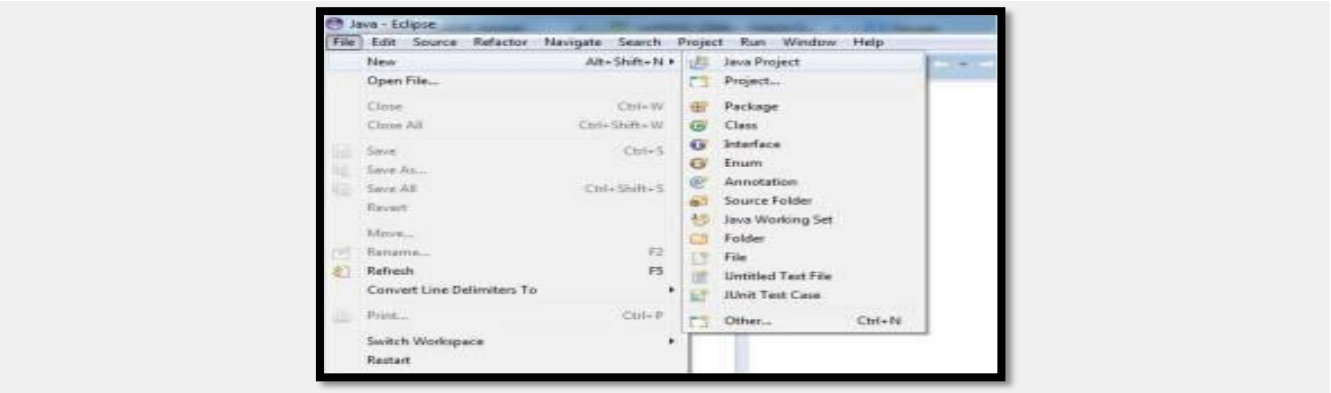
AIM:

To simulate a cloud scenario using cloudsim and run a scheduling algorithm.

PROCEDURE:

Step by step installation of cloud sim into eclipse

1. Open up Eclipse and go to Menu Section, then click File, keep on clicking New and finally select javaproject. It is shown as in the Figure1



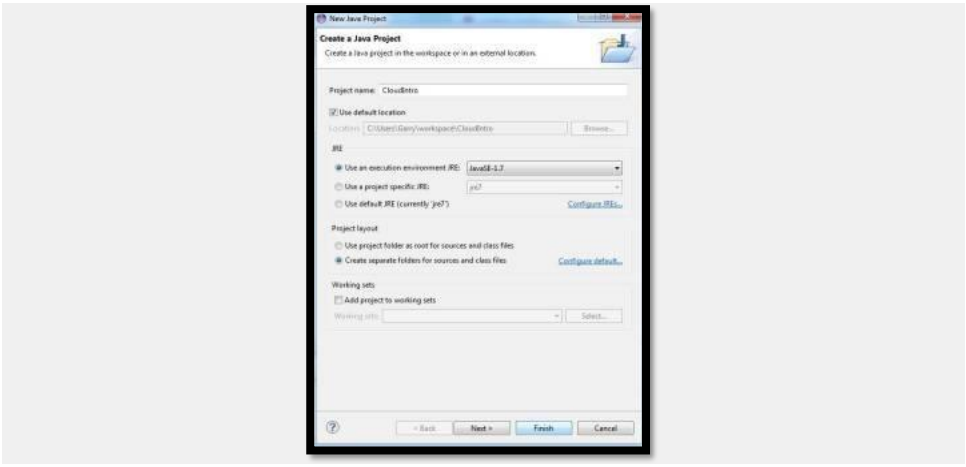
Open eclipse and select java projectOpen up Eclipse and Click on java project

2. A new window will get open. Put a foot on to the following steps:-2.1.Enter project name. (I have named it as CloudIntro)

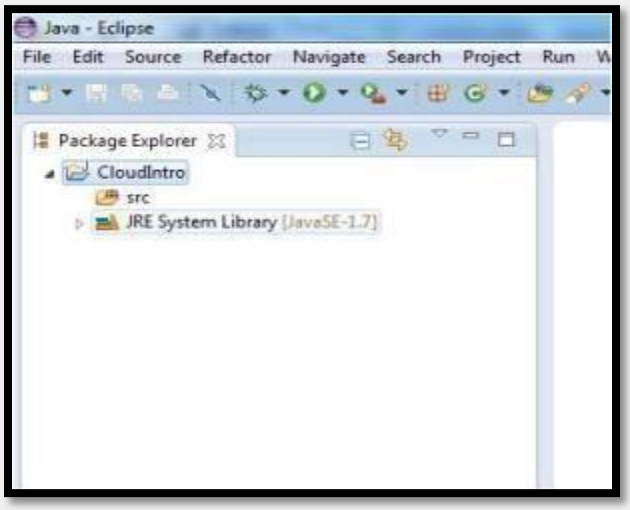
In the next line you will see the path where your project will be created as it as shown in the Figure2.

Next You need to select the JRE environment.

Finally Click Finish

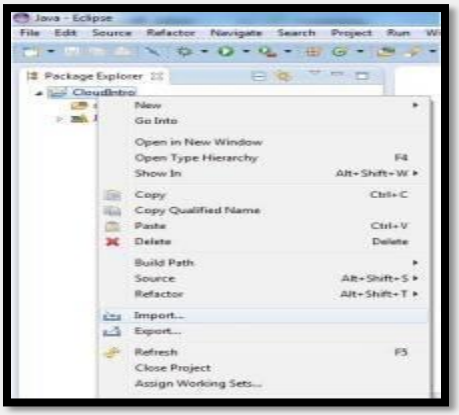


3. Once you hit finish. An empty project named CloudIntro will be created in the project List as shown inthe Figure3.



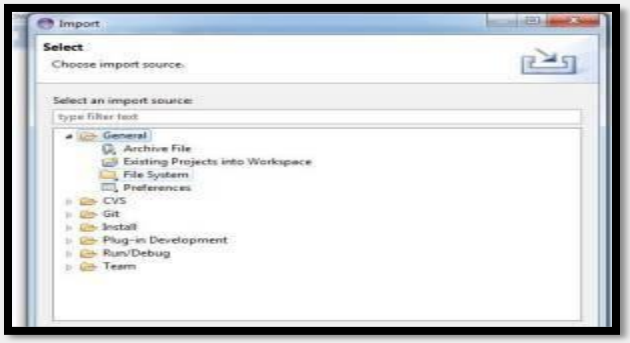
Project Folder Location

4. Next step is to go the project CloudIntro, right click on it. Click Import as shown in the Figure4.

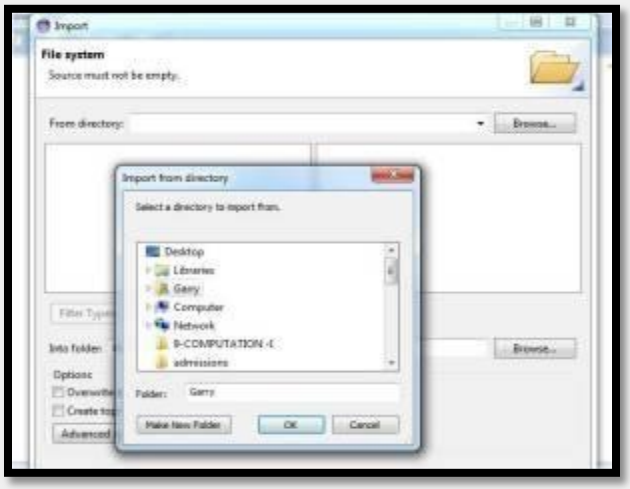


Import cloud sim tool files and subsequent folders

5. A new window will get open, now click File System as demonstrated in the Figure5.

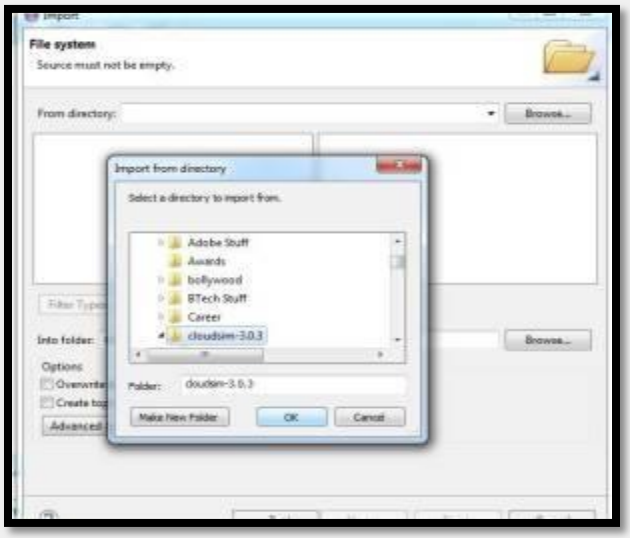


6. Next Step is to go to the directory where you have extracted your cloud sim tool. Figure6 is shown toguide you to get into the directory where your cloudsim folder is located.



Go to Directory to select Cloudsim (My system searching)

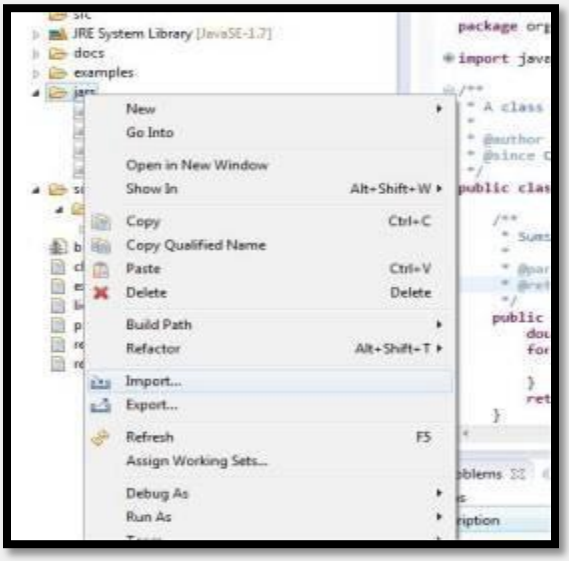
7. Select the cloudsim and click Finish as shown in the Figure7.



Select Cloudsim and Hit finish

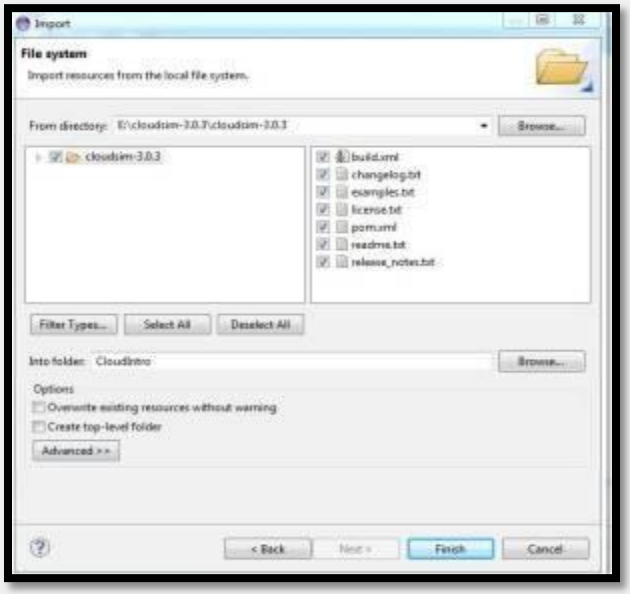
8. Now go to the link **http://commons.apache.org/proper/commonsmath/download_math.cgi . Download the file named as "**commons-math3-3.4.1-bin.zip". Unzip this file. We need jar files for mathfunctions.

9. Now go to the left side of the eclipse tool in the project bar. Go to jar and right click on it. Click import asshown in the Figure8.



Import jar files for math calculations

10. Now go to the folder where you have placed the downloaded and extracted file as described by point 8.Then all you have to do is select that jar file and hit finish as shown by the Figure9**.**



Import only jar

11. Finally the cloud sim is installed into your Eclipse environment.

PROGRAM FOR SCHEDULING ALGORITHM

```
package examples.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 java.util.Random;
import org.cloudbus.cloudsim.Cloudlet;
import org.cloudbus.cloudsim.CloudletSchedulerSpaceShared;
import org.cloudbus.cloudsim.CloudletSchedulerTimeShared;
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;
/**
* An example showing how to create
* scalable simulations.
*/
public class Simulation {
/** The cloudlet list. */
private static List<Cloudlet> cloudletList;
/** The vmlist. */
private static List<Vm> vmlist;
private static List<Vm> createVM(int userId, int vms) {
//Creates a container to store VMs. This list is passed to the broker later
LinkedList<Vm> list = new LinkedList<Vm>();
//VM Parameters
long size = 10000; //image size (MB)
int ram = 512; //vm memory (MB)
int mips = 1000;
long bw = 1000;
int pesNumber = 1; //number of cpus
String vmm = "Xen"; //VMM name
//create VMs
Vm[] vm = new Vm[vms];
```

```
for(int i=0;i<vms;i++){
vm[i] = new Vm(i, userId, mips, pesNumber, ram, bw, size, vmm,
new CloudletSchedulerSpaceShared());
//for creating a VM with a space shared scheduling policy for
cloudlets:
//vm[i] = Vm(i, userId, mips, pesNumber, ram, bw, size, vmm, new
CloudletSchedulerSpaceShared());
list.add(vm[i]);
}
return list;
}
private static List<Cloudlet> createCloudlet(int userId, int cloudlets){
// Creates a container to store Cloudlets
        LinkedList<Cloudlet> list = new LinkedList<Cloudlet>();
//cloudlet parameters
long length = 1000;
long fileSize = 300;
long outputSize = 300;
int pesNumber = 1;
      UtilizationModel utilizationModel = new UtilizationModelFull();
Cloudlet[] cloudlet = new Cloudlet[cloudlets];
for(int i=0;i<cloudlets;i++){
Random r= new Random();
cloudlet[i] = new Cloudlet(i, length +r.nextInt(2000), pesNumber,
fileSize, outputSize, utilizationModel, utilizationModel, utilizationModel);
// setting the owner of these Cloudlets
cloudlet[i].setUserId(userId);
list.add(cloudlet[i]);
}
return list;
}
////////////////////////// STATIC METHODS ///////////////////////
/**
* Creates main() to run this example
*/
public static void main(String[] args) {
Log.printLine("Starting CloudSimExample6...");
try {
// First step: Initialize the CloudSim package. It should be called
// before creating any entities.
                    int num_user = 3;  // number of grid users
                  Calendar calendar = Calendar.getInstance();
                boolean trace_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
```

```
Datacenter datacenter0 = createDatacenter("Datacenter_0");
Datacenter datacenter1 = createDatacenter("Datacenter_1");
//Third step: Create Broker
                DatacenterBroker broker = createBroker();
int brokerId = broker.getId();
//Fourth step: Create VMs and Cloudlets and send them to broker
                vmlist = createVM(brokerId,10); //creating 20 vms
cloudletList = createCloudlet(brokerId,40); // creating 40 cloudlets
broker.submitVmList(vmlist);
broker.submitCloudletList(cloudletList);
// Fifth step: Starts the simulation
CloudSim.startSimulation();
                // Final step: Print results when simulation is over
List<Cloudlet> newList = broker.getCloudletReceivedList();
CloudSim.stopSimulation();
printCloudletList(newList);
                //Print the debt of each user to each datacenter
datacenter0.printDebts();
datacenter1.printDebts();
                Log.printLine("CloudSimExample6 finished!");
}
catch (Exception e)
{
e.printStackTrace();
Log.printLine("The simulation has been terminated due to an
unexpected error");
}
}
private static Datacenter createDatacenter(String name){
        // Here are the steps needed to create a PowerDatacenter:
// 1. We need to create a list to store one or more
//                Machines
List<Host> hostList = new ArrayList<Host>();
// 2. A Machine contains one or more PEs or CPUs/Cores. Therefore, should
//                create a list to store these PEs before creating
//                a Machine.
List<Pe> peList1 = new ArrayList<Pe>();
int mips = 1000;
// 3. Create PEs and add these into the list.
//for a quad-core machine, a list of 4 PEs is required:
peList1.add(new Pe(0, new PeProvisionerSimple(mips))); // need to store Pe
id and MIPS Rating
        peList1.add(new Pe(1, new PeProvisionerSimple(mips)));
        peList1.add(new Pe(2, new PeProvisionerSimple(mips)));
        peList1.add(new Pe(3, new PeProvisionerSimple(mips)));
//Another list, for a dual-core machine
List<Pe> peList2 = new ArrayList<Pe>();
        peList2.add(new Pe(0, new PeProvisionerSimple(mips)));
        peList2.add(new Pe(1, new PeProvisionerSimple(mips)));
```

```
//4. Create Hosts with its id and list of PEs and add them to the list of machines
int hostId=0;
int ram = 2048; //host memory (MB)
long storage = 1000000; //host storage
int bw = 10000;
hostList.add(
new Host(
                                hostId,
                        new RamProvisionerSimple(ram),
                        new BwProvisionerSimple(bw),
                                storage,
                                peList1,
new VmSchedulerTimeShared(peList1)
)
); // This is our first machine
hostId++;
hostList.add(
new Host(
                                hostId,
                        new RamProvisionerSimple(ram),
                        new BwProvisionerSimple(bw),
                                storage,
                                peList2,
new VmSchedulerTimeShared(peList2)
)
); // Second machine
//To create a host with a space-shared allocation policy for PEs to VMs:
//hostList.add(
//                      new Host(
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();
}
return datacenter;
}
//We strongly encourage users to develop their own broker policies, to submit vms
and cloudlets according
//to the specific rules of the simulated scenario
private static DatacenterBroker createBroker(){
```

```
DatacenterBroker broker = null;
try {
                    broker = new DatacenterBroker("Broker");
} catch (Exception e) {
e.printStackTrace();
return null;
}
return broker;
}
/**
* Prints the Cloudlet objects
* @param list  list of Cloudlets
*/
@SuppressWarnings("deprecation")
private static void printCloudletList(List<Cloudlet> list) {
int size = 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 + indent +
"Time" + indent + "Start Time" + indent + "Finish Time" +indent+"user id"+indent);
DecimalFormat dft = new DecimalFormat("###.##");
for (int i = 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 + indent +
dft.format(cloudlet.getActualCPUTime()) +
indent + indent + dft.format(cloudlet.getExecStartTime())+
indent + indent + indent +
dft.format(cloudlet.getFinishTime())+indent
+cloudlet.getUserId());

}
}
}
}
```

RESULT:

Thus the cloudsim was created successfully and executed a scheduling algorithm.

EX.No: 6	FIND A PROCEDURE TO TRANSFER THE FILES FROM ONE VIRTUALMACHINE TO ANOTHER VIRTUAL MACHINE

AIM:

To transfer the files from one virtual machine to another virtual machine.

PROCEDURE:

Transferring files between different types of virtual machines which are provided by the SCS.

Often times it is useful to be able to transfer files between the **guest** machine and the **host** machine.

On this page you will find instructions for the following methods:

- Creating a Shared Folder in VirtualBox
- Dragging and Dropping Files in VirtualBox
- Managing Files with NextCloud

Creating a Shared Folder in VirtualBox

- A shared folder is a folder which makes its files available on both the guest machine *and* the host machine **atthe same time**. Creating a shared folder between the guest and the host allows you to easily manage files which should be present on both machines.
- The course virtual machines are ready to use shared folders right away, but
- if you are using the virtual machine on your personal computer you will need to specify which folder to use as shared storage.If you are using a course VM on a lab computer,
- it is likely that a shared folder has already been setup for you. On the desktop of your course VM you should notice a foldertitled *SharedFolders*.
- Inside of this you will find any folders that have been shared between the course VM and lab computers.You should see two folders that have already been configured for you: **Z_DRIVE** and **Temp**.
- *Z_DRIVE* gives you access to your Windows Account Z:\ drive. This is storage that is persistent to your SCSaccount and available as a network drive on the lab computers.

Shared Folders on Personal Computers

- If you are using your own personal machine, you will need to configure VirtualBox to look in the right place for your shared files.First, click on the guest machine you intend to share files with. From there, you can select the guest *Settings* and navigate to *Shared Folders* on the left side menu.
- To create a new shared folder,either click the *New Folder* icon on the right menu **or** right click the empty list of shared folders and click *Add Shared Folder*. From here, there are six options:

- **Folder Path:** The folder name on the **host** machine. Click the drop down menu
- and navigate to the folder you would like to share.

- **Folder Name:** This is the name of the folder as it will appear on the **guest** machine.

- **Read-Only:** If you check read-only, the **guest** machine will be unable to write changes to the folder. This is valuable when you only want to send files *to* the virtual machine, but do not want to risk having the files modified by the guest.

- **Mount Point:** Unless you already know about mount points, leave this blank.

- **Make Permanent:** If you check this, the shared folder will be a permanent **machine folder**. If it is not checked, the folder will not be shared after a shutdown.

  On the course virtual machines, when you load into the desktop, you should see a folder labelled *SharedFolders*. In there you will see any folders that are currently mounted and being shared.

Dragging and Dropping Files in VirtualBox

If you only need to transfer a few files quickly, you can simply drag and drop the files in.

On the top bar of the running guest machine, click on *Devices > Drag and Drop* and make sure that *Bidirectional* is selected.

You can also drag files from the guest machine into the host. To do this, simply open the file browser on the host to where you would like to drop the files and drag the files from the virtual machine into the file browser of the host.

File transfers should be pretty quick; if the virtual machine seems stuck when transferring, simply cancel the transfer and try again.

Managing Files with NextCloud

On any virtual machine, including VirtualBox, VMWare, or the virtual machines hosted on the SCS OpenStack, you can access the SCS NextCloud services to move files between multiple machines andyour SCS Windows Account storage.

NextCloud offers you all of your SCS storage in one remote location, similar to how you might use other file hosting services like Dropbox or Google Drive.

Before trying to use NextCloud, you should check that you can access the service by logging in here.

If you can access the NextCloud services, you can browse the various file storage services available to you:

- **Linux Home:** These are the files from your SCS Linux Account
- **Windows Home:** These are the files from your SCS Windows Account and your lab Z:\ drive.
- **NextCloud:** In addition to the other storage accounts provided to you by the SCS, you can also upload up to20GB of files directly to NextCloud.

Downloading NextCloud Files to a VM or Other PC

Once your files are uploaded you will be able to download those files onto any machine which can connectto NextCloud. First, log in to your preferred NextCloud interface (eg. the web interface). N to the folder which contains the files you would like to download. Once you are in the target folder, click the checkbox next to each file you would like to download. Above the file listing you should notice the context bar changing to tell you how many files you have selected and a button labelled *Actions*. Click *Actions* > *Download*.

If you have selected a single file, it will prompt you to confirm the download.
If you have chosen more than one file, NextCloud will place all of the selected files into a *zip archive*.
Before you can use the files, you will need to extract them from the archive.
Once you have downloaded your file, or extracted your archive, you are ready to use your files on your machine.

RESULT:

Thus the procedure to transfer the files from one virtual machine to another virtual machine was studiedsuccessfully.

EX.No: 7	INSTALL HADOOP SINGLE NODE CLUSTER AND SIMPLE APPLICATION LIKE WORDCOUNT

AIM:

To install Hadoop single node cluster and simple application like wordcount.

PROCEDURE:

Install Hadoop

Step 1: Click here to download the Java 8 Package. Save this file in your home directory.

Step 2: Extract the Java Tar File.

***Command*:** tar -xvf jdk-8u101-linux-i586.tar.gz

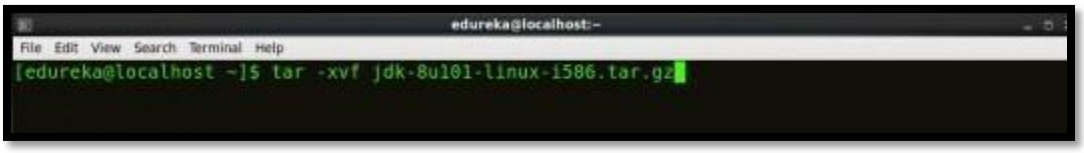


Fig: Hadoop Installation – Extracting Java Files

Step 3: Download the Hadoop 2.7.3 Package.

***Command*:**wgethttps://archive.apache.org/dist/hadoop/core/hadoop-2.7.3/hadoop- 2.7.3.tar.gz

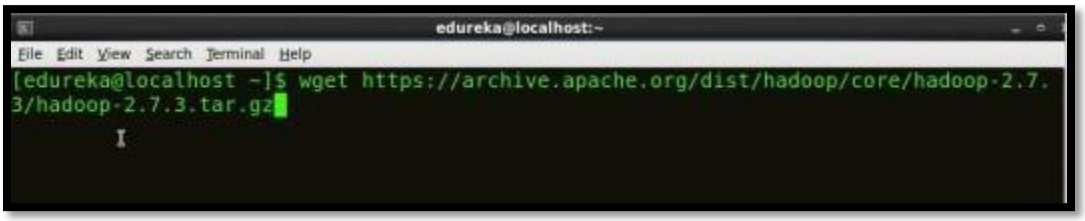


Fig: Hadoop Installation – Downloading Hadoop

Step 4: Extract the Hadoop tar File.

Command: tar -xvf hadoop-2.7.3.tar.gz

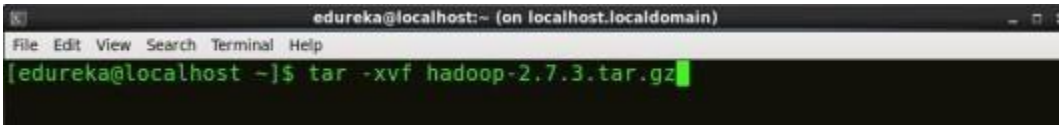


Fig: Hadoop Installation – Extracting Hadoop Files **Step 5:** Add the Hadoop and Java paths in the bash file (.bashrc). Open**. bashrc** file. Now, add Hadoop and Java Path as shown below.

***Command*:** vi .bashrc

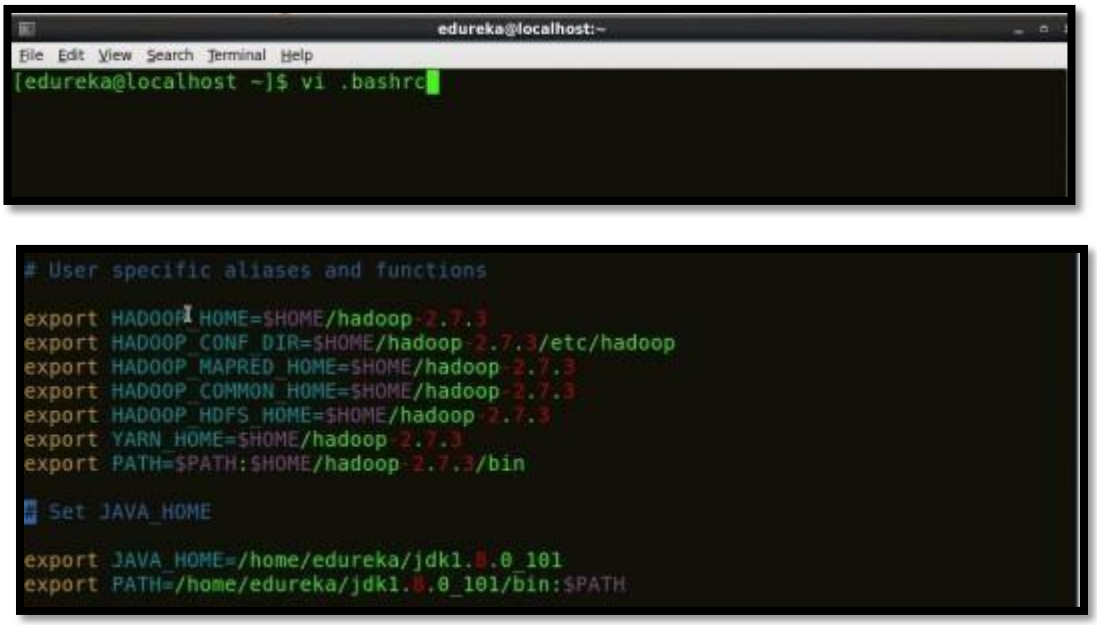


Fig: Hadoop Installation – Setting Environment Variable

Then, save the bash file and close it.For applying all these changes to the current Terminal, execute the source command.

***Command*:** source .bashrc

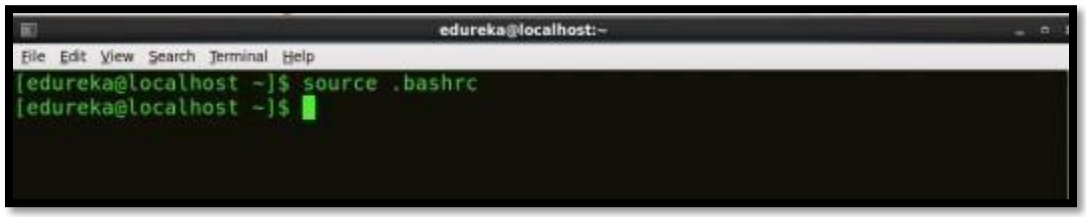


Fig: Hadoop Installation – Refreshing environment variables

To make sure that Java and Hadoop have been properly installed on your system and can beaccessed through the Terminal, execute the java -version and hadoop version commands.

***Command*:** java -version

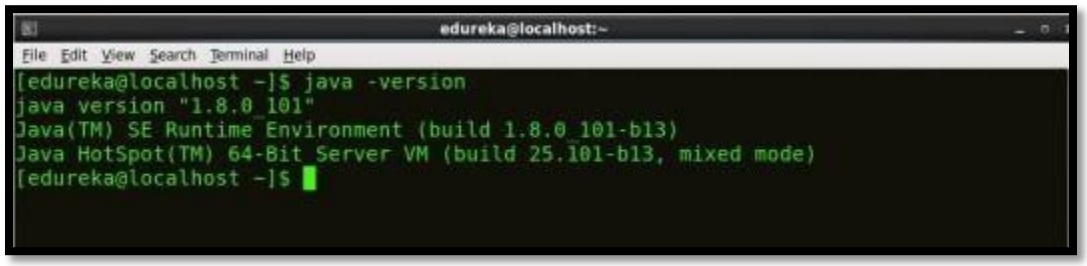


Fig: Hadoop Installation – Checking Java Version

***Command*:** hadoop version

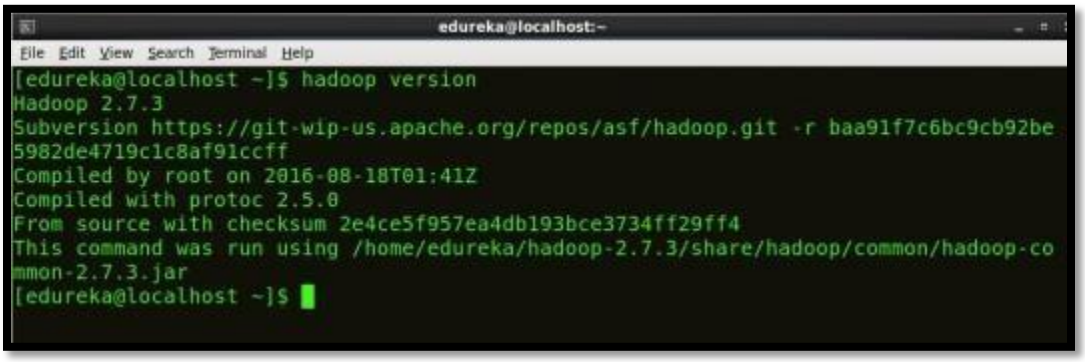


Fig: Hadoop Installation – Checking Hadoop Version

Step 6: Edit the **Hadoop Configuration files**.

Command: cd hadoop-2.7.3/etc/hadoop/

Command: ls
All the Hadoop configuration files are located in **hadoop-2.7.3/etc/hadoop** directory as you can seein the snapshot below:

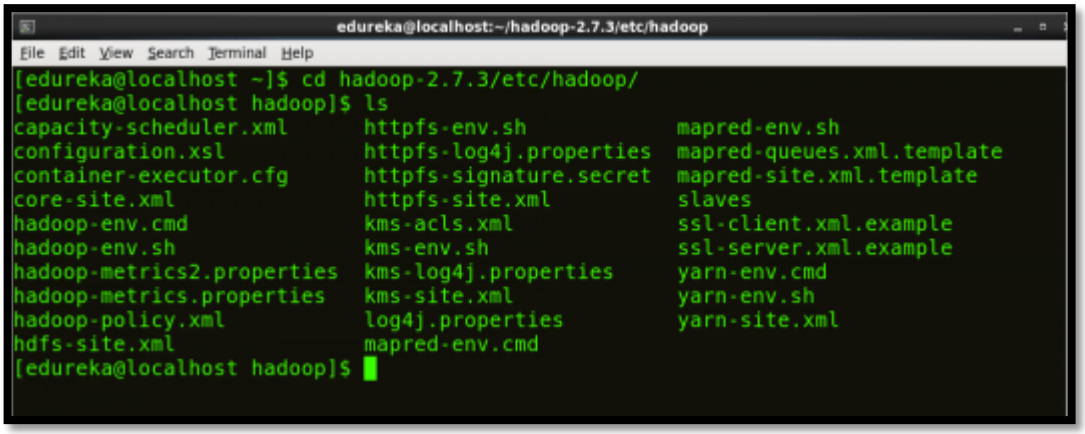


Fig: Hadoop Installation – Hadoop Configuration Files

Step 7: Open *core-site.xml* and edit the property mentioned below inside configuration tag:

core-site.xml informs Hadoop daemon where NameNode runs in the cluster. It contains configuration settings of Hadoop core such as I/O settings that are common to HDFS & MapReduce.

***Command*:** vi core-site.xml

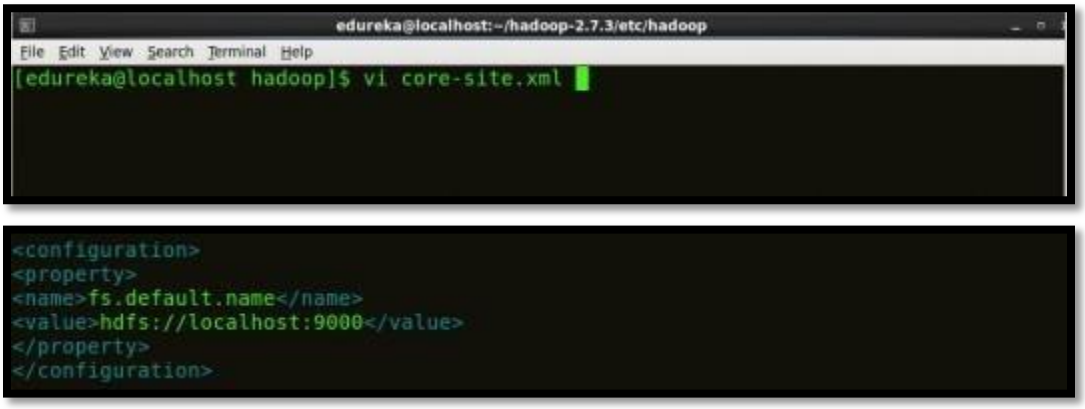


Fig: Hadoop Installation – Configuring core-site.xml

```
1   <?xmlversion="1.0"encoding="UTF-8"?>
2   <?xml-stylesheettype="text/xsl"href="configuration.xsl"?>
3   <configuration>
4   <property>
```

Step 8: Edit*hdfs-site.xml*and edit the property mentioned below inside **configuration tag:**

hdfs-site.xml contains configuration settings of HDFS daemons (i.e. NameNode, DataNode, Secondary NameNode). It also includes the replication factor and block size of HDFS.

***Command*:** vi hdfs-site.xml

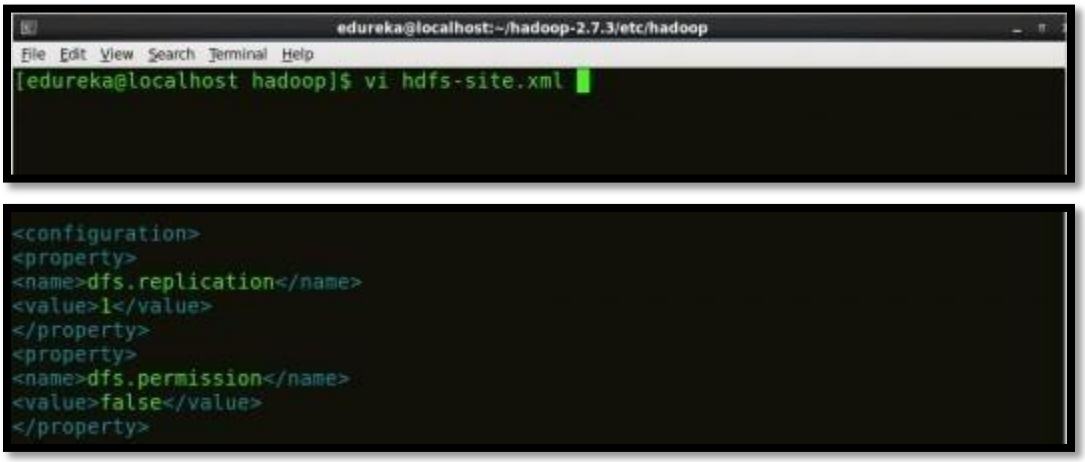


Fig: Hadoop Installation – Configuring hdfs-site.xml

```
<?xmlversion="1.0"encoding="UTF-8"?>
<?xml-stylesheettype="text/xsl"href="configuration.xsl"?>
<configuration>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
```

Step 9: Edit the *mapred-site.xml* file and edit the property mentionedbelow **inside configuration tag:**

mapred-site.xml contains configuration settings of MapReduce application like number of JVM that can run in parallel, the size of the mapper and the reducer process, CPU cores available for a process,etc.In some cases, mapred-site.xml file is not available. So, we have to create themapred-site.xml file using mapred-site.xml template.

***Command*:** cp mapred-site.xml.template mapred-site.xml

***Command*:** vi mapred-site.xml.

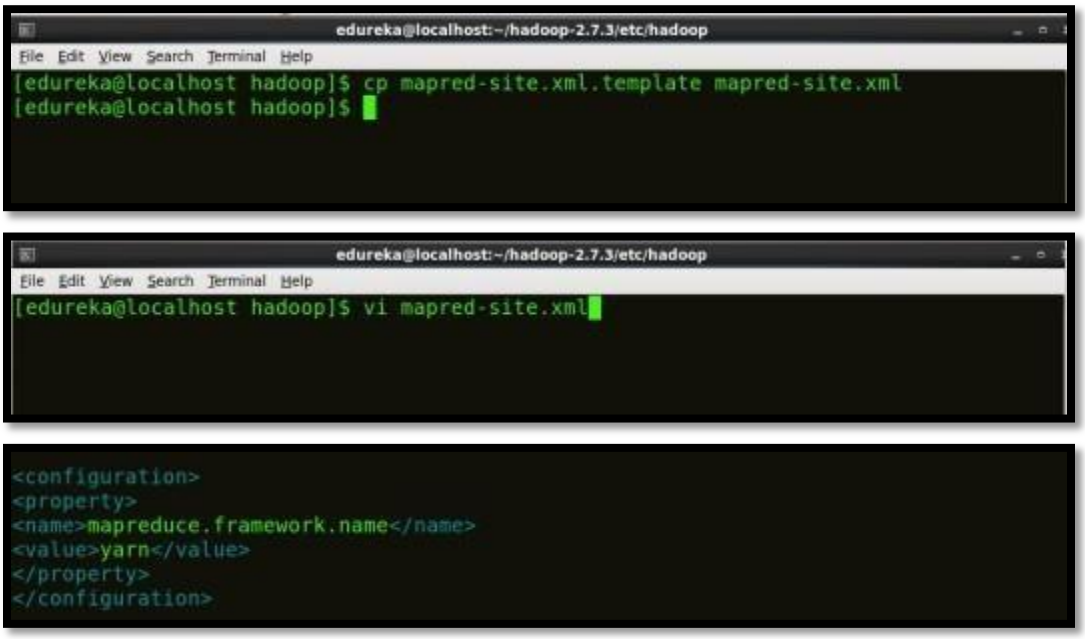
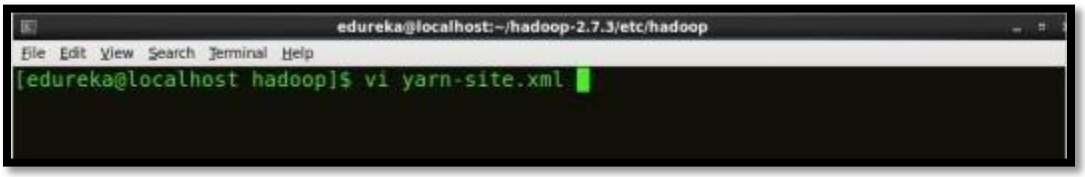


Fig: Hadoop Installation – Configuring mapred-site.xml

```
1   <?xmlversion="1.0"encoding="UTF-8"?>
2   <?xml-stylesheettype="text/xsl"href="configuration.xsl"?>
3   <configuration>
4   <property>
```



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

Fig: Hadoop Installation – Configuring yarn-site.xml

```
1
2                    <?xmlversion="1.0">
                       <configuration>
3
                         <property>
4          <name>yarn.nodemanager.aux-services</name>
5                <value>mapreduce_shuffle</value>
6                          </property>
                           <property>
7
```

Step 11: Edit *hadoop-env.sh* and add the Java Path as mentioned below:

hadoop-env.sh contains the environment variables that are used in the script to run Hadoop like Java home path, etc.

***Command*:** vi hadoop–env.sh

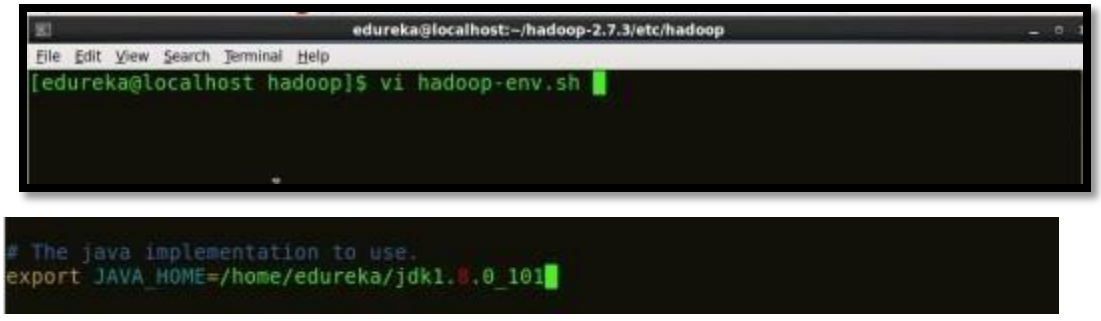


Fig: Hadoop Installation – Configuring hadoop-env.sh

Step 12: Go to Hadoop home directory and format the NameNode. ***Command*:** cd

***Command*:** cd hadoop-2.7.3

***Command*:** bin/hadoop namenode -format

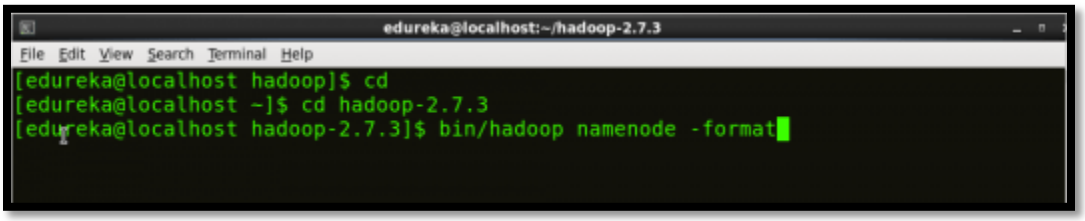


Fig: Hadoop Installation – Formatting NameNode

This formats the HDFS via NameNode. This command is only executed for the first time. Formatting the file system means initializing the directory specified by the dfs.name.dir variable.

Never format, up and running Hadoop filesystem. You will lose all your data stored intheHDFS.

Step 13: Once the NameNode is formatted, go to hadoop-2.7.3/sbin directory and start all the daemons.

Command: cd hadoop-2.7.3/sbin

Either you can start all daemons with a single command or do it individually.

Command: ./start-all.sh

The above command is a combination of ***start-dfs.sh, start-yarn.sh*** &***mr-jobhistory-daemon.sh***

Or you can run all the services individually as below:

Start NameNode:

The NameNode is the centerpiece of an HDFS file system. It keeps the directory tree of all files stored in the HDFS and tracks all the file stored across the cluster.

Command: ./hadoop-daemon.sh start namenode

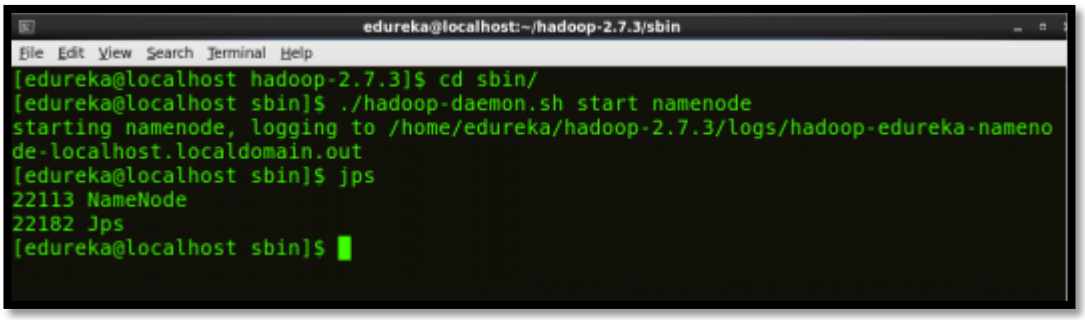


Fig: Hadoop Installation – Starting NameNode

StartDataNode:

On startup, a DataNode connects to the Namenode and it responds to the requests from the Namenode for differentoperations.

Command: ./hadoop-daemon.sh start datanode

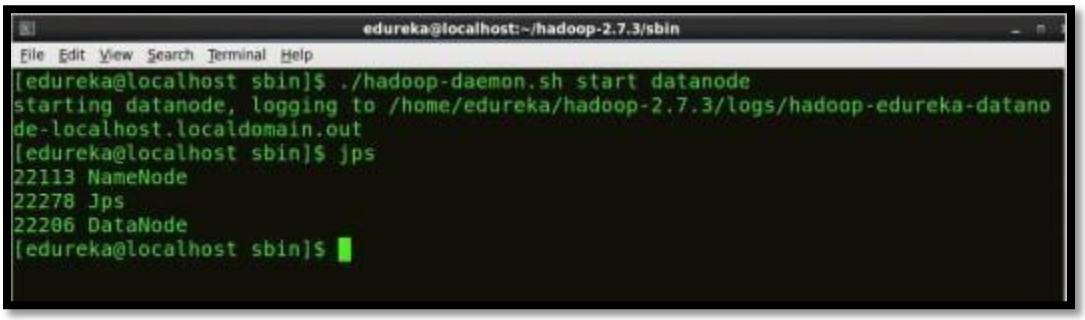


Fig: Hadoop Installation – Starting DataNode

Start ResourceManager:

ResourceManager is the master that arbitrates all the available cluster resources and thus helps in managing the distributed applications running on the YARN system. Its work is to manage each NodeManagers and the each application's ApplicationMaster.

Command: ./yarn-daemon.sh start resourcemanager

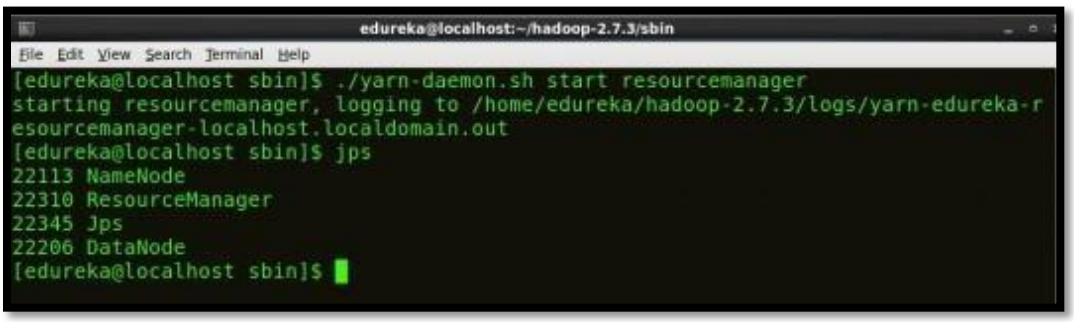


Fig: Hadoop Installation – Starting ResourceManager

Start NodeManager:

The NodeManager in each machine framework is the agent which is responsible for managing containers, monitoring their resource usage and reporting the same to the ResourceManager.

Command: ./yarn-daemon.sh start nodemanager

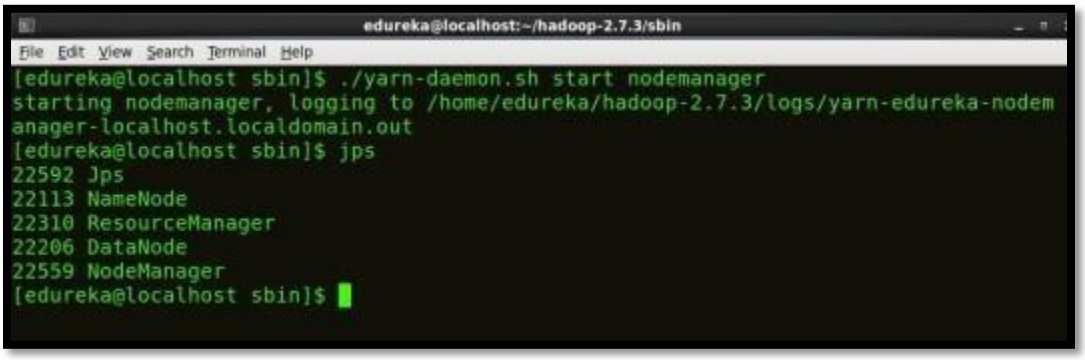


Fig: Hadoop Installation – Starting NodeManager

Start JobHistoryServer:

JobHistoryServer is responsible for servicing all job history related requests from client.

***Command*:** ./mr-jobhistory-daemon.sh start historyserver

Step 14: To check that all the Hadoop services are up and running, run the below command.

Command: jps

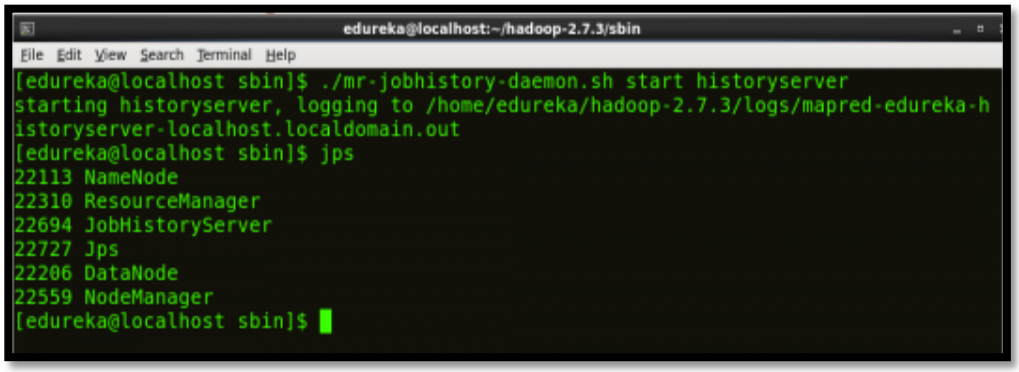


Fig: Hadoop Installation – Checking Daemons

Step 15: Now open the Mozilla browser and go to **localhost**:**50070/dfshealth.html** to check the NameNode interface.

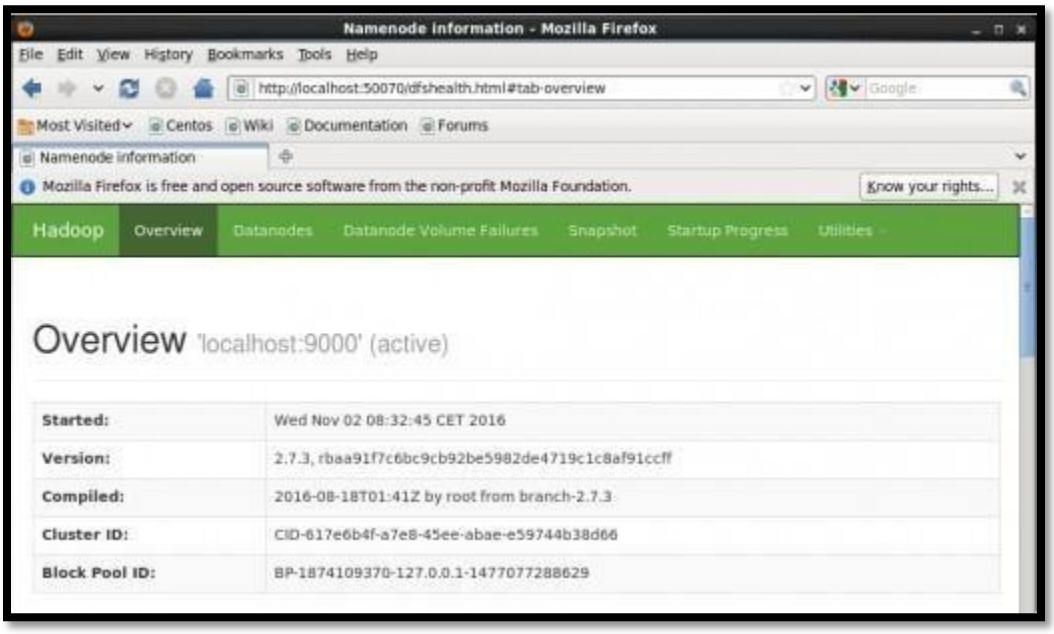


Fig: Hadoop Installation – Starting WebUI

RESULT:

Thus the Hadoop single node cluster was successfully created and executed.

EX.No: 8&9	CREATING AND EXECUTING YOUR FIRST CONTAINER USING DOCKER AND RUN A CONTAINER FROM DOCKER HUB

AIM:

To write a program to run a container from Docker hub.

PROCEDURE:

Run docker -h,

```
$ docker -h
Flag shorthand -h has been deprecated, please use --help

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

…
        ManagementCommands:
  builder     Manage builds
  config      Manage Docker configs
  container   Manage containers
  engine      Manage the docker engine
  image       Manage images
  network     Manage networks
  node        Manage Swarm nodes
  plugin      Manage plugins
  secret      Manage Docker secrets
  service     Manage services
  stack       Manage Docker stacks
  swarm       Manage Swarm
  system      Manage Docker
  trust       Manage trust on Docker images
  volume      Manage volumes
```

- The Docker command line can be used to manage several features of the Docker Engine. In this lab, we will mainly focus on the container command.

```
docker version

Client:
 Version:      19.03.6
...
```

```
Server: Docker Engine - Community
 Engine
 Version:  19.03.5
  ...
```

Step 1: Run your first container

We are going to use the Docker CLI to run our first container.

1. Open a terminal on your local computer
2. Run docker container run -t ubuntu top

Use the docker container run command to run a container with the ubuntu image using the top command.

The -t flags allocate a pseudo-TTY which we need for the top to work correctly.

```
$ docker container run -it ubuntu top
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
aafe6b5e13de: Pull complete
0a2b43a72660: Pull complete
18bdd1e546d2: Pull complete
8198342c3e05: Pull complete
f56970a44fd4: Pull complete
Digest: sha256:f3a61450ae43896c4332bda5e78b453f4a93179045f20c81810
43b26b5e79028
 Status: Downloaded newer image for ubuntu:latest
```

The docker run command will result first in a docker pull to download the ubuntu image onto your host. Once it is downloaded, it will start the container.
The output for the running container should look like this:

```
top - 20:32:46 up 3 days, 17:40,  0 users,  load average: 0.00, 0.01, 0.00
Tasks:  1 total,  1 running,  0 sleeping,  0 stopped,  0 zombie
%Cpu(s):  0.0 us,  0.1 sy,  0.0 ni, 99.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  2046768 total,  173308 free,  117248 used,  1756212 buff/cache
KiB Swap:  1048572 total,  1048572 free,       0 used.  1548356 avail Mem

PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
  1 root      20   0   36636   3072   2640 R   0.3  0.2   0:00.04 top
```

3. Inspect the container with docker container exec

The docker container exec command is a way to "enter" a running container's namespaces with a new process.

Open a new terminal. On cognitiveclass.ai, select Terminal > New Terminal.

Using play-with-docker.com, to open a new terminal connected to node1,

click "Add New Instance" on the

left hand side, then ssh from node2 into node1 using the IP that is listed by 'node1 '.

For example:

```
[node2] (local) root@192.168.0.17 ~
$ ssh 192.168.0.18
[node1] (local) root@192.168.0.18 ~
$
```

In the new terminal, use the docker container ls command to get the ID of the running container you
just created.

```
$ docker container ls
CONTAINER ID   IMAGE     COMMAND     CREATED          STATUS       PORTS    NAMES
 b3ad2a23fab3   ubuntu      "top"     29 minutes ago  Up 29 minutes         goofy_nobel
```

Then use that id to run bash inside that container using the docker container exec command.
Since we are using bash and want to interact with this container from our terminal,
use -it flags to run using interactive mode while allocating a psuedo-terminal.

```
$ docker container exec -it b3ad2a23fab3 bash
root@b3ad2a23fab3:/#
```

From the same termina, run ps -ef to inspect the running processes.

```
root@b3ad2a23fab3:/# ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 20:34 ?        00:00:00 top
root        17     0  0 21:06 ?        00:00:00 bash
root        27    17  0 21:14 ?        00:00:00 ps –ef
```

You should see only the top process, bash process and our ps process.

For comparison, exit the container, and run ps -ef or top on the host. These commands

will work on linux or mac. For windows, you can inspect the running processes using tasklist.

```
root@b3ad2a23fab3:/# exit
exit
$ ps -ef
# Lots of processes!
```

- Clean up the container running the top processes by typing: <ctrl>-c, list all

containers and remove the containers by their id.

```
docker ps -a
```

```
docker rm <CONTAINER ID>
```

Step 2: Run Multiple Containers

1. Explore the Docker Hub

The Docker Hub is the public central registry for Docker images, which contains community and official images.

2. Run an Nginx server

Let's run a container using the official Nginx image from the Docker Hub.

```
$ docker container run --detach --publish 8080:80 --name nginx nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
36a46ebd5019: Pull complete
57168433389f: Pull complete
332ec8285c50: Pull complete
Digest: sha256:c15f1fb8fd55c60c72f940a76da76a5fccce2fefa0dd9b17967b9e40b0355316
Status: Downloaded newer image for nginx:latest
5e1bf0e6b926bd73a66f98b3cbe23d04189c16a43d55dd46b8486359f6fdf048
```

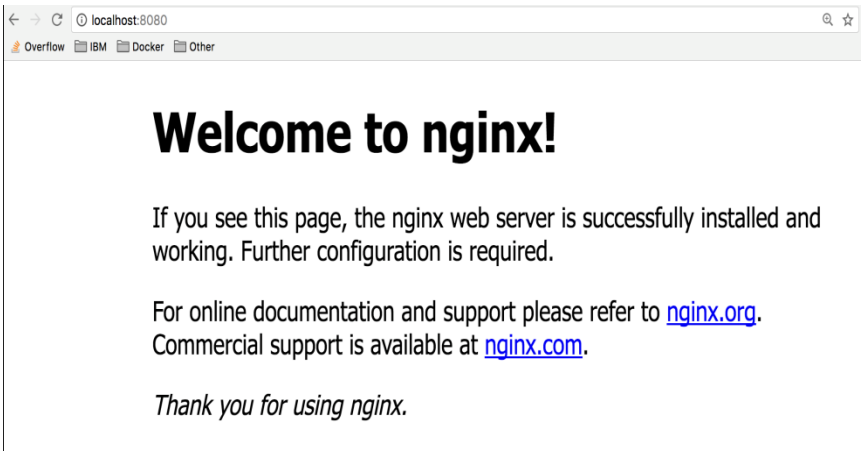
3. Access the nginx server on localhost:8080.

```
curl localhost:8080
```

will return the HTML home page of Nginx,

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
```

4. If you are using play-with-docker, look for the 8080 link near the top of the page, or if you run a Docker client with access to a local browser,



5. Run a mongo DB server

Now, run a mongoDB server. We will use the official mongoDB image from the Docker Hub.

Instead of using the latest tag (which is the default if no tag is specified), we will use a specific version of the mongo image.

```
$ docker container run --detach --publish 8081:27017 --name mongo mongo:4.4
Unable to find image mongo:4.4 locally
4.4: Pulling from library/mongo
d13d02fa248d: Already exists
bc8e2652ce92: Pull complete
3cc856886986: Pull complete
c319e9ec4517: Pull complete
b4cbf8808f94: Pull complete
cb98a53e6676: Pull complete
f0485050cd8a: Pull complete
ac36cdc414b3: Pull complete
61814e3c487b: Pull complete
523a9f1da6b9: Pull complete
3b4beaef77a2: Pull complete
Digest: sha256:d13c897516e497e898c229e2467f4953314b63e48d4990d3215d876ef9d1fc7c
Status: Downloaded newer image for mongo:4.4
d8f614a4969fb1229f538e171850512f10f490cb1a96fca27e4aa89ac082eba5
```

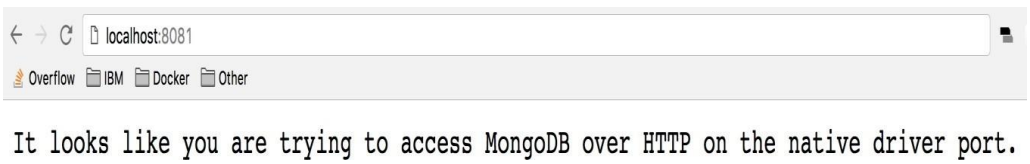
6. Access localhost:8081 to see some output from mongo.

curl localhost:8081

which will return a warning from MongoDB,

```
It looks like you are trying to access MongoDB over HTTP on
the native driver port.
```

7. If you are using play-with-docker, look for the 8080 link near the top of the page.



8.Check your running containers with `docker container ls`

```
$ docker container ls
CONTAINER ID    IMAGE    COMMAND    CREATED    STATUS    PORTS    NAMES
d6777df89fea    nginx    "nginx -g 'daemon ..."    Less than a second ago    Up 2 seconds
0.0.0.0:8080->80/tcp    nginx ead80a0db505    mongo    "docker-entrypoint..."    17 seconds ago
Up 19 seconds    0.0.0.0:8081->27017/tcp    mongo af549dccd5cf    ubuntu    "top"    5 minutes ago
 Up 5 minutes    priceless_kepler
```

Step 3: Clean Up

Completing this lab results in a bunch of running containers on your host. Let's clean these up.

1. First get a list of the containers running using docker container ls.

```
$ docker container ls
CONTAINER ID        IMAGE        COMMAND        CREATED        STATUS        PORTS        NAMES
d6777df89fea        nginx        "nginx -g 'daemon ..."    3 minutes ago        Up 3 minutes        0.0.0.0:8080->80/tcp
 nginx
ead80a0db505        mongo        "docker-entrypoint..."    3 minutes ago        Up 3 minutes        0.0.0.0:8081->2701
7/tcp    mongoa
f549dccd5cf        ubuntu        "top"                8 minutes ago        Up 8 minutes        priceless_kepler
```

2. Next, run docker container stop [container id] for each container in the list.

You can also use the names of the containers that you specified before.

```
$ docker container stop d67 ead af5
d67
ead
af5
```

3. Remove the stopped containers

docker system prune is a really handy command to clean up your system.

It will remove any stopped containers, unused volumes and networks, and dangling images.

```
$ docker system prune
WARNING! This will remove:
    - all stopped containers
    - all volumes not used by at least one container
    - all networks not used by at least one container
    - all dangling images
Are you sure you want to continue? [y/N] y
Deleted Containers:
7872fd96ea4695795c41150a06067d605f69702dbcb9ce49492c9029f0e1b44b
60abd5ee65b1e2732ddc02b971a86e22de1c1c446dab165462a08b037ef7835c
31617fdd8e5f584c51ce182757e24a1c9620257027665c20be75aa3ab6591740

Total reclaimed space: 12B
```

RESULT:

Thus the above container program was successfully created and executed from Docker hub.