|  |  |
| --- | --- |
| **Ex.No:01** | **INSTALLING VIRTUAL BOX/VMWARE WITH DIFFERENT FLAVORS OF OPERATING SYSTEMS** |
| **Date:** |

#### AIM:

To Install Virtualbox/VMware workstation with different flavors of linux or windows os on top of

windows7 or 8

#### PROCEDURE:

For Installing a VMware workstation this is the direct link to download htt[ps://www.vmware.c](http://www.vmware.com/go/getplayer-win)o[m/go/getplayer-win](http://www.vmware.com/go/getplayer-win)

#### Lets get the process Started.

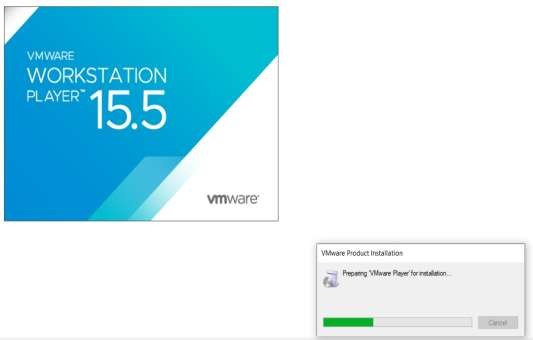
Step 1 – Run the installer

Start the installer by double clicking it. You might see User Account Control Warning. Click Yes to continue.



VMware Player 15 – Installation – User Account Control Warning

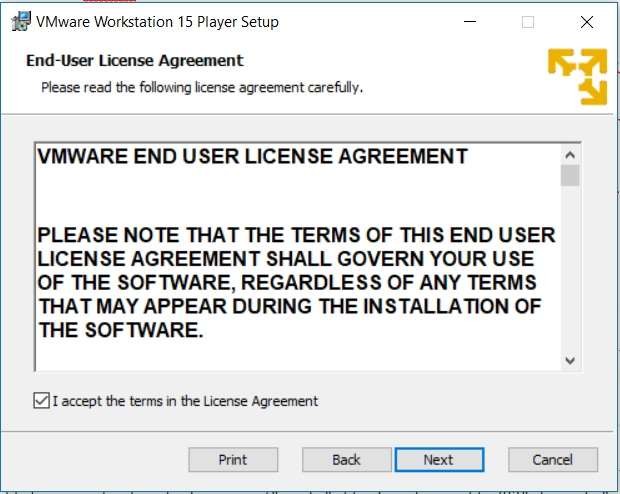
Then, you will see a splash screen. It will prepare the system for installation and then the installation wizard opens.



VMware Player 15.5 Installation – Initial Splash Screen

VMware Player 15.5 Installation – Setup Wizard

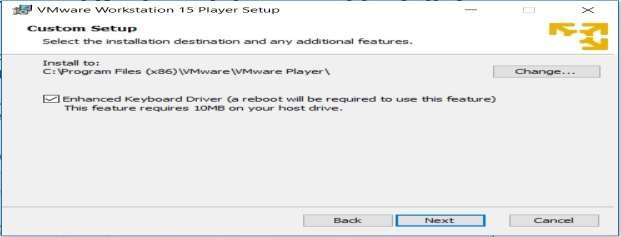
Click next and accept the license terms and click next again to move on to the next screen.



VMware Player 15 Installation – End User Agreement

Step 2 – Custom setup – Enhanced Keyboard driver and Installation directory

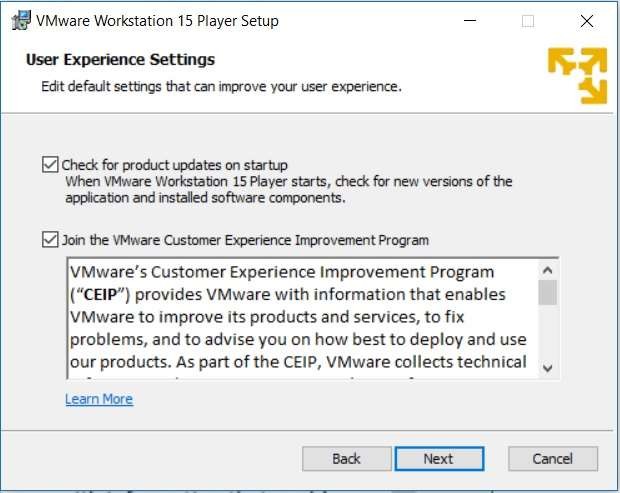
In this dialog box, please select the folder in which you want to install the application. Leave it as it is. Also check the box Enhanced Keyboard Drivers option. Click next.



VMware Player 15 Installation – Custom Setup – Enhanced Keyboard Driver

Step 3 – User Experience Settings

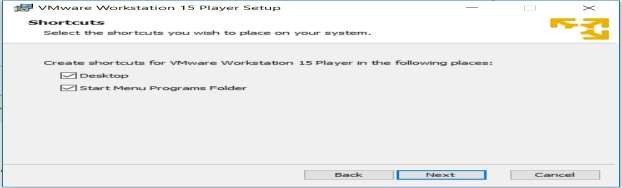
Check the options for Check the product update at Startup and Join the VMware Customer Program. Normally leave it as it is. You can unchecked it if you so desire. Click next



VMware Player 15 Installation – User Experience Settings

Step 4 – Select where the shortcuts will be installed

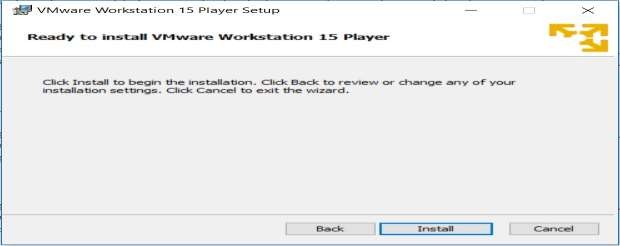
Check the box where the shortcut to run the application will be created. Leave it as it is. Click on next.



VMware Player 15 Installation – Shortcuts

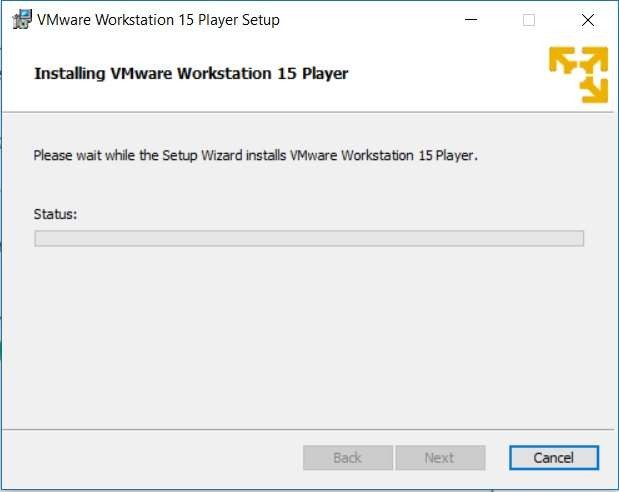
Step 5 – Ready to install

Now the installation wizard is ready to install. Click on install to begin the installation.



VMware Player 15 Installation – Ready to Install

Installation begins, wait for it to complete.



VMware Player 15 Installation – Installation in Progress

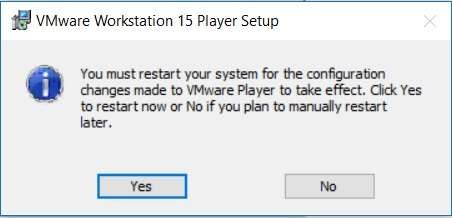
After sometime, you will see installation compete message. You are done.



VMware Player 15.5 Installation – Installation Complete

Click on Finish to complete the installation.

You will be asked to restart your system. Click on Yes to restart. Click No, if you want to restart later. But you must restart before using the application, else some features will not work properly.



VMware Player 15 Installation – Reboot Required

Step 6 – License

Now run the application. You should see a desktop icon. Double click on that or use the start menu to navigate to VMware Player option.

Once you run the application for the first time, you will be asked for license. Select the option Use VMware Workstation Player 15 for free for non-commercial use.

Click continue.



**Installing Oracle Virtual Box**

This is free for personal non-commercial use, so you can download your own version of this software. First, go to this https://[www.virtualbox.org/wiki/Downloads](http://www.virtualbox.org/wiki/Downloads)

Installing Virtual Box is just like installing any other application on your Windows OS, so you shouldn’t have any problem with the installation.

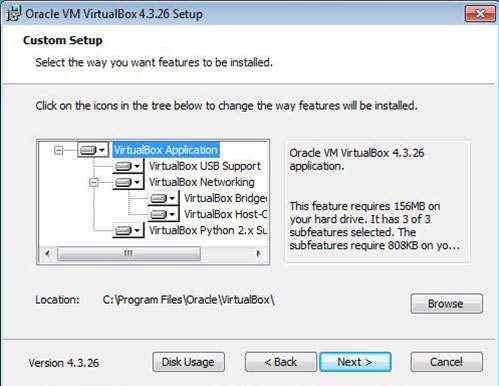
To start the installation, double-click on the installer file you’ve downloaded.



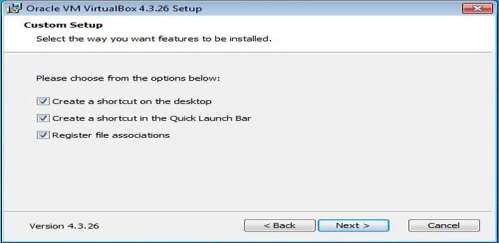
In the Welcome window, click **Next**:



Select the installation folder and click **Next**:



Select the way you want features to be installed and then click **Next**:



Click **Yes** to install Oracle VirtualBox interfaces:



After a minute or two the installation process should finish.

#### Install Ubuntu on Virtual Box as a VM:

**System Configuration of the Host Machine**

In order to install Ubuntu on VirtualBox, you should have a physical computer with at least 4GB of RAM (Random Access Memory), a hard disk drive with at least 30GB of free space (SSD is preferred due to its higher performance).

Your CPU (Central Processor Unit) must support Intel VT-x or AMD-v hardware virtualization features which must also be enabled in UEFI/BIOS. This point is especially important if you are looking for how to install Ubuntu 64-bit on Virtual Box.

#### Downloading the Installation Image

You need to download the Ubuntu distribution for installing Ubuntu on Virtual Box. [Go to](https://ubuntu.com/download/desktop) [the official Ubuntu website](https://ubuntu.com/download/desktop) and download the necessary version of the Ubuntu installer.

Let’s download Ubuntu 18.04.2 LTS – this is the latest long term support (LTS) Ubuntu version available at this moment. You can find version numbers that are higher than 18.04.2, but they may not offer long term support yet.

Five-year support is provided for Ubuntu LTS distributions (both Ubuntu Desktop and Ubuntu Server). Ubuntu LTS is more widely tested, enterprise-focused and compatible with new hardware.

Click the green *Download* button and save the ISO file to the custom location. In our case, the file name is *ubuntu-18.04.2-desktop-amd64.iso*. Ubuntu 18 is provided only as 64-bit editions.



#### Creating a New VM

In order to create a new virtual machine for installing Ubuntu on VirtualBox, open VirtualBox and click *New* (*Machine > New*) or press *Ctrl+N*.



In the *Create Virtual Machine* screen, set the options for a new VM. In our example of installing Ubuntu on VirtualBox, the new VM options are the following:

*Name*: Ubuntu18x64

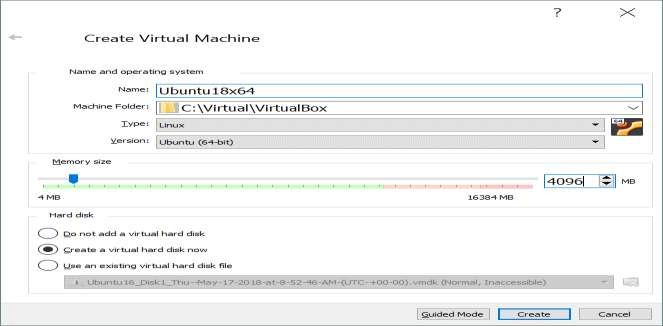
*Machine Folder*: C:\Virtual\VirtualBox (try to use disk D, E or other non-system partitions if you have them).

*Type*: Linux

*Version*: Ubuntu (64-bit)

*Memory size*: Set at least 1 GB of RAM. As our physical machine used in this example has 16 GB of RAM, we can set 4 GB of RAM for a virtual machine to install Ubuntu on VirtualBox. You should leave enough memory for your host operating system to operate normally.

Select the *Create a virtual hard disk now* option. Hit *Create* to continue.



On the next *Create Virtual Hard Disk* screen, set the virtual disk file location, for example,

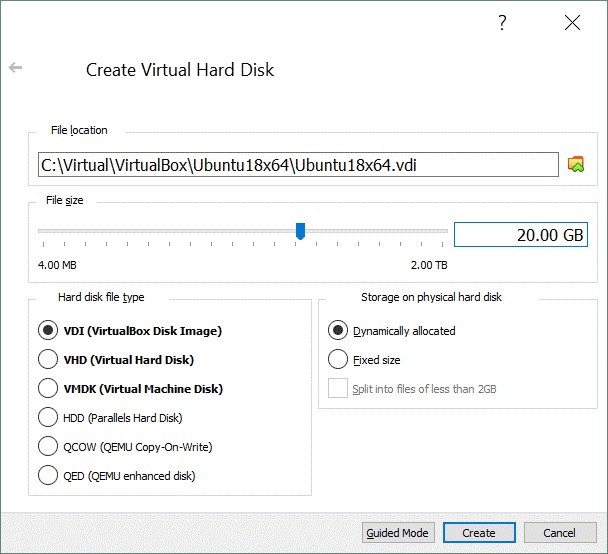
*C:\Virtual\VirtualBox\Ubuntu18x64\Ubuntu18x64.vdi*

The *file size* of the virtual disk: 20 GB or more.

*Hard disk file type*: VDI (VirtualBox Disk Image). Let’s select the native VirtualBox virtual disk format.

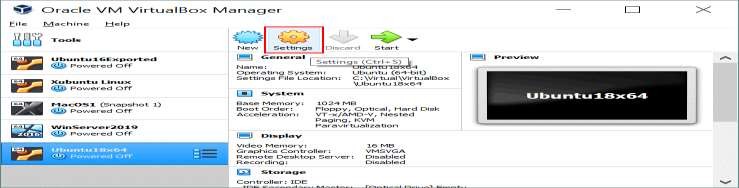
*Storage on physical hard disk*: Dynamically allocated. This option allows you to save space on your physical disk until the virtual disk grows to its maximum allocated size.

Hit *Create* to finish creating a new VM to install Ubuntu on VirtualBox.



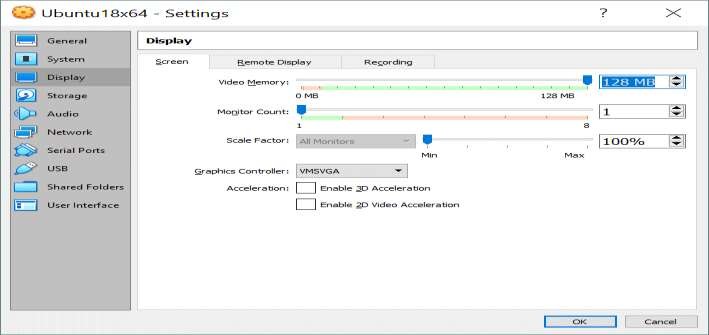
#### VM Configuration

A new virtual machine to install Ubuntu on VirtualBox has now been created and its name is displayed in the list of VMs in the main VirtualBox window. You need to edit VM settings after VM creation. Select your new VM (*Ubuntu18x64* in this case) and click *Settings* (*Machine > Settings* or press*Ctrl+S*).



In the *Settings* window, go to the *Display* section and select the *Screen* tab. Set video memory to 128 MB. Otherwise the Ubuntu installer may hang on some installation steps, keyboard may not response etc. You can enable 3D acceleration.

Hit *OK* to save settings.



#### Select the Boot Disk Image

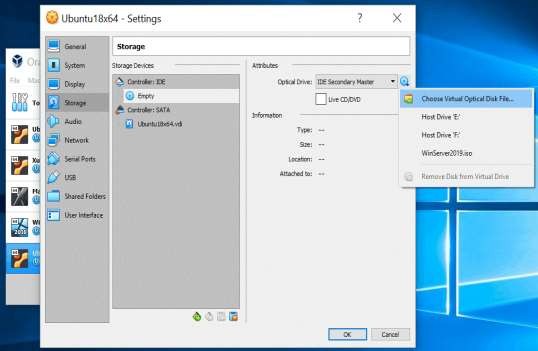
You don’t need to burn the ISO image onto a DVD disk as you would for installing an operating system on a physical machine. You can mount the ISO image to the virtual DVD drive of the virtual machine and boot a VM from this media.

Let’s insert the *ubuntu-18.04.2- desktop-amd64.iso* image that was downloaded from the official Ubuntu website before, into a virtual DVD drive of the *Ubuntu18x64* VM.

Open your VM settings and go to the *Storage* section. Select your virtual controller used for connecting a virtual DVD drive (by default a virtual DVD drive is empty). Click the *Empty* status and in the right pane near the *IDE Secondary Master*, click the disc icon.

In the menu that appears, click *Choose Virtual Optical Disk File* and browse your Ubuntu installation ISO image file (*ubuntu-18.04.2- desktop-amd64.iso*).

Hit *OK* to save settings. Now your VM is ready to install Ubuntu on VirtualBox.



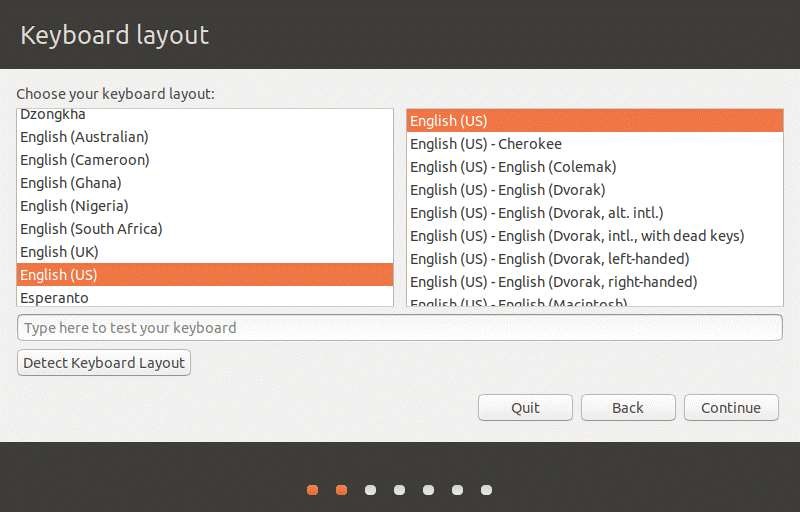
Install Ubuntu on VirtualBox VMs

Once the new VM is prepared for installing Ubuntu on VirtualBox, start the VM(*Machine> Start*). The VM boots from the ISO Ubuntu installation image. The first screen that you can see after booting is the Welcome screen.

In the left pane select *Language* for displaying information in the installer interface. English is selected in the current example. Then click *Install Ubuntu*.



**Keyboard layout**. Choose your keyboard layout. Let’s select English (US).



**Updates and other software**

There are a few options to choose from on this screen. *Normal installation*. A web browser, utilities, office applications and media players are installed.

*Minimal installation*. Only the main components including a web browser and basic utilities are installed.

Let’s select the normal installation. Other options:

*Download updates while installing Ubuntu*. The Ubuntu team is always working towards making Linux better. That’s why after downloading the installer, some updates may be already available.

You can automatically download and install updates right during Ubuntu installation, letting you save time after OS installation. Let’s select this option.

*Install third-party software for graphics and Wi-Fi hardware and additional media formats*. Tick this check box if you would like to install additional software, such as proprietary Wi-Fi drivers, video drivers, some TTF fonts etc.



**Installation type**

This screen contains options for preparing a disk for Ubuntu installation. *Erase disk and install Ubuntu*. This is the default option. All disk space will be automatically allocated to Ubuntu. If you select *Erase disk and install Ubuntu* on VirtualBox VMs, one big */dev/sda1* partition is created on */dev/sda*.

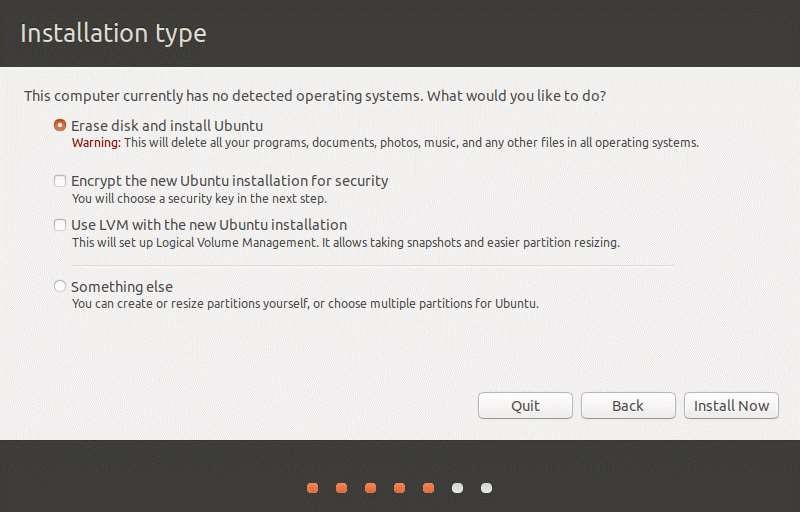
This */dev/sda1* partition with ext4 file system is mounted to the / directory (root directory), though a separate swap partition is not created. Attention: All data on the virtual disk will be erased there is no reason to worry about it, however, because an empty virtual disk created previously is being used for installing Ubuntu on VirtualBox.

There are some additional options:

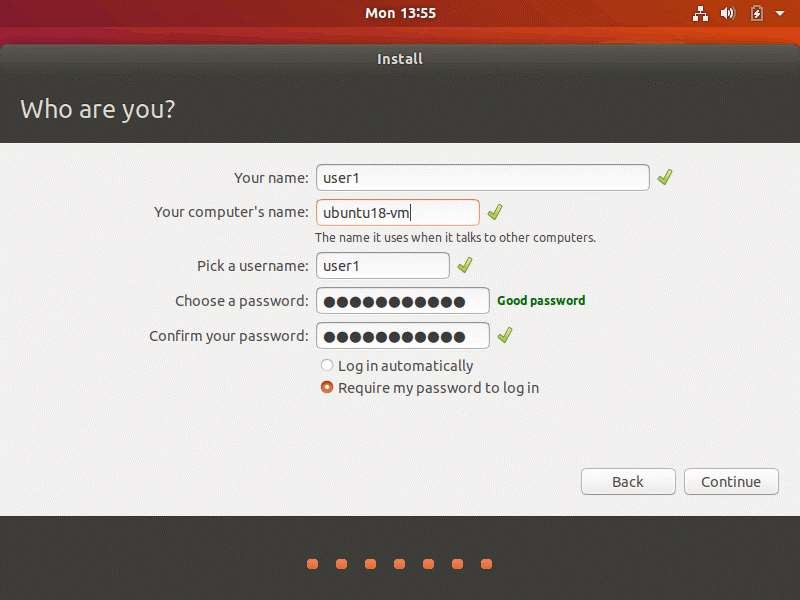
* Encrypt the new Ubuntu installation for security.
* Use LVM (Logical Volume Management) with the new Ubuntu installation.

*Something else*. Use this option for manual creation of the partition table on your virtual disk which is used to install Ubuntu on VirtualBox.

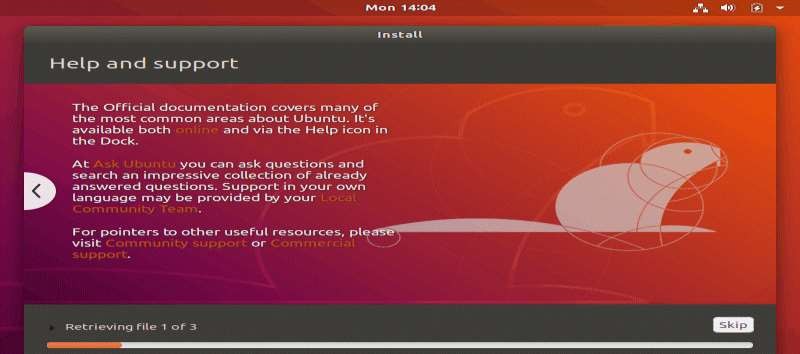
Click *Install Now* when you are ready to continue. Then on the confirmation screen, hit *Continue*.



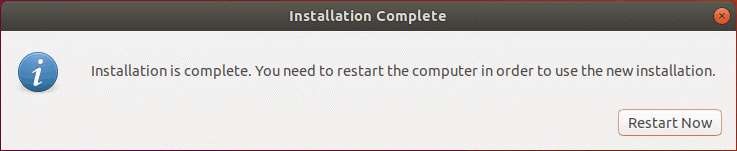
Enter your user name, computer’s name, and set the password. Select *Require my password to log in* for a higher level of security. In our example, the username is *user1* and the computer’s name is *ubuntu18-vm*.



As you can see, useful tips are displayed on the screen during the installation process.

When installation is complete, you will see a notification window. You have to restart your VM

with Ubuntu on VirtualBox.



|  |  |  |
| --- | --- | --- |
| Particulars | Marks Allotted | Marks Obtained |
| Performance | 50 |  |
| Viva | 10 |  |
| Record | 15 |  |
| Total | 75 |  |

**RESULT:**

Thus the installation of Virtual machine and a flavor of Linux OS is completed. This experiment belongs to CO1 and it is mapped with PO1, PO2, PO3, PO4, PO5, PO6, PSO1, PSO2 and PSO3.

|  |  |
| --- | --- |
| **Ex.No: 02** | **INSTALL A ‘C’ COMPILER IN A VIRTUAL MACHINE AND EXECUTE SIMPLE PROGRAMS** |
| **Date:** |

### AIM:

To install a C Compiler in a virtual machine created using virtual box and execute the

programs.

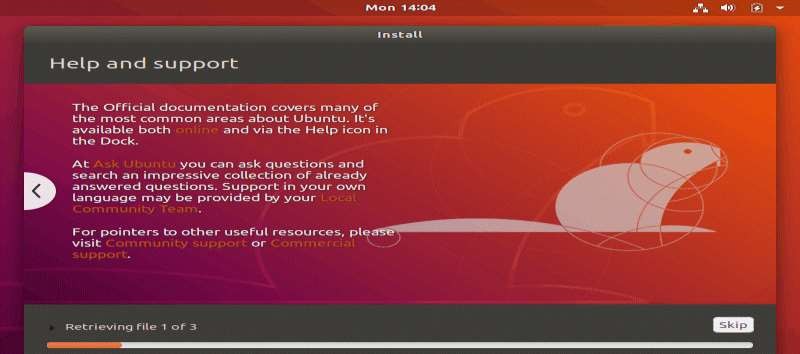
### PROCEDURE:

Install Ubuntu on VirtualBox VMs

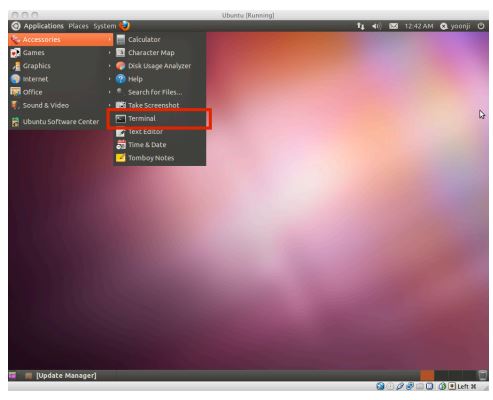
Once the new VM is prepared for installing Ubuntu on VirtualBox, start the VM (*Machine> Start*). The VM boots from the ISO Ubuntu installation image. The first screen that you can see after booting is the Welcome screen.

In the left pane select *Language* for displaying information in the installer interface. English is selected in the current example. Then click *Install Ubuntu*.

### 



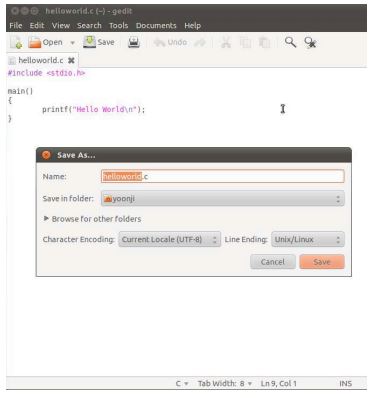
After installing ubuntu, open terminal and install gcc compiler

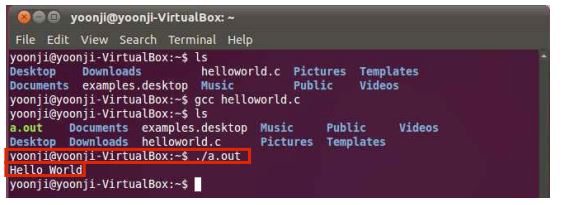


In terminal type

$ sudo apt install gcc

Then open one notepad and write one simple c program and save it as helloworld.c. Then execute the program as mentioned below.





|  |  |  |
| --- | --- | --- |
| Particulars | Marks Allotted | Marks Obtained |
| Performance | 50 |  |
| Viva | 10 |  |
| Record | 15 |  |
| Total | 75 |  |

##### **RESULT:**

Thus the GCC compiler has been successfully installed and executed a simple program. This experiment belongs to CO2 and it is mapped with PO1, PO2, PO3, PO4, PO5, PO6, PSO1, PSO2 andPSO3.

|  |  |
| --- | --- |
| **Ex.No:03** | **INSTALLING GOOGLE APP ENGINE AND CREATING HELLO WORLD APP** |
| **Date:** |

#### AIM:

To install the Google App Engine Software Development Kit (SDK) and running a simple “hello world” application using Python / Java.

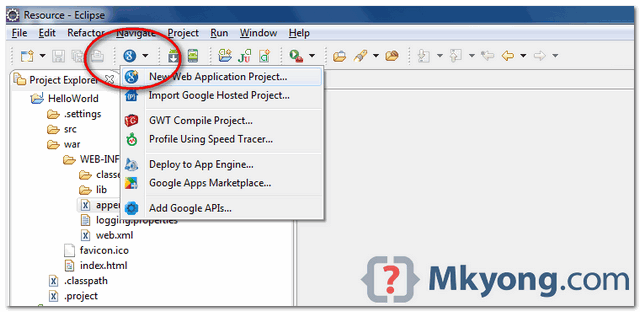
##### **PROCEDURE:**

Use **Eclipse** to create a **Google App Engine** (GAE) **Java** project (hello world example), run it locally, and deploy it to Google App Engine account.

**Tools used:**

1. JDK1.6
2. Eclipse3.7+GooglePluginforEclipse
3. Google App Engine Java SDK1.6.3.1
4. Install Google Plug in for Eclipse
5. Create New Web Application Project

In Eclipse toolbar, click on the Google icon, and select “NewWebApplication Project”



Figure–New Web Application Project

#### 

Figure – Deselect the “Google Web ToolKit“, and link your GAE Java SDK via the “configure SDK” link.

Click finished, Google Plug in for Eclipse will generate a sample project automatically.

1. HelloWorld

Review the generated project directory.

#### 

1. Run the local host

Access URL http://localhost:8888/, see output

#### 

#### 

|  |  |  |
| --- | --- | --- |
| Particulars | Marks Allotted | Marks Obtained |
| Performance | 50 |  |
| Viva | 10 |  |
| Record | 15 |  |
| Total | 75 |  |

#### RESULT:

Thus the Google App Engine is installed and executed the hello world application. This experiment belongs to CO2 and it is mapped with PO1, PO2, PO3, PO4, PO5, PO6, PSO1, PSO2 and PSO3.

|  |  |
| --- | --- |
| **Ex.No:04** | **LAUNCHING WEB APPLICATIONS USING**  **GAE LAUNCHER** |
| **Date:** |

#### AIM:

To launch the web application using Google Application Engine Launcher using Cloud SDK.

#### PROCEDURE:

Pre-Requisites: Python 2.5.4

If you don't already have Python 2.5.4 installed in your computer, download and Install Python 2.5.4 from:

<http://www.python.org/download/releases/2.5.4/>

## Download and Install

You can download the Google App Engine SDK by going to: <http://code.google.com/appengine/downloads.html>

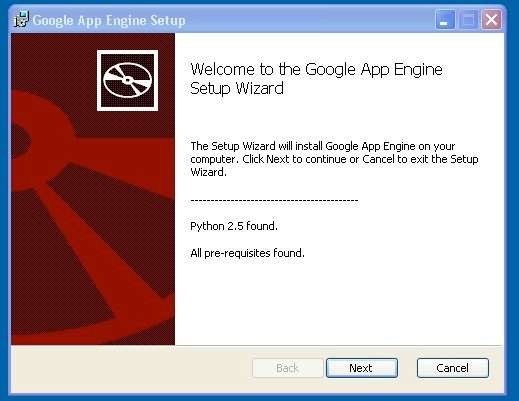
and download the appropriate install package.



Download the Windows installer – the simplest thing is to download it to your Desktop or another folder that you remember.



Double Click on the **Google Application Engine** installer.



Click through the installation wizard, and it should install the App Engine. If you do not have Python 2.5, it will install Python 2.5 as well.

Once the install is complete you can discard the downloaded installer



## **Making your First Application**

Now you need to create a simple application. We could use the “+” option to have the launcher make us an application but instead we will do it by hand to get a better sense of what is going on.

Make a folder for your Google App Engine applications. I am going to make the Folder on my Desktop called “**apps**” – the path to this folder is:

## C:\Documents and Settings\csev\Desktop\apps

And then make a sub-­‐folder in within **apps** called “**ae-01-trivial**” – the path to this folder would be:

## C:\ Documents and Settings \csev\Desktop\apps\ae-01-trivial

Using a text editor such as JEdit (www.jedit.org), create a file called **app.yaml** in the

**ae-01-trivial** folder with the following contents:

application: ae-01-trivial version: 1 runtime: python api\_version: 1

handlers:

- url: /.\*

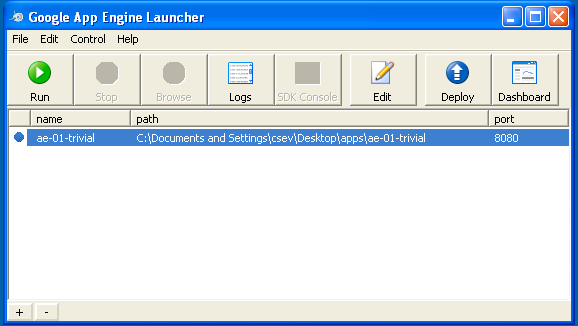
script: index.py

**Note:** Please do not copy and paste these lines into your text editor – you might end up with strange characters – simply type them into your editor.

Then create a file in the **ae-01-trivial** folder called **index.py** with three lines in it:

print 'Content-Type: text/plain' print ' ' print 'Hello there Chuck'

Then start the **Google App Engine Launcher** program that can be found under **Applications**. Use the **File -> Add Existing Application** command and navigate into the **apps** directory and select the **ae-01-trivial** folder. Once you have added the application, select it so that you can control the application using the launcher.



Once you have selected your application and press **Run**. After a few moments your application will start and the launcher will show a little green icon next to your application. Then press **Browse** to open a browser pointing at your application which is running at **http://localhost:8080/**

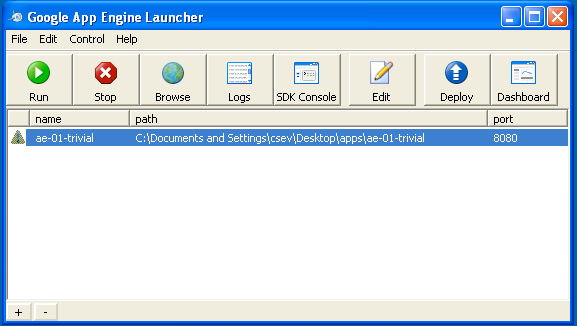
Paste **http://localhost:8080** into your browser and you should see your application as follows:



Just for fun, edit the **index.py** to change the name “Chuck” to your own name and press Refresh in the browser to verify your updates.

## **Dealing With Errors**

With two files to edit, there are two general categories of errors that you may encounter. If you make a mistake on the **app.yaml** file, the App Engine will not start and your launcher will show a yellow icon near your application:



## **Shutting Down the Server**

To shut down the server, use the Launcher, select your application and press the

**Stop** button.

**Alternate Method**

Download and install CloudSDK:

[Download the SDK](https://cloud.google.com/sdk/docs)

**Note**: If you already have the Cloud SDK installed, update it by running the following command:

gcloud components update

Create a new project:

Replace [YOUR\_PROJECT\_ID]with a [string of characters](https://cloud.google.com/sdk/gcloud/reference/projects/create#PROJECT_ID) that uniquely identifies your project. For example, my-project-24.

gcloud projects create [YOUR\_PROJECT\_ID] --set-as-default

Verify the project was created:

gcloud projects describe [YOUR\_PROJECT\_ID]

You see project details that might look like the following:

createTime: year-month-hour lifecycleState: ACTIVE name: project-name

parent:

id: '433637338589'

type: organization projectId: project-name-id

projectNumber: 499227785679

Initialize your App Engine app with your project and choose its region:

When prompted, select the [region](https://cloud.google.com/compute/docs/regions-zones) where you want your App Engine application located.

gcloud app create --project=[YOUR\_PROJECT\_ID]

**Caution**: You cannot change an app's region once it has been set.

Make sure billing is enabled for your project. A billing account needs to be linked to your project in order for the application to be deployed to App Engine.

[Enable billing](https://console.cloud.google.com/projectselector/billing?lang=python3&st=true)

**Note**: Running this sample app does not exceed your [free quota.](https://cloud.google.com/appengine/quotas)

Install the following prerequisites:

[Download and install Git](https://git-scm.com/).

Run the following command to install the [gcloud component](https://cloud.google.com/sdk/docs/managing-components) that includes the App Engine extension for Python3:

gcloud components install app-engine-python

Prepare your environment for Python development. It is recommended that you have the latest version of Python, pip, and other related tools installed on your system. For instructions, refer to the [Python Development Environment Setup Guide](https://cloud.google.com/python/setup).

This quick start demonstrates a simple Python app written with the [Flask](http://flask.pocoo.org/) web framework that can be deployed to App Engine. Although this sample uses Flask, you can use any web frame work that satisfies the requirements above. Alternative frameworks include [Django,](https://www.djangoproject.com/) [Pyramid,](http://www.pylonsproject.org/) [Bottle](http://bottlepy.org/), and[web.py.](http://webpy.org/)

# 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 an app to the Google Cloud.

1. 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](https://github.com/GoogleCloudPlatform/python-docs-samples/archive/master.zip) as a zip file and extract it.

1. Change to the directory that contains the sample code.

cd python-docs-samples/appengine/standard\_python3/hello\_world

# Run Hello World on your local machine

To run the Hello World app on your local computer:

[Mac OS / LinuxWindows](https://cloud.google.com/appengine/docs/standard/python3/quickstart#mac-os--linux)

1. Create an isolated Python environment in a directory external to your project and activate it:

python3 -m venv env source env/bin/activate

1. Navigate to your project directory and install dependencies:

cd***YOUR\_PROJECT***

pip install -r requirements.txt

1. Run the application:

pythonmain.py

1. In your web browser, enter the following address:

http://localhost:8080

**Note:** If you are using Cloud Shell, in the toolbar, click **Web Preview** and select **Preview on port 8080** instead.

The **Hello World** message from the sample app displays on the page. In your terminal window, press **Ctrl+C** to exit the web server.

# Deploy and run Hello World on App Engine

To deploy your app to the App Engine standard environment:

1. Deploy the Hello World app by running the following command

from the standard\_python3/hello\_world directory:

gcloud appdeploy

Learn about the [optional flags.](https://cloud.google.com/appengine/docs/standard/python3/quickstart#deploy_and_run_hello_world_on_app_engine)

1. Launch your browser to view the app athttps://***PROJECT\_ID***[.***REGION\_ID***](https://cloud.google.com/appengine/docs/standard/python3/quickstart#appengine-urls).r.appspot.com

gcloud app browse

where ***PROJECT\_ID***represents your Google Cloud project ID.

This time, the page that displays the Hello World message is delivered by a web server running on an App Engine instance.

**Congratulations!** You've deployed your first Python 3 app to App Engine standard environment!

See the following sections for information about cleaning up as well as links to possible next steps that you can take.

# Clean up

To avoid incurring charges, you can delete your Cloud project to stop billing for all the resources used within that project.

**Caution**: Deleting a project has the following effects:

**Everything in the project is deleted.** If you used an existing project for this tutorial, whenyou delete it, you also delete any other work you've done in theproject.

**Custom project IDs are lost.** When you created this project, you might have created a custom project ID that you want to use in the future. To preserve the URLs that use the project ID, such as an app spot.com URL, delete selected resources inside the project instead of deleting the whole project.

In the Cloud Console, go to the **Manage resources** page. [Go to the Manage resources page](https://console.cloud.google.com/iam-admin/projects)

In the project list, select the project that you want to delete and then click **Delete**.

In the dialog, type the project ID and then click **Shut down** to delete the project.

|  |  |  |
| --- | --- | --- |
| Particulars | Marks Allotted | Marks Obtained |
| Performance | 50 |  |
| Viva | 10 |  |
| Record | 15 |  |
| Total | 75 |  |

**RESULT :**

Thus the GAE launcher used to launch the web applications successfully. This experiment belongs to

CO2 and it is mapped with PO1, PO2, PO3, PO4, PO5, PO6, PSO1, PSO2 and PSO3

|  |  |
| --- | --- |
| **Ex.No:05** | **SIMULATE A CLOUD SCENARIO USING CLOUDSIM AND RUN A SCHEDULING ALGORITHM THAT IS NOT PRESENT IN CLOUDSIM** |
| **Date:** |

#### AIM:

To simulate a cloud scenario using CloudSim and run a scheduling algorithm that is not present in CloudSim.

#### PROCEDURE:

The Above Problem Requires the Following

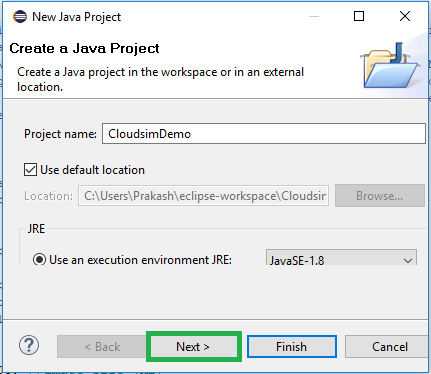
Eclipse IDE

CloudsimPackage

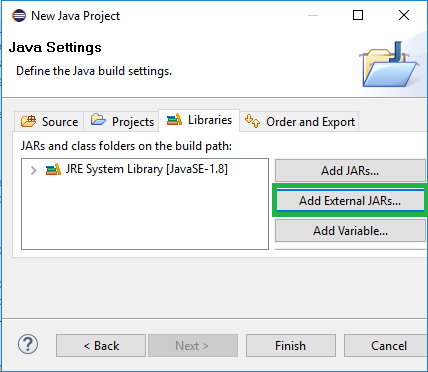
**Step1:**

Open the Eclipse IDE Create New JavaProject

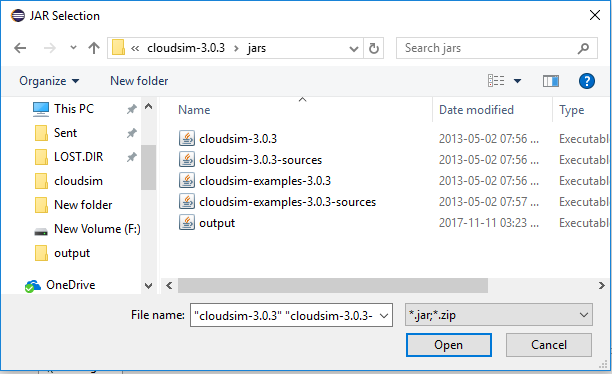
Give Name As “CloudsimDemo” and ClickNext



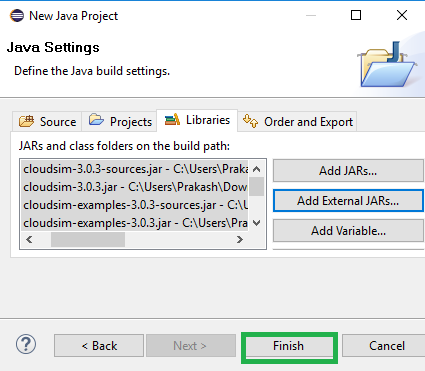
Goto Library and Click Add external JarsAnd



Browse to cloudsim Folder add All Jar Files



Click Finish



**Step2:**

* Create New Class Called “First” And Write the Following JavaCode
* This Java Code Consist For Creating Data Center , Data Center Broker ,VMs, Cloudlets
* The Space Shared Policy Is Used

**First.java**

**Import** java.io.File;

**Import** java.io.FileNotFoundException; **import** java.io.FileOutputStream;

**Import** java.io.IOException;

**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.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;

public class First {

/\*\* The cloudlet list. \*/

private static List<Cloudlet>cloudletList;

/\*\* The vmlist. \*/

private static List<Vm>vmlist;

private static List<Vm>createVM(intuserId, intvms, intidShift) {

//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) intmips = 250;

long bw = 1000;

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

//create VMs

Vm[] vm = new Vm[vms];

for(inti=0;i<vms;i++){

vm[i] = new Vm(idShift + i, userId, mips, pesNumber, ram, bw, size, vmm, new CloudletSchedulerSpaceShared());

list.add(vm[i]);}

return list;

}

private static List<Cloudlet>createCloudlet(intuserId, int cloudlets, intidShift){

// Creates a container to store Cloudlets LinkedList<Cloudlet> list = new LinkedList<Cloudlet>();

//cloudlet parameters long length = 40000; long fileSize = 300; long outputSize = 300; intpesNumber = 1;

UtilizationModelutilizationModel = new UtilizationModelFull(); Cloudlet[] cloudlet = new Cloudlet[cloudlets]; for(inti=0;i<cloudlets;i++){

cloudlet[i] = new Cloudlet(idShift + i, length, 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 Clouddemo...");

try {

// First step: Initialize the CloudSim package. It should be called

// before creating any entities. intnum\_user = 2;

// number ofusers

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");

//Third step: Create Broker

DatacenterBroker broker = createBroker("Broker\_0"); intbrokerId = broker.getId();

//Fourth step: Create VMs and Cloudlets and send them to broker vmlist = createVM(brokerId, 2, 0); //creating 2 vms

cloudletList = createCloudlet(brokerId, 8, 0); // creating 8 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); Log.printLine("Clouddemo finished!");

}

catch (Exception e)

{e.printStackTrace();

Log.printLine("The simulation has been terminated due to an unexpected error");

}

}

private static Datacenter createDatacenter(Stringname){

// Here are the steps needed to create aPowerDatacenter:

// 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>(); intmips =1000;

// 3. Create PEs and add these into the list.

//for a dual-core machine, a list of 2 PEs is required:

peList1.add(new Pe(0, new PeProvisionerSimple(mips))); // need to store Pe id and MIPS peList1.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 inthostId=0;

int ram = host storage intbw = 10000;

new Host( hostId,

new RamProvisionerSimple(ram), new BwProvisionerSimple(bw), storage, peList1,

new VmSchedulerSpaceShared(peList1))

);

// 5. Create a DatacenterCharacteristics object that stores the properties of a data center: architecture, //OS, list ofMachines, allocation policy:

//time- or space-shared, time zoneand its price (G$/Pe time unit).

String arch = "x86"; // system architecture

String os = "Linux"; // operating system

String vmm = "Xen";

double time\_zone =10.0; // time zone this resource located

doublecost=3.0; // the cost of using processing in this resource

double costPerMem = 0.05; // the cost of using memory in this resource

doublecostPerStorage = 0.1; // the cost of using storage in this resource

double costPerBw=0.1; // the cost of using bw in this resource

LinkedList<Storage>storageList =newLinkedList<Storage>();

//we are not adding SAN devices bynow

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;

}

//to the specific rules of the simulated scenario

private static DatacenterBrokercreateBroker(String name){

DatacenterBroker broker = null; try {

broker = new DatacenterBroker(name);

} catch (Exception e) { e.printStackTrace(); return null;

}

return broker;

}

/\*\*

* Prints the Cloudletobjects
* @paramlist list ofCloudlets

\*/

private static void printCloudletList(List<Cloudlet> list) { int size = list.size();

Cloudlet cloudlet;

String indent=" "; Log.printLine(); Log.printLine("========== OUTPUT==========");

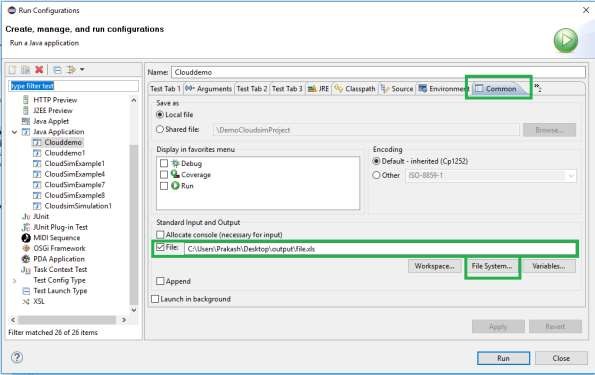
Log.printLine("Cloudlet ID" + indent + "STATUS" + indent +

"Data center ID" + indent + "VM ID" + indent + indent + "Time" + indent + "Start Time" + indent + "Finish Time");

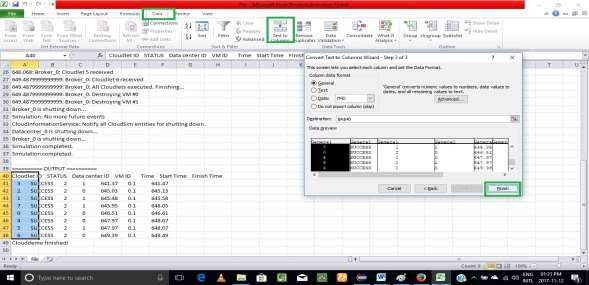
}

**Step3:**

* Change the RunConfiguration
* Go to Common , In the Standard Input and OutputSection
* Check File Checkbox And Add the file Path Using TheFileSystem
* Let The Path Be “C:\Users\Prakash\Desktop\output/file.xls” And ClickRun



**Step 4:**



* Open The Output File And Convert Text To Columns
* Plot The Graph Time Versus Cloudlets
* Change the Time shared policy to Space shared And obtain the graph
* Compare The Results Of The Timeshared And Space Shared

|  |  |  |
| --- | --- | --- |
| Particulars | Marks Allotted | Marks Obtained |
| Performance | 50 |  |
| Viva | 10 |  |
| Record | 15 |  |
| Total | 75 |  |

**RESULT:**

Thus the scheduling task in cloud sim is executed successfully. This experiment belongs to CO4 and it is mapped with PO1, PO2, PO3, PO4, PO5, PO6, PSO1, PSO2 and PSO3.

|  |  |
| --- | --- |
| **Ex.No: 06** | [**TRANSFER FILES FROM ONE VIRTUAL MACHINES TO**](https://www.wintips.org/how-to-transfer-virtualbox-virtual-machines-to-another-host-easily/) **ANOTHER VIRTUAL MACHINE** |
| **Date:** |

#### AIM:

#### To Find a procedure to transfer the files from one virtual machine to another virtual machine.

#### PROCEDURE:

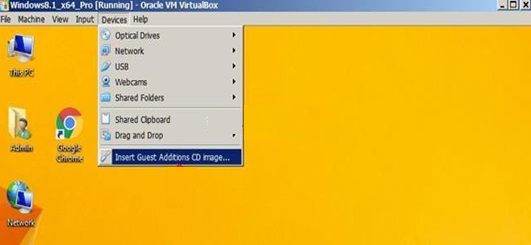
Step1.InstallGuestAdditionsontheGuestmachine.

Step2. Configure File Sharing on VirtualBox.

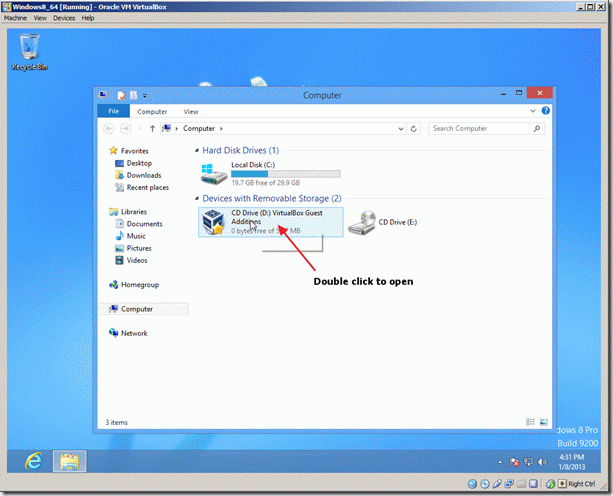
###### **Step 1**

###### Install Guest Additions on the Guest machine.

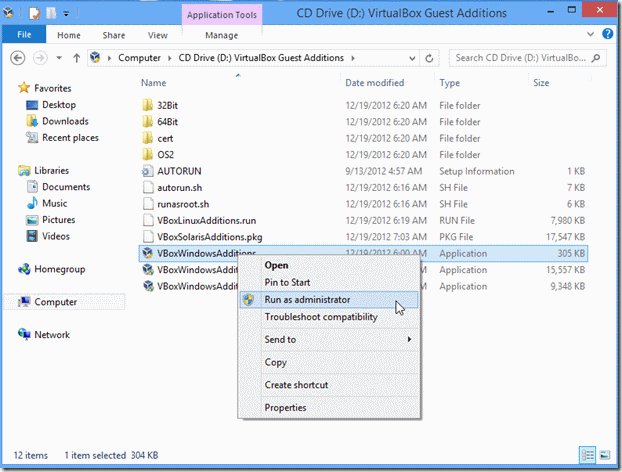
1. Start the Virtual box Guest Machine(OS).
2. From Oracle's VM VirtualBox main menu, select **Devices**>**Install Guest Additions**



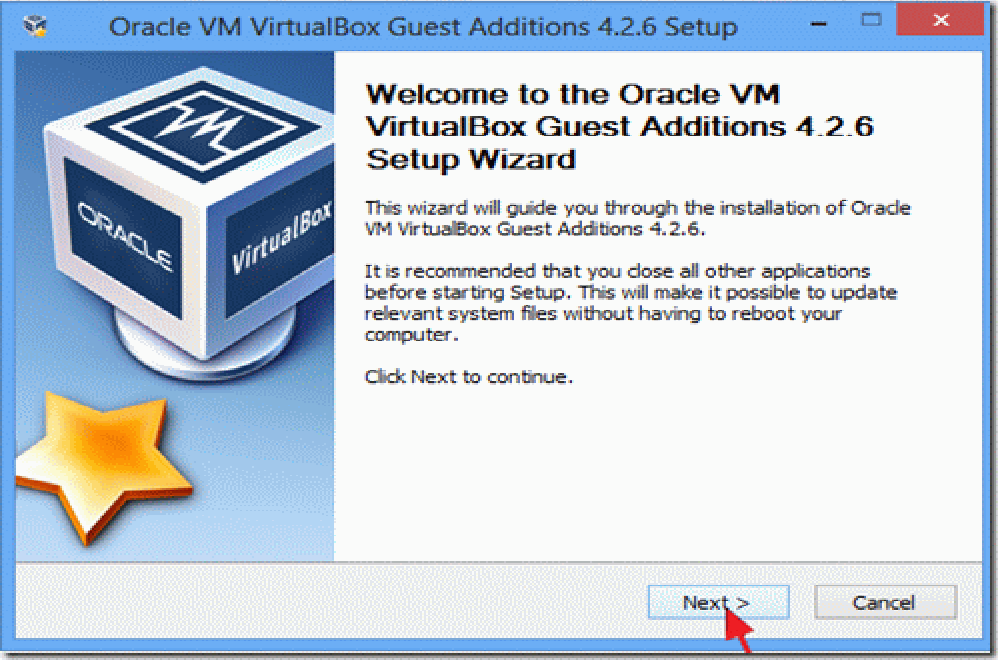
1. Open Windows Explorer.
2. Double click at the "CD Drive(X:) VirtualBox Guest additions" to explore its contents.



1. Right click at "VBox Windows Additions" application and from the pop-up menu, choose "**Run as administrator**".



1. Press **Next** and then follow the on screen instructions to complete the Guest Additions installation.

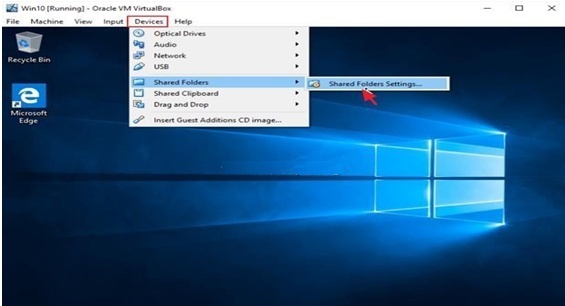


When the setup is completed, choose **Finish** and **restart** the Virtual box guest machine.

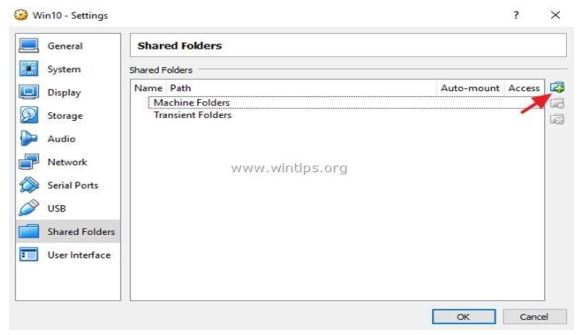
###### **Step2.**

###### Setup File Sharing on VirtualBox Guest Machine.

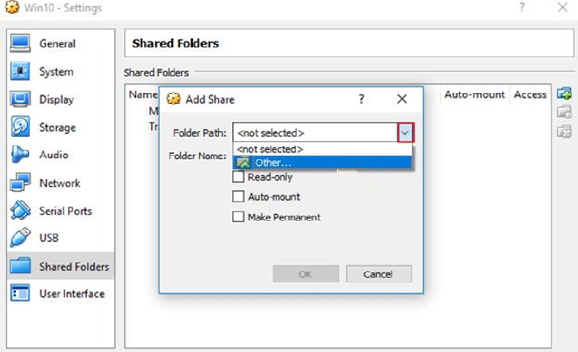
From VirtualBox menu click **Devices** and choose **SharedFolders**->**Shared Folder Settings**



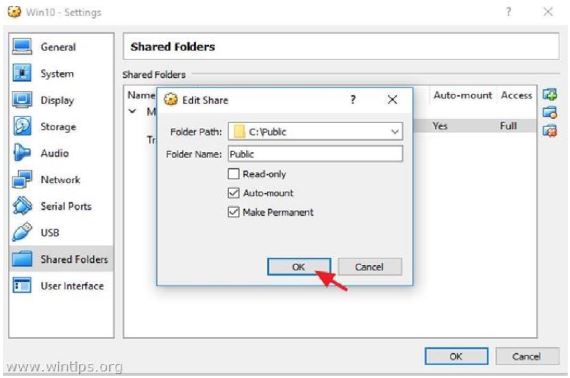
Click the **Add new shared folder** icon.



Click the drop-down arrow and select **Other**.



Now, in the 'Add Share' options, type a name (if you want) at the 'Folder Name box, click the **Auto Mount** and the **Make Permanent** checkboxes and click **OK** twice to close the Shared Folder Settings



You 're done! To access the shared folder from the Guest OS, open Windows Explorer and under the 'Network locations' you should see a new network drive that corresponds to the shared folder on the Host OS.

|  |  |  |
| --- | --- | --- |
| Particulars | Marks Allotted | Marks Obtained |
| Performance | 50 |  |
| Viva | 10 |  |
| Record | 15 |  |
| Total | 75 |  |

**RESULT:**

Thus the transfer of virtual machine from one host to another host was successfully completed. This experiment belongs to CO4 and it is mapped with PO1, PO2, PO3, PO4, PO5, PO6, PSO1, PSO2 and PSO3.

|  |  |
| --- | --- |
|  | **INSTALL SINGLE NODE HADOOP CLUSTER**  **AND EXECUTE THE WORD COUNT APPLICATION** |
| **Date :** |

**AIM:**

To install the single node Hadoop cluster on the physical machine and execute the word count application.

**PROCEDURE:**

Hadoop pseudo node setup created for following environment:

Ubuntu Linux 64-bit JDK 1.8.0\_05

Hadoop 2.7.x stable release

***Prerequisites:***

* Installing Java v1.8
* Configuring SSHaccess.

**sudo apt-get install vim**

**1) Installing Java:**

Hadoop is a framework written in Java for running applications on large clusters of

commodity hardware. Hadoop needs Java 6 or above to work.

Step 1:

Download Jdk tar.gz file for linux-62 bit, extract it into

“/usr/local” boss@solaiv[]# cd /opt

boss@solaiv[]# sudo tar xvpzf /home/itadmin/Downloads/jdk-8u5-linux-

x64.tar.gz boss@solaiv[]# cd /opt/jdk1.8.0\_05

Step 2:

Open the “/etc/profile” file and Add the following line as per the version set a environment for Java

Use the root user to save the /etc/proflie or use gedit instead of vi .

The 'profile' file contains commands that ought to be run for login shells

boss@solaiv[]# sudo vi /etc/profile

#--insert JAVA\_HOME JAVA\_HOME=/opt/jdk1.8.0\_05

#--in PATH variable just append at the end of the line

PATH=$PATH:$JAVA \_HOME/bin

#--Append JAVA\_HOME at end of the exportstatement export PATH JAVA\_HOME

save the file using by pressing “Esc” key followed by :wq!

Step 3: Update the java alternatives

By default OS will have a open jdk. Check by “java -version”. You will be prompt “openJDK”

If you also have openjdk installed then you'll need to update the java alternatives:

If your system has more than one version of Java, configure which one your system causes by entering the following command in a terminal window

By default OS will have a open jdk. Check by “java -version”. You will be prompt “Java HotSpot(TM) 64-Bit Server”

boss@solaiv[]# update-alternatives --install "/usr/bin/java" java "/opt/jdk1.8.0\_05/bin/java" 1

boss@solaiv[]# update- alternatives --config java --type selection number:

boss@solaiv[]# java –version

**2) configure ssh**

HadooprequiresSSHaccesstomanageitsnodes,i.e.remotemachinesplusyourlocal machine if you want to use Hadoop on it (which is what we want to do in this short tutorial). For our single-node setup of Hadoop, we therefore need to configure SSH access to local host. The need to create a Password-less SSH Key generation based authentication is so that the master node can then login to slave nodes (and the secondary node) to start/stop them easily without any delays for authentication.

If you skip this step, then have to provide password Generate an SSH key for the user. Then Enable password-less SSH access to you

#### sudo apt-get install openssh-server

You will be asked to enter password, root@solaiv[]# ssh localhost

root@solaiv[]# ssh-keygen root@solaiv[]# ssh-copy-id

-i localhost

After above 2 steps, You will be connected without password, root@solaiv[]# ssh localhost

root@solaiv[]# exit

**3) Hadoop installation**

Now Download Hadoop from the official Apache, preferably a stable release version of Hadoop 2.7.x and extract the contents of the Hadoop package to a location of your choice.

We chose location as “/opt/”

Step 1: Download the tar.gz file of latest version Hadoop ( hadoop-2.7.x) from the official site .

Step 2: Extract(untar) the downloaded file from this commands to /opt/bigdata

root@solaiv[]# cd /opt

root@solaiv[/opt]# sudo tar xvpzf /home/itadmin/Downloads/hadoop- 2.7.0.tar.gz root@solaiv[/opt]# cd hadoop-2.7.0/

Like java, update Hadop environment variable in /etc/profile boss@solaiv[]# sudo vi /etc/profile

#--insert HADOOP\_PREFIX

HADOOP\_PREFIX=/opt/ha doop-2.7.0

#--in PATH variable just append at the end of the line PATH=$PATH:$HADOOP\_PREFIX/bin

#--Append HADOOP\_PREFIX at end of the export statement export PATH JAVA\_HOME HADOOP\_PREFIX

save the file using by pressing “Esc” key followed by :wq!

Step 3: Sourcethe

/etc/profile boss@solaiv[]# source/etc/profile

Verify Hadoop installation boss@solaiv[]# cd $HADOOP\_PREFIX boss@solaiv[]# bin/hadoop version

**Modify the Hadoop Configuration Files**

In this section, we will configure the directory where Hadoop will store its configuration files, the network ports it listens to, etc. Our setup will use Hadoop Distributed File System,(HDFS), even though we are using only a single local machine.

Add the following properties in the various hadoop configuration files which is available under $HADOOP\_PREFIX/etc/hadoop/

core-site.xml, hdfs-site.xml, mapred-site.xml & yarn-site.xml Update Java, hadoop path to the Hadoop environment file

boss@solaiv[]# cd $HADOOP\_PREFIX/etc/hadoop boss@solaiv[]# vi hadoop-env.sh

Paste following line at beginning of the file

export JAVA\_HOME=/usr/local/jdk1.8.0\_05 export HADOOP\_PREFIX=/opt/hadoop-2.7.0

Modify the **core-site.xml**

boss@solaiv[]# cd $HADOOP\_PREFIX/etc/hadoop boss@solaiv[]# vi core-site.xml

Paste following between <configuration> tags

#### <configuration>

<property>

<name>fs.defaultFS</name>

<value>hdfs://localhost:9000</value>

</property>

#### </configuration>

Modify the **hdfs-site.xml**

boss@solaiv[]# vi hdfs-site.xml

Paste following between <configuration> tags

#### <configuration>

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

#### </configuration>

YARN configuration - Single Node modify the **mapred- site.xml**

boss@solaiv[]# cp mapred-site.xml.template mapred-site.xml boss@solaiv[]# vi mapred-site.xml

Paste following between <configuration> tags

#### <configuration>

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

#### </configuration>

Modiy yarn-site.xml boss@solaiv[]# vi yarn-site.xml

Paste following between <configuration> tags

#### <configuration>

<property><name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuf fle</value></property>

#### </configuration>

Formatting the HDFS file-system via the NameNode

The first step to starting up your Hadoop installation is formatting the Hadoop files system which is implemented on top of the local file system of our “cluster” which includesonlyourlocalmachine.WeneedtodothisthefirsttimeyousetupaHadoop cluster.

Do not format a running Hadoop file system as you will lose all the data currently in the cluster (in HDFS)

root@solaiv[]# cd

$HADOOP\_PREFIX

root@solaiv[]# bin/hadoop namenode –format

Start NameNode daemon and DataNode daemon: (port 50070) root@solaiv[]# sbin/start-dfs.sh

To know the running daemons jut type jps or

usr/local/jdk1.8.0\_05/bin/jps Start ResourceManager daemon and NodeManager daemon: (port 8088)

root@solaiv[]# sbin/start-yarn.sh

To stop the running process root@solaiv[]# sbin/stop-dfs.sh

To know the running daemons jut type jps or

/usr/local/jdk1.8.0\_05/bin/jps Start ResourceManager daemon and NodeManager daemon: (port 8088)

root@solaiv[]# sbin/stop-yarn.sh

Make the HDFS directories required to execute MapReduce jobs:

$ bin/hdfs dfs -mkdir /user

$ bin/hdfs dfs -mkdir /user/mit

Copy the input files into the distributed filesystem:

$ bin/hdfs dfs -put <input-path>/\* /input

Run some of the examples provided:

$ bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.5.1.jar grep /input /output '(CSE)

Examine the output files:

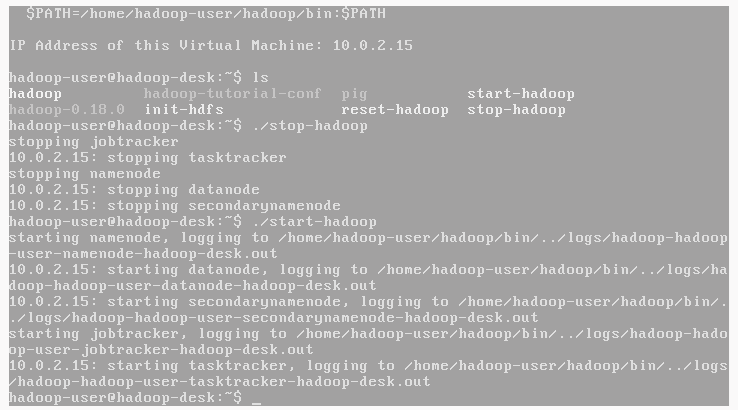
Copy the output files from the distributed filesystem to the local filesystem and examine them:

$ bin/hdfs dfs -get output output $ cat output/\*

or

View the output files on the distributed filesystem: $ bin/hdfs dfs -cat /output/\*

To check the installation of Hadoop is completed. Check with the following commands.



**Word count program to demonstrate the use of Map and Reduce tasks**

##### **Procedure:**

1. Format the path.
2. Start the dfs and check the no. of nodes running.
3. Start the yarn and check the no. of nodes running.
4. Open the browser and check whether the hadoop is installed correctly.
5. Add a file and check whether we can view the file.
6. Implement the grep command for the file added and see the result.
7. Implement the word count command for the file added and see the result.
8. After completing the process stop dfs and yarn properly.

##### **Commands:**

Install the hadoop cluster by using the commands

1. $sudochown–Rhgee.gee/opt/
2. $nano yarn–site.xml

<configuration>

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

</property>

</configuration>

1. $cd$Hadoop\_prefix
2. $bin/Hadoopnamenode\_format
3. $s.bin/start\_dfs.sh
4. Ips

##### **PROGRAM:**

Package hadoop; import java.util.\* ;

import java.io.IOException;

importjava.io.IOException;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.conf.\* ;

import org.apache.hadoop.io.\* ;

import org.apache.hadoop.m apred.\* ;

import org.apache.hadoop.util.\* ;

public class ProcessUnits

{

public static class E\_EMapper extends MapReduceBaseimplements Mapper<LongWritable Text,

Text, IntWritable>

{

public void m ap(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException

{

String line = value.toString(); String lasttoken = null;

StringTokenizer s = new StringTokenizer(line,"\t"); String year = s.nextToken(); while(s.hasMoreTokens())

{

lasttoken=s.nextToken();

}

intavgprice = Integer.parseInt(lasttoken); output.collect(new Text(year), new IntWritable(avgprice));

}

}

public static class E\_EReduce extends MapReduceBaseimplements Reducer< Text, IntWritable, Text, IntWritable>

{

public void reduce( Text key, Iterator <IntWritable> values, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException

{

int m axavg=30; intval=Integer.MIN\_VALUE; while (values.hasNext())

{

if((val=values.next().get())>m axavg)

{

output.collect(key, new IntWritable(val));

}

}

}

}

public static void m ain(String args[])throws Exception

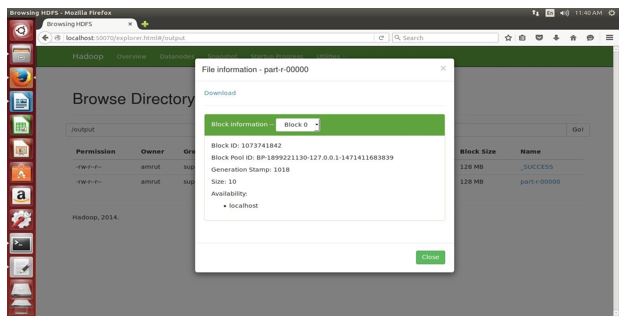
{

JobConfconf = new JobConf(Eleunits.class); conf.setJobNam e("m ax\_eletricityunits"); conf.setOutputKeyClass(Text.class); conf.setOutputValueClass(IntWritable.class); conf.setMapperClass(E\_EMapper.class); conf.setCombinerClass(E\_EReduce.class); conf.setReducerClass(E\_EReduce.class); conf.setInputForm at(TextInputFormat.class); conf.setOutputForm at(TextOutputFormat.class); FileInputFormat.setInputPaths(conf, new Path(args[0])); FileOutputFormat.setOutputPath(conf, new Path(args[1])); JobClient.runJob(conf);

}

}

##### **OUTPUT:**



|  |  |  |
| --- | --- | --- |
| Particulars | Marks Allotted | Marks Obtained |
| Performance | 50 |  |
| Viva | 10 |  |
| Record | 15 |  |
| Total | 75 |  |

**RESULT:**

Thus the installation of single node Hadoop cluster and executing the word count application is successfully completed. This experiment belongs to CO5 and it is mapped with PO1, PO2, PO3, PO4, PO5, PO6, PSO1, PSO2 and PSO3.

|  |  |
| --- | --- |
| **Ex.No:8** | **CREATING AND EXECUTING YOUR FIRST CONTAINER USING DOCKER** |
| **Date :** |