****

**DEDEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**LAB MANUAL**



**Regulation 2021**

**CCS335 – CLOUD COMPUTING LABORATORY**

**Year / Semester : V / III**

**PREPARED BY**

**Mrs. R. ARUNADEVI, M.E.,**

**Assistant Professor / CSE**

**OBJECTIVES:**

* To understand the principles of cloud architecture, models and infrastructure.
* To understand the concepts of virtualization and virtual machines.
* To gain knowledge about virtualization Infrastructure.
* To explore and experiment with various Cloud deployment environments.
* To learn about the security issues in the cloud environment.

**OUTCOMES:**

**On completion of this course, the students will be able to:**

**CO1:** Understand the design challenges in the cloud.

**CO2:** Apply the concept of virtualization and its types.

**CO3:** Experiment with virtualization of hardware resources and Docker.

**CO4:** Develop and deploy services on the cloud and set up a cloud environment.

**CO5:** Explain security challenges in the cloud environment.

**LIST OF EXPERIMENTS**

1. Install Virtualbox/VMware/ Equivalent open source cloud Workstation with different flavours of Linux or Windows OS on top of windows 8 and above.
2. Install a C compiler in the virtual machine created using a virtual box and execute Simple Programs.
3. Install Google App Engine. Create a hello world app and other simple web applications using python/java.
4. Use the GAE launcher to launch the web applications.
5. Simulate a cloud scenario using CloudSim and run a scheduling algorithm that is not present in CloudSim.
6. Find a procedure to transfer the files from one virtual machine to another virtual machine.
7. Install Hadoop single node cluster and run simple applications like wordcount.
8. Creating and Executing Your First Container Using Docker.
9. Run a Container from Docker Hub.

**EX NO. : 1**

**Install Virtualbox / VMware Workstation with different flavours of linux or windows OS on top of windows7 or 8.**

**Aim:**

To Install Virtualbox/VMware/ Equivalent open source cloud Workstation with different flavours of Linux or Windows OS on top of windows 8 and above.

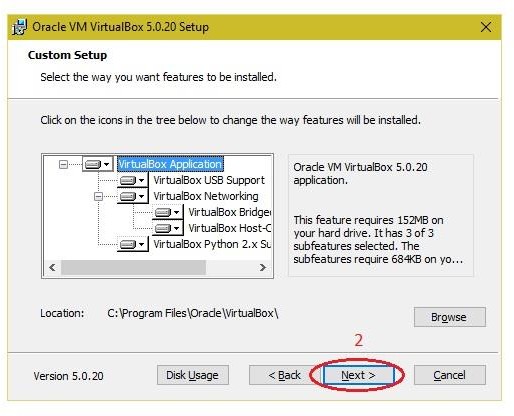
# PROCEDURE:

**Steps to install Virtual Box:**

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



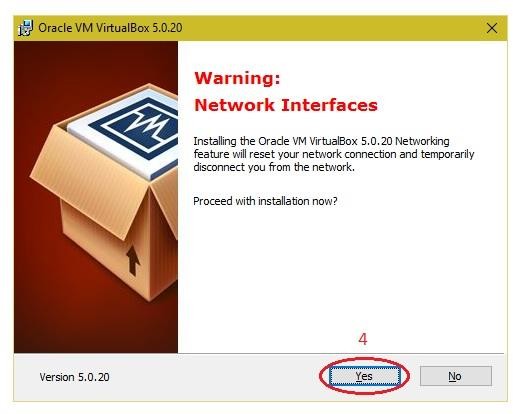
1. Click the next button.



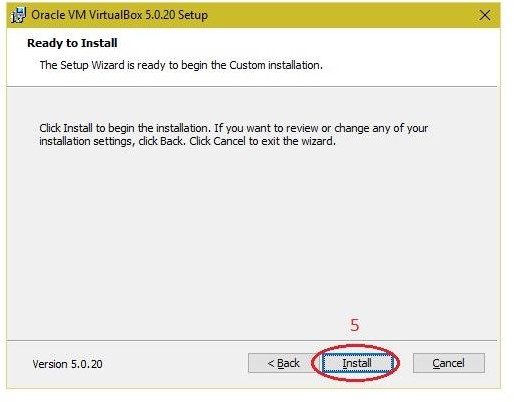
1. Click the next button.



1. Click the YES button.



1. Click the install button…

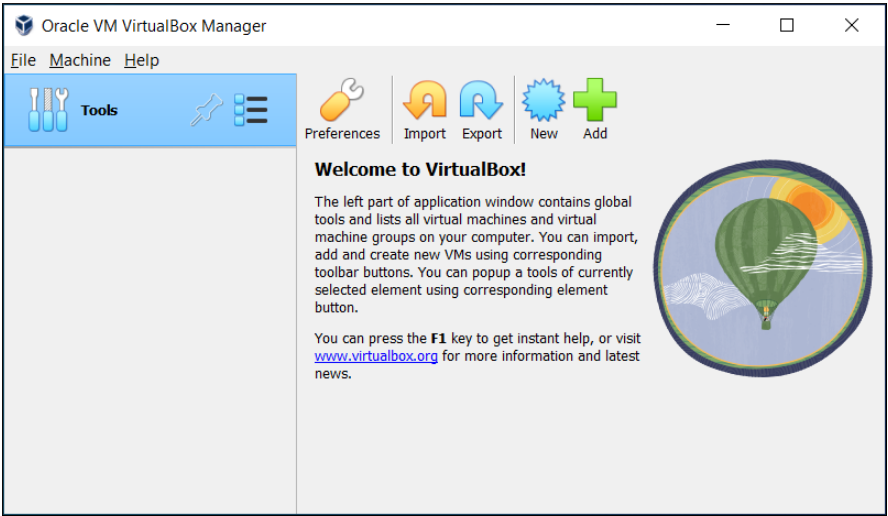
****

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



**Steps to Create Virtual Machine and Install LINUX OS on it:**

* 1. VirtualBox Manager, Showing Welcome Screen After Initial Startup



The above Virtual Box Manager might look before you have created some virtual machines.

* 1. VirtualBox Manager Window shows, After Creating Virtual Machines: The main

components of the VirtualBox Manager window are as follows:

**The machine list.** The left pane of the **VirtualBox Manager** window lists all your virtual machines. If you have not yet created any virtual machines, this list is empty.

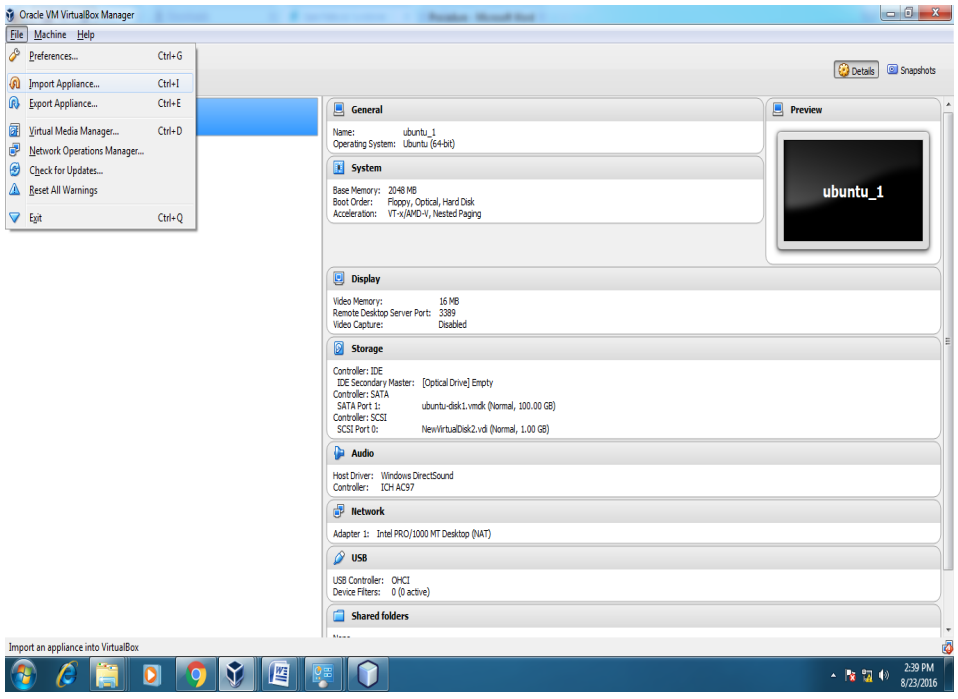
**The Details pane.** The pane on the right displays the properties of the currently selected virtual machine. If you do not have any machines yet, the pane displays a welcome message. The toolbar buttons on the Details pane can be used to create and work with virtual machines.

**Help Viewer.** A window that displays context-sensitive help topics for VirtualBox Manager Tasks.



**VirtualBox Manager Details Pane, Including Toolbar**

****

****

#### VirtualBox Manager Toolbar

**A toolbar at the top of the Details pane contains buttons that enable you to configure the selected virtual machine, or to create a new virtual machine.**

**The toolbar includes the following buttons:**

**New. Creates a new virtual machine, and adds it to the machine list.**

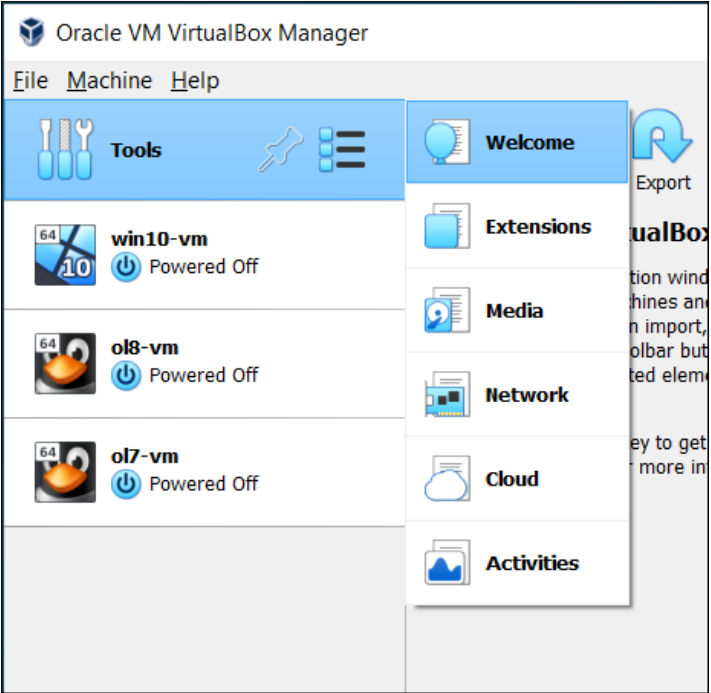
**Add. Adds an existing virtual machine to the machine list.**

**Settings. Displays the Settings window for the virtual machine, enabling you to make configuration changes.**

**Discard. For a running virtual machine, discards the saved state for the virtual machine and closes it down.**

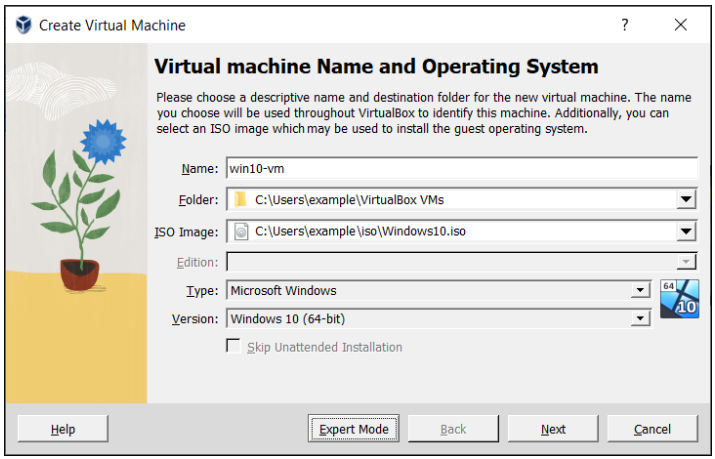
**Show/Start. For a running virtual machine, Show displays the virtual machine window. For a stopped virtual machine, Start displays options for powering up the virtual machine.**

**Global Tools Menu**

****

* 1. Creating Your First Virtual Machine:

1. Create Virtual Machine Wizard: Name and Operating System



1. Use this page to specify a name and operating system (OS) for the virtual machine and to change the storage location used for VMs. You can also choose to disable the unattended guest operating system install feature.

**The following fields are available on this wizard page:**

* **Name.** A name for the new VM. The name you enter is shown in the machine list of VirtualBox Manager and is also used for the virtual machine's files on disk.

Be sure to assign each VM an informative name that describes the OS and software running on the VM.

For example, a name such as Windows 10 with Visio.

* **Folder.** The location where VMs are stored on your computer, called the *machine folder*. The default folder location is shown.

Ensure that the folder location has enough free space, especially if you intend to use the snapshots feature. See also [Section 10.1.1, “The Machine Folder”](https://www.virtualbox.org/manual/ch10.html#vboxconfigdata-machine-folder).

* **ISO Image.** Select an ISO image file. The image file can be used to install an OS on the new virtual machine or it can be attached to a DVD drive on the new virtual machine.
* **Type and Version.** These fields are used to select the OS that you want to install on the new virtual machine.

The supported OSes are grouped into types. If you want to install something very unusual that is not listed, select the **Other** type. Depending on your selection, Oracle VM VirtualBox will enable or disable certain VM settings that your guest OS may require. This is particularly important for 64-bit guests. See [Section 3.1.2, “64-bit Guests”](https://www.virtualbox.org/manual/ch03.html#intro-64bitguests). It is therefore recommended to always set this field to the correct value.

If an ISO image is selected and Oracle VM VirtualBox detects the operating system for the ISO, the **Type** and **Version** fields are populated automatically and are disabled.

* **Skip Unattended Installation.** Disables unattended guest OS installation, even if an ISO image is selected that supports unattended installation. In that case, the selected ISO image is mounted automatically on the DVD drive of the new virtual machine and user interaction is required to complete the OS installation.

The unattended installation step in the wizard is skipped.

### Note: This option is disabled if you do not select an installation medium in the ISO Image field.

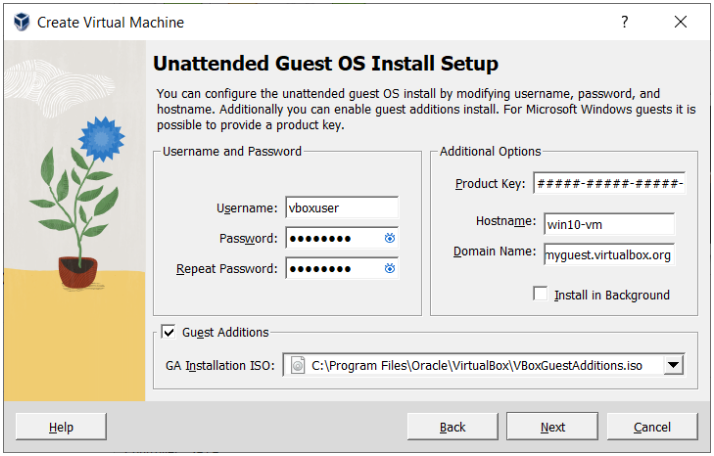
1. Click **Next** to go to the next wizard page.

### (Optional) Create Virtual Machine Wizard: Unattended Guest OS Install: Unattended guest OS installation enables you to install the OS on a virtual machine automatically.

### Note: This page is optional. It is not displayed if you have selected the Skip Unattended Installation option on the initial wizard page.

1. Use this page to set up the required parameters for unattended guest OS installation and to configure automatic installation of the Oracle VM VirtualBox Guest Additions.

**Creating a Virtual Machine: Unattended Guest OS Installation**



The following fields are available on this wizard page:

**Username & Password.** Enter the credentials for a default user on the guest OS.

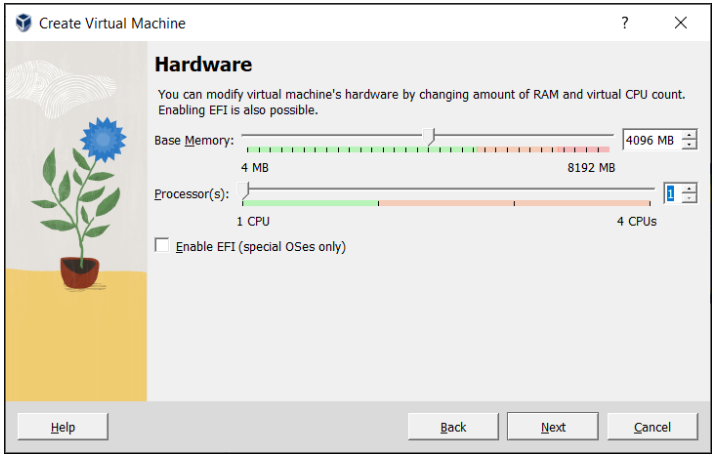
**Guest Additions.** Enables automatic installation of the Guest Additions, following installation of the guest OS. Use the drop-down list to select the location of the ISO image file for the Guest Additions.

**Additional Options.** The following options enable you to perform extra configuration of the guest OS:

* + **Product Key.** For Windows guests only. Enter the product key required for Windows installation.
  + **Hostname.** Host name for the guest. By default, this is the same as the VM name.
  + **Domain Name.** Domain name for the guest.
  + **Install in Background.** Enable headless mode for the VM, where a graphical user interface is not shown.

Click **Next** to go to the next wizard page.

### Create Virtual Machine Wizard: Hardware

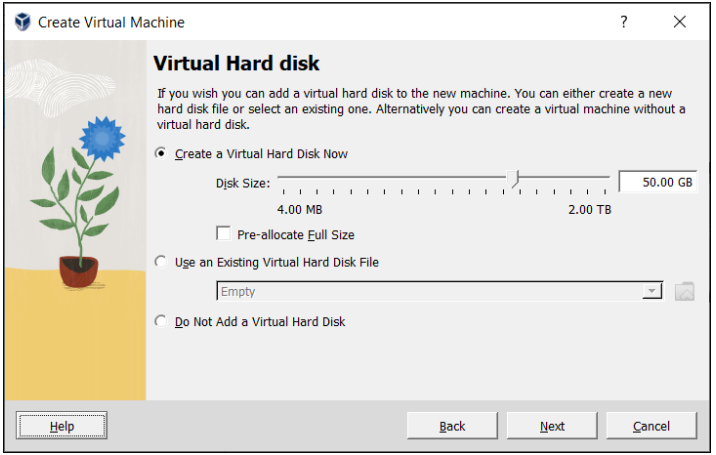


### Caution: A guest OS may require at least 1 or 2 GB of memory to install and boot up. For best performance, more memory than that may be required.

As with other **Create Virtual Machine** wizard settings, you can change this setting later, after you have created the VM. **Processor(s).** Select the number of virtual processors to assign to the VM. It is not advised to assign more than half of the total processor threads from the host machine. **Enable EFI.** Enables Extensible Firware Interface (EFI) booting for the guest OS.

Click **Next** to go to the next wizard page.

### Create Virtual Machine Wizard: Virtual Hard Disk



### Note: You can skip attaching a virtual hard disk file to the new virtual machine you are creating. But you will then need to attach an hard disk later on, in order to install a guest operating system.

* 1. Running Your Virtual Machine:

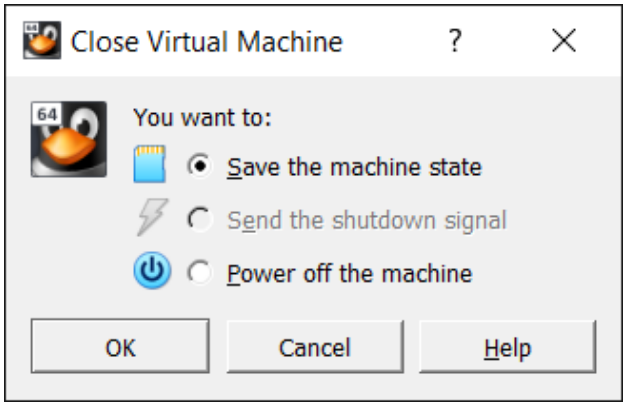
**Starting a New VM for the First Time:**When you start a VM for the first time the OS installation process is started automatically, using the ISO image file specified in the Create Virtual Machine wizard. Follow the onscreen instructions to install your OS.**Capturing and Releasing Keyboard and Mouse:** Host Key Setting on the Virtual Machine Taskbar, To release keyboard ownership, press the Host key. As explained above, this is typically the **right Ctrl key.** 

**Resizing the Machine's Window:** To enable scaled mode, **press Host key+C**, or select **Scaled Mode** from the View menu in the VM window. To leave scaled mode, **press Host key+C** again. The aspect ratio of the guest screen is preserved when resizing the window.

To ignore the aspect ratio, press Shift during the resize operation.

5. **Saving the State of the Machine:** When you click on the Close button of your virtual machine window, at the top right of the window, just like you would close any other window on your system, Oracle VM VirtualBox asks you whether you want to save or power off the VM. As a shortcut, you can also press Host key + Q.

**Closing Down a Virtual Machine**



**Warning**

This is equivalent to pulling the power plug on a real computer without shutting it down properly. If you start the machine again after powering it off, your OS will have to reboot completely and may begin a lengthy check of its virtual system disks. As a result, this should not normally be done, since it can potentially cause data loss or an inconsistent state of the guest system on disk.

**APPLICATIONS:** There are various applications of cloud computing in today’s network world. Many search engines and social websites are using the concept of cloud computing like [www.amazon.com,](http://www.amazon.com/) hotmail.com, facebook.com, linkedln.com etc. the advantages of cloud computing in context to scalability is like reduced risk, low cost testing, ability to segment the customer base and auto-scaling based on application load.

### RESULT:

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

**EX.NO.:2**

**Install a C compiler in the virtual machine created using virtual box and execute Simple Programs**

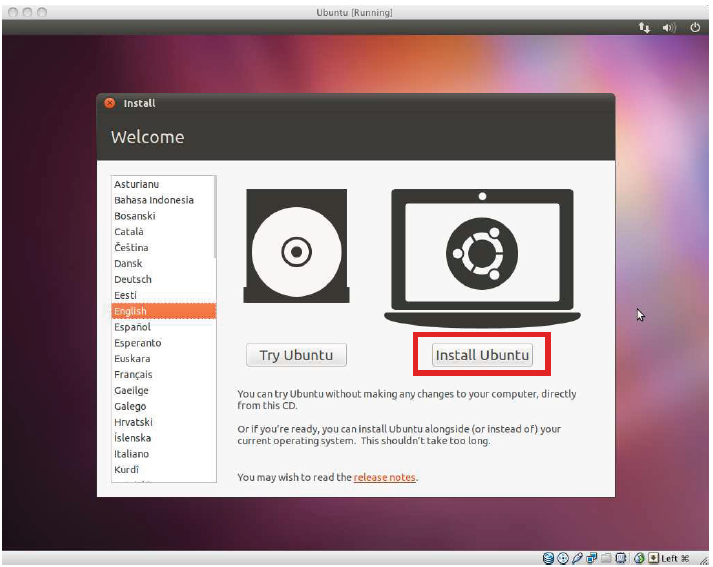
**Aim:**

To Install a C compiler in the virtual machine created using virtual box and execute simple programs.

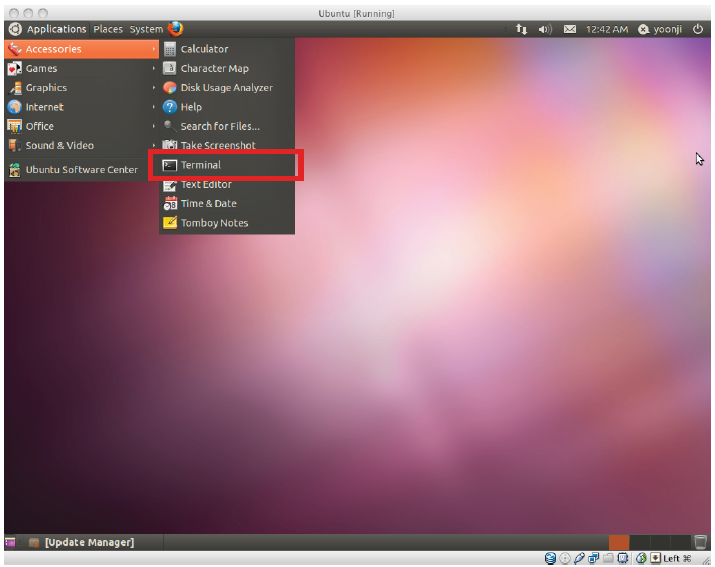
# PROCEDURE:

**Steps to Create Virtual Machine and Install :**

1. Create Virtual Machine Wizard: Name and Operating System, follow the steps same as above experiment.
2. Install Linux using Virtual Box 🡪 Choose Ubuntu from left column and click Start 🡪 Click Install Ubuntu



**C Programming on Linux:** 1. Open Terminal (Applications-Accessories-Terminal)



2. Open gedit by typing “**gedit &** “ on terminal: (Can also use any other Text Editor application),

Type the following on gedit (or any other text editor) : **Save this "linearSearch.c"**

#include <stdio.h>

int main() { int array[100], search, c, n;

printf("Enter number of elements in array\n"); scanf("%d", &n); printf("Enter %d integer(s)\n", n);

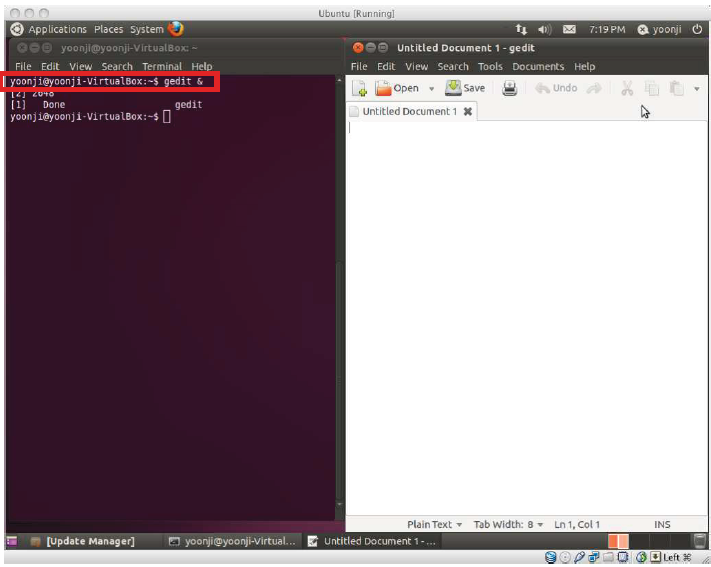
for (c = 0; c < n; c++) scanf("%d", &array[c]);

printf("Enter a number to search\n"); scanf("%d", &search);

for (c = 0; c < n; c++) { if (array[c] == search) /\* If required element is found \*/ {

printf("%d is present at location %d.\n", search, c+1); break; } }

if (c == n) printf("%d isn't present in the array.\n", search); return 0;}

****

4. **Compile and Run C Program:** $ gcc linearSearch.c $ ./a.out

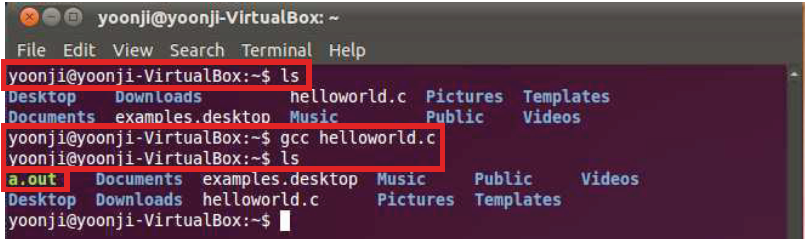
**Output:** Enter number of elements of an array: 8

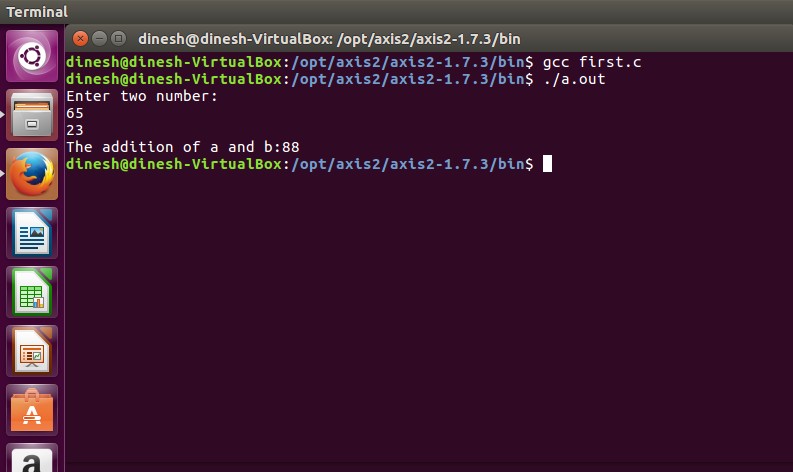
Enter elements: 2 3 5 7 8 6 4 1

Enter item to search: 1

Item found at location 8

5. **Steps to install GCC: $** sudo apt install gcc **$**GCC — version **$**cd DIRECTORY\_PATH



****

# APPLICATIONS: Simply running all programs in grid environment.

**RESULT:**

Thus the GCC compiler installed and a simple C program executed successfully.

**EX NO.:3**

**Install Google App Engine. Create *hello world* app and other simple web applications using python/java**.

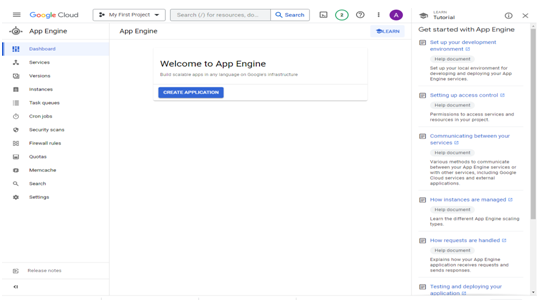
**Aim:**

To Install Google App Engine and create hello world app and other simple web applications using python/java.

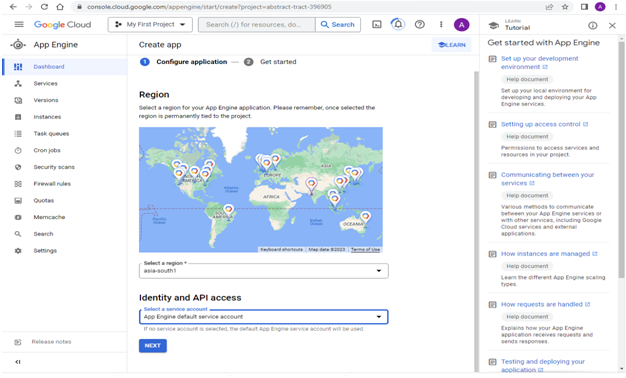
**Procedure:**

1. Download Google App Engine or create an account in Google Cloud Platform

(*from* Free Tier Account).



1. Select the downloaded file (double click) 🡪 select next 🡪 select next(language preferences) 🡪 location to save python/java programs.
2. Download python latest version and install in the local system.
3. Create a web application after selecting the language either java / python 🡪 select the region as given below:



**5. Create a Python app in the App Engine flexible environment:** This quickstart demonstrates how to create and deploy an app that displays a short message. The sample application uses the latest version of Python 3. You can use the same code sample for Python runtime version 3.8 and later, by specifying the version in your app.yaml.

**Before you begin:**

1. In the Google Cloud console, on the project selector page, select or create a Google Cloud project.
2. Make sure that billing is enabled for your Google Cloud project.
3. Enable the Cloud Build API.
4. Enable the API. Install the Google Cloud CLI.
5. To initialize the gcloud CLI, run the following command:

*gcloud init*

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

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

When prompted, select the region where you want to locate your App Engine application.

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

Install the following prerequisites:

* Download and install Git.

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

*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.

This quickstart demonstrates the latest Python 3 app written with the Flask web framework that can be deployed to App Engine. Although this sample uses Flask, you can use any web framework that satisfies the requirements above. Alternative frameworks include Django, Pyramid, Bottle, and web.py.

**Download the Hello World app:** We've created a simple Hello World app for App Engine so you can quickly get a feel for deploying an app 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 that contains the sample code.

*cd python-docs-samples/appengine/flexible/hello\_world*

Run Hello World on your local machine:

* To run the Hello World app on your local computer:

Use PowerShell to run your Python packages.

* Locate your installation of PowerShell.
* Right-click on the shortcut to PowerShell and start it as an administrator.

**Create an isolated Python environment.**

*python -m venv env*

*.\env\Scripts\activate*

Navigate to your project directory & install dependencies. If you're not in the directory that contains the sample code, navigate to the directory that contains the hello\_world sample code. Then, install dependencies:

*cd YOUR\_SAMPLE\_CODE\_DIR*

*pip install -r requirements.txt*

**Run the application:** python main.py

In your web browser, enter the following address: http://localhost:8080

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 flexible environment:

Deploy the Hello World app by running the following command from the hello\_world directory:

*gcloud app deploy*

Learn about the optional flags.

Launch your browser to view the app at https://PROJECT\_ID.REGION\_ID.r.appspot.com

*gcloud app browse*

where PROJECT\_ID represents your Google Cloud project ID.

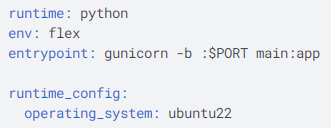
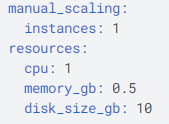
This time, the page that displays the Hello World message is delivered by a web server running on an App Engine instance.

Successfully deployed your first App Engine app to the App Engine flexible environment!

#### main.py:

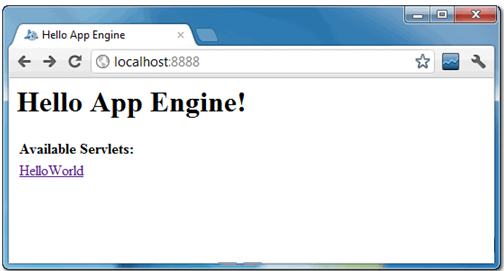
**

#### app.yaml

* *

**requirements.txt**

Flask==2.1.1  
gunicorn==20.1.0

**

**Result:**

Thus the *hello world* app and other simple web applications using python were created successfully.

**EX NO.:4**

**Use GAE launcher to launch the web applications.**

**Aim:**

To use Google App Engine - GAE launcher to launch the web applications.

**Procedure:**

You can use Google App Engine to host a static website. Static web pages can contain client- side technologies such as HTML, CSS, and JavaScript. Hosting your static site on App Engine can cost less than using a traditional hosting provider, as App Engine provides a free tier.

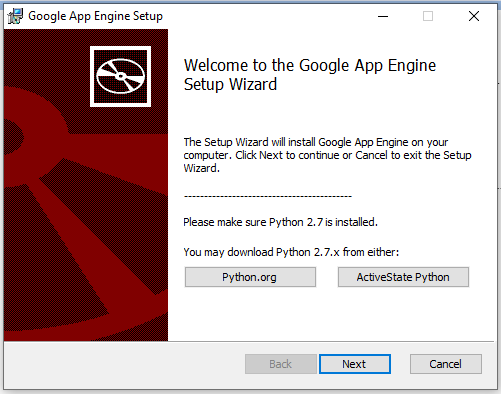
Sites hosted on App Engine are hosted on the REGION\_ID.r.appspot.com subdomain, such as

[my-project-id].uc.r.appspot.com. After you deploy your site, you can map your own domain name to your App Engine-hosted website.

1. Create a new Cloud Console project or retrieve the project ID of an existing project to use: Go to the Project page

2. Install and then initialize the Google Cloud SDK:

* Download the SDK
* Creating a website to host on Google App Engine
* Basic structure for the project



**This guide uses the following structure for the project:**

* app.yaml: Configure the settings of your App Engine application.
* www/: Directory to store all of your static files, such as HTML, CSS, images, and JavaScript.
* css/: Directory to store stylesheets.
* style.css: Basic stylesheet that formats the look and feel of your site.
* images/: Optional directory to store images.
* index.html: An HTML file that displays content for your website.
* js/: Optional directory to store JavaScript files.
* Other asset directories.

**Create the app.yaml file in your application's root directory:**

1. Create a directory that has the same name as your project ID. You can find your project ID in the Console.

2. In directory that you just created, create a file named app.yaml.

3. Edit the app.yaml file and add the following code to the file:

runtime: python27

api\_version: 1

threadsafe:

true handlers:

**-url: /**

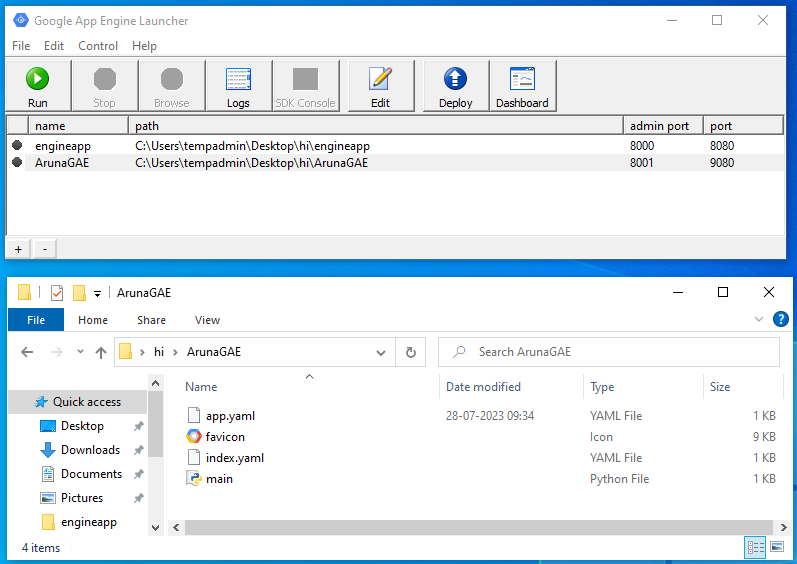
static\_files: www/index.html

upload: www/index.html

**-url: /(.\*)**

static\_files: www/\1

upload: www/(.\*)



**4. Creating the index.html file**

<html> <head> <title>Hello, world!</title>

<link rel="stylesheet" type="text/css" href="/css/style.css">

</head> <body>

<h1>Hello, world!</h1>

<p> This is a simple static HTML file that will be served from Google App Engine. </p>

</body>

</html>

To deploy your app, run the following command from within the root directory of your application where the app.yaml file is located:

*gcloud app deploy*

1. Google App Engine Launcher 🡪 Select File 🡪 Create New Application 🡪 Application Name [User\_Name] 🡪 Choose the path[folder to save the application\_package] 🡪 Select any Port[local\_Host] 🡪 Create to finish.
2. Go the Folder / Application\_Package[Open] 🡪 Following Files to edit the source code:

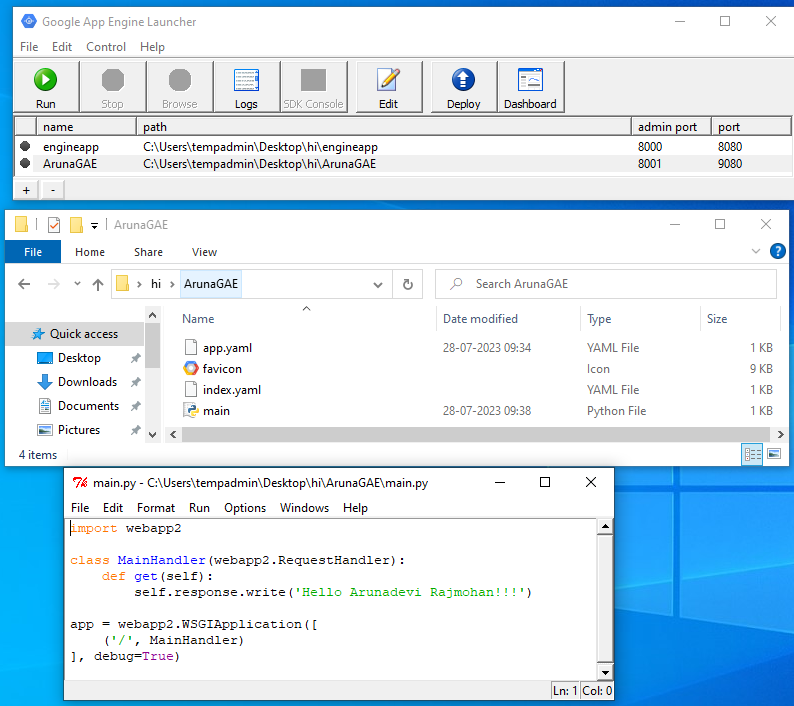
- app.yaml - favicon[Icon] - index.yaml - main.py

**Google App Engine Launcher:**

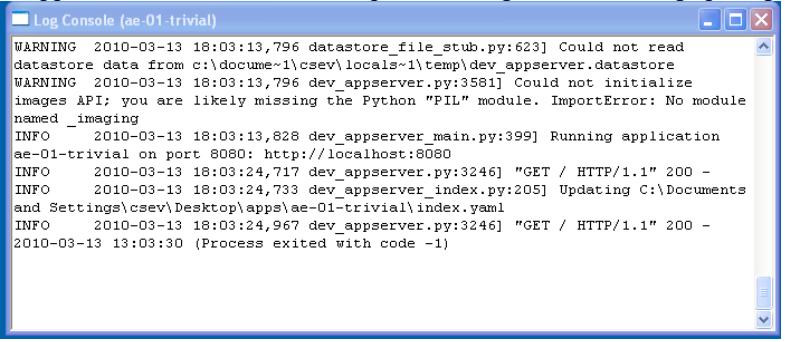
1. Select your Application 🡪 Run 🡪 it will deploy your application in the Local\_Host[where your selected while create]

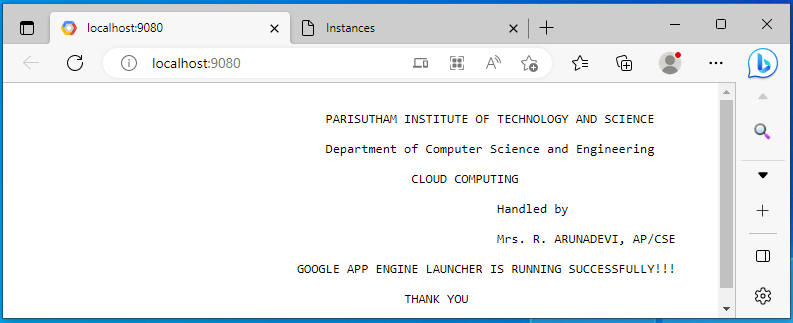


1. Open a browser and type: [http://localhost:9080[PORT/NOT ADMIN\_PORT]/](http://localhost:9080[PORT/NOT%20ADMIN_PORT%5d/) 🡪 Select the browser icon in GAE.

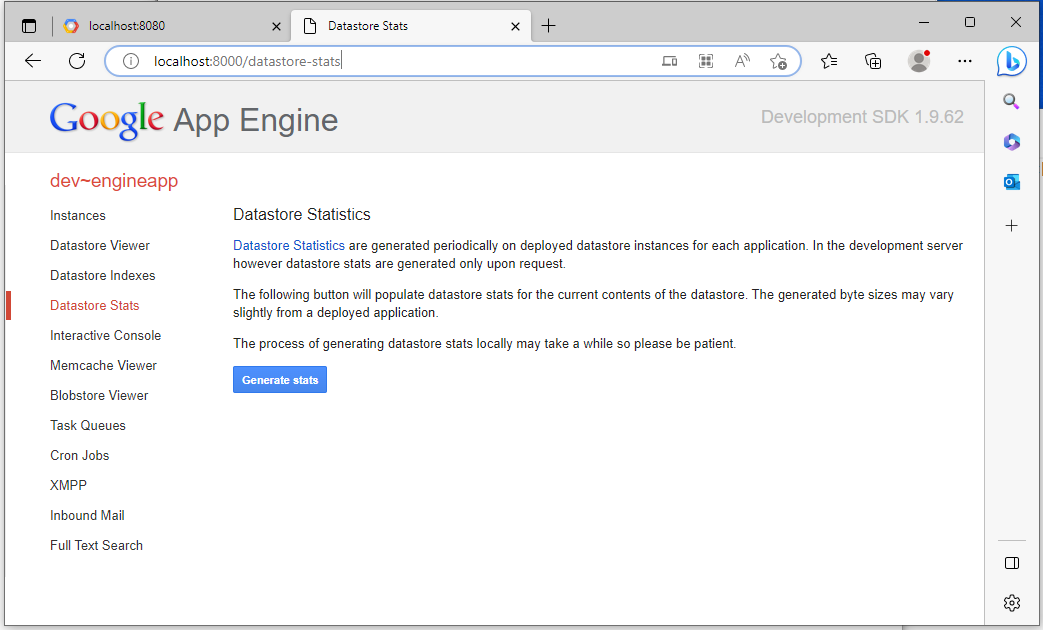


**NOTE:** You can watch the internal log of the actions that the web server is performing when you are interacting with your application in the browser. Select your application in the Launcher and press the Logs button to bring up a log window:





GAE instances / dashboard



**Shutting Down the Server:**  To shut down the server, use the Launcher, select your application and press the Stop button.

**Result:**

Thus a simple web application is launched using GAE launcher successfully.

**EX. NO.:5**

**Simulate a cloud scenario using CloudSim and run a scheduling algorithm that is not present in CloudSim.**

**Aim:**

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

**Steps: How to use CloudSim in Eclipse**

CloudSim is written in Java. The knowledge you need to use CloudSim is basic Java programming and some basics about cloud computing. Knowledge of programming IDEs such as Eclipse or NetBeans is also helpful. It is a library and, hence, CloudSim does not have to be installed. Normally, you can unpack the downloaded package in any directory, add it to the Java classpath and it is ready to be used. Please verify whether Java is available on your system.

**To use CloudSim:**

1. Download CloudSim installable files

from *https://code.google.com/p/cloudsim/downloads/list and unzip*

1. Open Eclipse
2. Create a new Java Project: File -> New
3. Import an unpacked CloudSim project into the new Java Project:

The first step is to initialise the CloudSim package by initialising the CloudSim library, as follows

CloudSim.init(num\_user, calendar, trace\_flag)

1. Data centres are the resource providers in CloudSim; hence, creation of data centres is a second step. To create Datacenter, you need the DatacenterCharacteristics object that stores the properties of a data centre such as architecture, OS, list of machines, allocation policy that covers the time or spaceshared, the time zone and its price:
2. Datacenter datacenter9883 = new Datacenter(name, characteristics, new VmAllocationPolicySimple(hostList), s
3. The third step is to create a broker:
4. DatacenterBroker broker = createBroker();
5. The fourth step is to create one virtual machine unique ID of the VM, userId ID of the VM’s owner, mips, number Of Pes amount of CPUs, amount of RAM, amount of bandwidth, amount of storage, virtual machine monitor, and cloudletScheduler policy for cloudlets:
6. Vm vm = new Vm(vmid, brokerId, mips, pesNumber, ram, bw, size, vmm, new CloudletSchedulerTimeShared())
7. Submit the VM list to the broker: broker.submitVmList(vmlist)
8. Create a cloudlet with length, file size, output size, and utilisation model:

Cloudlet cloudlet = new Cloudlet(id, length, pesNumber, fileSize, outputSize, utilizationModel, utilizationMode

1. Submit the cloudlet list to the broker: broker.submitCloudletList(cloudletList) Sample Output from the Existing Example:

Starting

CloudSimExample1... Initialising...

Starting CloudSim version 3.0 Datacenter\_0 is starting...

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>null

Broker is starting... Entities started.

: Broker: Cloud Resource List received with 1 resource(s) 0.0: Broker: Trying to Create VM #0 in Datacenter\_0

: Broker: VM #0 has been created in Datacenter #2, Host #0 0.1: Broker: Sending cloudlet 0 to VM #0

* 1. : Broker: Cloudlet 0 received

: Broker: All Cloudlets executed. Finishing 400.1: Broker: Destroying

VM #0

Broker is shutting down... Simulation: No more future events

CloudInformationService: Notify all CloudSim entities for shutting down. Datacenter\_0 is shutting down...

Broker is shutting down Simulation completed. Simulation completed.

|  |  |  |  |
| --- | --- | --- | --- |
| ========== OUTPUT ========== |  | | |
| Cloudlet ID STATUS Data center ID Finish Time 0 SUCCESS 2  0.1 400.1 | VM ID | Time 0 | Start Time  400 |

\*\*\*\*\*Datacenter: Datacenter\_0\*\*\*\*\* User id

Debt

3 35.6

CloudSimExample1 finished!

# RESULT:

Thus the simulation was successfully executed.

**EX.NO:6**

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

**Aim:**

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

**Procedure:**

* + 1. You can copy few (or more) lines with *copy & paste* mechanism.

For this you need [to share clipboard between host OS and guest OS](https://superuser.com/questions/42134/virtualbox-shared-clipboard), installing Guest Addition on both the virtual machines (probably setting *bidirectional* and restarting them). You *copy* from *guest OS* in the clipboard that is shared with the *host OS*.

Then you *paste* from the *host OS* to the second *guest OS*.

* + 1. You can enable drag and drop too with the same method (Click on the machine, settings, general, advanced, drag and drop: set to *bidirectional* )
    2. You can have common *Shared Folders* on both virtual machines and use one of the directory shared as buffer to copy.

Installing Guest Additions you have the possibility to set [Shared Folders too](https://www.virtualbox.org/manual/ch04.html#sharedfolders). As you put a file in a shared folder from *host OS* or from *guest OS*, is immediately visible to the other. (Keep in mind that can arise some problems for date/time of the files when there are different clock settings on the different virtual machines).

*If you use the same folder shared on more machines you can exchange files directly copying them in this folder.*

* + 1. You can use usual method to copy files between 2 different computer with client-server application. (e.g. scp with sshd active for linux, winscp... [you](http://en.wikipedia.org/wiki/Comparison_of_SSH_servers) [can get some info about SSH](http://en.wikipedia.org/wiki/Comparison_of_SSH_servers) [servers e.g. here](http://en.wikipedia.org/wiki/Comparison_of_SSH_servers))

You need an active server (sshd) on the receiving machine and a client on the sending machine. Of course you need to have the authorization setted (via password or, better, via an [automatic authentication method](https://help.github.com/articles/generating-ssh-keys)).

**Note:** many Linux/Ubuntu distribution install sshd by default: you can see if it is running with pgrep sshd from a shell. You can install with sudo apt-get install openssh-server.

* + 1. You can mount part of the file system of a virtual machine via [NFS](http://en.wikipedia.org/wiki/Network_File_System) or [SSHFS](http://en.wikipedia.org/wiki/SSHFS) on the other, or you can share file and directory with Samba. You may find interesting the article [Sharing files between guest and](https://www.virtualbox.org/wiki/Sharing_files_on_OSE) [host without](https://www.virtualbox.org/wiki/Sharing_files_on_OSE) [VirtualBox shared folders](https://www.virtualbox.org/wiki/Sharing_files_on_OSE) with detailed step by step instructions.

**Step 1: Create Network**

**Network:**Yes, the network in here is our own local network. So, your instances will be not

mixed up with the others. You can imagine this as your own LAN (Local Area Network) in the

cloud.

1. Go to Network > Networks and then click Create Network.

2. In Network tab, fill Network Name for example internal and then click Next.

3. In Subnet tab,

1. Fill Network Address with appropriate CIDR, for example 192.168.1.0/24. Use private

network CIDR block as the best practice.

2. Select IP Version with appropriate IP version, in this case IPv4.

3. Click Next.

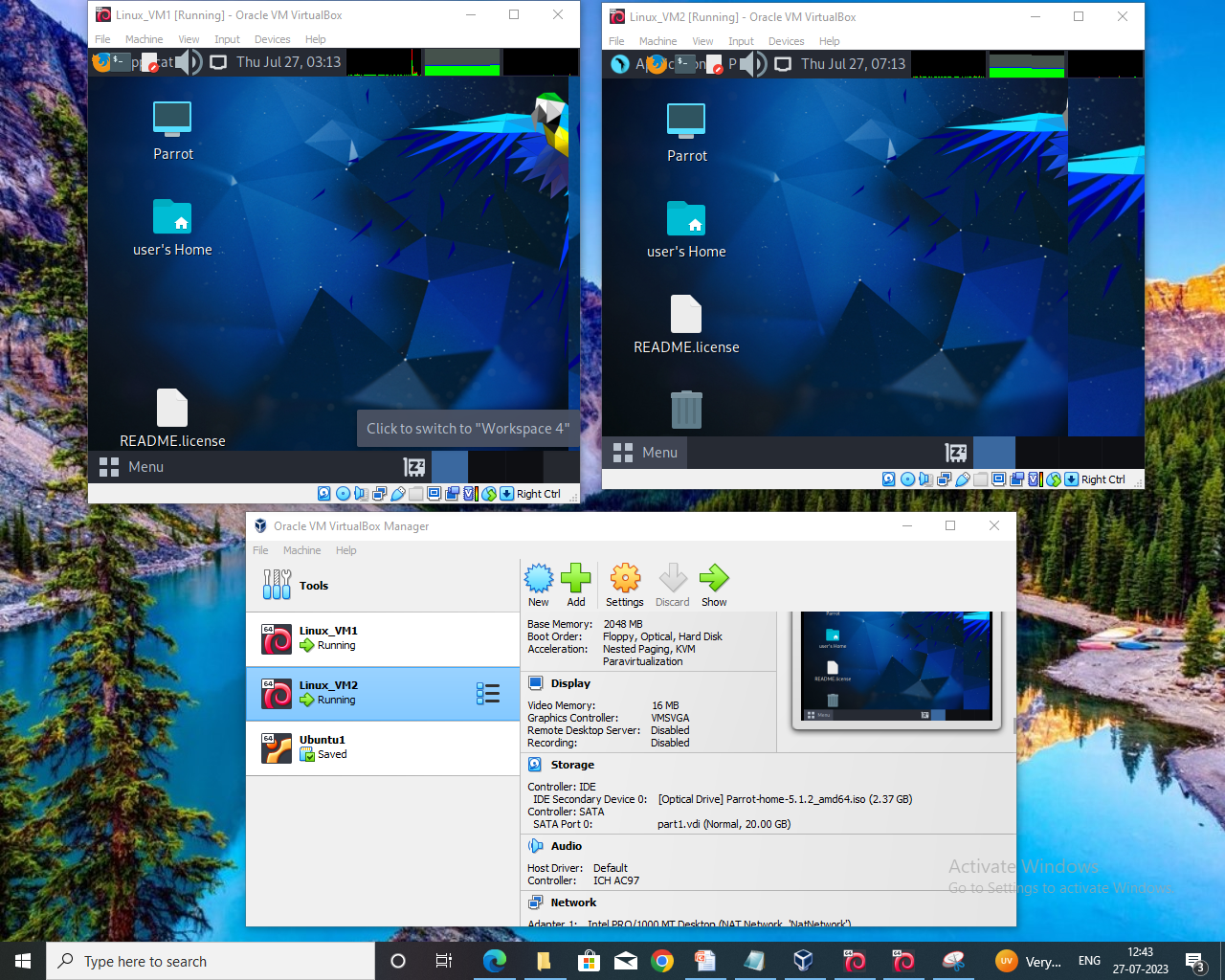
4. In Subnet Details tab, fill DNS Name Servers with 8.8.8.8 (Google DNS) and then

click Create.

Step 2: Create Instance

**Steps:**

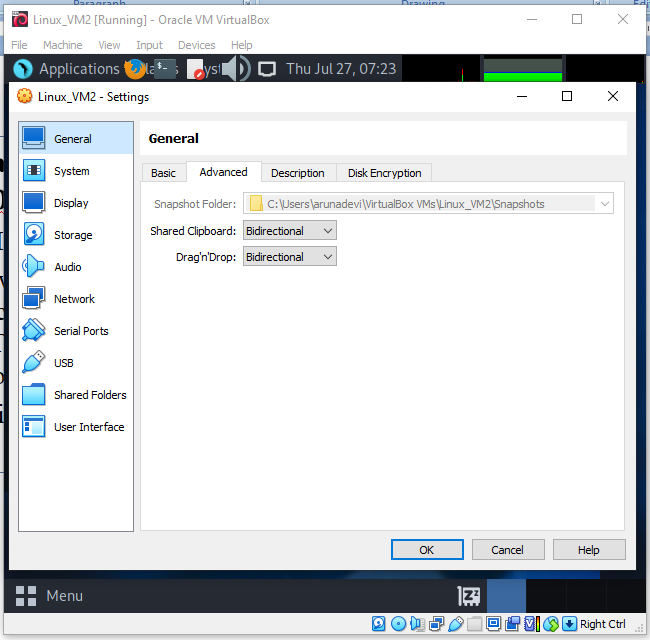
1. Open two virtual machines to share clipboard between host OS and guest OS, installing Guest Addition on both the virtual machines (probably setting bidirectional and restarting them).
2. Check the IP address of your current local system, in order to create NAT network.
3. Deploy Virtual MachineVM1: Click File > New Virtual Machine. The New Virtual Machine Wizard appears.
4. Repeat Step 2 for Virtual MachineVM2.



**Shared Clipboard in VirtualBox:** you can set up a **Shared Clipboard**and **Drag'n'Drop**:

1. Select your **VM**, right-click and select Settings.
2. You should now see the Shared Clipboard and Drag'n'Drop dropdown options. You can choose from Host to Guest, Guest to Host, and Bidirectional. There's also the default option, Disabled, which prevents your attempts to copy files to VirtualBox.
3. Select Bidirectional in both drop-down boxes to enable two-way file sharing.
4. Usual method to copy files between 2 different computer with client-server application. (e.g. scp with sshd active for linux, winscp.. get some info about SSH(Secure Shell or Secure Socket Shell, is a network protocol) servers e.g. here).
5. SCP command or secure copy allows the secure transferring of files between the local host and the remote host or between two remote hosts.
6. **Note:** Many Linux/Ubuntu distribution install sshd by default: you can see if it is running with pgrep sshd from a shell. If not then you can install with

$ sudo apt-get install openssh-server



**Start SSH Service(LINUX):**

If transfer connection refused: **$ sudo service ssh status**

Port using: **$grep Port /etc/ssh/ssh\_config**

Port 22 🡪Use Is Closed: **$sudo lsof -i -n -P | grep LISTEN**

If SSH isn’t installed on LINUX:

**$sudo iptables-save** # display IPv4 rules

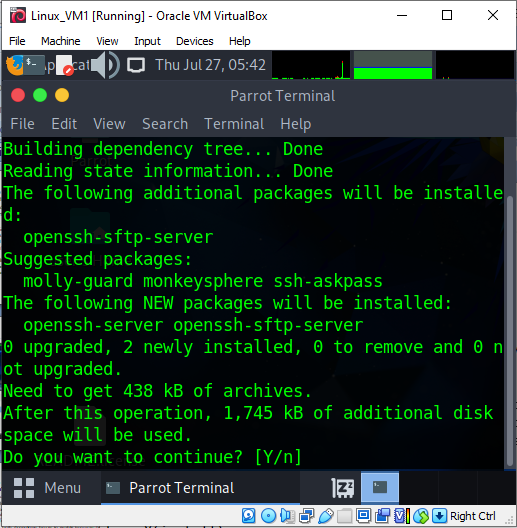
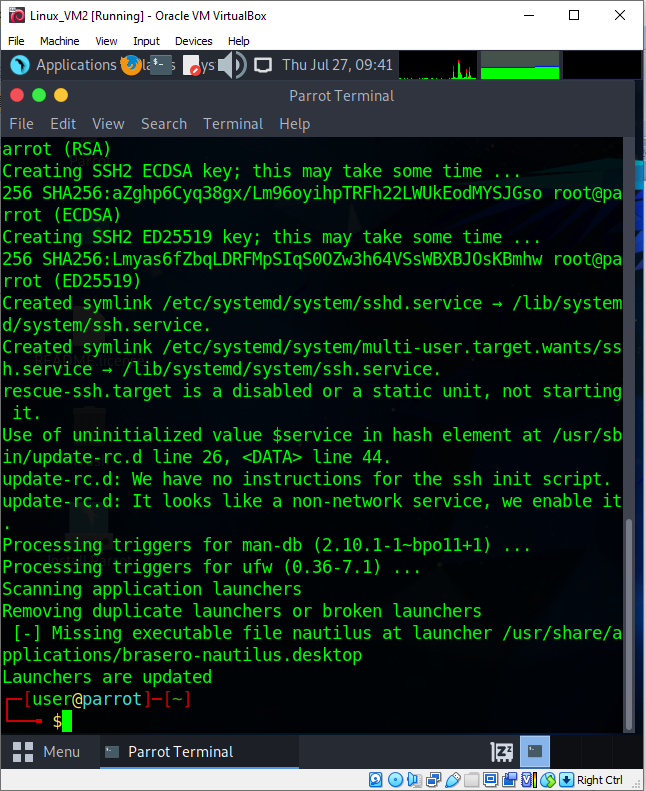
**$sudo ip6tables-save** # display IPv6 rules

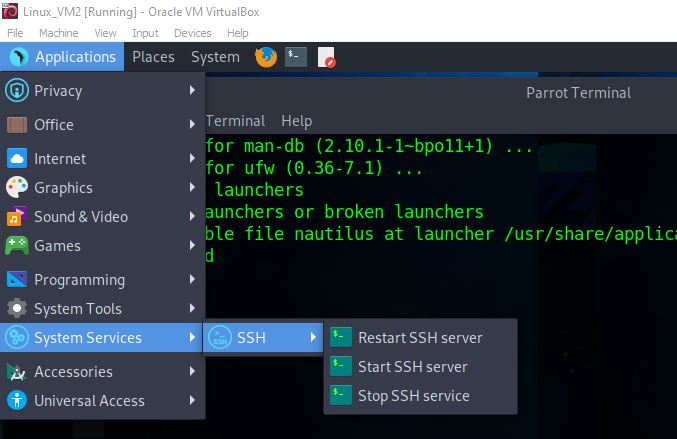
To install: **$sudo apt install openssh-server**

Type:Y(install)

Type:yes(to open server for access)

**$sudo service ssh restart**

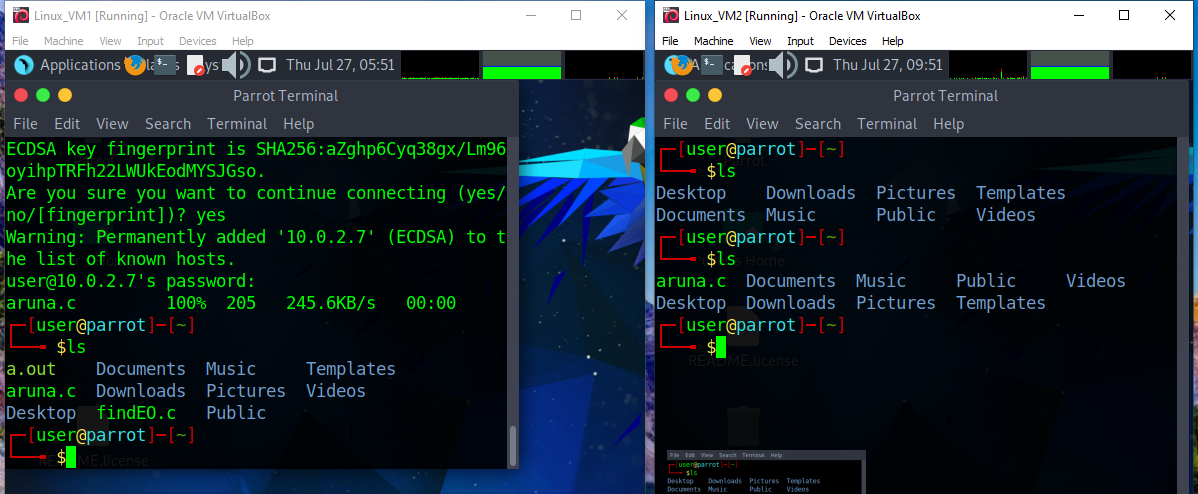




**To transfer file from one VM to another VM us the below code:**

**$ scp [file\_name] [user\_name]@[IP\_address] :/[current\_path$pwd]**

**After Migration:**

****

The above screenshot shows that aruna.c file transfered from Linux\_VM1 to Linux\_VM2 successfully.

# APPLICATIONS:

Easily migrate your virtual machine from one pc to another.

**Result:**

Thus the file transfer between VM was successfully completed.

**EX.NO:7**

**Install Hadoop single node cluster and run simple applications like wordcount.**

**Aim:**

To Install Hadoop single node cluster and run simple applications like wordcount.

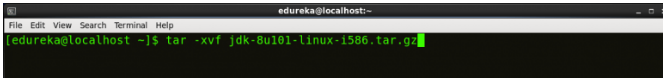
**Steps:**

**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.

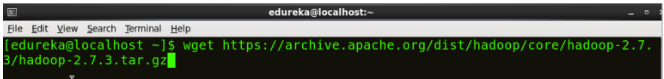
Command: tar -xvf jdk-8u101-linux-i586.tar.gz



*Fig: Hadoop Installation – Extracting Java Files*

**Step 3:** **Download the Hadoop 2.7.3 Package.**

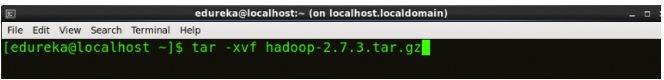
***Command*:** wget <https://archive.apache.org/dist/hadoop/core/hadoop-2.7.3/hadoop-> 2.7.3.tar.gz



*Fig: Hadoop Installation – Downloading Hadoop*

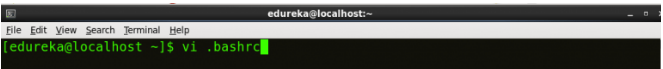
**Step 4:** Extract the Hadoop tar File.

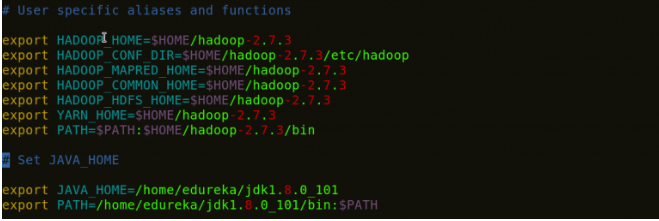
***Command***: tar -xvf hadoop-2.7.3.tar.gz



*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



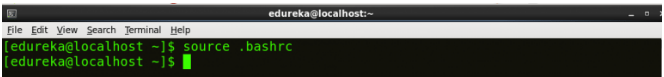


*Fig: Hadoop Installation – Setting Environment Variable*

Then, save the bash file and close it.

For applying all these changes to the current Terminal, execute the source command.

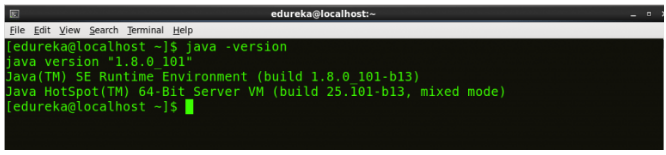
***Command*:** source .bashrc



*Fig: Hadoop Installation – Refreshing environment variables*

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

***Command*:** java -version

**

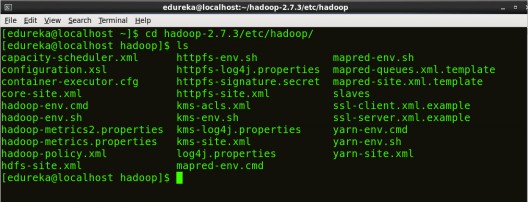
*Fig: Hadoop Installation – Checking Java Version*

***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

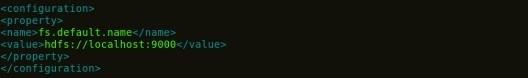
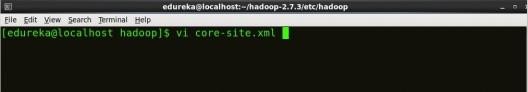
All the Hadoop configuration files are located in **hadoop-2.7.3/etc/hadoop** directory as you can see in the snapshot below:

*Fig: Hadoop Installation – Hadoop Configuration File*

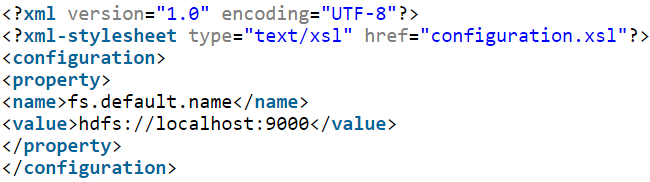
**Step 7:** Open *core-site.xml* and edit the property mentioned below inside configuration tag:

***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

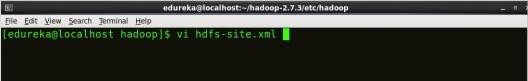


*Fig: Hadoop Installation – Configuring core-site.xml*

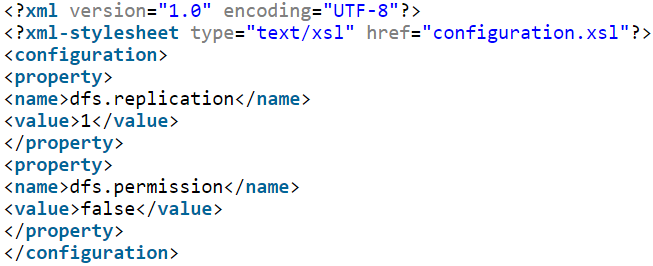
**

**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*

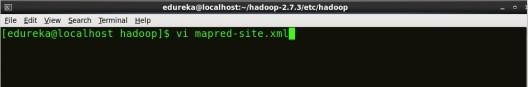
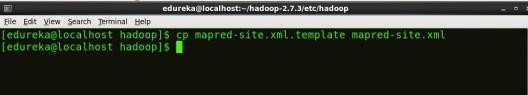
**

**Step 9:** Edit the *mapred-site.xml* file and edit the property mentioned below

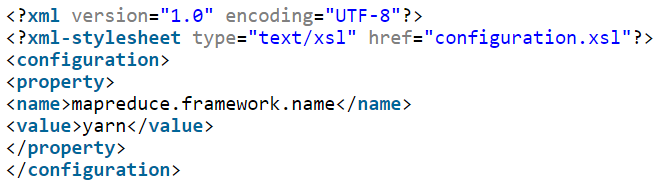
**inside configuration tag:** *mapred-site.xml* contains configuration settings of MapReduce application like number of JVM that can run in parallel, the size of the mapper and the reducer process, CPU cores available for a process, etc.

In some cases, mapred-site.xml file is not available. So, we have to create the mapred- site.xml file using mapred-site.xml template.

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

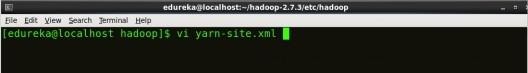
***Command*:** vi mapred-site.xml.

*Fig: Hadoop Installation – Configuring mapred-site.xml*

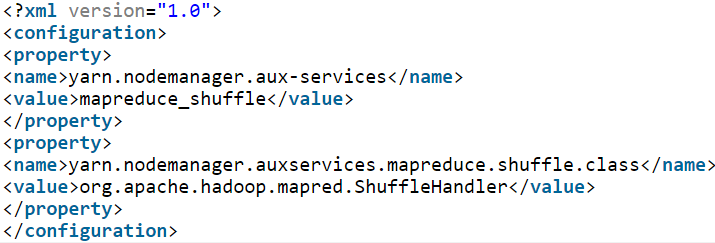


**Step 10:** Edit *yarn-site.xml* and edit the property mentioned below inside configuration tag: *yarn-site.xml* contains configuration settings of ResourceManager and NodeManager like application memory management size, the operation needed on program & algorithm, etc.

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



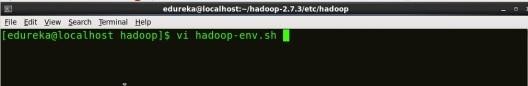
*Fig: Hadoop Installation – Configuring yarn-site.xml*

**

**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 in the HDFS.

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

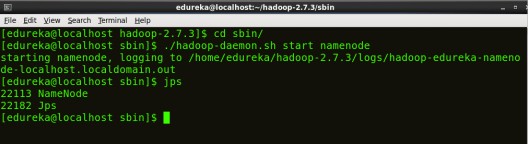
The above command is a combination of ***start-dfs.sh, start-yarn.sh*** & ***mr-jobhistory- daemon.sh***

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.

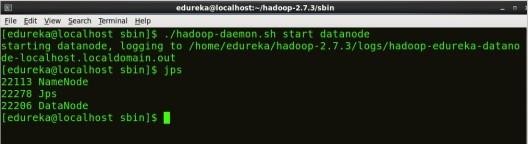
***Command:*** ./hadoop-daemon.sh start namenode



## Start DataNode:Fig: Hadoop Installation – Starting NameNode

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

***Command:*** ./hadoop-daemon.sh start datanode

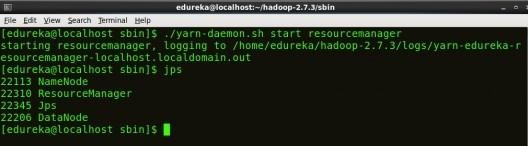


*Fig: Hadoop Installation – Starting DataNode*

## 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

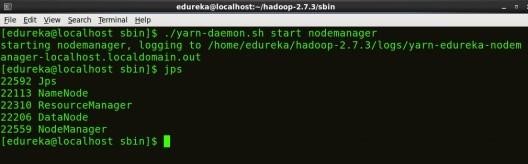


*Fig: Hadoop Installation – Starting ResourceManager*

## 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*

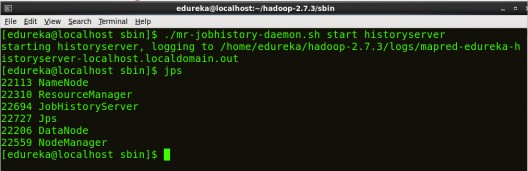
## Start JobHistoryServer:

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.**

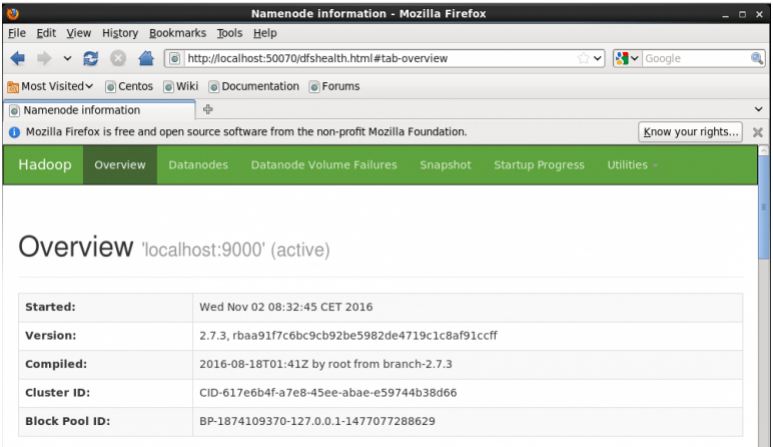
***Command:*** jps



*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*

Successfully installed a single node Hadoop cluster.

**Result:**

Thus the Hadoop one cluster was installed and simple applications executed successfully.

**Steps:**

**Creating First Docker Application Step-by-Step**

1. Getting Docker

You have two options:

Option a) Install Docker on your machine

Access the Docker website to find the installation file: version for Mac, version for Windows.

For Windows and macOS, we install the Docker Desktop utility, which includes: Docker Engine, Docker CLI client, Docker Compose, Docker Content Trust, Kubernetes, and Credential Helper.

For Linux it varies by distribution, the exact information can be found on the Docker Engine, in the “Installation per distro” tab.

Select the system you are using and follow the instructions to run the Docker application on your computer for the first time.

Option b) Play with Docker

If you do not want to install the docker on your computer, you can use a Docker playground which allows users to run Docker commands. Register on this website and start using Docker.

### 

#### 2. Containerizing an App:

Access to the Docker environment. It could be Docker Desktop on Windows or MacOS. It could be Play with Docker in the cloud.  So it’s time to build a Docker image containing the code of your application. In example, use the React application code, but remember you can use other languages, like Angular, JavaScript, HTML, & more.

##### **2.1. Creating a Dockerfile:** Assume you have your application code in a folder on your computer. So it’s time to create a Dockerfile.

The first thing we need to do is to define what image we want to build from. Here we will use the latest version of node available from the [Docker Hub](https://hub.docker.com/_/node):

FROM node:latest

Next, we create a directory to hold the application code inside the image:

#Create directory in container image for app code

RUN mkdir /usr/src/app

To bundle your app’s source code inside the Docker image, use the COPY instruction:

#Copy app code (./my-app) to /usr/src/app/ in container image

COPY ./my-app /usr/src/app/

Next, we set a directory that will be the working directory for your application:

#Set working directory

WORKDIR /usr/src/app

To install npm dependencies from packages.json, run the following command:

#Install dependencies from packages.json

RUN npm install

Our app binds to port 3000 so you’ll use the EXPOSE instruction to have it mapped by the docker daemon:

#Set container listen port

EXPOSE 3000

Finally, we define a command to run the application inside the container:

#Command for container to execute

ENTRYPOINT [ "npm" , "start" ]

Your Dockerfile should now look like this:

FROM node:latest

#Create directory in container image for app code

RUN mkdir /usr/src/app

#Copy app code (./my-app) to /usr/src/app/ in container image

COPY ./my-app /usr/src/app/

#Set working directory

WORKDIR /usr/src/app

#Install dependencies from packages.json

RUN npm install

#Set container listen port

EXPOSE 3000

#Command for container to execute

ENTRYPOINT [ "npm" , "start" ]

##### **2.2. Building the Image:** Go to the directory that has your Dockerfile and run the following command to build the Docker image.

docker build -t node-web-app .

The **-t** flag allows you to tag an image to make it easier to find later.

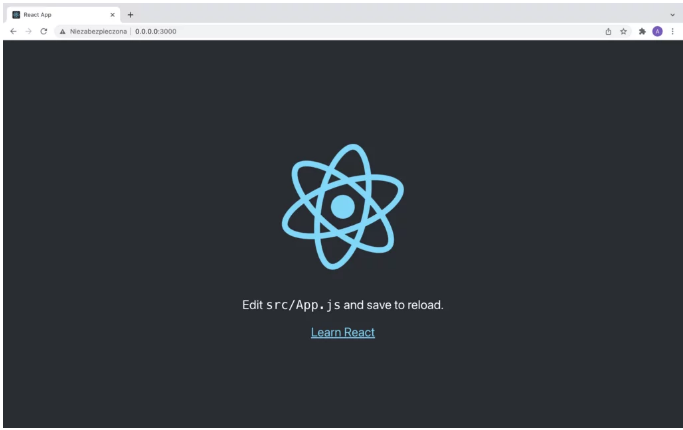
#### ****3. Running a Containerized App****

Now, if you’ve been following along you’ll have an image on your local machine with the app and the dependencies inside of it. And you can use that local image and run a container from it.

docker run -p 3000:3000 –name my-app -d node-web-app

Running your image with the **-p** flag redirects a public port to a private port inside the container. The **-d** flag runs the container in detached mode, leaving the container running in the background. The flag **–name**allows you to give a name to the container you are creating. As the last argument, we give the name of the image from which we want to run the container.

To check if the application is actually running, you need to go to your browser and enter [http://localhost:3000](http://localhost:3000/).



#### ****Managing a Containerized App (Useful Docker Command List)****

So, we’ve got a web app running inside a container. Well, we call what we’ve done a containerized app, and it’s kind of like a fast, lightweight virtual machine, meaning, we can stop and restart it, just like we can with a VM.

Finally, I prepared a list of commands that may be useful in managing the docker:

1. **Build an image from a Dockerfile**: docker build
2. **List images**: docker image ls
3. **List all existing containers**: docker ps -a
4. **Stop a container**: docker stop [container name]
5. **Delete a container**: docker rm [container name]
6. **Display logs of a container**: docker logs [container name]
7. **Run a command in a running container**: docker exec