

EX.NO:1

**INSTALL VIRTUAL BOX/VMWARE WORKSTATION
WITH DIFFERENT FLAVOURS OF LINUX OR WINDOWS**

DATE:

AIM:

To Install Virtual box/VMware Workstation with different flavours of linux or windows OS on top of windows7 or 8 Procedure

Virtual Box installation

1.First we need to download Virtual Box from <https://www.virtualbox.org>. I've downloaded Virtual Box 5.1.14

2. Run the executable and follow the prompts to complete the installation. We don't really need to change anything for our purposes, and can accept the defaults. Before completing the wizard you will get a warning that the network connection will temporarily be interrupted, so make sure you're not doing anything that would be impacted, like being half-way through downloading a 16GB file that can't be resumed



2. Create an Ubuntu virtual machine

1. Download the latest Ubuntu release from <https://www.ubuntu.com/download/desktop>.

I've downloaded Ubuntu 16.04.1

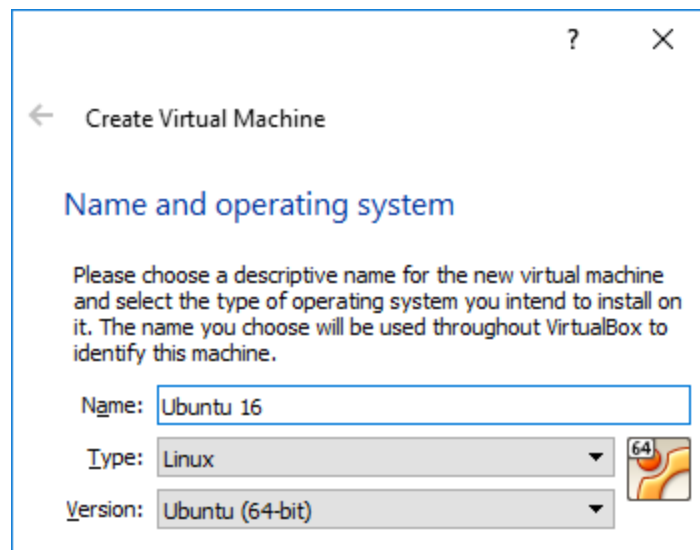
2. Open Virtual Box and click **New**



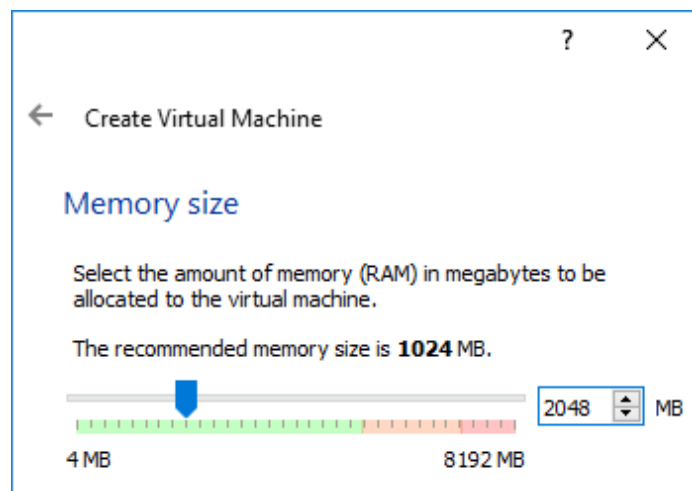
3. Type the *Name* for the virtual machine, like Ubuntu 16. VirtualBox will try to predict the *Type* and *Version* based on the name you enter. Otherwise, select:

- Type: Linux
- Version: Ubuntu (64-bit)

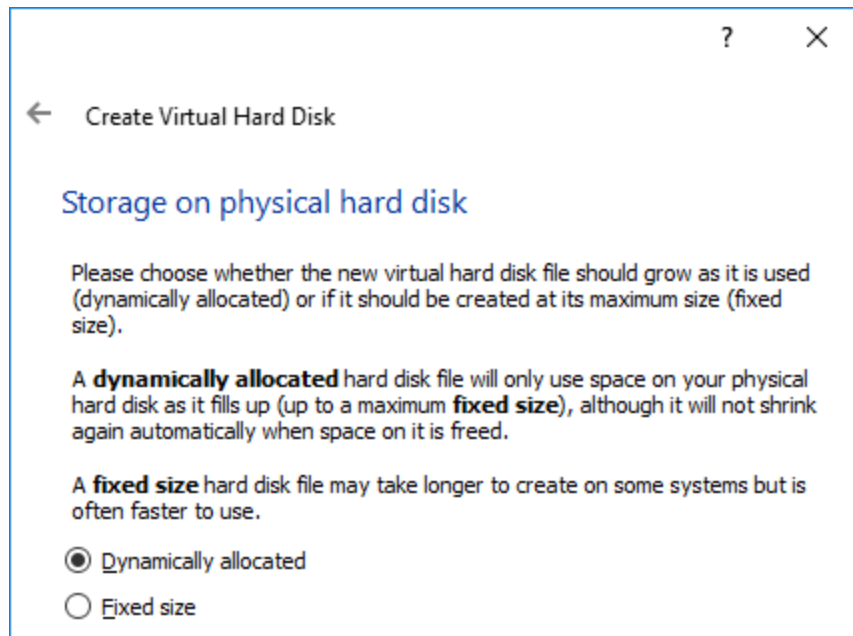
and click **Next**.



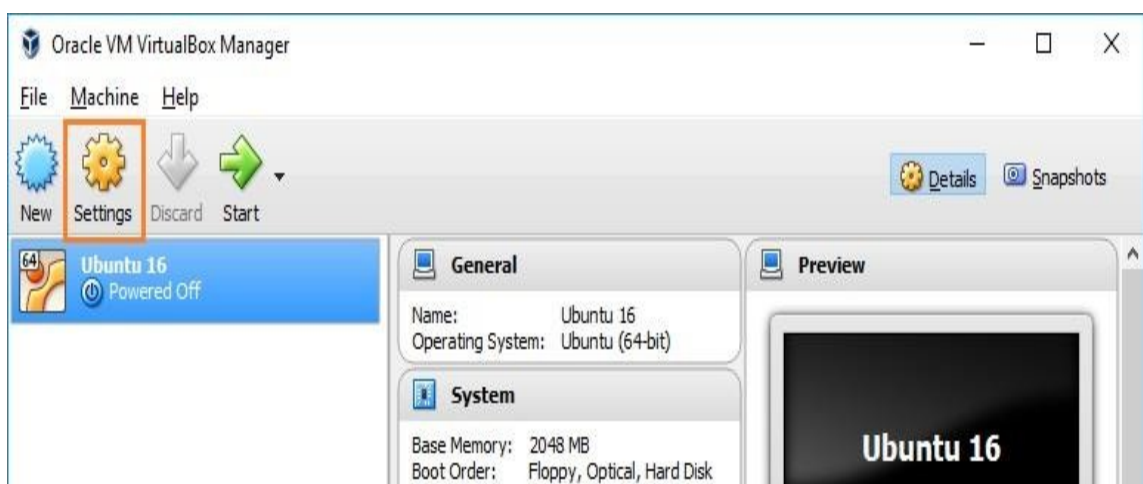
4. Next we need to specify how much memory to allocate the virtual machine. According to the Ubuntu [system requirements](#) we need 2GB, but I'd recommend more if your host can handle it. Basically the higher you can set the memory without severely impacting your host machine, the better the performance of the guest machine. If you're not sure, stick with 2GB.



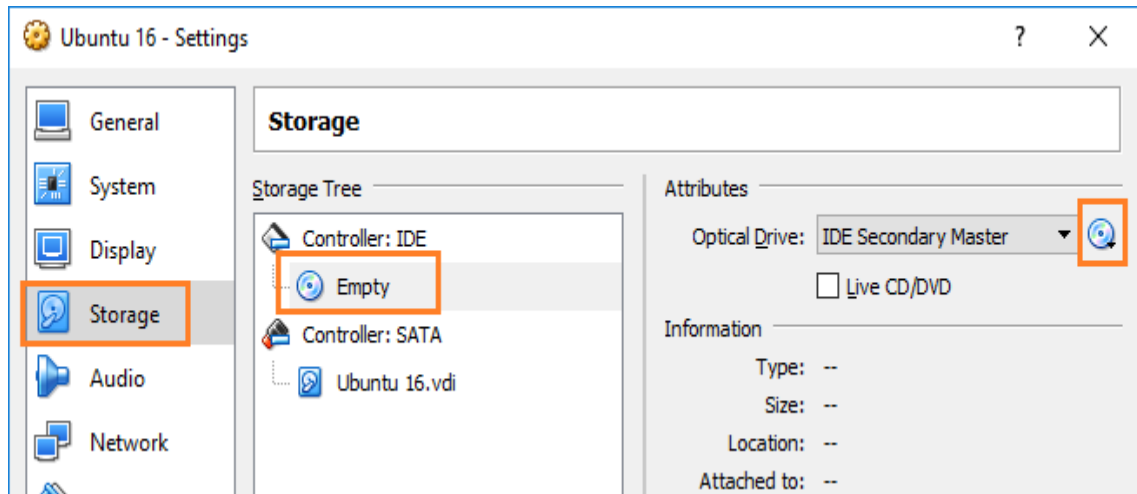
5. On the **Hardware** screen select **Create a virtual hard disk now** and click **Create**
6. Accept the default option **VDI** for **Hard disk file type** (or change it if you wish...) and click **Next**
7. Next we are prompted for **Storage on physical hard disk**. The options are *Dynamically allocated* and *Fixed size*. We'll use the default of **Dynamically allocated**. Click **Next**



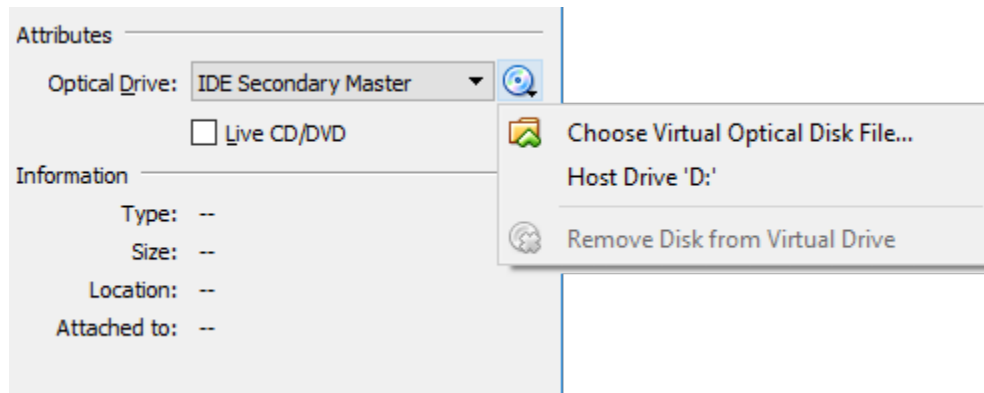
8. Choose the hard disk size and storage location. The Ubuntu system requirements recommend 25GB. Remember, we choose *Dynamically allocated* as our storage option in the last step, so we won't consume all this disk space immediately. Rather, VirtualBox will allocate it as required, up to the maximum 25GB we specified. Click **Create**
9. The wizard will finish and we are returned to the main VirtualBox window. Click **Settings**



10. In the left pane select **Storage**, then in the right select the CD icon with the word Empty beside it.



11. Under *Attributes* click the CD icon (highlighted in the screenshot above) and select **Choose Virtual Optical Disk File** and browse to the downloaded file ubuntu-16.04.1- desktop-amd64.iso

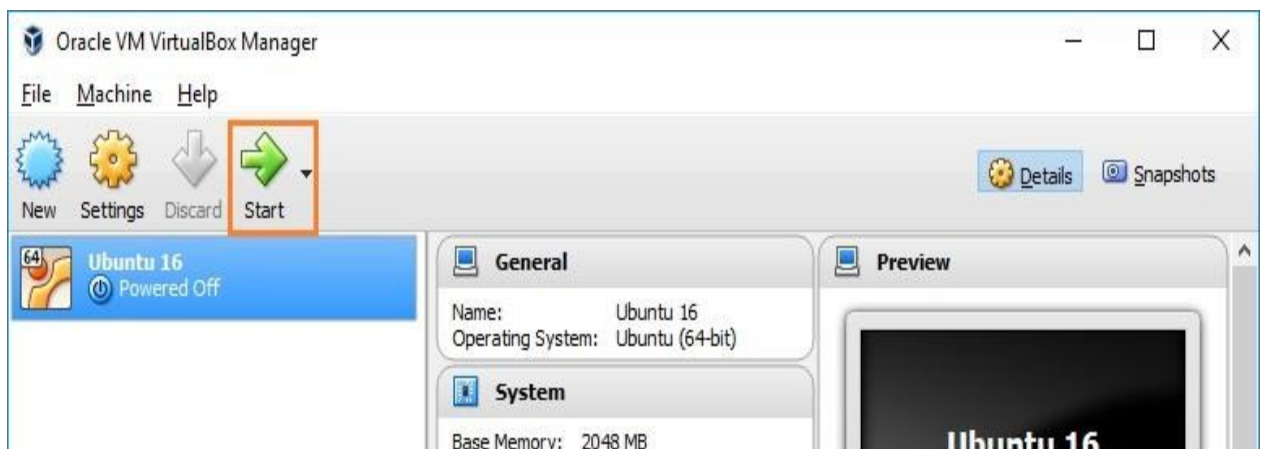


12. Click **OK** to close the *Settings* dialog window. The virtual machine should now be ready to start.

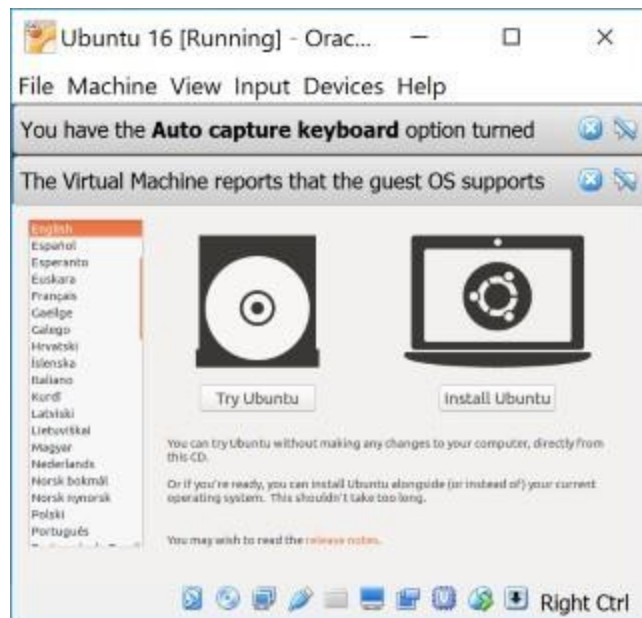
Install Ubuntu

In VirtualBox your VM should be showing as *Powered Off*, and the optical drive configured to point to the Ubuntu ISO file we downloaded previously.

1. In VirtualBox, select the virtual machine **Ubuntu 16** and click **Start**. VirtualBox will launch a new window with the vm and boot from the iso.

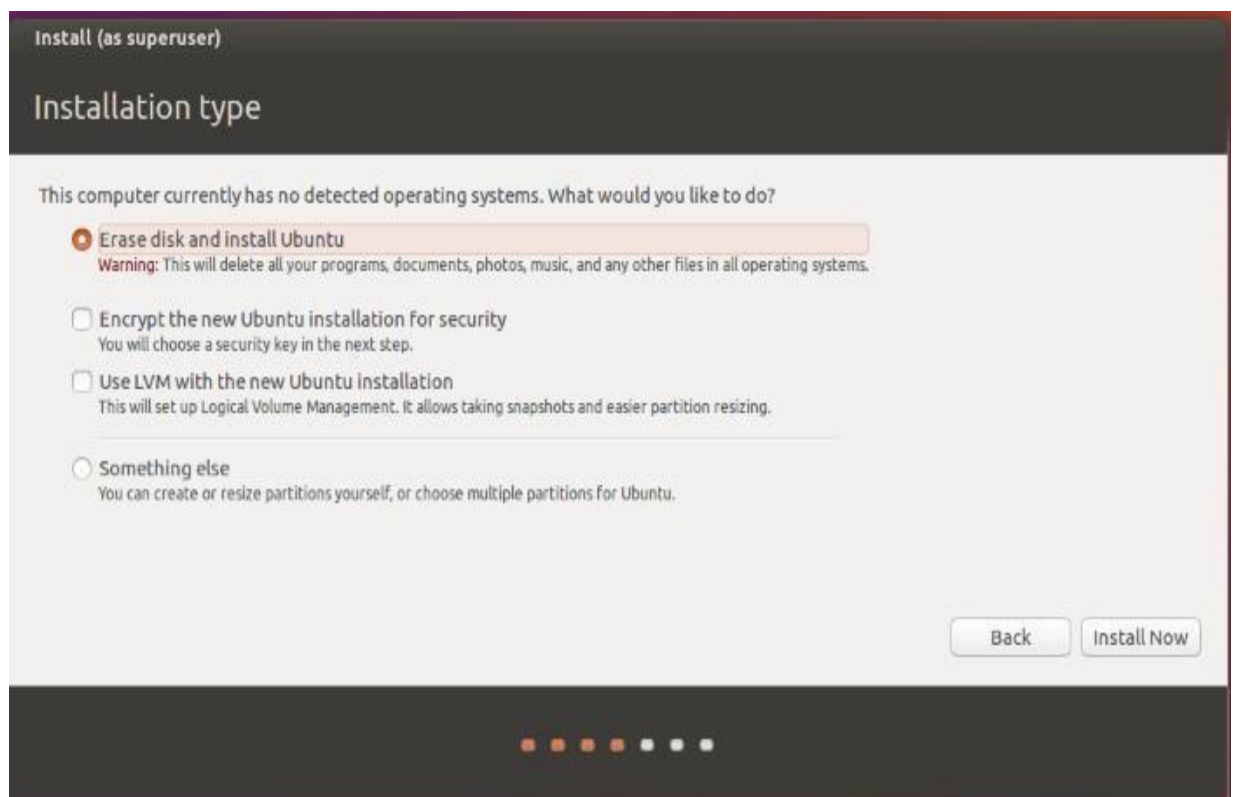


2. Click **Install Ubuntu**

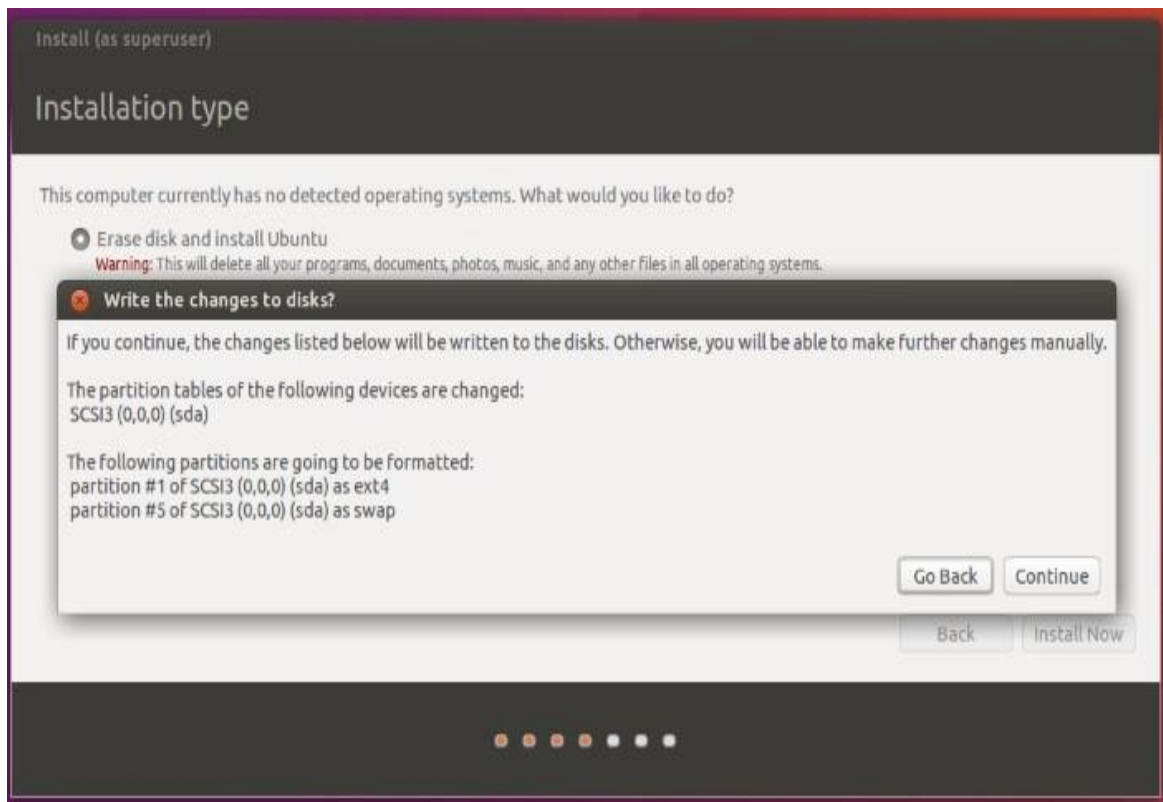


3. Select **Download updates while installing Ubuntu** and click **Continue**

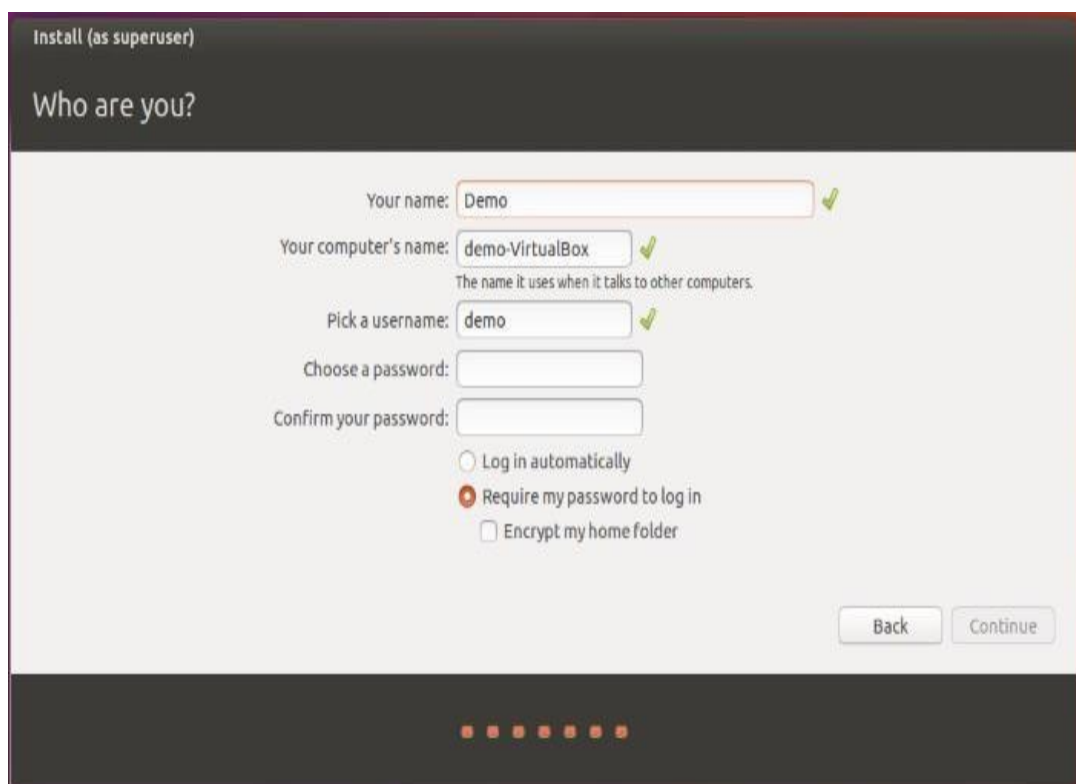
4. On the next screen accept the default of **Erase disk and install Ubuntu** and click **Install Now**



5. You will be prompted with a warning saying the changes will be written to disk. Click **Continue**



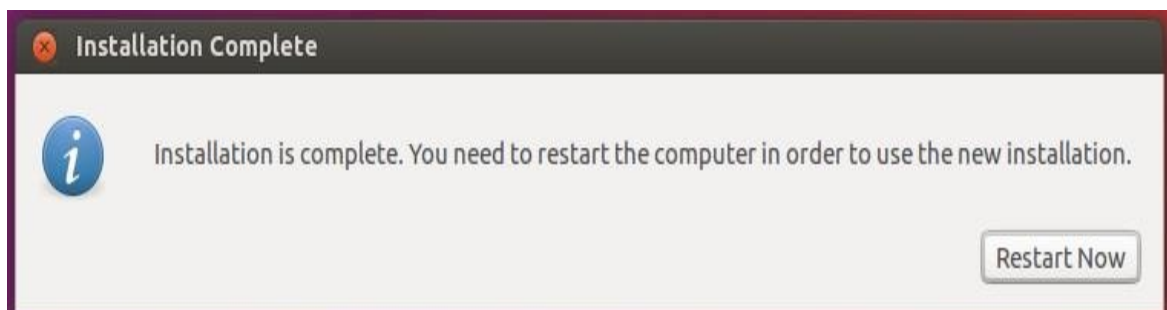
6. Select your timezone and click **Continue**
7. Select your keyboard layout. I accepted the default of **English (US)** and click **Continue**
8. Enter a username and password, then click **Continue**



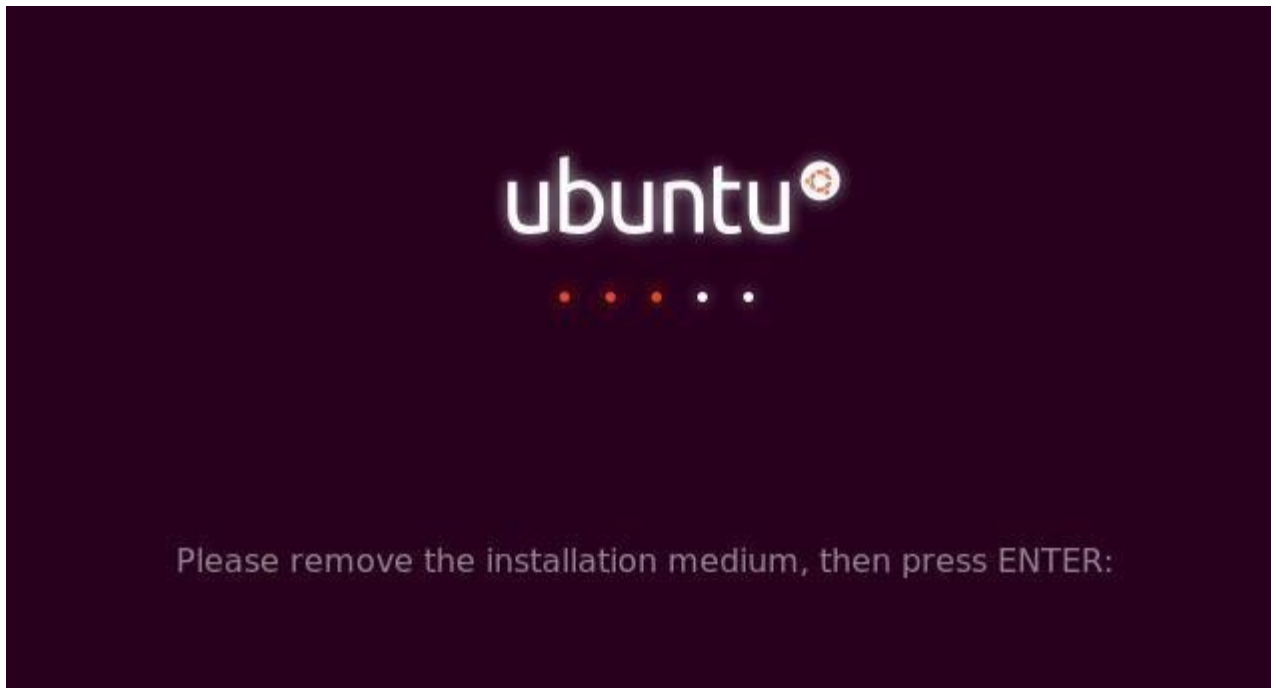
9. The Ubuntu installation may take several minutes to run, so have another coffee.



10. When the installation is finished you will be prompted to restart. Save and close anything else you may have open and click **Restart Now**



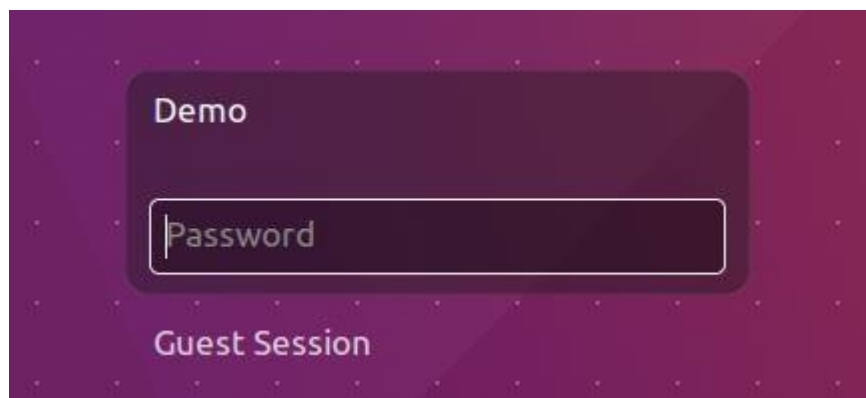
11. Now when the vm reboots you may see this message.



From the menu select **Machine > Settings**.

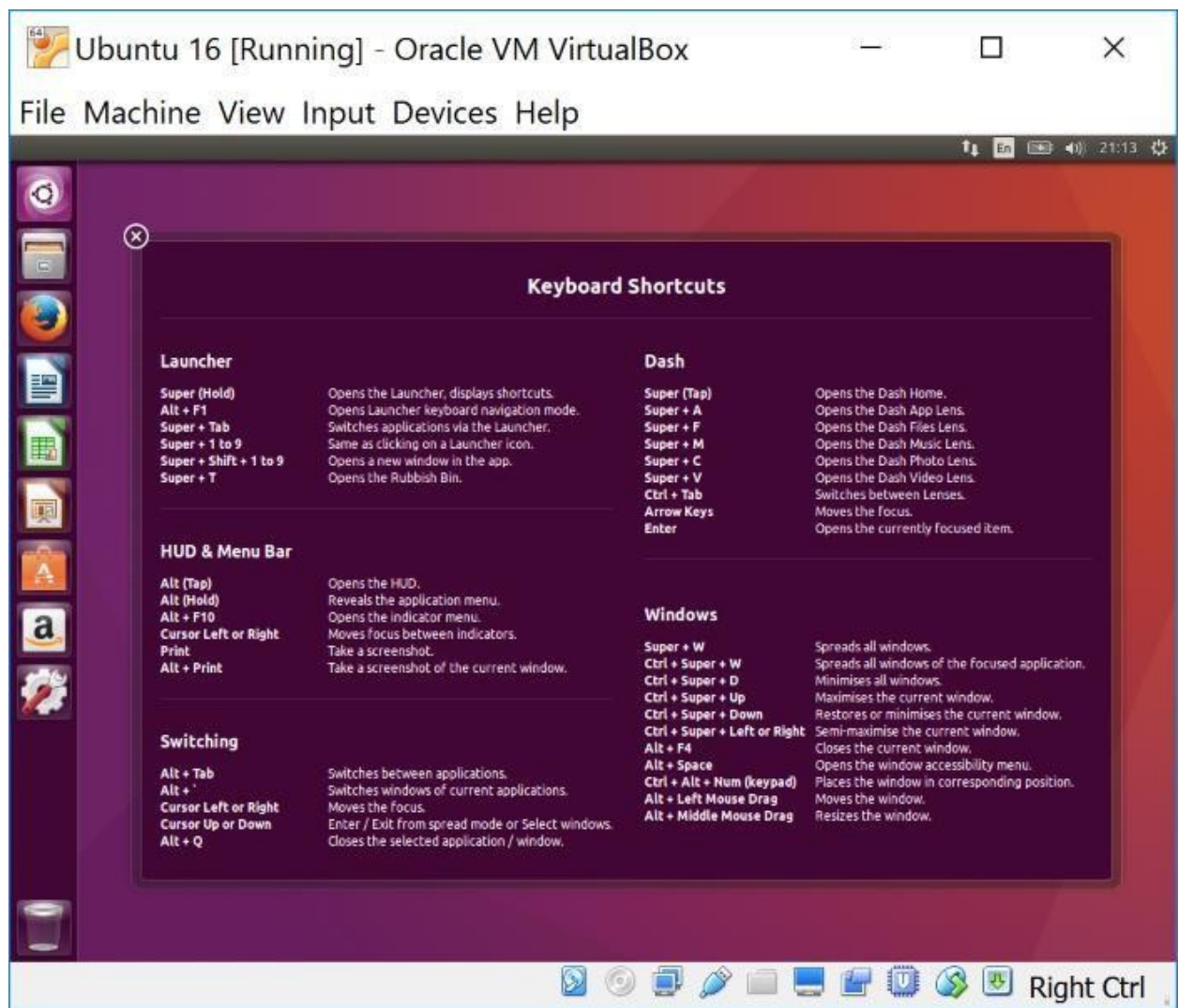
Navigate back into the **Storage** settings where we previously selected the iso file. If the Ubuntu iso file is still there, remove it. Otherwise close the *Settings* window and in the vm press **Enter** to proceed.

12. If all went well the VM should boot to the Ubuntu login screen. Enter your password to continue.



Ubuntu should run normally in the VirtualBox environment. If everything is far too small, you can adjust the 'zoom' by selecting *View > Scale Factor > 200%*.

Have fun!



Result:

Thus the Virtual box/Oracle Virtual machine has installed Successfully.

EX.NO:2	INSTALL A C COMPILER IN THE VIRTUAL MACHINE AND EXECUTE A SAMPLE PROGRAM.
DATE:	

Aim :

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

Procedure:

step1:

Install the centos or ubuntu in the VMware or Oracle Virtual Machine as per previous commands.

Step 2:

Login into the VM of installed OS.

Step 3:

If it is ubuntu then, for gcc installation

```
$ sudo add-apt-repository ppa:ubuntu-toolchain-r/test
```

```
$ sudo apt-get update
```

```
$ sudo apt-get install gcc-6 gcc-6-base
```

Step 4:

Write a sample program like

Welcome.cpp

```
#include<iostream.h>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
cout<<"Hello world";
```

```
return 0;
```

```
}
```

Step 5:

First we need to compile and link our program. Assuming the source code is saved in a file `welcome.cpp`, we can do that using GNU C++ compiler `g++`, for example `g++ -Wall -o welcome welcome.cpp` and output can be executed by `./welcome`

Result:

Thus the GCC compiler has installed and executed in this sample program successfully.

EX.NO:3

**INSTALL GOOGLE APP ENGINE. CREATE HELLO WORLD
APP AND OTHER SIMPLE WEB APPLICATIONS USING
PYTHON/JAVA.**

DATE:


AIM:

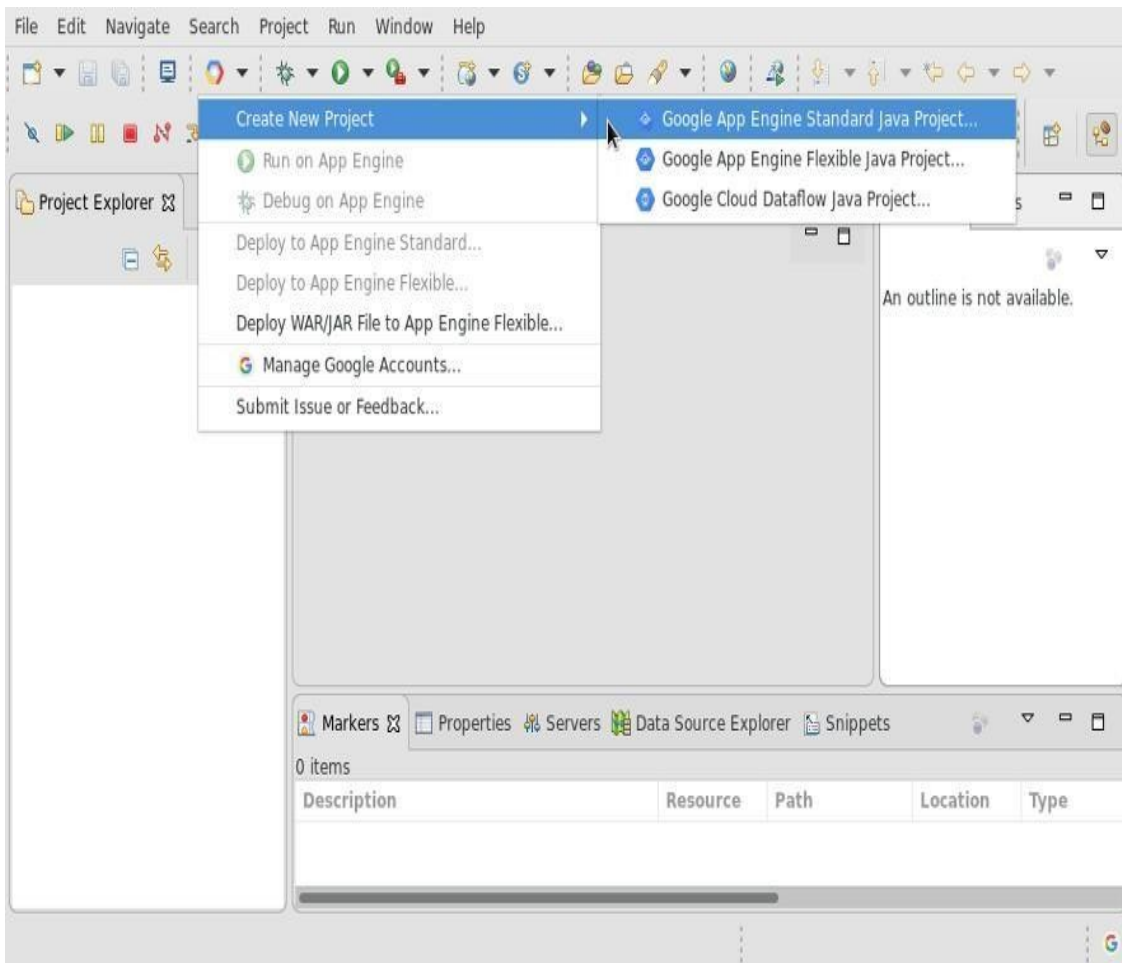
To install Google App Engine. Create *hello world* app and other simple web applications using python/java.

PROCEDURE:

To create a new App Engine standard project in Eclipse

Steps to creation:

1. Click the **Google Cloud Platform** toolbar button  .
2. Select Create New Project > Google App Engine Standard Java Project.



New App Engine Standard Project

Create a new Eclipse project for App Engine standard environment development.

Project name: my-project

☒ Use default location

Location: /home/elharo/workspaces/qa6/my-project Browse...

Java version: Java 8, Servlet 3.1

Java package: com.example

App Engine service: default

☐ Create as Maven project

Maven project coordinates

Group ID:

Artifact ID:

Version: 0.1.0-SNAPSHOT

< Back Next > Cancel Finish

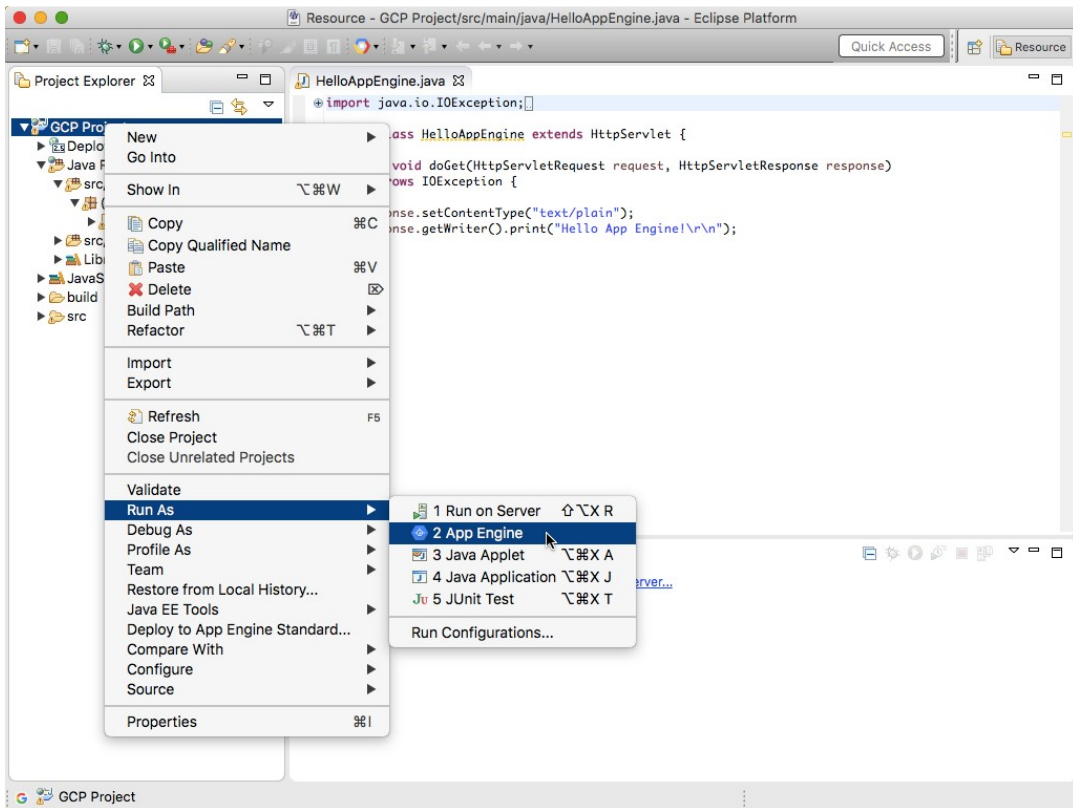
3. To create a Maven-based App Engine project, check **Create as Maven Project** and enter a Maven **Group ID** and **Artifact ID** of your choosing to set the coordinates for this project. The **Group ID** is often the same as the package name, but does not have to be. The **Artifact ID** is often the same as or similar to the project name, but does not have to be.
4. Click **Next**.
5. Select any libraries you need in the project.
6. Click **Finish**.

The wizard generates a native Eclipse project, with a simple servlet, that you can run and deploy from the IDE.

Running the project locally

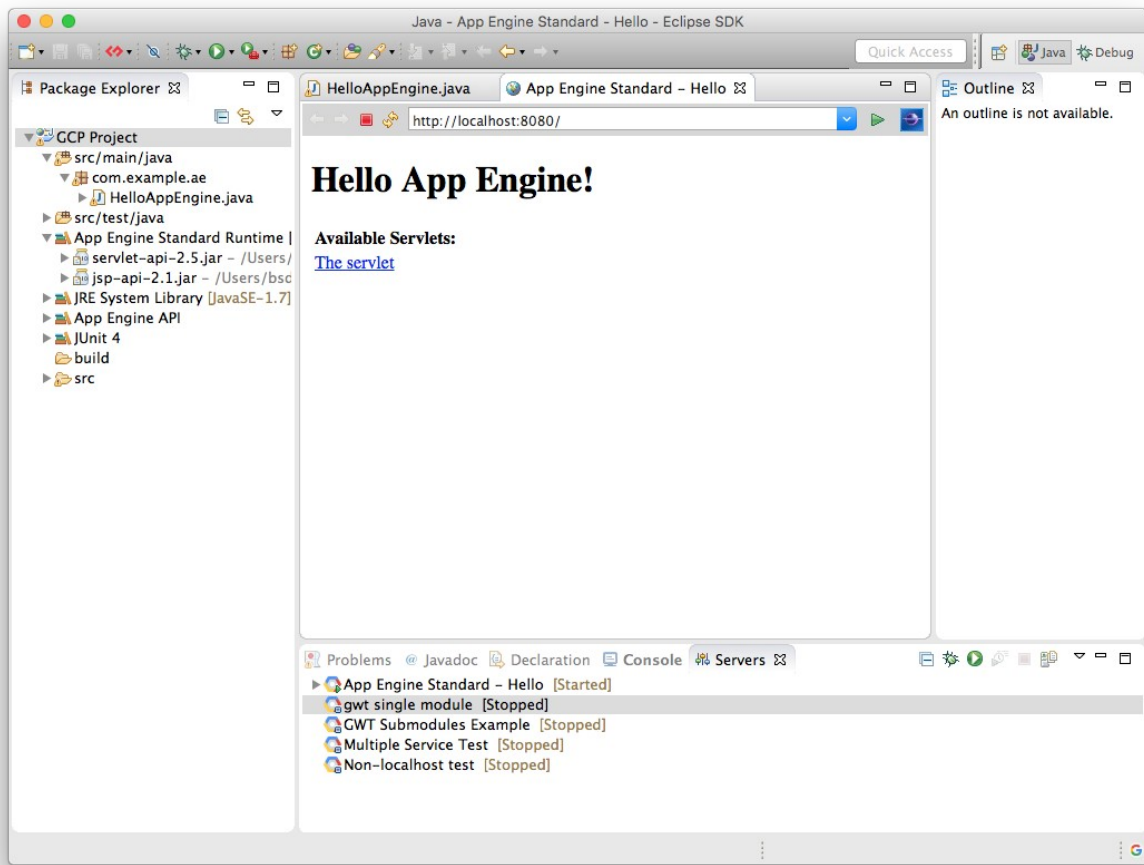
Steps for Running the Project locally:

1. Select the project in the Project Explorer or Package Explorer.
2. Open the context menu.
3. Select **Run As > App Engine**.



1. Log messages appear in the console as the server starts up.
2. Eclipse opens its internal web browser to your application. You can also open an external browser and navigate to <http://localhost:8080>. Either way, you'll see a static HTML page with a link to the servlet.

Note: You may see a message that says, **Port 8080 already in use**. If so, you can [run your application on a different host or port](#).



Debugging the project locally

To debug your project locally, complete the [running the project locally](#) steps, except select **Debug As > App Engine** instead of **Run As > App Engine** in the context menu.

The server stops at the breakpoints you set and shows the Eclipse debugger view.

Running App Engine Standard Apps on a Different Host or Port

To run your App Engine standard application on a different host or port:

1. Right-click your project.
2. Select **Run As > Run on Server**.



Note: You can also select **Debug As > Debug on Server** to debug your application on a different host or port.

1. In the dialog, select **Manually define a new server**.
2. Select **App Engine Standard** as the **server type**.
3. Enter the hostname in the **Server's host name** field.
4. Enter the port in the **Server port** field.
5. Click **Finish**.

Configuring Eclipse

To configure Cloud Tools for Eclipse to use Objectify:

1. In Eclipse, select **Run > Run Configurations**.
2. In the **Run Configurations** dialog, select an existing **App Engine Local Server** launch configuration, or click the **New launch configuration** button to create one.
3. Select the **Cloud Platform** tab of your run configuration.
4. Select an account.
5. Select a project to assign a project ID to be used in the local run. It doesn't matter which project you select because you won't actually connect to it.
6. As an alternative, if you aren't logged in or don't have a Cloud Project, you can instead set the **GOOGLE_CLOUD_PROJECT** environment variable to a legal string, such as **MyProjectId**, in the **Environment** tab of the run configuration.

Result:

Thus the Google App Engine has installed and executed in this sample program effectively.

EX.NO:4

**USE GAE LAUNCHER TO LAUNCH THE WEB
APPLICATIONS**

DATE:

AIM:

To use GAE launcher to launch the web applications(Eclipse)

Procedure:

Step1:

Install an Eclipse and create GAE web application as per previous commands.

Step2 :

Deploying App Engine Standard Applications from Eclipse

The steps of creating a new App Engine app in the Google Cloud Console, authenticating with Google, and deploying your project to App Engine.

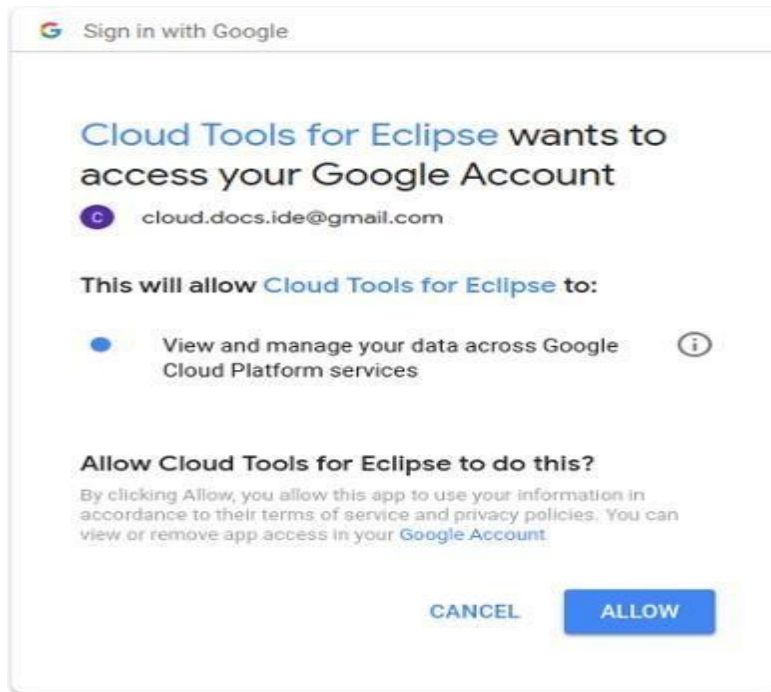
Before you begin

You need a Google Cloud project with an App Engine application to deploy to. If you don't already have one, use the Google Cloud Console to set up your Cloud project:

1. Select or create a new Cloud project.
2. Sign in to a Google account that is used to deploy your project to App Engine.
3. Select **File > Sign in to Google**.

If you see **Manage Google Accounts** instead of the **Sign in to Google** option, that means you are already signed in, so you can skip these account sign in steps.

4. Your system browser opens outside of Eclipse and asks for the permissions it needs to manage your App Engine Application.



5. Click **Allow** and close the window. Eclipse is now signed into your account.
6. Ensure that the `appengine-web.xml` file is in the `WEB-INF` folder of your web application.
7. Ensure that the project has the *App Engine Project* facet. If you created it using the wizard, it should already have this facet. Otherwise:
8. Right click the project in the Package Explorer to bring up the context menu.
9. Select **Configure > Convert to App Engine Project**.

Deploy the Project to App Engine

To deploy the project to App Engine standard environment:

1. Right click the project in the Package Explorer to open the context menu.
2. Select **Deploy to App Engine Standard**.
3. A dialog pops up.
4. Select the account you want to deploy with, or add a new account.
5. The list of projects the account has access to loads. Select the one you want to deploy to.
6. Click OK.

Deployment Parameters for "my-web-app"



Account:

Project: *You can create a new Google Cloud Platform project in the [Cloud Console](#). Then refresh this list.*

Name	ID
gcp-web-app-test	gcp-web-app-test
my-gcp-test-project	my-gcp-test-project-111
my-new-app	my-new-app-206616
new-project-test	new-project-test-206616
prod-web-app	prod-web-app
sample-web-app	sample-web-app-206616
staging-web-app	staging-web-app

Version:

☒ Promote the deployed version to receive all traffic

☒ Stop previous version

☒ Include optional App Engine configuration files

▶ **Advanced**

A background job launches that deploys the project to App Engine. The output of the job is visible in the Eclipse Console view.

By default, App Engine stops the previous version of your application and immediately promotes your new code to receive all traffic. If you'd rather manually promote it later using [gcloud](#) or the [Google Cloud Console](#), uncheck **Promote the deployed version to receive all traffic**. If you don't want to stop the previous version, uncheck **Stop previous version**.

Result:

Thus the Google App Engine has launched in this sample program agreeably.

EX.NO:5

DATE:

**SIMULATE A CLOUD SCENARIO USING CLOUDSIM AND
RUN A SCHEDULING ALGORITHM THAT IS NOT IN
CLOUDSIM**

Aim :

To Simulate a cloud scenario using CloudSim and run a scheduling algorithm.

Procedure:

1. Before you start, It is essential that the cloudsim should already installed/setup on your local computer machine. In case you are yet to install it, you may follow the process of [Cloudsim setup using Eclipse IDE](#)

```
public static void main(String[] args)
```

2. The main() method is the pointer from where the execution of this example starts
3. There are eleven steps that are followed in each example with some variation in them, specified as follows:

Step1 :Set the Number of users for the current simulation. This user count is directly

```
int num_user = 1; // number of cloud users  
Calendar calendar = Calendar.getInstance();  
boolean trace_flag = false;
```

proportional to a number of brokers in the current simulation.

- **Step 2:** Initialize the simulation, provided with the current time, number of users and trace flag.

```
CloudSim.init(num_user, calendar, trace_flag);
```

```
Datacenter datacenter0 = createDatacenter("Datacenter_0");
```

- **Step 3:** Create a Datacenter.

4. where the createDatacenter() method itself initializes the various datacenter

characteristics along with the host list. This is the most important entity without this there is no way the simulation of hosting the virtual machine is applicable.

```
private static Datacenter createDatacenter(String name)
{
    List<Host> hostList = new ArrayList<Host>();
    List<Pe> peList = new ArrayList<Pe>();

    int mips = 1000;

    peList.add(new Pe(0, new PeProvisionerSimple(mips))); int
    hostId = 0;

    int ram = 2048; // host memory (MB) long
    storage = 1000000; // host storage int bw
    = 10000;

    hostList.add(
        new Host(
            hostId,
            new RamProvisionerSimple(ram),
            new BwProvisionerSimple(bw),
            storage,
            peList,
```



```

        new VmSchedulerTimeShared(peList)
    )
    );

    String arch =
    "x86"; String
    os = "Linux";
    String vmm =
    "Xen"; double
    time_zone =
    10.0; double
    cost = 3.0;

    double costPerMem
    = 0.05; double
    costPerStorage =
    0.001;

    double costPerBw = 0.0;

    LinkedList<Storage> storageList = new
    LinkedList<Storage>(); DatacenterCharacteristics
    characteristics = new DatacenterCharacteristics(arch, os,
    vmm, hostList,

```

Step:4 Create a datacenter broker

```

        time_zone cost costPerMem
DatacenterBroker broker = createBroker();
int brokerId = broker.getId();

```

Where the createBroker() method initializes the entity object from DatacenterBroker class

```

private static DatacenterBroker createBroker()
{
    try {
        DatacenterBroker broker = null;
        datacenter = new Datacenter(name,
        try {
            characteristics = new
            broker = new DatacenterBroker("Broker");
        } catch (Exception e) {
            VmAllocationPolicySimple(hostList),
            e.printStackTrace();
            storageList, 0);
            return null;
        } catch (Exception e) {
            return broker;
        }
        e.printStackTrace();
    }

    return datacenter;
}

```

- **Step 5:** Create a Virtual Machine(s).

```
vmList = new
ArrayList<Vm>(); int vmid
= 0;

int mips =
1000; long
size = 10000;
9oint ram =
512; long bw
= 1000;

int pesNumber = 1;

Vm vm = new Vm(vmid, brokerId, mips, pesNumber, ram, bw, size, vmm,
new CloudletSchedulerTimeShared());

vmList.add(vm);
```

- **Step 6:** Submit Virtual Machine to Datacenter broker.

```
broker.submitVmList(vmList);
```

- **Step 7:** Create Cloudlet(s) by specifying their characteristics.

```
cloudletList = new ArrayList<Cloudlet>();

int id = 0;
long length =
400000; long
fileSize = 300;
long
outputSize =
300;
UtilizationModel utilizationModel = new UtilizationModelFull();

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

cloudlet.setUserId(brokerId);
cloudlet.setVmId(vmid);
```

- **Step 8:** Submit Cloudlets to Datacenter broker.

```
broker.submitCloudletList(cloudletList);
```

- **Step 9:** Send call to Start Simulation.

```
CloudSim.startSimulation();
```

- **Step 10:** Once no more event to execute, send the call to Stop Simulation.

```
CloudSim.stopSimulation();
```

- **Step 11 :** Finally, print the final status of the Simulation.

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

Where printCloudletList() method formats the output to correctly display it on the console.

```
private static void printCloudletList(List<Cloudlet> list)  
{  
    int size = list.size();  
    Cloudlet cloudlet;  
    String indent = "    ";  
    Log.println();  
  
    Log.println("===== OUTPUT =====");  
  
    Log.println("Cloudlet ID" + indent + "STATUS" + indent  
               + "Data center ID" + indent + "VM ID" +  
               indent + "Time" + indent  
               + "Start Time" + indent + "Finish Time");
```

```

DecimalFormat dft = new DecimalFormat("###.##"); for
(int i = 0; i < size; i++)

{

    cloudlet = list.get(i);

    Log.print(indent + cloudlet.getCloudletId() + indent +
    indent);

    if (cloudlet.getCloudletStatus() == Cloudlet.SUCCESS)
    {

        Log.print("SUCCESS");
        Log.println(indent + indent +

        cloudlet.getResourceId()

        + indent + indent + indent +

        cloudlet.getVmId())

```

Once you Run the example the output for cloudsimeExample1.java will be displayed like:

```

+ indent + indent +
dft.format(cloudlet.getActualCPUTime())

+ indent + indent +
dft.format(cloudlet.getExecStartTime())

```

```

eclipse-workspace - Cloudsim/examples/org/cloudbus/cloudsim/examples/CloudSimExample1.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer Type Hierarchy
Cloudsim
  JRE System Library [jre1.8.0_201]
  examples
    org.cloudbus.cloudsim.examples
      CloudSimExample1.java
      CloudSimExample2.java
      CloudSimExample3.java
      CloudSimExample4.java
      CloudSimExample5.java
      CloudSimExample6.java
      CloudSimExample7.java
      CloudSimExample8.java
    org.cloudbus.cloudsim.examples.network
      org.cloudbus.cloudsim.examples.network.datacenter
      org.cloudbus.cloudsim.examples.power
      org.cloudbus.cloudsim.examples.power.planetia
      org.cloudbus.cloudsim.examples.power.random
      workload.planetia
  sources
  Referenced Libraries
  docs
  jars
    build.xml
    changelog.txt
    examples.txt
    license.txt
    pom.xml
    readme.txt
    release_notes.txt

Problems Javadoc Declaration Console Progress Call Hierarchy
<terminated> CloudSimExample1 (3) [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (02-Jun-2019, 3:35:59 PM)
Starting CloudSimExample1...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.1: Broker: VM #0 has been created in Datacenter #2, Host #0
0.1: Broker: Sending cloudlet 0 to VM #0
400.1: Broker: Cloudlet 0 received
400.1: 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   VM ID   Time   Start Time   Finish Time
0            SUCCESS   2              0       400    0.1          400.1
CloudSimExample1 finished!

```

Result :

Thus the Simulation of a cloud scenario using CloudSim and run a scheduling algorithm has implemented successfully

EX.NO:6

DATE:

**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:

Step 1: Open Opennebula service from root user and view in localhost:9869

```
root@linux:~$ /etc/init.d/opennebula-sunstone restart
```

Step 2: Create oneimage, onetemplate and one vm as like earlier Creating oneimage

```
oneadmin@linux:~/datastores$ oneimage create --name "Ubuntu" --path  
"/home/linux/Downloads/source/tubuntu1404-5.0.1.qcow2c" --driver qcow2 -- datastore default
```

Creating One Template:

```
oneadmin@linux:~/datastores$ onetemplate create --name "ubuntu1" --cpu 1 --vcpu 1 --  
memory 1024 --arch x86_64 --disk "Ubuntu" --nic "private" --vnc --ssh
```

Instantiating OneVm (oneemplate)

```
oneadmin@linux:~/datastores$ onetemplate instantiate "ubuntu1"
```

**Step 3: To perform a migration. We use onevm command with VM id as VID = 0 to
host02(HID=1)**

```
oneadmin@linux:~/datastores$ onevm migrate --live 0 1
```

This will move the VM from host01 to host02. The onevm list shows something like the following

```
oneadmin@linux:~/datastores$ onevm list
```

ID	USER	GROUP	NAME	STAT	CPU	MEM	HOSTNAME	TIME
0	oneadmin	oneadmin	one-0	runn	0	0k	host02	00:00:48

Result :

Thus the virtual machine transfer the files from one virtual machine to another virtual machine from one node to the other has executed successfully.

EX.NO:7

DATE:

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`

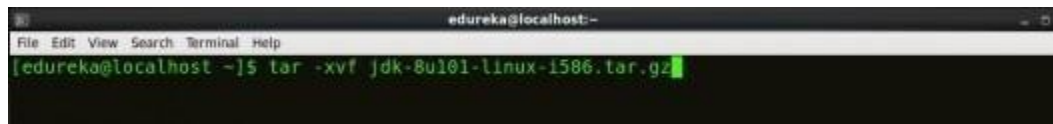
A terminal window titled 'edureka@localhost:~' with a menu bar (File, Edit, View, Search, Terminal, Help). The command '[edureka@localhost ~]\$ tar -xvf jdk-8u101-linux-i586.tar.gz' is entered and executed, with a green cursor at the end of the line.

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`

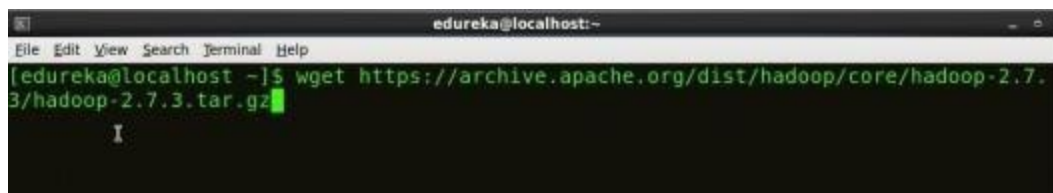
A terminal window titled 'edureka@localhost:~' with a menu bar (File, Edit, View, Search, Terminal, Help). The command '[edureka@localhost ~]\$ wget https://archive.apache.org/dist/hadoop/core/hadoop-2.7.3/hadoop-2.7.3.tar.gz' is entered and executed, with a green cursor at the end of the line.

Fig: Hadoop Installation – Downloading Hadoop

Step 4: Extract the Hadoop tar File.

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

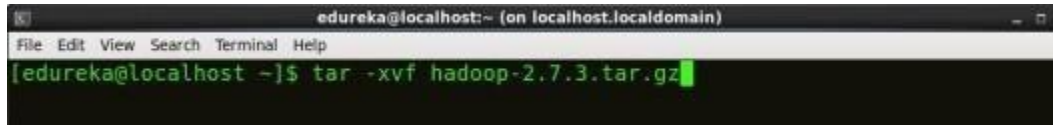
A terminal window titled 'edureka@localhost:~ (on localhost.localdomain)' with a menu bar (File, Edit, View, Search, Terminal, Help). The command '[edureka@localhost ~]\$ tar -xvf hadoop-2.7.3.tar.gz' is entered and the cursor is at the end of the line.

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

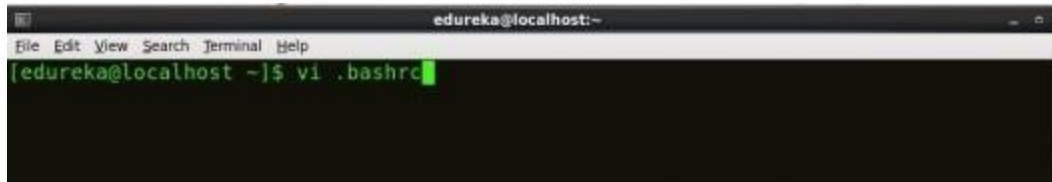
A terminal window titled 'edureka@localhost:~' with a menu bar (File, Edit, View, Search, Terminal, Help). The command '[edureka@localhost ~]\$ vi .bashrc' is entered and the cursor is at the end of the line.A screenshot of the .bashrc file content. It shows a section for user-specific aliases and functions, followed by export statements for Hadoop environment variables (HADOOP_HOME, HADOOP_CONF_DIR, HADOOP_MAPRED_HOME, HADOOP_COMMON_HOME, HADOOP_HDFS_HOME, YARN_HOME, and PATH) and a section for setting JAVA_HOME (export JAVA_HOME=/home/edureka/jdk1.8.0_101 and export PATH=/home/edureka/jdk1.8.0_101/bin:\$PATH).

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

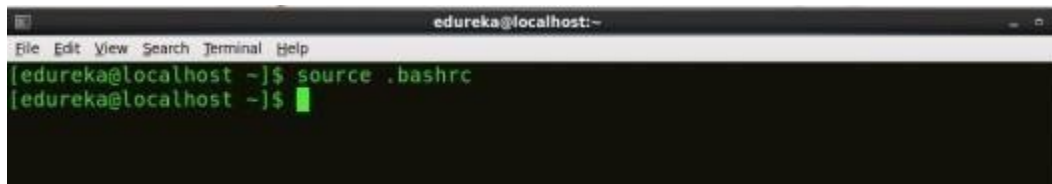
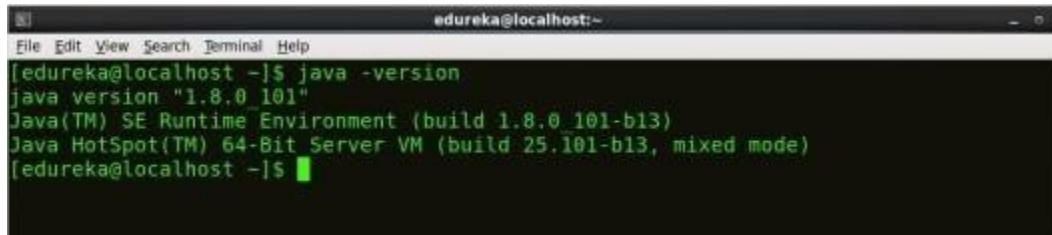
A terminal window titled 'edureka@localhost:~' with a menu bar (File, Edit, View, Search, Terminal, Help). The command '[edureka@localhost ~]\$ source .bashrc' is entered, and the prompt changes to '[edureka@localhost ~]\$'.

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

A screenshot of a terminal window titled 'edureka@localhost:~'. The terminal has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The command '[edureka@localhost ~]\$ java -version' has been entered and executed. The output is displayed in green text on a black background: 'java version "1.8.0_101"', 'Java(TM) SE Runtime Environment (build 1.8.0_101-b13)', and 'Java HotSpot(TM) 64-Bit Server VM (build 25.101-b13, mixed mode)'. The prompt '[edureka@localhost ~]\$' is followed by a green cursor.

```
edureka@localhost:~  
File Edit View Search Terminal Help  
[edureka@localhost ~]$ java -version  
java version "1.8.0_101"  
Java(TM) SE Runtime Environment (build 1.8.0_101-b13)  
Java HotSpot(TM) 64-Bit Server VM (build 25.101-b13, mixed mode)  
[edureka@localhost ~]$
```

Fig: Hadoop Installation – Checking Java Version

Command: `hadoop version`

```
edureka@localhost:~$ hadoop version
Hadoop 2.7.3
Subversion https://git-wip-us.apache.org/repos/asf/hadoop.git -r baa91f7c6bc9cb92be5982de4719c1c8af91ccff
Compiled by root on 2016-08-18T01:41Z
Compiled with protoc 2.5.0
From source with checksum 2e4ce5f957ea4db193bce3734ff29ff4
This command was run using /home/edureka/hadoop-2.7.3/share/hadoop/common/hadoop-common-2.7.3.jar
[edureka@localhost ~]$
```

Fig: Hadoop Installation – Checking Hadoop Version

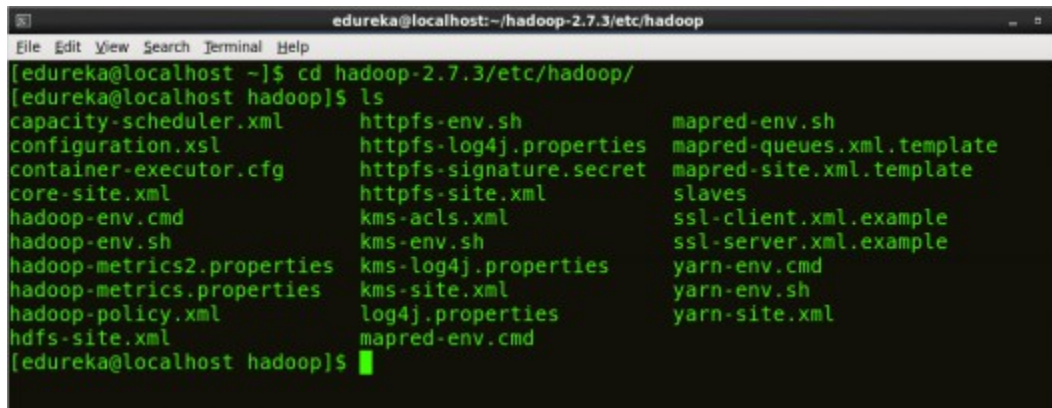
Step 6: Edit the **Hadoop Configuration files**.

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:



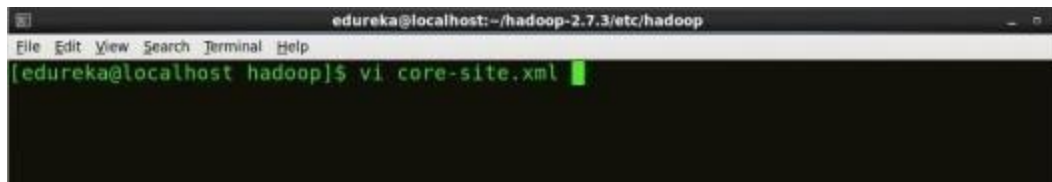
```
edureka@localhost:~/hadoop-2.7.3/etc/hadoop
File Edit View Search Terminal Help
[edureka@localhost ~]$ cd hadoop-2.7.3/etc/hadoop/
[edureka@localhost hadoop]$ ls
capacity-scheduler.xml      httpfs-env.sh              mapred-env.sh
configuration.xml           httpfs-log4j.properties   mapred-queues.xml.template
container-executor.cfg      httpfs-signature.secret   mapred-site.xml.template
core-site.xml               httpfs-site.xml            slaves
hadoop-env.cmd              kms-acls.xml               ssl-client.xml.example
hadoop-env.sh               kms-env.sh                 ssl-server.xml.example
hadoop-metrics2.properties kms-log4j.properties      yarn-env.cmd
hadoop-metrics.properties  kms-site.xml               yarn-env.sh
hadoop-policy.xml           log4j.properties          yarn-site.xml
hdfs-site.xml               mapred-env.cmd
```

Fig: Hadoop Installation – Hadoop Configuration Files


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

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

Command: vi core-site.xml

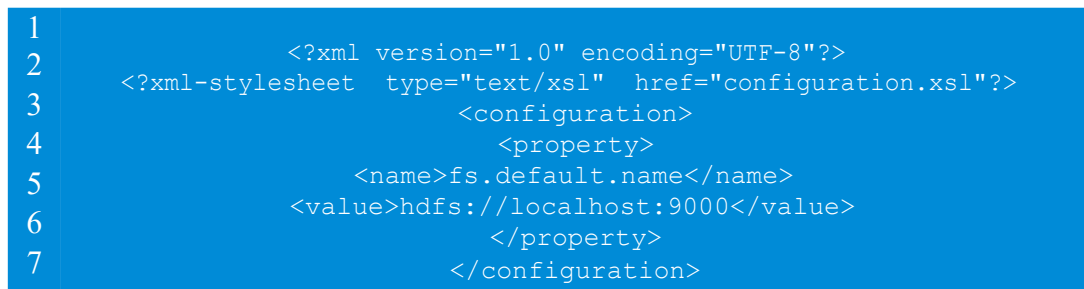


```
edureka@localhost:~/hadoop-2.7.3/etc/hadoop
File Edit View Search Terminal Help
[edureka@localhost hadoop]$ vi core-site.xml
```



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

Fig: Hadoop Installation – Configuring core-site.xml

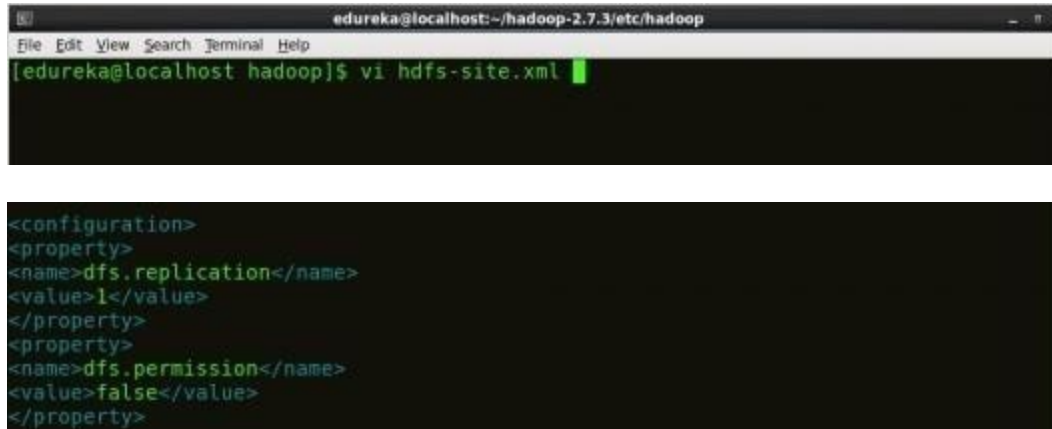


```
1      <?xml version="1.0" encoding="UTF-8"?>
2      <?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
3          <configuration>
4              <property>
5                  <name>fs.default.name</name>
6                  <value>hdfs://localhost:9000</value>
7                  </property>
8              </configuration>
```

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.

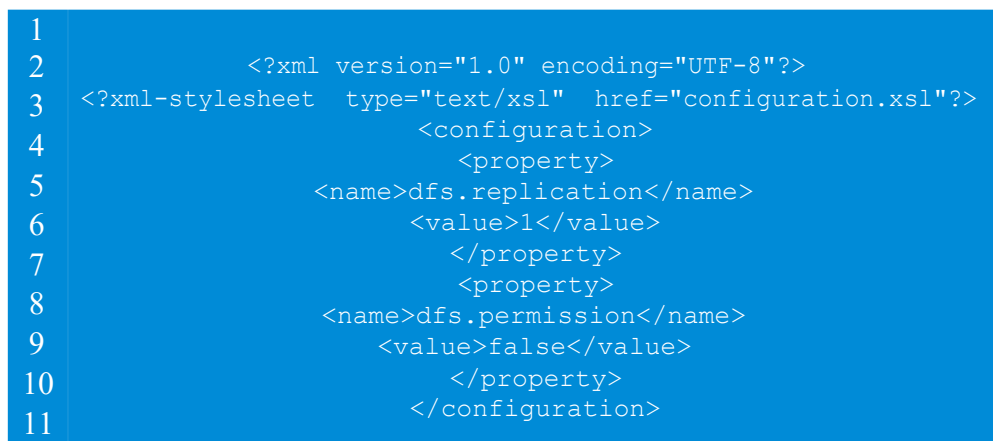
Command: vi hdfs-site.xml



```
edureka@localhost:~/hadoop-2.7.3/etc/hadoop
File Edit View Search Terminal Help
[edureka@localhost hadoop]$ vi hdfs-site.xml

<configuration>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
<property>
<name>dfs.permission</name>
<value>false</value>
</property>
```

Fig: Hadoop Installation – Configuring hdfs-site.xml



```
1
2      <?xml version="1.0" encoding="UTF-8"?>
3  <?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
4      <configuration>
5          <property>
6              <name>dfs.replication</name>
7              <value>1</value>
8          </property>
9          <property>
10             <name>dfs.permission</name>
11             <value>false</value>
12         </property>
13     </configuration>
```

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

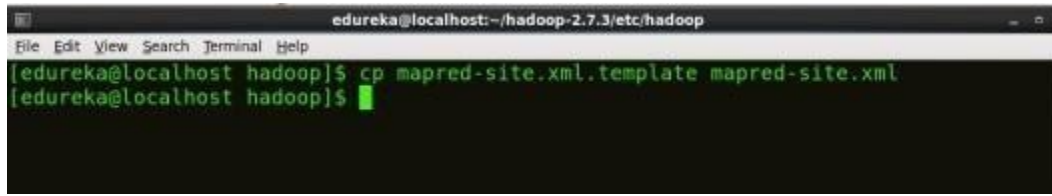
inside configuration tag:

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.

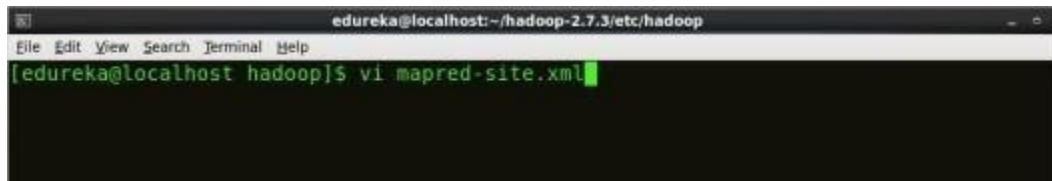
In some cases, *mapred-site.xml* file is not available. So, we have to create the *mapred-site.xml* file using *mapred-site.xml* template.

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


Command: vi mapred-site.xml.



```
edureka@localhost:~/hadoop-2.7.3/etc/hadoop
File Edit View Search Terminal Help
[edureka@localhost hadoop]$ cp mapred-site.xml.template mapred-site.xml
[edureka@localhost hadoop]$
```

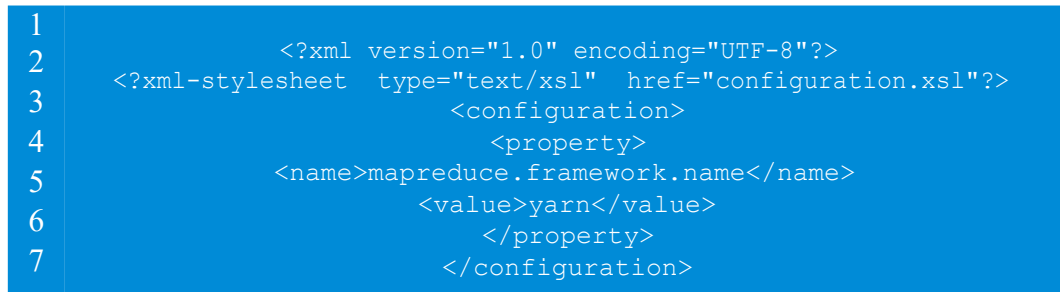


```
edureka@localhost:~/hadoop-2.7.3/etc/hadoop
File Edit View Search Terminal Help
[edureka@localhost hadoop]$ vi mapred-site.xml
```



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

Fig: Hadoop Installation – Configuring mapred-site.xml

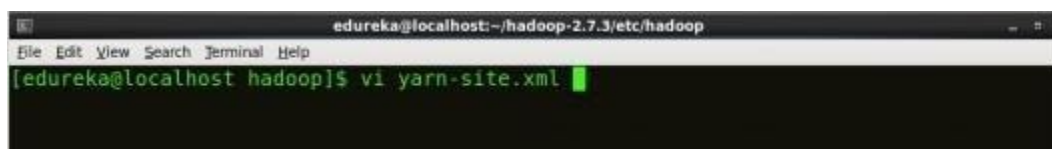


```
1      <?xml version="1.0" encoding="UTF-8"?>
2      <?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
3          <configuration>
4              <property>
5                  <name>mapreduce.framework.name</name>
6                  <value>yarn</value>
7              </property>
          </configuration>
```

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



```
edureka@localhost:~/hadoop-2.7.3/etc/hadoop
File Edit View Search Terminal Help
[edureka@localhost hadoop]$ vi yarn-site.xml
```



```

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

```

Fig: Hadoop Installation – Configuring yarn-site.xml

```

1
2
3      <?xml version="1.0">
4      <configuration>
5          <property>
6              <name>yarn.nodemanager.aux-services</name>
7              <value>mapreduce_shuffle</value>
8          </property>
9          <property>
10             <name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</
11             name>
12             <value>org.apache.hadoop.mapred.ShuffleHandler</value>
13         </property>
14     </configuration>

```

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*



```

# The java implementation to use.
export JAVA_HOME=/home/edureka/jdk1.8.0_101

```

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

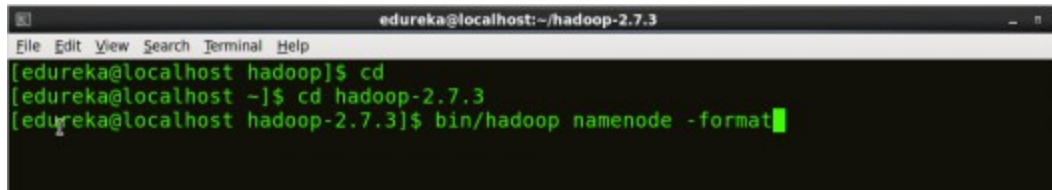
A terminal window titled 'edureka@localhost: ~/hadoop-2.7.3' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows three lines of commands: '[edureka@localhost hadoop]\$ cd', '[edureka@localhost ~]\$ cd hadoop-2.7.3', and '[edureka@localhost hadoop-2.7.3]\$ bin/hadoop namenode -format' with a green cursor at the end.

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

```
edureka@localhost:~/hadoop-2.7.3/sbin
File Edit View Search Terminal Help
[edureka@localhost hadoop-2.7.3]$ cd sbin/
[edureka@localhost sbin]$ ./hadoop-daemon.sh start namenode
starting namenode, logging to /home/edureka/hadoop-2.7.3/logs/hadoop-edureka-namenode-localhost.localdomain.out
[edureka@localhost sbin]$ jps
22113 NameNode
22182 Jps
[edureka@localhost sbin]$
```

Fig: Hadoop Installation – Starting NameNode

Start DataNode:

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`

```
edureka@localhost:~/hadoop-2.7.3/sbin
File Edit View Search Terminal Help
[edureka@localhost sbin]$ ./hadoop-daemon.sh start datanode
starting datanode, logging to /home/edureka/hadoop-2.7.3/logs/hadoop-edureka-datanode-localhost.localdomain.out
[edureka@localhost sbin]$ jps
22113 NameNode
22278 Jps
22206 DataNode
[edureka@localhost sbin]$
```

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`

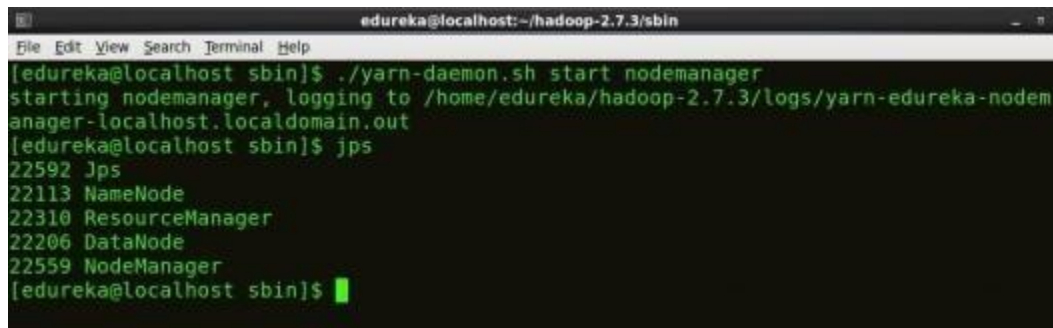
```
edureka@localhost:~/hadoop-2.7.3/sbin
File Edit View Search Terminal Help
[edureka@localhost sbin]$ ./yarn-daemon.sh start resourcemanager
starting resourcemanager, logging to /home/edureka/hadoop-2.7.3/logs/yarn-edureka-resourcemanager-localhost.localdomain.out
[edureka@localhost sbin]$ jps
22113 NameNode
22310 ResourceManager
22345 Jps
22206 DataNode
[edureka@localhost sbin]$
```

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`



```
edureka@localhost:~/hadoop-2.7.3/sbin
File Edit View Search Terminal Help
[edureka@localhost sbin]$ ./yarn-daemon.sh start nodemanager
starting nodemanager, logging to /home/edureka/hadoop-2.7.3/logs/yarn-edureka-nodemanager-localhost.localdomain.out
[edureka@localhost sbin]$ jps
22592 Jps
22113 NameNode
22310 ResourceManager
22206 DataNode
22559 NodeManager
[edureka@localhost sbin]$
```



[See Batch Details](#)

Fig: Hadoop Installation – Starting NodeManager

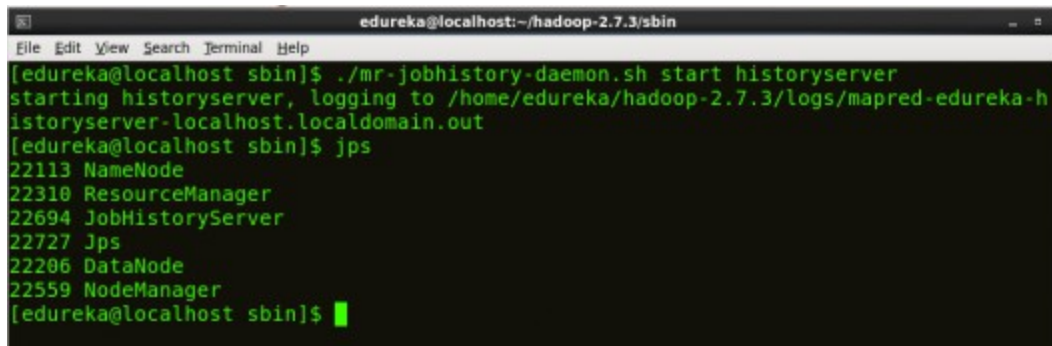
Start JobHistoryServer:

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

A terminal window titled 'edureka@localhost:~/hadoop-2.7.3/sbin' shows the execution of the 'jps' command. The output lists several processes: NameNode (PID 22113), ResourceManager (PID 22310), JobHistoryServer (PID 22694), Jps (PID 22727), DataNode (PID 22206), and NodeManager (PID 22559). The prompt returns to the user after the command is executed.

```
edureka@localhost:~/hadoop-2.7.3/sbin
[edureka@localhost sbin]$ ./mr-jobhistory-daemon.sh start historyserver
starting historyserver, logging to /home/edureka/hadoop-2.7.3/logs/mapred-edureka-h
istoryserver-localhost.localdomain.out
[edureka@localhost sbin]$ jps
22113 NameNode
22310 ResourceManager
22694 JobHistoryServer
22727 Jps
22206 DataNode
22559 NodeManager
[edureka@localhost sbin]$
```

Fig: Hadoop Installation – Checking Daemons

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

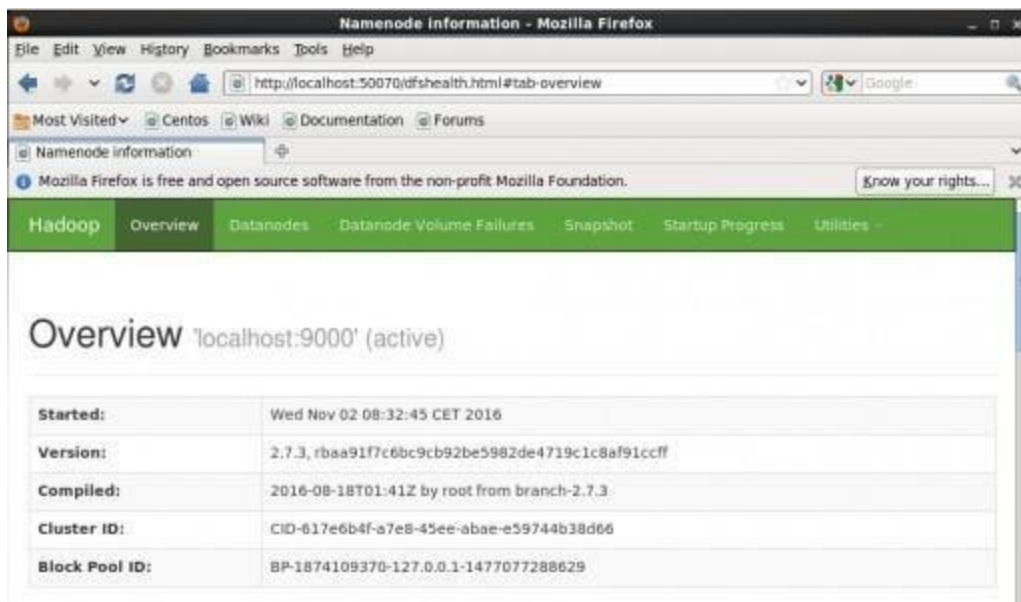


Fig: Hadoop Installation – Starting WebUI

Congratulations, you have successfully installed a single node Hadoop cluster

Result:

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

EX.NO:8

Creating And Executing Your First Container Using Docker.

DATE:

AIM:

To create and execute a container using docker..

STEP:1 Installing Docker on the System

To begin, you will need to install Docker on your system.

Docker provides installers for [Windows](#), [macOS](#), and various flavors of Linux, making it accessible to a wide Range of users.

Below are the commands you can use to install Docker on Ubuntu:

```
1.sudo apt update
```

```
2.sudo apt install apt-transport-https ca-certificates curl software-properties-common
```

```
3.curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

```
4.sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
```

```
5.sudo apt update
```

```
6.sudo apt install docker-ce
```

Once the installation is complete, you can verify it by checking the Docker version and making sure the Docker daemon is running.

STEP:2 Verifying the Installation and Accessing the Docker CLI

For those using the Ubuntu operating system, you can verify the Docker installation by running the following command:

```
1.docker --version
```

```
2.sudo systemctl status docker
```

With Docker successfully installed, you can now access the Docker command-line interface (CLI) to start creating and managing containers.

The CLI provides a set of commands for interacting with Docker, allowing you to build, run, and manage containers with ease.

STEP:3 Crafting Your First Dockerfile

Some of the key concepts in Docker revolve around creating a Dockerfile, which is a text document that contains all the commands a user could call on the command line to assemble an image.

The Dockerfile contains all the information Docker needs to build the image. Let's take a look at how to define a simple Dockerfile and some best practices for writing it.

STEP:4 Defining a Simple Dockerfile

First, let's start by creating a basic Dockerfile.

In this example, we'll create a Dockerfile that simply prints "Hello, World!" when run as a container.

1.FROM alpine

2.CMD echo "Hello, World!"

When defining a simple Dockerfile, it's important to keep it as minimal as possible.

Only include the necessary dependencies and commands required for your application to run within the container.

This helps to keep the image size small and reduces the attack surface, making it more secure.

STEP:5 Best Practices for Writing Dockerfiles

Dockerfiles should follow best practices to ensure consistency, maintainability, and reusability.

One of the best practices is to use the official base images from Docker Hub, as they are well-maintained and regularly updated. It's also important to use specific versions of the base images to avoid unexpected changes.

1.FROM node:14

2.COPY ./app

3.WORKDIR /app

4.RUN npm install

5.CMD ["npm", "start"]

Best practices for writing Dockerfiles also include using a .dockerignore file to specify files and directories to exclude from the context when building the image.

This helps to reduce the build context and improve build performance.

Some additional best practices for writing Dockerfiles include avoiding running commands as root, using multi-stage builds for smaller images, and using environment variables for configuration.

STEP6: Building and Running Your Container

To build and run your Docker container, you will need to follow a few simple steps.

First, you will need to build the [Docker image](#) from your Dockerfile.

Once the image is built, you can run your container using the Docker run command. In this section, we will walk through each step in detail.

Building the Docker Image from Your Dockerfile

To build the Docker image from your Dockerfile, you will need to navigate to the directory where your Dockerfile is located and run the following command:

```
docker build -t your-image-name .
```

This command will build the Docker image using the instructions specified in your Dockerfile. Once the build process is complete, you will have a new Docker image ready for use.

Running Your Docker Container

To run your Docker container, you will need to use the Docker run command followed by the name of the image you want to run.

For example:

```
docker run your-image-name
```

Running this command will start a new container based on the specified image.

Depending on your application, you may need to specify additional options for the `docker run` command, such as port bindings or environment variables.

```
docker run -p 8080:80 your-image-name
```

Your Docker container is now up and running, ready to serve your application to the world.

Managing Your Docker Container

Unlike traditional virtual machines, where you need to manually install and configure software, Docker containers are designed to be easily managed and manipulated.

Let's take a look at some key ways to manage your Docker containers.

Monitoring Container Performance

With Docker, you can easily monitor the performance of your containers using built-in commands.

By running `docker stats`, you can view real-time CPU, memory, and network usage for all running containers.

This can help you identify any resource bottlenecks and optimize your container performance.

Stopping, Starting, and Removing Containers

The Docker CLI provides simple commands for stopping, starting, and removing containers.

The command

```
docker stop [container_name]
```

will gracefully stop a running container, while

```
docker start [container_name]
```

will restart a stopped container.

To remove a container entirely, use the command


```
docker rm [container_name]
```

Additionally, you can use the `docker ps` command to [list all running containers](#), and `docker ps -a` to see all containers, including those that are stopped.

This gives you full visibility and control over your containers.

RESULT:

Thus a container is created and executed in a docker successfully.