Ex. No: 1	**INSTALL VIRTUALBOX / VMWARE WORKSTATION WITH DIFFERENT FLAVOURS OF LINUX OR WINDOWS OS ON TOP OF WINDOWS7 OR 8**
Date:	

AIM:

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

SOFTWARE/TOOLS USED:

- Oracle VirtualBox 6.x / VMware Workstation
- OpenNebula Sandbox 5.0 (OVA File)
- Windows 7 / 8 (Host OS)
- Guest OS (Ubuntu / CentOS / Windows IO / others)
- Web Browser (for OpenNebula access)

APPLICATIONS:

Cloud computing technologies such as virtualization are widely used in:

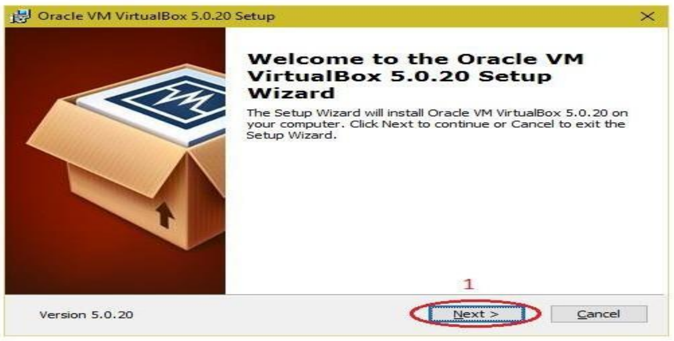
- Hosting scalable web applications (e.g., Amazon AWS, Microsoft Azure)
- Running isolated testing environments for software development
- Enabling Infrastructure as a Service (IaaS)
- Supporting educational labs and training environments
- Business continuity and disaster recovery solutions

Social platforms like Facebook, LinkedIn, and services such as Hotmail and Amazon use cloud infrastructure for performance, scalability and reliability.

PROCEDURE:

Steps to install Virtual Box:

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

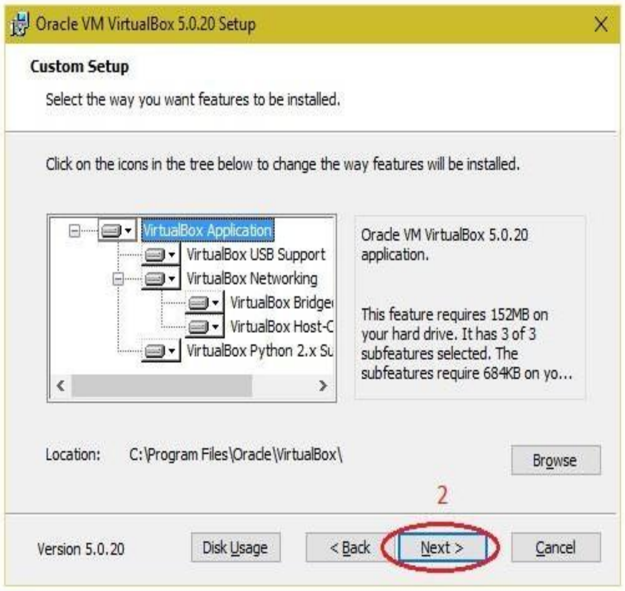

Oracle VM VirtualBox 5.0.20 Setup

Welcome to the Oracle VM VirtualBox 5.0.20 Setup Wizard

The Setup Wizard will install Oracle VM VirtualBox 5.0.20 on your computer. Click Next to continue or Cancel to exit the Setup Wizard.

Version 5.0.20

2. Click the next button


Oracle VM VirtualBox 5.0.20 Setup

Custom Setup

Select the way you want features to be installed.

Click on the icons in the tree below to change the way features will be installed.

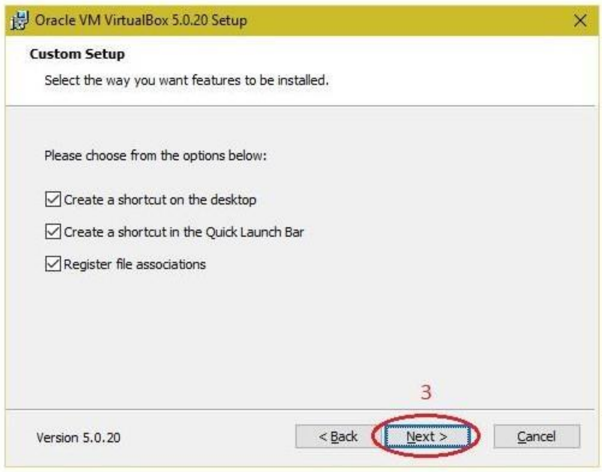
- VirtualBox Application
  - VirtualBox USB Support
  - VirtualBox Networking
    - VirtualBox Bridge
    - VirtualBox Host-C
  - VirtualBox Python 2.x Su

Oracle VM VirtualBox 5.0.20 application.

This feature requires 152MB on your hard drive. It has 3 of 3 subfeatures selected. The subfeatures require 684KB on yo...

Location: C:\Program Files\Oracle\VirtualBox\

3. Click the next button


Oracle VM VirtualBox 5.0.20 Setup

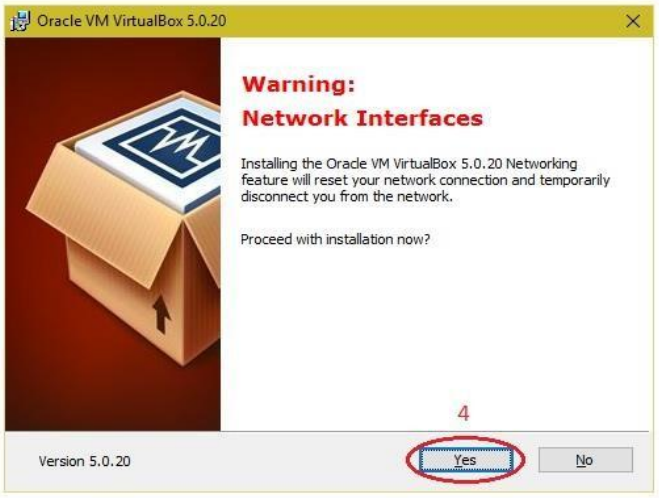
Custom Setup

Select the way you want features to be installed.

Please choose from the options below:

- [✓] Create a shortcut on the desktop
- [✓] Create a shortcut in the Quick Launch Bar
- [✓] Register file associations

4. Click the YES button



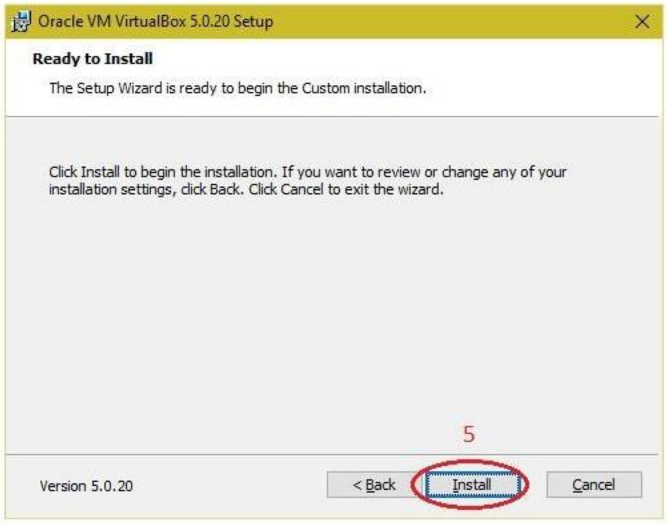
Oracle VM VirtualBox 5.0.20

Warning:
Network Interfaces

Installing the Oracle VM VirtualBox 5.0.20 Networking feature will reset your network connection and temporarily disconnect you from the network.

Proceed with installation now?

5. Click the install button



Oracle VM VirtualBox 5.0.20 Setup

Ready to Install

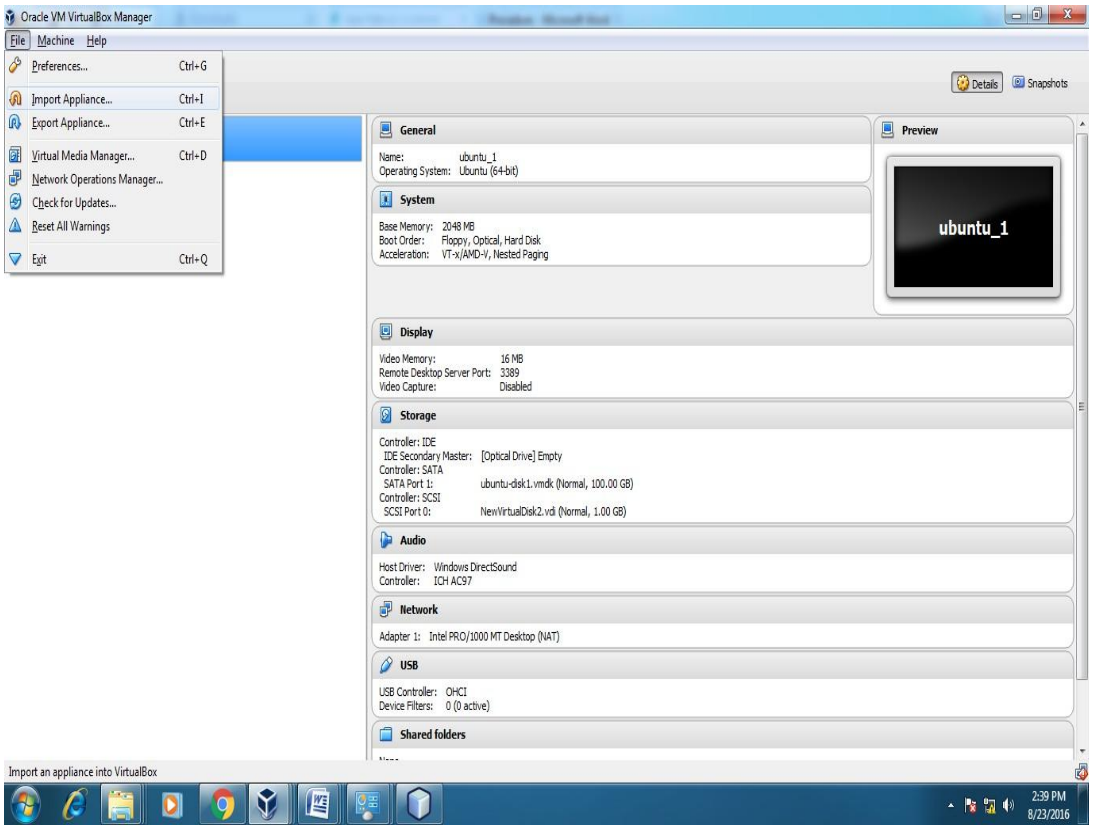
The Setup Wizard is ready to begin the Custom installation.

Click Install to begin the installation. If you want to review or change any of your installation settings, click Back. Click Cancel to exit the wizard.

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

Steps to import Open nebula sandbox:

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


Oracle VM VirtualBox Manager
File
Machine
Help
Preferences... Ctrl+G
Import Appliance... Ctrl+I
Export Appliance... Ctrl+E
Virtual Media Manager... Ctrl+D
Network Operations Manager...
Check for Updates...
Reset All Warnings
Exit Ctrl+Q
Details
Snapshots
Preview
ubuntu_1
2:39 PM
8/23/2016

General
Name: ubuntu_1
Operating System: Ubuntu (64-bit)

System
Base Memory: 2048 MB
Boot Order: Floppy, Optical, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging

Display
Video Memory: 16 MB
Remote Desktop Server Port: 3389
Video Capture: Disabled

Storage
Controller: IDE
IDE Secondary Master: [Optical Drive] Empty
Controller: SATA
SATA Port 1: ubuntu-disk1.vmdk (Normal, 100.00 GB)
Controller: SCSI
SCSI Port 0: NewVirtualDisk2.vdi (Normal, 1.00 GB)

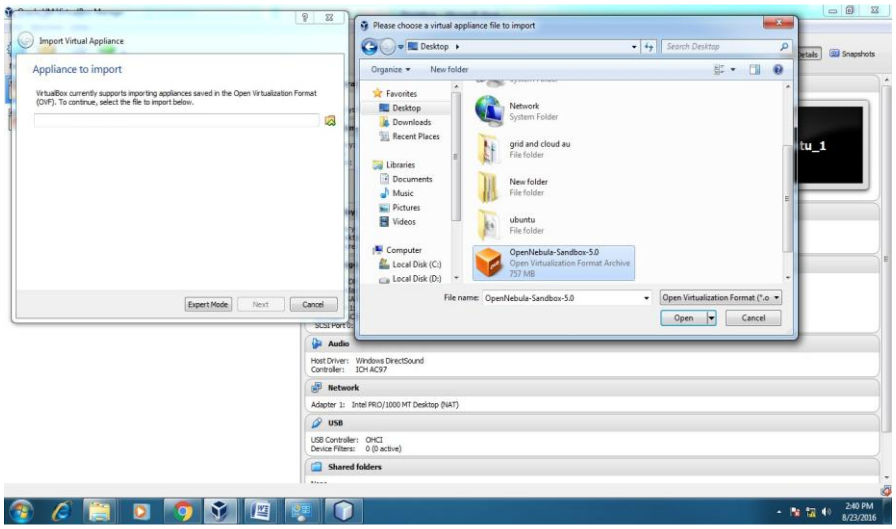
Audio
Host Driver: Windows DirectSound
Controller: ICH AC97

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

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

Shared folders

Import an appliance into VirtualBox



VirtualBox currently supports importing appliances saved in the Open Virtualization Format (OVF). To continue, select the file to import below.

File name: OpenNebula-Sandbox-5.0 | Open Virtualization Format (*.o

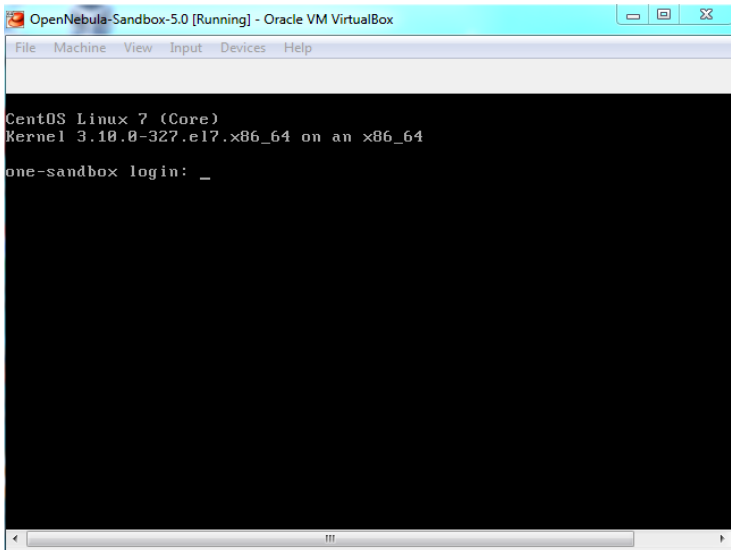
Host Driver: Windows DirectSound
Controller: ICH AC97

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

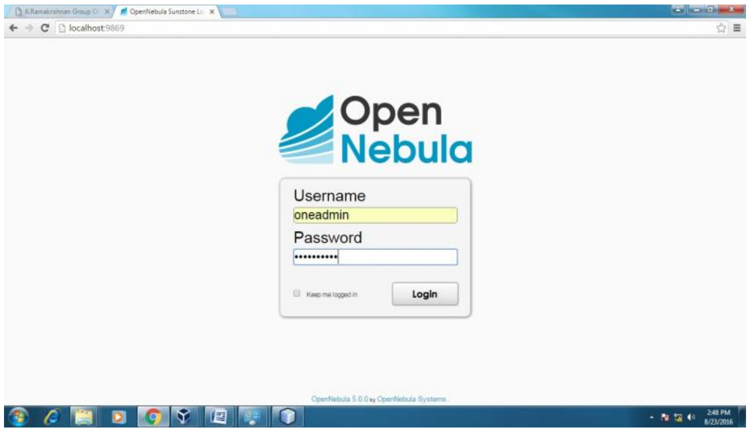
USB Controller: OHCI
Device Filters: 0 (0 active)

Steps to create Virtual Machine through opennebula

1. Open Browser, type localhost:9869
2. Login using username: oneadmin, password: opennebula
3. Click on instances, select VMs then follow the steps to create Virtaul machine
    a. Expand the + symbol
    b. Select user oneadmin
    c. Then enter the VM name,no.of instance, cpu.
    d. Then click on create button.
    e. Repeat the steps the C,D for creating more than one VMs.

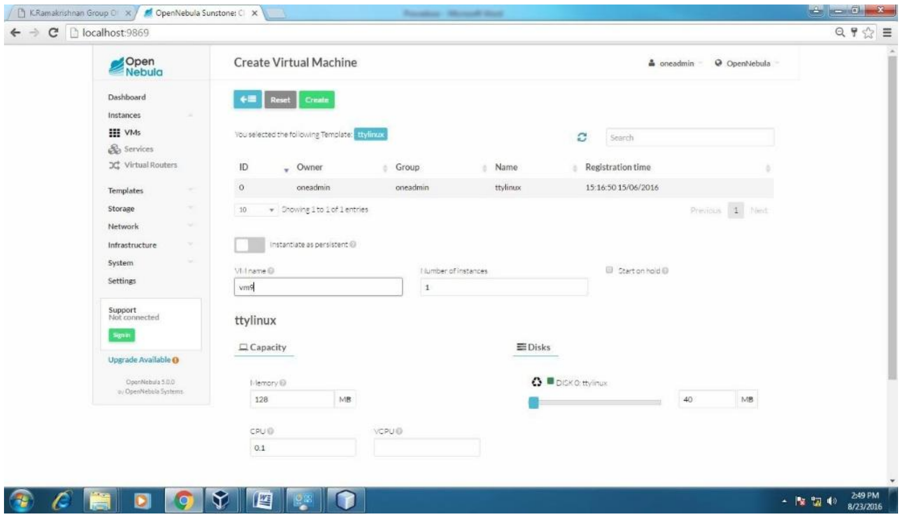


```
CentOS Linux 7 (Core)
Kernel 3.10.0-327.el7.x86_64 on an x86_64

one-sandbox login: _
```


localhost:9869

Username: oneadmin

Password: ••••••••••


localhost:9869

Create Virtual Machine

You selected the following Template: ttylinux

ID	Owner	Group	Name	Registration time
0	oneadmin	oneadmin	ttylinux	15:16:50 15/06/2016

Showing 1 to 1 of 1 entries

VM name: vm9

Number of instances: 1

ttylinux

Capacity: Memory 128 MB; CPU 0.1; VCPU

Disks: DISK 0: ttylinux 40 MB

RESULT:

Thus, the installation and configuration of VirtualBox and OpenNebula were successfully carried out, and multiple virtual machines of different configurations were created and tested.

Ex. No: 2	**INSTALL A C COMPILER IN THE VIRTUAL MACHINE CREATED USING VIRTUAL BOX AND EXECUTE SIMPLE PROGRAMS**
Date:	

AIM:

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

SOFTWARE/TOOLS USED:

- Oracle VirtualBox
- Ubuntu-based Virtual Machine Image (ubuntu_gt6.ova)
- GCC (GNU Compiler Collection)
- Terminal
- Gedit (Text Editor)

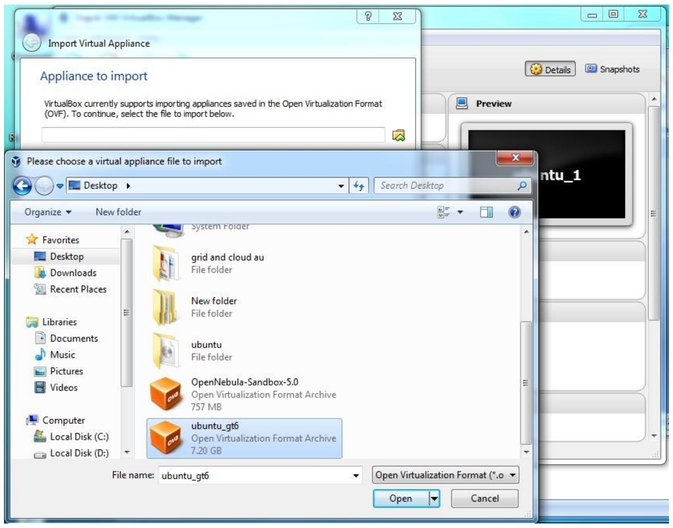
APPLICATIONS:

- Enables compilation and testing of C programs in a virtualized Linux environment.
- Provides a controlled environment for simulating cloud/grid programming and software development.
- Supports safe experimentation with C programming without affecting the host system.
- Encourages hands-on practice with basic software installation, compilation, and command-line operation.

PROCEDURE:

Steps to import .ova file:

1. Open Virtual box
2. File import Appliance
3. Browse ubuntu_gt6.ova file
4. Then go to setting, select Usb and choose USB 1.1
5. Then Start the ubuntu_gt6
6. Login using username: dinesh, password:99425.



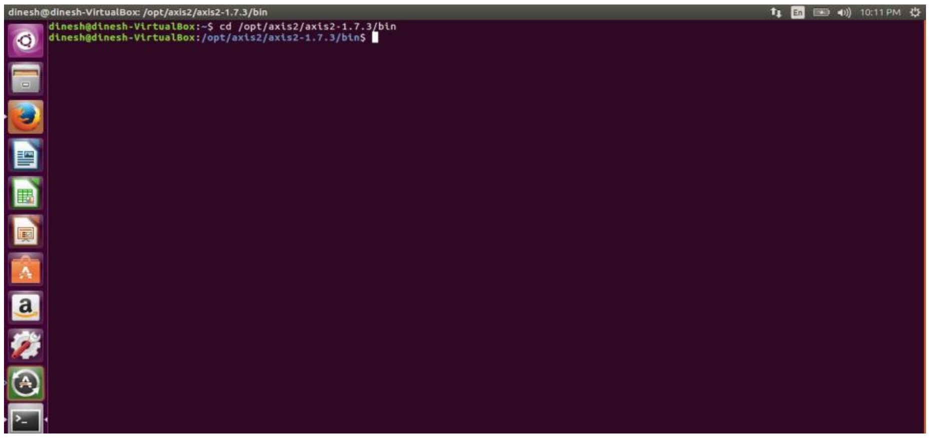
VirtualBox currently supports importing appliances saved in the Open Virtualization Format (OVF). To continue, select the file to import below.

File name: ubuntu_gt6

Steps to run c program:

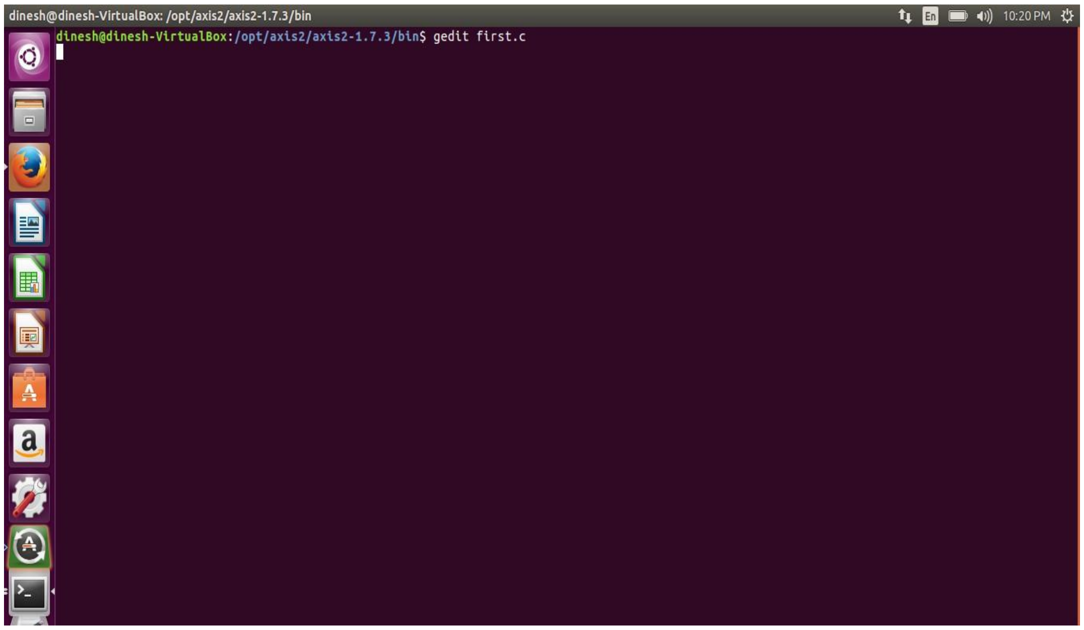
1. Open the terminal
2. Type cd /opt/axis2/axis2-1.7.3/bin then press enter
3. gedit hello.c
4. gcc hello.c
5. ./a.out

1. Type cd /opt/axis2/axis2-1.7.3/bin then press enter



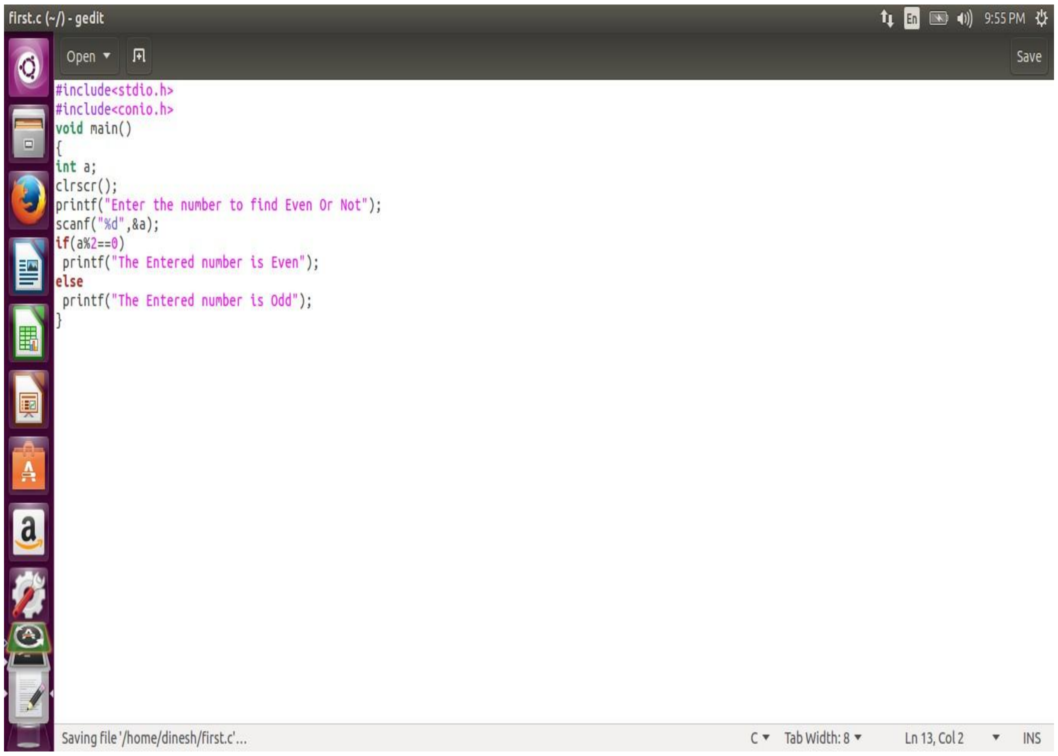
```
dinesh@dinesh-VirtualBox: /opt/axis2/axis2-1.7.3/bin
dinesh@dinesh-VirtualBox:~$ cd /opt/axis2/axis2-1.7.3/bin
dinesh@dinesh-VirtualBox:/opt/axis2/axis2-1.7.3/bin$
```

2. **Type gedit first.c**



```
dinesh@dinesh-VirtualBox: /opt/axis2/axis2-1.7.3/bin
dinesh@dinesh-VirtualBox:/opt/axis2/axis2-1.7.3/bin$ gedit first.c
```

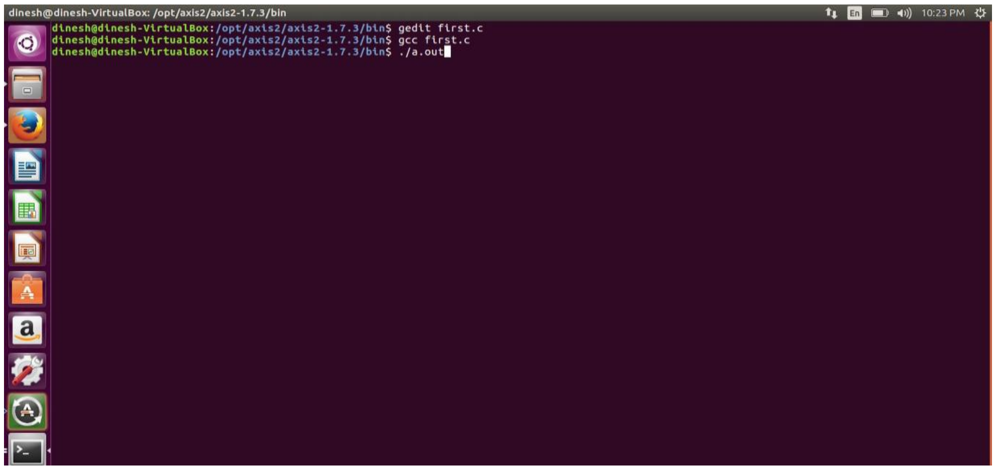
3. **Type the c program**



```
#include<stdio.h>
#include<conio.h>
void main()
{
int a;
clrscr();
printf("Enter the number to find Even Or Not");
scanf("%d",&a);
if(a%2==0)
 printf("The Entered number is Even");
else
 printf("The Entered number is Odd");
}
```

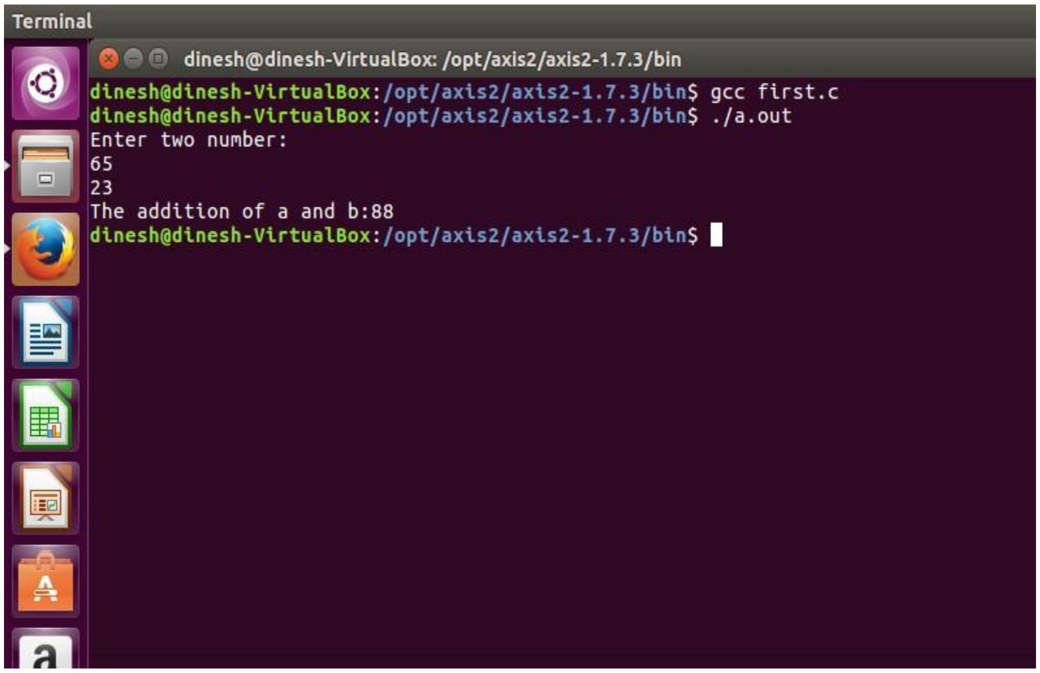
Saving file '/home/dinesh/first.c'...

4. Running the C program



```
dinesh@dinesh-VirtualBox: /opt/axis2/axis2-1.7.3/bin
dinesh@dinesh-VirtualBox:/opt/axis2/axis2-1.7.3/bin$ gedit first.c
dinesh@dinesh-VirtualBox:/opt/axis2/axis2-1.7.3/bin$ gcc first.c
dinesh@dinesh-VirtualBox:/opt/axis2/axis2-1.7.3/bin$ ./a.out
```

5. Display the output:



```
Terminal
dinesh@dinesh-VirtualBox: /opt/axis2/axis2-1.7.3/bin
dinesh@dinesh-VirtualBox:/opt/axis2/axis2-1.7.3/bin$ gcc first.c
dinesh@dinesh-VirtualBox:/opt/axis2/axis2-1.7.3/bin$ ./a.out
Enter two number:
65
23
The addition of a and b:88
dinesh@dinesh-VirtualBox:/opt/axis2/axis2-1.7.3/bin$
```

RESULT:

The simple C programs were successfully compiled and executed in the Ubuntu virtual machine created using Oracle VirtualBox. The process verified the proper installation and working of the GCC compiler within the virtual environment.

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.

SOFTWARE/TOOLS USED:

- Google App Engine Java SDK
- Eclipse IDE with Google Plugin for Eclipse
- Java Development Kit (JDK)
- Web Browser (for accessing localhost and deployed URL)
- Internet Connection (for deployment to App Engine)

APPLICATIONS:

- Used for developing and deploying scalable web applications on the cloud using Google App Engine.
- Supports hands-on experience in cloud-based application development using Java/Python.
- Demonstrates the process of local hosting and live deployment using a platform-as-a-service (PaaS) model.
- Useful for beginners to understand how cloud environmentswork for web application hosting and scalability.

PROCEDURE:

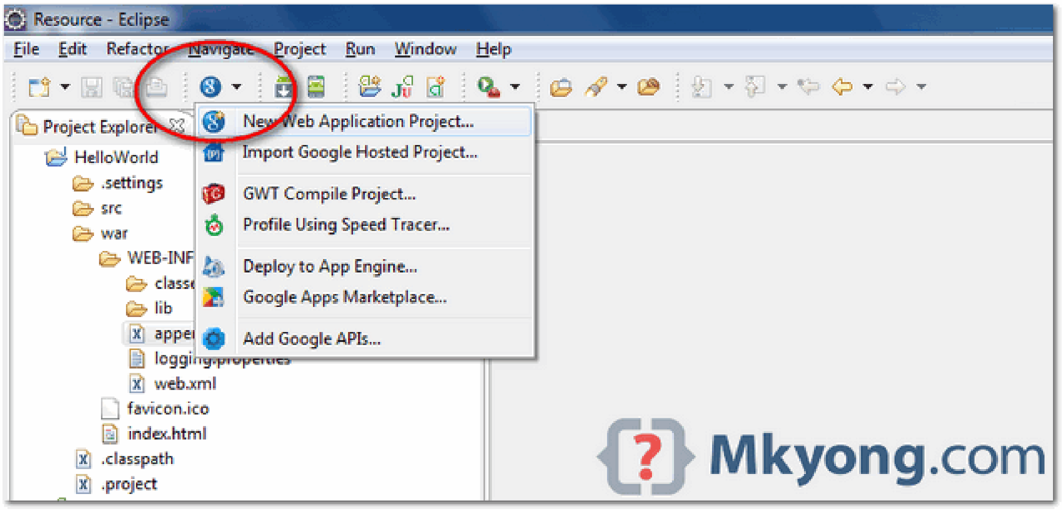
1. Install Google Plugin for Eclipse

If the Google App Engine Java SDK together with "**Google Plugin for Eclipse**", then go to step 2, Otherwise, get the Google App Engine Java SDK and extract it.

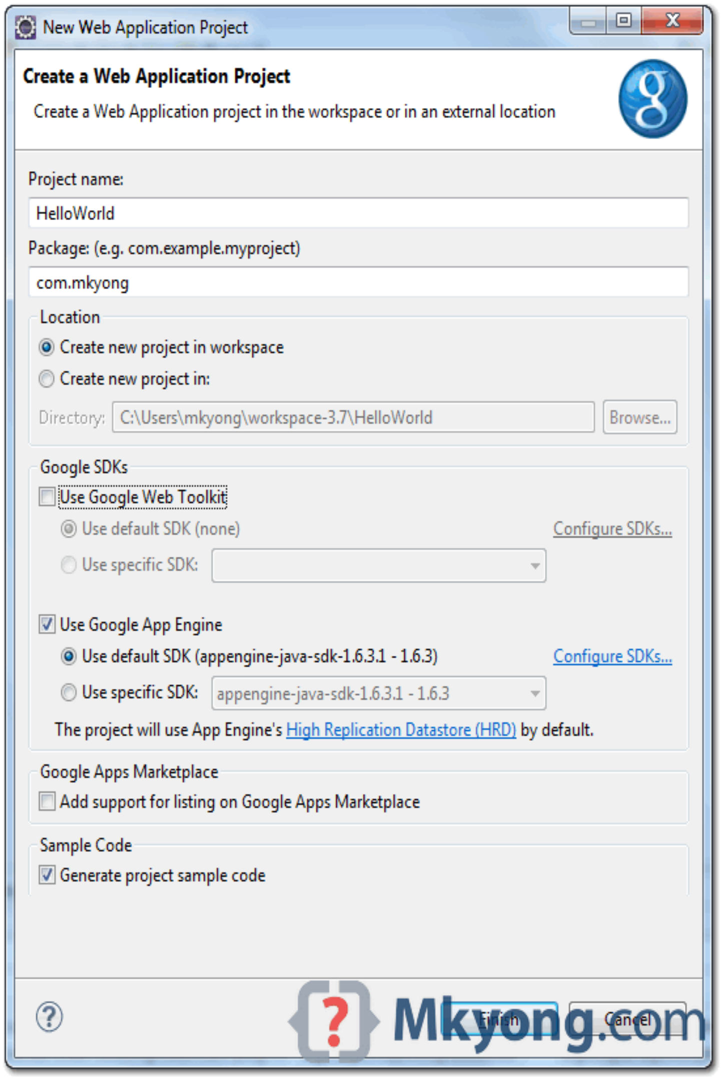
2. Create New Web Application Project

In Eclipse toolbar, click on the Google icon, and select "**New Web Application Project…**"

Figure – New Web Application Project



Deselect the "**Google Web ToolKit**", and link your GAE Java SDK via the "**configure SDK**" link.


New Web Application Project
Create a Web Application Project
Create a Web Application project in the workspace or in an external location

Project name: HelloWorld

Package: (e.g. com.example.myproject) com.mkyong

Location
- (●) Create new project in workspace
- ( ) Create new project in:

Directory: C:\Users\mkyong\workspace-3.7\HelloWorld

Google SDKs
- [ ] Use Google Web Toolkit
  - (●) Use default SDK (none)
  - ( ) Use specific SDK:
- [✓] Use Google App Engine
  - (●) Use default SDK (appengine-java-sdk-1.6.3.1 - 1.6.3)
  - ( ) Use specific SDK: appengine-java-sdk-1.6.3.1 - 1.6.3

The project will use App Engine's High Replication Datastore (HRD) by default.

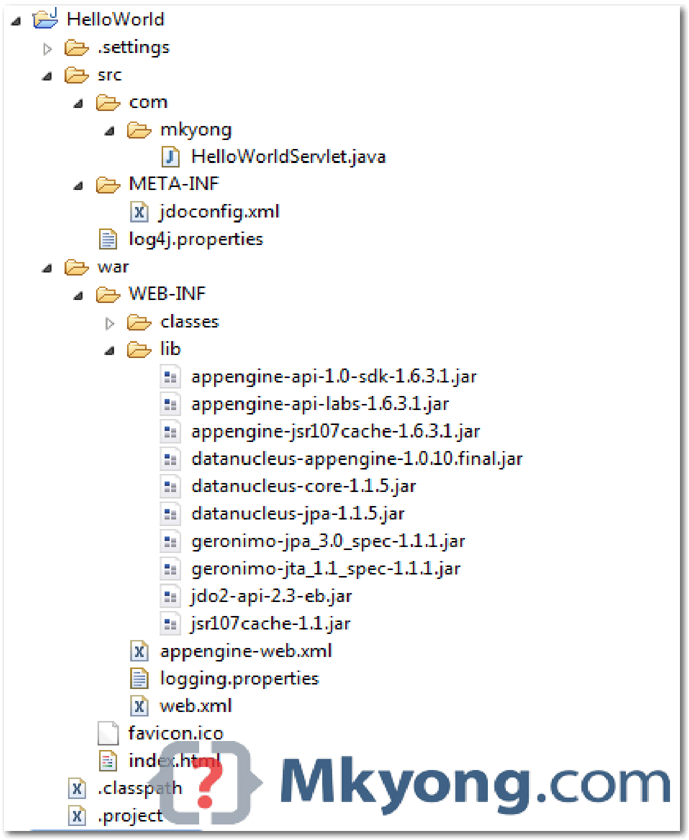
Google Apps Marketplace
- [ ] Add support for listing on Google Apps Marketplace

Sample Code
- [✓] Generate project sample code

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

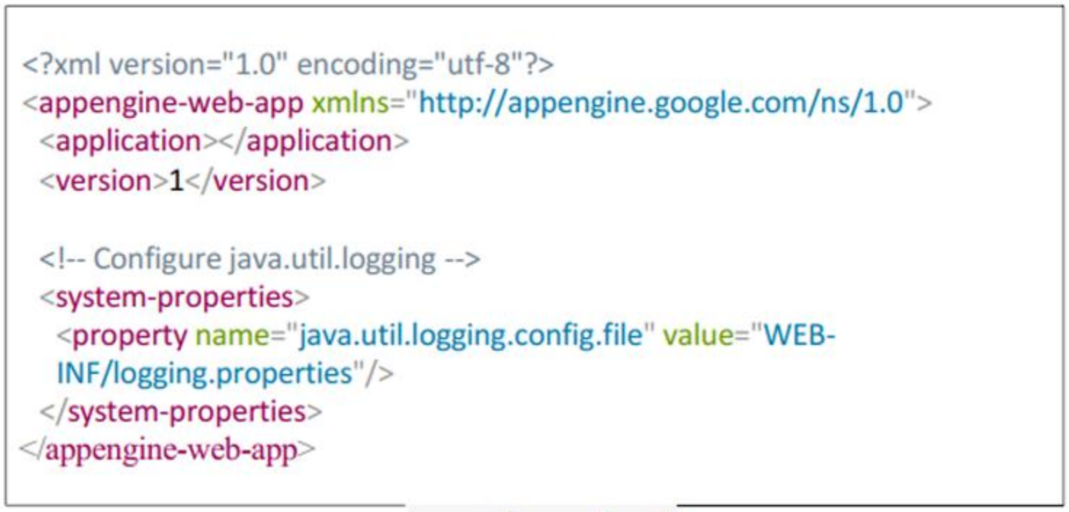
3. Hello World

Review the generated project directory.



```
HelloWorld
  .settings
  src
    com
      mkyong
        HelloWorldServlet.java
    META-INF
      jdoconfig.xml
    log4j.properties
  war
    WEB-INF
      classes
      lib
        appengine-api-1.0-sdk-1.6.3.1.jar
        appengine-api-labs-1.6.3.1.jar
        appengine-jsr107cache-1.6.3.1.jar
        datanucleus-appengine-1.0.10.final.jar
        datanucleus-core-1.1.5.jar
        datanucleus-jpa-1.1.5.jar
        geronimo-jpa_3.0_spec-1.1.1.jar
        geronimo-jta_1.1_spec-1.1.1.jar
        jdo2-api-2.3-eb.jar
        jsr107cache-1.1.jar
      appengine-web.xml
      logging.properties
      web.xml
    favicon.ico
    index.html
  .classpath
  .project
```

4. Copy the file "appengine-web.xml", Google App Engine need this to run and deploy the application.



```
<?xml version="1.0" encoding="utf-8"?>
<appengine-web-app xmlns="http://appengine.google.com/ns/1.0">
  <application></application>
  <version>1</version>

  <!-- Configure java.util.logging -->
  <system-properties>
    <property name="java.util.logging.config.file" value="WEB-
    INF/logging.properties"/>
  </system-properties>
</appengine-web-app>
```

appengine-web.xml

4. Run it local

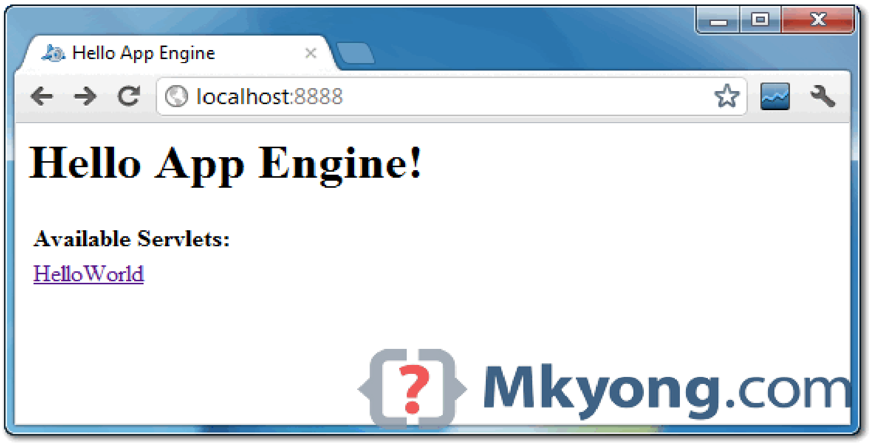
Right click on the project and run as “**Web Application**“.

```
//...
INFO: The server is running at http://localhost:8888/
30 Mac 2012 11:13:01 PM com.google.appengine.tools.development.DevAppServerImpl
start INFO: The admin console is running at http://localhost:8888/_ah/admin
```

Copy

Access URL http://localhost:8888/,

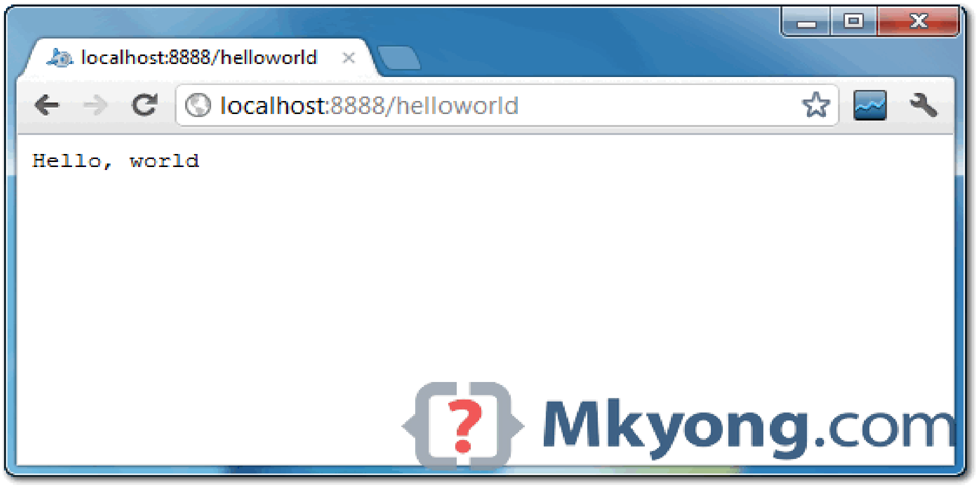
OUTPUT:



Hello App Engine!

Available Servlets:
HelloWorld

and also the hello world servlet – http://localhost:8888/helloworld

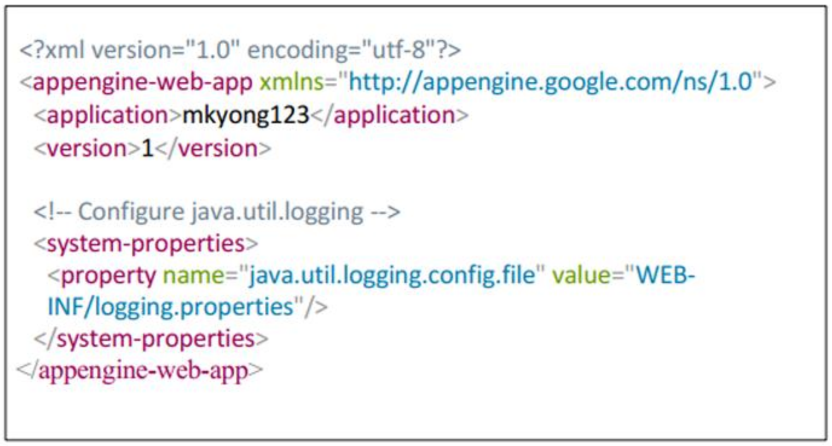


```
Hello, world
```

5. Deploy to Google App Engine

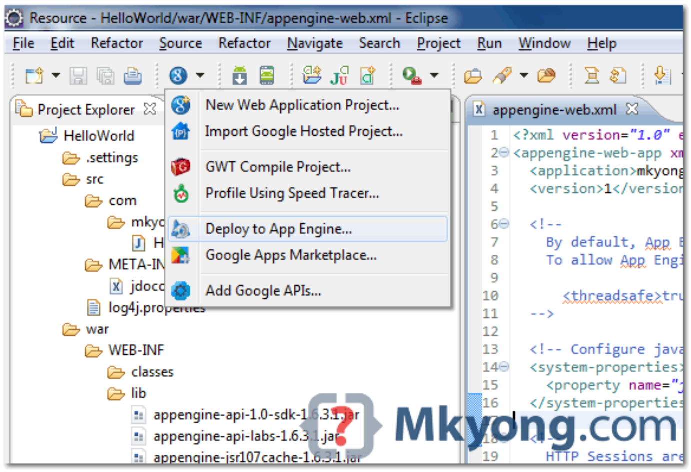
Register an account on https://appengine.google.com/, and create an application ID for your web application.

In this demonstration, I created an application ID, named "mkyong123", and put it in appengine web.xml

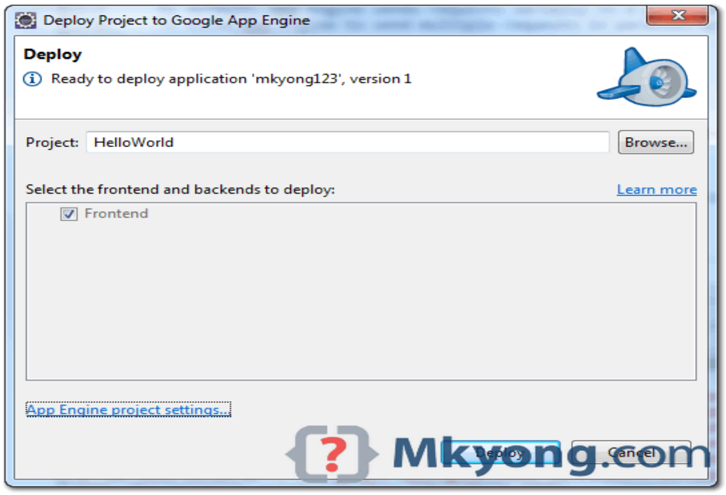


```xml
<?xml version="1.0" encoding="utf-8"?>
<appengine-web-app xmlns="http://appengine.google.com/ns/1.0">
  <application>mkyong123</application>
  <version>1</version>

  <!-- Configure java.util.logging -->
  <system-properties>
    <property name="java.util.logging.config.file" value="WEB-
    INF/logging.properties"/>
  </system-properties>
</appengine-web-app>
```

7. Copy the file
8. To deploy, do the following steps:
9. Click on GAE deploy button on the toolbar.


Resource - HelloWorld/war/WEB-INF/appengine-web.xml - Eclipse
File Edit Refactor Source Refactor Navigate Search Project Run Window Help
Project Explorer
HelloWorld
.settings
src
com
META-IN
log4j.properties
war
WEB-INF
classes
lib
appengine-api-1.0-sdk-1.6.3.1.jar
appengine-api-labs-1.6.3.1.jar
appengine-jsr107cache-1.6.3.1.jar
New Web Application Project...
Import Google Hosted Project...
GWT Compile Project...
Profile Using Speed Tracer...
Deploy to App Engine...
Google Apps Marketplace...
Add Google APIs...
appengine-web.xml
Mkyong.com

10. Sign in with your Google account and click on the Deploy button



Deploy Project to Google App Engine

Deploy

Ready to deploy application 'mkyong123', version 1

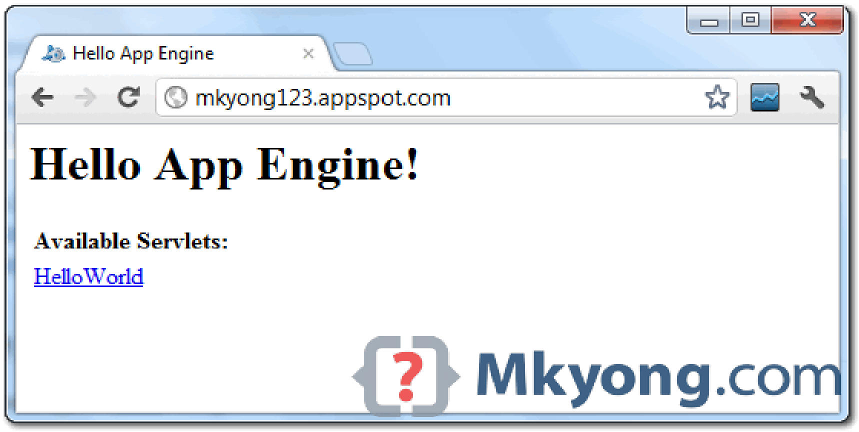
Project: HelloWorld

Select the frontend and backends to deploy:

[✓] Frontend

11. If everything is fine, the hello world web application will be deployed to this URL – http://mkyong123.appspot.com/

Ex. No: 4	**SIMULATE A CLOUD SCENARIO USING CLOUDSIM AND RUN A SCHEDULING ALGORITHM THAT IS NOT PRESENT IN CLOUDSIM**
Date:	



Hello App Engine!

Available Servlets:

HelloWorld

Result:

The Google App Engine environment was successfully set up using Eclipse and the required SDK. A simple "Hello World" web application was developed, tested locally at http://localhost:8888/ and successfully deployed to Google App Engine.

AIM:

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

SOFTWARE/TOOLS USED:

- CloudSim: A framework used for simulating cloud computing environments, primarily written in Java.
- Eclipse IDE: A development environment used for Java programming, where CloudSim is integrated and the simulation is executed.

APPLICATIONS:

- Cloud Computing Simulation: CloudSim allows the simulation of cloud computing resources such as virtual machines (VMs), data centers, and cloudlets.
- Scheduling Algorithms: Custom scheduling algorithms can be implemented and tested in CloudSim, such as those for VM allocation and resource scheduling.
- Cloud Resource Management: Helps in simulating resource allocation in cloud environments for understanding load balancing and efficiency improvements.

PROCEDURE:

Step 1 : Download CloudSim

1. Visit the official CloudSim GitHub repository:
https://github.com/Cloudslab/cloudsim
2. Click on the green Code button —o Download ZIP.
3. Extract the ZIP to a known location (e.g., or --/CloudSim).

Step 2: Open Eclipse and Create a Java Project

l. Open Eclipse IDE.
2. Go to File > New > Java Project.
3. Name the project, e.g., CloudSimTest.
4. Click Finish.

Step 3: Add CloudSim Libraries to the project

1.Right-click on the newly created project in the Package Explorer.
2. Select Build Path > Configure Build Path.

3.GO to the Libraries tab.

4. Click Add External JARs....

5.Browse to the extracted cloudsim folder, then add the following .jar files:

- cloudsim-XX.X.jar (usually found in the jars or Iib folder)
- Any Other dependencies (e.g., commons-math3-X.X.jar)
-

6.Click Apply and Close

Step 4: Create a Sample Simulation Class

1. Right-click the src folder > New > Class.
2. Name it FirstSimulation.
3. Check public static void main(StringIl args).
4. Click Finish.

Step 5: Write CloudSim Simulation Code

Paste the following basic example to simulate one datacenter, one VM, and one

Cloudlet:

```
import org.cloudbus.cloudsim.*;
import org.cloudbus.cloudsim.core.CloudSim;
import java. util. * ;
public class FirstSimulation {
public static void main (String[] args)
try{
    // Step 1 : Initialize the Cloud-Sim package
    int numUsers = l;
    Calendar calendar Calendar -getlnstance();
    boolean traceFlag false;
    CloudSim.init(numUsers, calendar, traceFlag);
    // Step 2: Create Datacenter
    Datacenter datacenterO createDatacenterCDatacenter_0");
    // Step 3: Create Broker
    DatacenterBroker broker new
    int brokerld = broker.getld();
    // Step 4: Create one virtual machine
    vmList = new
    Vm vm •new brokerld, 1000, l, 512, 1000, 1000, "Xen", new CloudletSchedulerTimeShared());
    vmList.add(vm);
    // Step 5: Submit VM list to the broker
    broker. submit V mL ist(vmList);
    // Step 6: Create one Cloudlet
    List<Cloudlet> cloudletList = new ArrayList<>();
```

```
UtilizationModel utilizationModel = new UtilizationModelFull();
Cloudlet cloudlet new 40000, l, 300, 300, utilizationModel,
utiliz.ationModel, utilizationModel);
cloudlet. set Userld( brokerld);
cloudlet.setVmld(vm.getld());
cloudletList.add(c10LK11et);

// Step 7: Submit cloudlet list to the broker
broker. submitCloudletList(cloudletList);
// Step 8: Stan simulation
CloudSim.startSimulation();
// Step 9: Print results
List<Cloudlet> newList broker.getCloudletReceivedList();
CloudSim.stopSimuIation();
printCloudletList(newList);
catch (Exception e) {
e. printStackTraceO;
private static Datacenter createDatacenter(String name) throws Exception
List<Host> hostList = ncw ArrayList<>();
// A Machine contains one or more PEs or CPUs/Cores
List<Pe> peList = new ArrayList<>();
peList.add(new Pe(0, new PeProvisionerSimple(1000))); // need to store Pe id and MIPS
Host host = new Host( 0, new RamProvisionerSimple(2048), new BwProvisionerSimple( 10000),1000000,
peList, new VmSchedulerTimeShared(peList)
String arch "x86";
String os = "Linux";
String vmm "Xcn";
// system architecture
// operating system
double timeZone 10.0; // time zone this resource located
double cost = 3.0; // the cost of using processing in this resource
double costPerMem = 0.05;
double costPerStorage = 0.001 ;
double costPerBw 0.0;
DatacentetCharacteristics characteristics new DatacenterCharacteristics( arch, os, vmm, hostList,
timcZonc, cost, costPerMcm, costPerStoragc, costPerBwn return new Datacenter(name, characteristics,
new VmAllocationPolicySimple(hostList), new 0);
private static void printCloudletList(List<Cloudlet> list) {
String indent = " ";
System.out.println("Cloudlet Execution Results:");
System.out.println( "CloudletID" + indent + "STATUS" + indent +
"DataCenterID" + indent + "VMID" + indent + "Time" + indent + "StartTime" + indent + "FinishTime");
for (Cloudlet cloudlet : list) {
        + indent);
        if(cloudlet.getStatus() Cloudlet.SUCCESS) {
                System.out.println("SUCCESS" + indent + cloudlet.getResourceId() + indent + indent +
                cloudlet. getActuaICPUTime() + indent +
                cloudlet.getExecStartTime() + indent +
                cloudlet.getF inishTime());
        }
```

Ex. No: 5	**GAE LAUNCHER TO LAUNCH WEB APPLICATIONS**
Date:	

```
        }
      }
    }
```

Step 6: Run the Simulation

1. Right-click on FirstSimulation.java.
2. Click Run As > Java Application.
3. We can see output in the Console showing Cloudlet execution results.

RESULT:

- The simulation successfully executed with a custom scheduling algorithm that was not present in CloudSim.
- The output demonstrated the behavior of cloudlets, VMS, and data centers in the simulated cloud environment, including their statuses, IDs, and execution times.
- The data center and broker were properly initialized and executed, with cloudlets successfully processed by the virtual machines.

AIM:

To use GAE (Google App Engine) Launcher to launch the web applications.

SOFTWARE/TOOLS USED:

- Google App Engine (GAE) Launcher
- Python (runtime environment)

- JEdit / Notepad++ / Any Text Editor
- Web Browser (e.g., Chrome, Firefox)
- Operating System: Windows

APPLICATIONS:

- A basic web application using Python and Google App Engine.
- index.py: Python script to display a simple message.
- app.yaml: Configuration file for GAE specifying application settings and URL handler.

PROCEDURE:

STEP 1

Create the Project Directory:

Create a folder named **apps** at the desired location, such as on the Desktop.
C:\Documents and Settings\csev\Desktop\apps

STEP 2

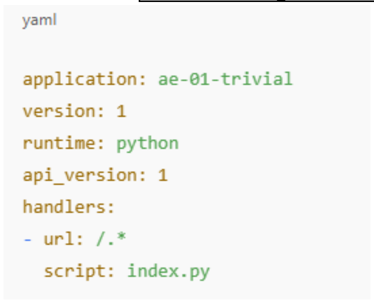
Inside the apps folder, create a subfolder named **ae-01-trivial**:
C:\Documents and Settings\csev\Desktop\apps\ae-01-trivial

STEP 3

Create the Configuration File (app.yaml)

- Open a plain text editor (e.g., JEdit or Notepad++).
- In the ae-01-trivial folder, create a new file named app.yaml.
- Type the following content exactly as shown:

```
application: ae-01-trivial
version: 1
runtime: python
api_version: 1
handlers:
- url: /.*
  script: index.py
```



```yaml
application: ae-01-trivial
version: 1
runtime: python
api_version: 1
handlers:
- url: /.*
  script: index.py
```

STEP 4

Create the Application Script (index.py):

- In the same folder (ae-01-trivial), create a new file named **index.py**.
- Add the following Python code:

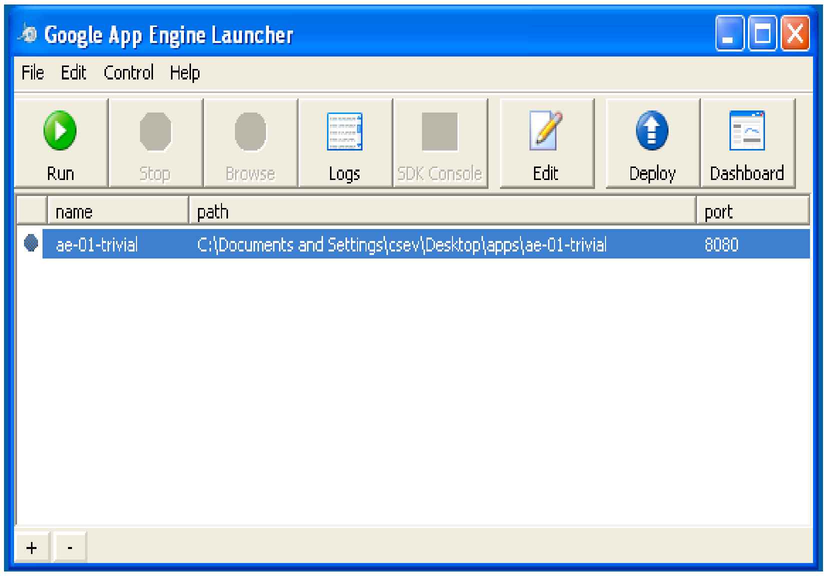
```
print 'Content-Type: text/plain'
print ''
print 'Hello there Chuck'
```

STEP 5

Launch the Application Using Google App Engine Launcher:

- Open the **Google App Engine Launcher** from the system's application menu.
- Select **File > Add Existing Application**, then navigate to:
C:\Documents and Settings\csev\Desktop\apps\ae-01-trivial
- Select the folder to add the application.
- Once added, select the application in the launcher interface.

STEP 6

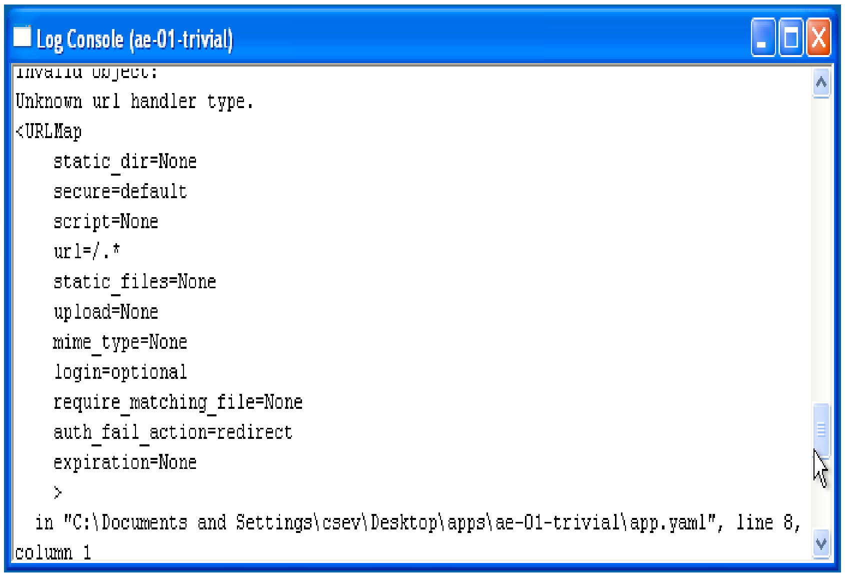


name	path	port
ae-01-trivial	C:\Documents and Settings\csev\Desktop\apps\ae-01-trivial	8080

Running and Testing the Application

- After the application has been selected in the Google App Engine Launcher, click the **Run** button. The application will start within a few seconds, and a **green icon** will be displayed next to the application name, indicating successful launch.
- Click the **Browse** button in the launcher. This action will open the default web browser and load the application at the following URL: http://localhost:8080/
- Alternatively, open a web browser and manually enter the URL in the address bar to access the application output.
- To modify the displayed output, open the **index.py** file located in the application folder and replace the name **"Chuck"** with the desired name.
- Save the modified file and refresh the browser window to verify that the updated content is displayed.

STEP 7



```
invalid object:
Unknown url handler type.
<URLMap
    static_dir=None
    secure=default
    script=None
    url=/.*
    static_files=None
    upload=None
    mime_type=None
    login=optional
    require_matching_file=None
    auth_fail_action=redirect
    expiration=None
    >
  in "C:\Documents and Settings\csev\Desktop\apps\ae-01-trivial\app.yaml", line 8,
column 1
```

Monitoring the Application Logs

- To monitor the actions performed by the web server during interaction with the application, access the log files using the following steps:
  1. In the **Google App Engine Launcher**, select the running application from the list.
  2. Click the **Logs** button to open the log window.
- Each time the browser page is refreshed, the log window will display entries showing HTTP **GET** requests and the corresponding server responses being processed.

STEP 8

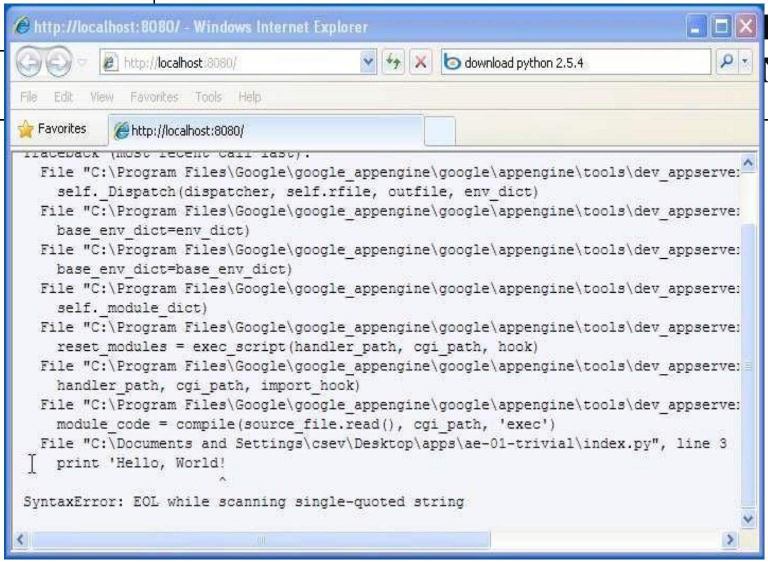
Handling Errors

While configuring the application, errors may occur in either the app.yaml file or the index.py script. These errors typically fall into the following categories:

- **Errors in app.yaml:**
  If the app.yaml file contains incorrect syntax or improper indentation, the application will fail to start. The Google App Engine Launcher will display a **yellow icon** next to the application name.
  - To troubleshoot the issue, open the **Log** window to review detailed error messages.
  - Correct the error in the app.yaml file and attempt to start the application again.
- **Errors in index.py:**
  If the Python script contains syntax errors, the application may still start, but a **Python traceback error** will be displayed in the browser when the application is accessed.
  - Examine the **last few lines** of the traceback message for information about the error.
  - Update the index.py file to fix the issue.

After saving the changes, **refresh the browser** to test the fix- **server restart is not required**.

Ex. No: 6	**ES BETWEEN TWO VIRTUAL NES**
Date:	


http://localhost:8080/ - Windows Internet Explorer
http://localhost:8080/
download python 2.5.4
File Edit View Favorites Tools Help
Favorites
http://localhost:8080/

```
Traceback (most recent call last):
  File "C:\Program Files\Google\google_appengine\google\appengine\tools\dev_appserve
    self._Dispatch(dispatcher, self.rfile, outfile, env_dict)
  File "C:\Program Files\Google\google_appengine\google\appengine\tools\dev_appserve
    base_env_dict=env_dict)
  File "C:\Program Files\Google\google_appengine\google\appengine\tools\dev_appserve
    base_env_dict=base_env_dict)
  File "C:\Program Files\Google\google_appengine\google\appengine\tools\dev_appserve
    self._module_dict)
  File "C:\Program Files\Google\google_appengine\google\appengine\tools\dev_appserve
    reset_modules = exec_script(handler_path, cgi_path, hook)
  File "C:\Program Files\Google\google_appengine\google\appengine\tools\dev_appserve
    handler_path, cgi_path, import_hook)
  File "C:\Program Files\Google\google_appengine\google\appengine\tools\dev_appserve
    module_code = compile(source_file.read(), cgi_path, 'exec')
  File "C:\Documents and Settings\csev\Desktop\apps\ae-01-trivial\index.py", line 3
    print 'Hello, World!
                        ^
SyntaxError: EOL while scanning single-quoted string
```

STEP 9

Shutting Down the Server

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

RESULT :

The web application was successfully created and deployed using Google App Engine Launcher. The application ran on http://localhost:8080 and displayed the message **“Hello there RAJE”** in the browser. Errors were debugged using the GAE Launcher’s log viewer.

AIM:

To identify and implement various procedures for transferring files between one virtual machine and another within a virtualized environment.

SOFTWARE/TOOLS USED:

- VirtualBox / VMware Workstation
- Guest Operating Systems (e.g., Ubuntu, Windows)
- OpenNebula (for VM orchestration)

- SSH (Secure Shell)
- SCP (Secure Copy Protocol)
- Samba / NFS / SSHFS (optional for network sharing)
- Web browser (for accessing OpenNebula UI)

APPLICATIONS:

This procedure enables seamless file transfer and migration between virtual machines hosted on the same or different physical hosts. It facilitates efficient VM mobility and resource utilization in cloud environments.

STEP 1 : Methods for File Transfer Between VMs.

STEP 2 : Clipboard Sharing:

- Enable bidirectional clipboard sharing using Guest Additions.
- Copy content from one VM and paste into another via the host clipboard.

STEP 3 : Drag and Drop:

- Navigate to Virtual Machine Settings → General → Advanced.
- Enable Drag and Drop (set to Bidirectional).

STEP 4 : Shared Folders:

- Configure a common shared folder accessible from both VMs.
- Install Guest Additions to enable folder sharing.
- Use the shared folder as a buffer to transfer files.

STEP 5 : Client-Server File Transfer

- Utilize tools such as scp with sshd running on the receiving VM.
- Confirm SSH server status using pgrep sshd.
- Install using sudo apt-get install openssh-server if not available.

STEP 6 :

Network File System (NFS), SSHFS, or Samba:

- Mount the file system of one VM onto another.
- Configure necessary sharing permissions.

Note: File permissions and user-level restrictions from the host operating system must be considered to allow access and execution privileges.

STEP 7 :

B. File Migration Procedure Using OpenNebula:

STEP 8 :

Open a browser and navigate to http://localhost:9869.

STEP 9 :

Log in with credentials:

- **Username:** oneadmin
- **Password:** opennebula

STEP 10:

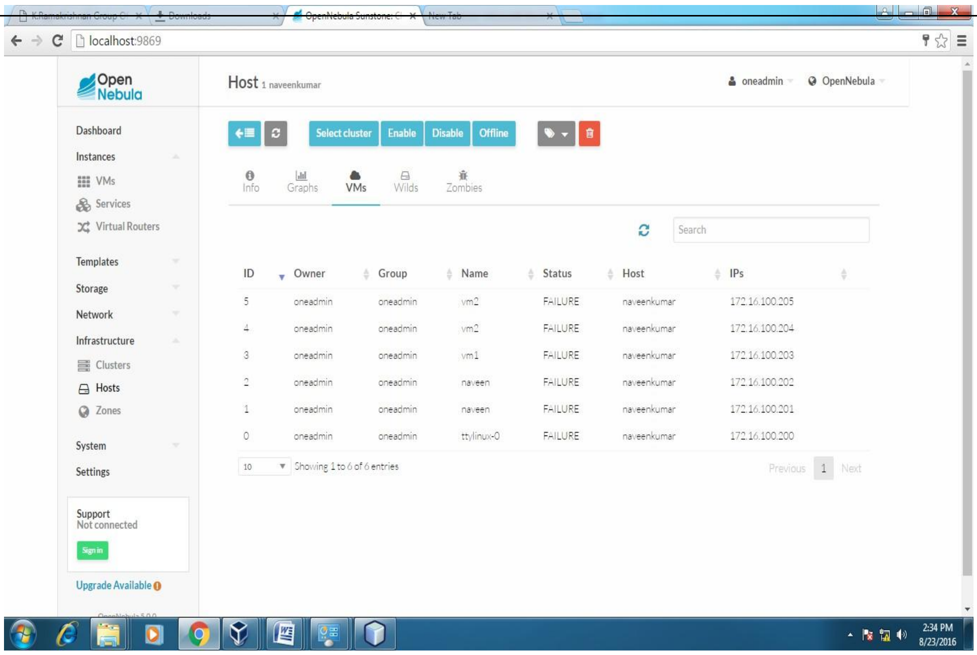
Perform the following steps to configure infrastructure:
a. Go to **Infrastructure** → **Clusters** → Select Cluster Name
b. Navigate through the **Host**, **Vnets**, and **Datastores** tabs, selecting all options
c. Under **Infrastructure**, click **Host** → **+ (Add New Host)** → Enter host name → **Create**

STEP 11 :

Migrate the VM:
a. Go to **Instances** → Select VM
b. Click the **8th icon** to display the dropdown
c. Select **Migrate** → Choose Target Host → Click **Migrate**

STEP 12 :

Before migration Host:SACET

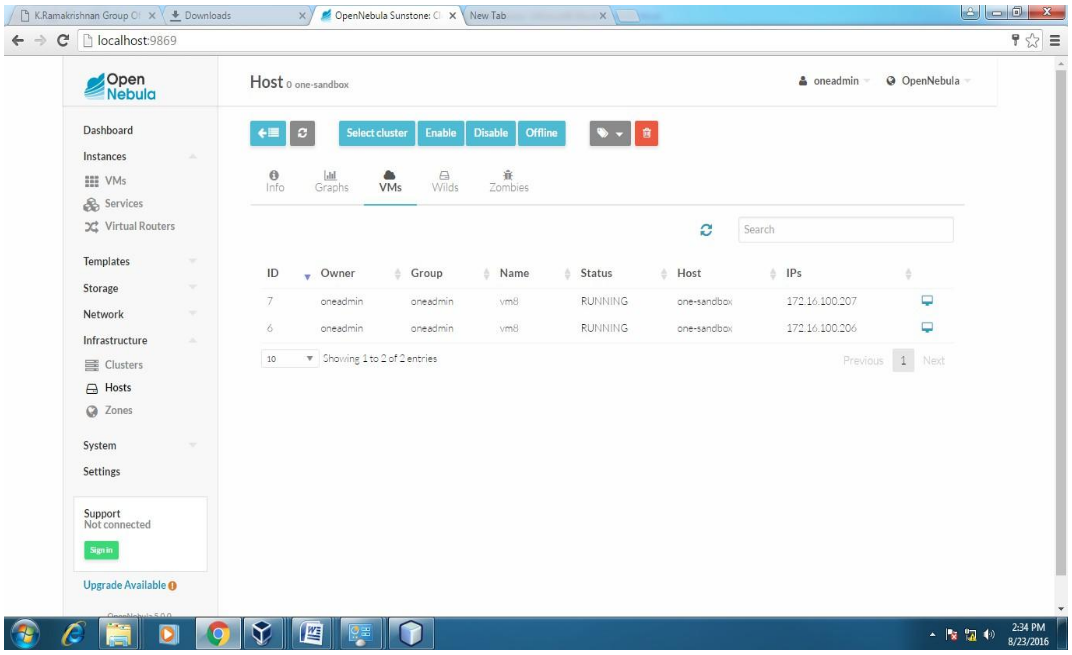


Host 1 naveenkumar

ID	Owner	Group	Name	Status	Host	IPs
5	oneadmin	oneadmin	vm2	FAILURE	naveenkumar	172.16.100.205
4	oneadmin	oneadmin	vm2	FAILURE	naveenkumar	172.16.100.204
3	oneadmin	oneadmin	vm1	FAILURE	naveenkumar	172.16.100.203
2	oneadmin	oneadmin	naveen	FAILURE	naveenkumar	172.16.100.202
1	oneadmin	oneadmin	naveen	FAILURE	naveenkumar	172.16.100.201
0	oneadmin	oneadmin	ttylinux-0	FAILURE	naveenkumar	172.16.100.200

10 Showing 1 to 6 of 6 entries Previous 1 Next

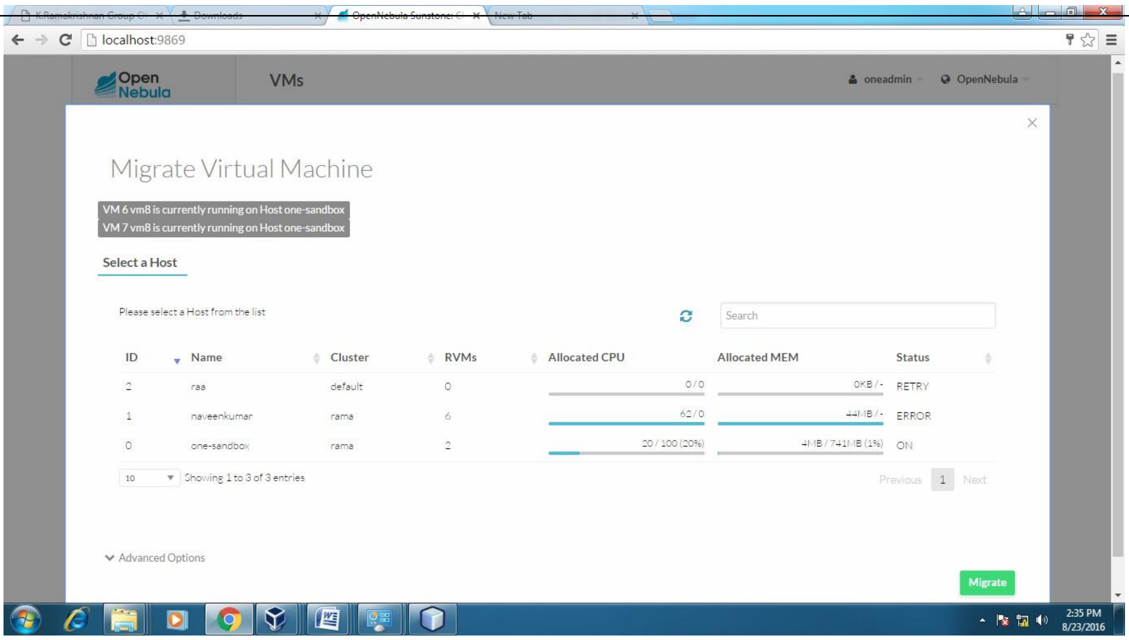
Host:one-sandbox



Host 0 one-sandbox

ID	Owner	Group	Name	Status	Host	IPs
7	oneadmin	oneadmin	vm8	RUNNING	one-sandbox	172.16.100.207
6	oneadmin	oneadmin	vm8	RUNNING	one-sandbox	172.16.100.206

10 Showing 1 to 2 of 2 entries Previous 1 Next



Migrate Virtual Machine

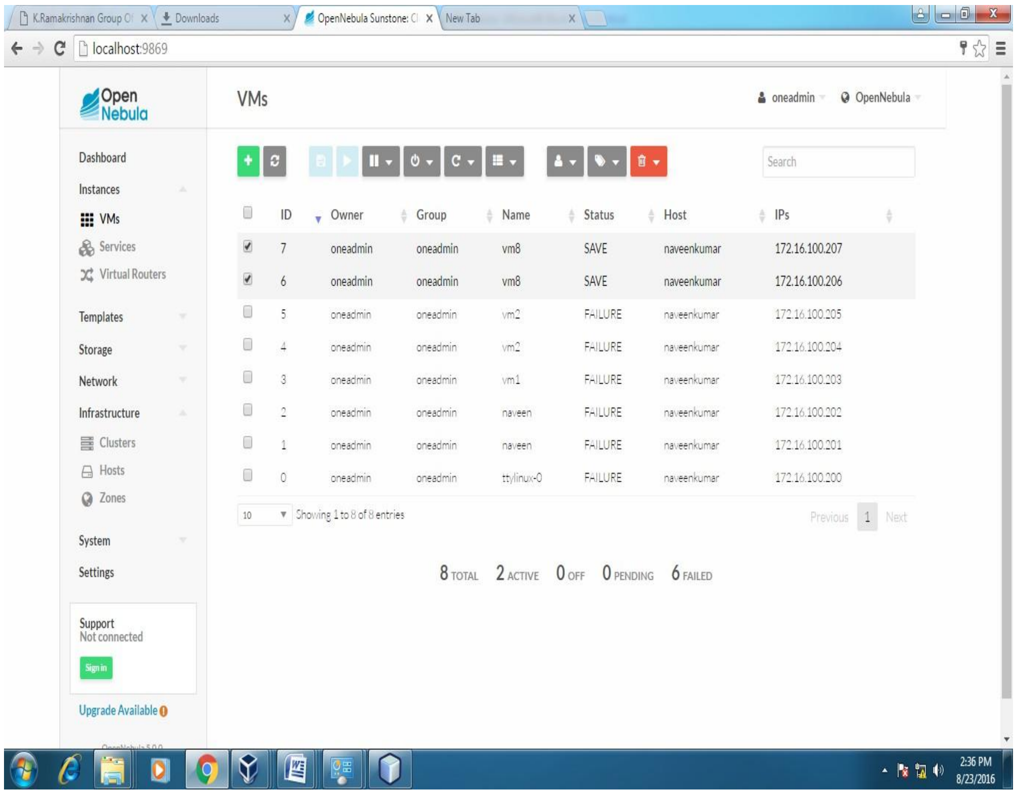
VM 6 vm8 is currently running on Host one-sandbox
VM 7 vm8 is currently running on Host one-sandbox

Select a Host

Please select a Host from the list

ID	Name	Cluster	RVMs	Allocated CPU	Allocated MEM	Status
2	raa	default	0	0 / 0	0KB / -	RETRY
1	naveenkumar	rama	6	62 / 0	44MB / -	ERROR
0	one-sandbox	rama	2	20 / 100 (20%)	4MB / 741MB (1%)	ON

Showing 1 to 3 of 3 entries

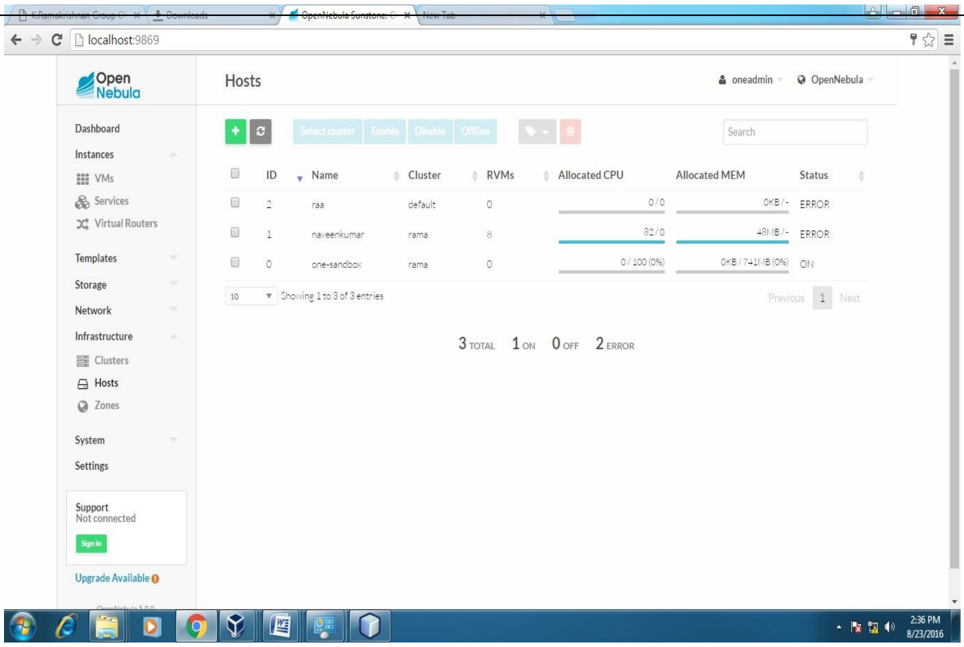


	ID	Owner	Group	Name	Status	Host	IPs
✓	7	oneadmin	oneadmin	vm8	SAVE	naveenkumar	172.16.100.207
✓	6	oneadmin	oneadmin	vm8	SAVE	naveenkumar	172.16.100.206
	5	oneadmin	oneadmin	vm2	FAILURE	naveenkumar	172.16.100.205
	4	oneadmin	oneadmin	vm2	FAILURE	naveenkumar	172.16.100.204
	3	oneadmin	oneadmin	vm1	FAILURE	naveenkumar	172.16.100.203
	2	oneadmin	oneadmin	naveen	FAILURE	naveenkumar	172.16.100.202
	1	oneadmin	oneadmin	naveen	FAILURE	naveenkumar	172.16.100.201
	0	oneadmin	oneadmin	ttylinux-0	FAILURE	naveenkumar	172.16.100.200

Showing 1 to 8 of 8 entries

8 TOTAL 2 ACTIVE 0 OFF 0 PENDING 6 FAILED

After Migration:

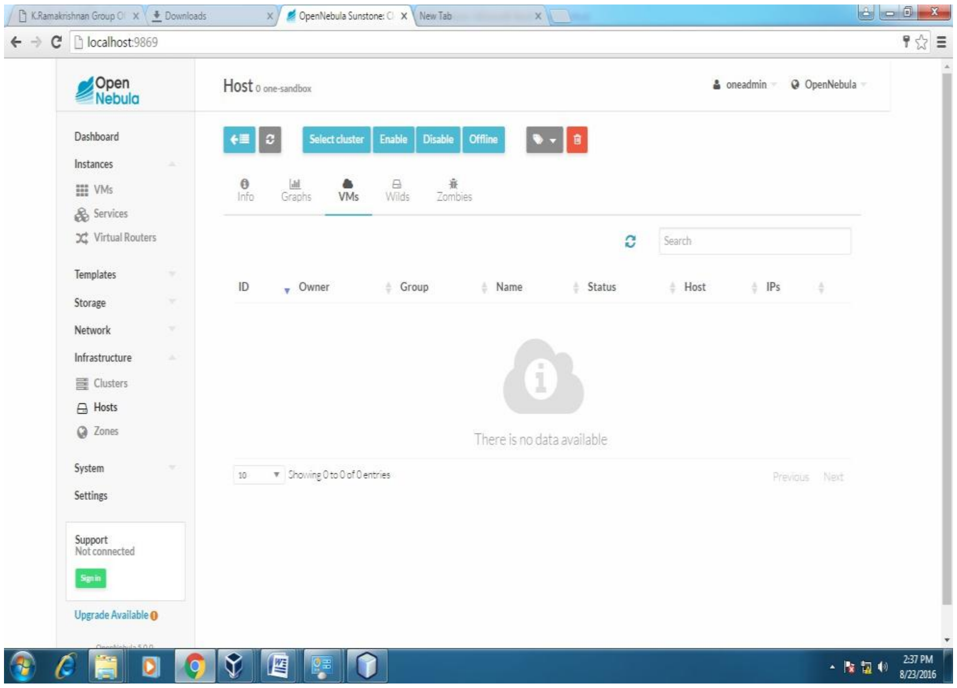

Hosts

ID	Name	Cluster	RVMs	Allocated CPU	Allocated MEM	Status
2	raa	default	0	0 / 0	0KB / -	ERROR
1	naveenkumar	rama	8	82 / 0	48MB / -	ERROR
0	one-sandbox	rama	0	0 / 100 (0%)	0KB / 7.41MB (0%)	ON

Showing 1 to 3 of 3 entries

3 TOTAL 1 ON 0 OFF 2 ERROR

Host:one-sandbox

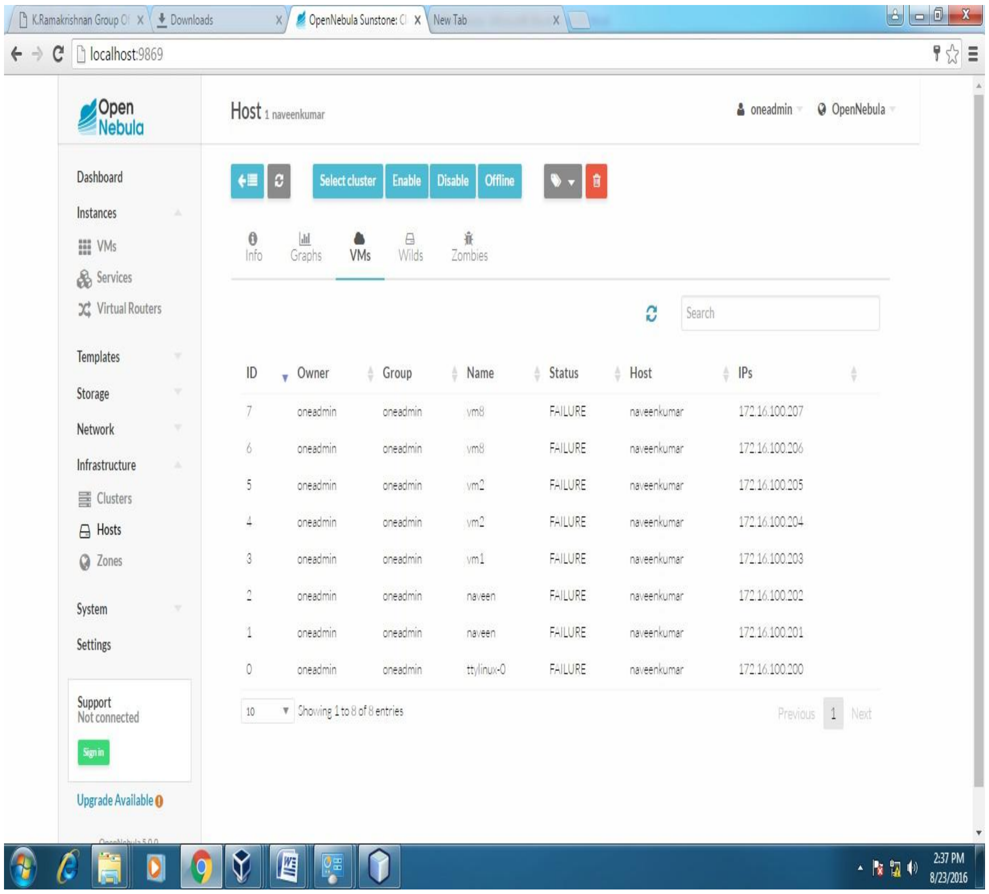

Host 0 one-sandbox

ID	Owner	Group	Name	Status	Host	IPs

There is no data available

Showing 0 to 0 of 0 entries

Host:SACET



localhost:9869

ID	Owner	Group	Name	Status	Host	IPs
7	oneadmin	oneadmin	vm8	FAILURE	naveenkumar	172.16.100.207
6	oneadmin	oneadmin	vm8	FAILURE	naveenkumar	172.16.100.206
5	oneadmin	oneadmin	vm2	FAILURE	naveenkumar	172.16.100.205
4	oneadmin	oneadmin	vm2	FAILURE	naveenkumar	172.16.100.204
3	oneadmin	oneadmin	vm1	FAILURE	naveenkumar	172.16.100.203
2	oneadmin	oneadmin	naveen	FAILURE	naveenkumar	172.16.100.202
1	oneadmin	oneadmin	naveen	FAILURE	naveenkumar	172.16.100.201
0	oneadmin	oneadmin	ttylinux-0	FAILURE	naveenkumar	172.16.100.200

Showing 1 to 8 of 8 entries

2:37 PM 8/23/2016

RESULT :

File transfer and virtual machine migration between two virtual machines were successfully executed using shared folders, clipboard, and OpenNebula orchestration tools.

Ex. No: 7	**LAUNCH VIRTUAL MACHINE USING TRYSTACK (ONLINE OPENSTACK DEMO VERSION)**
Date:	

AIM:

To demonstrate the procedure for launching a virtual machine using TryStack, an online demo version of OpenStack.

SOFTWARE/TOOLS USED:

- **TryStack** (OpenStack Online Demo Portal)
- **OpenStack Dashboard (Horizon GUI)**
- **Web Browser** (Google Chrome / Mozilla Firefox)
- **SSH Client** (Terminal in Linux/macOS or PuTTY for Windows)
- **Key Pair Generator** (e.g., ssh-keygen or PuTTYgen)

APPLICATION:

- Provisioning virtual machines in the cloud environment without local hardware.
- Demonstrating Infrastructure as a Service (IaaS) using OpenStack.
- Gaining practical exposure to OpenStack cloud management functions such as networking, instance creation, floating IP configuration, and security rules.
- Testing and deploying applications in a virtualized environment for software development, system administration, and cloud computing education.

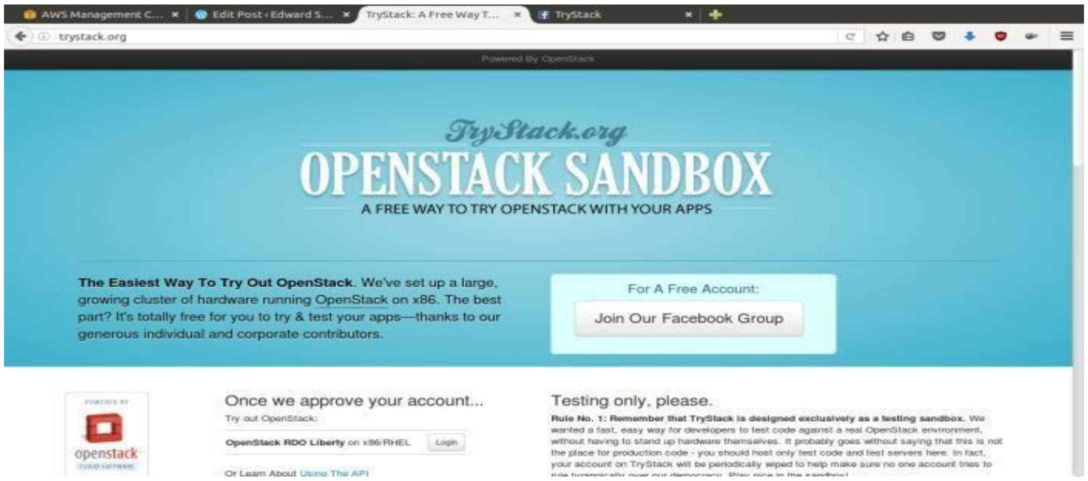
PROCEDURE:

OpenStack is an open-source cloud computing platform primarily utilized for deploying Infrastructure as a Service (IaaS) solutions, similar to commercial platforms such as Amazon Web Services (AWS). It enables the creation and management of scalable virtualized environments by offering compute, storage, and networking resources through a unified dashboard interface.

TryStack provides a free online demonstration environment for OpenStack, allowing users to explore OpenStack functionalities without local installation. Access to TryStack is granted upon membership approval through the official TryStack Facebook group, which may take a few days due to manual verification.

Once access is granted, logging into the TryStack web portal presents the **OpenStack Horizon Dashboard**, a web-based interface used for provisioning and managing cloud resources. The initial interface provides access to modules such as Compute, Network, and Identity.

This lab exercise involves deploying a virtual machine (referred to as an *instance*) that is configured with both internal networking and public internet access. The deployment architecture will follow a virtual topology where the instance is connected to a private subnet bridged to an external network via a virtual router, and the instance is assigned a floating (public) IP for remote access.



TryStack.org

OPENSTACK SANDBOX

A FREE WAY TO TRY OPENSTACK WITH YOUR APPS

The Easiest Way To Try Out OpenStack. We've set up a large, growing cluster of hardware running OpenStack on x86. The best part? It's totally free for you to try & test your apps—thanks to our generous individual and corporate contributors.

For A Free Account:

Join Our Facebook Group

Once we approve your account...

Try out OpenStack:

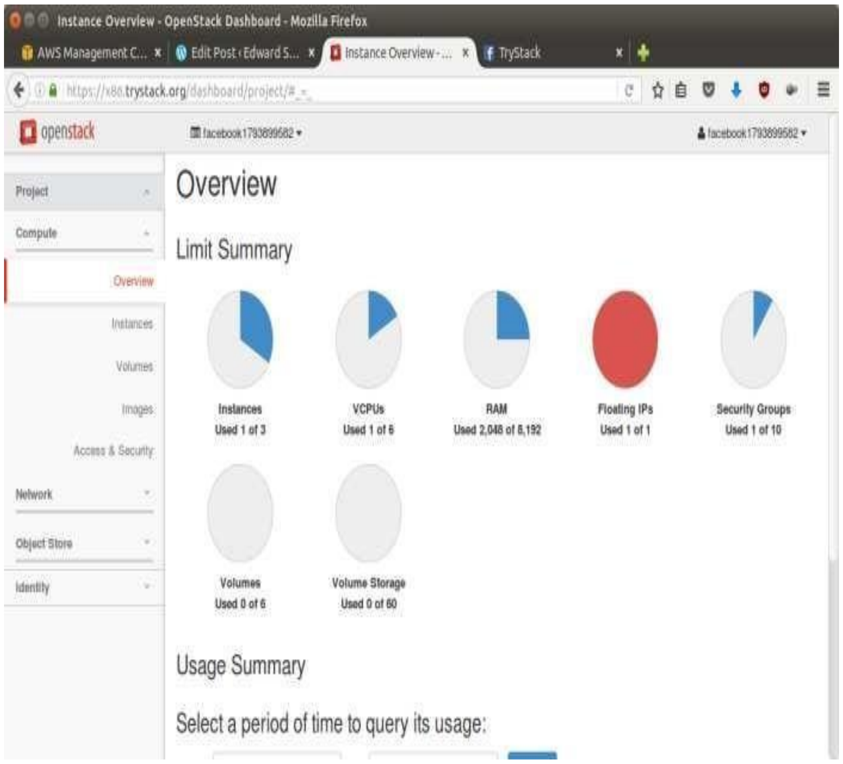
OpenStack RDO Liberty on x86/RHEL Login

Or Learn About Using The API

Testing only, please.

Rule No. 1: Remember that TryStack is designed exclusively as a testing sandbox. We wanted a fast, easy way for developers to test code against a real OpenStack environment, without having to stand up hardware themselves. It probably goes without saying that this is not the place for production code - you should host only test code and test servers here. In fact, your account on TryStack will be periodically wiped to help make sure no one account tries to

TryStack.org Homepage



Instance Overview - OpenStack Dashboard - Mozilla Firefox

https://x86.trystack.org/dashboard/project/#

Overview

Limit Summary

Instances Used 1 of 3

VCPUs Used 1 of 6

RAM Used 2,048 of 6,192

Floating IPs Used 1 of 1

Security Groups Used 1 of 10

Volumes Used 0 of 6

Volume Storage Used 0 of 60

Usage Summary

Select a period of time to query its usage:

OpenStack Compute Dashboard

This experiment demonstrates the procedure for provisioning and configuring a virtual machine instance using the TryStack online OpenStack environment. The objective is to deploy an instance within a logically isolated private network and configure it for internet accessibility through the assignment of a public (floating) IP address.
The resulting network topology will consist of:

- A private internal network (tenant network)
- A virtual router bridging the internal network with the external network
- One or more virtual instances attached to the internal network
- Public IP assignment enabling external SSH and HTTP access

This setup simulates a typical Infrastructure as a Service (IaaS) environment where virtualized resources are securely provisioned, networked, and exposed to the internet through controlled access mechanisms.

STEP 1 : Create a Virtual Network

To ensure logical isolation, a dedicated virtual network must be created. This network functions as a tenant-local LAN within the cloud environment.

STEP 2 :

1. Navigate to **Network > Networks** and select **Create Network**.
2. In the **Network** tab:
a. Specify the **Network Name**, e.g., internal.
b. Click **Next**.
3. In the **Subnet** tab:
a. Enter the **Network Address**, e.g., 192.168.1.0/24 (private CIDR block).
b. Select **IP Version** as IPv4.
c. Click **Next**.
4. In the **Subnet Details** tab:
a. Enter 8.8.8.8 under **DNS Name Servers**.
b. Click **Create** to finalize the network setup.

STEP 3 : Launch an Instance

1. An instance in OpenStack is a virtual machine provisioned on the cloud infrastructure.
2. Navigate to **Compute > Instances** and click **Launch Instance**.
3. In the **Details** tab:
4. Assign an **Instance Name**, e.g., Ubuntu-1.
5. Choose a **Flavor**, e.g., m1.medium.
6. Set **Instance Count** to 1.
7. Set **Instance Boot Source** to Boot from Image.
8. Select an **Image Name**, e.g., Ubuntu 14.04 amd64.
9. In the **Access & Security** tab:

10. Click the + icon under **Key Pair** to import a public key.
11. In the **Import Key Pair** dialog:
a. Enter the **Key Pair Name**, e.g., Client-Key.
b. Paste the **Public Key** from the local SSH key (e.g., from ~/.ssh/id_rsa.pub).
c. Click **Import Key Pair**.
12. Ensure the **default** security group is selected.
13. In the **Networking** tab:
14. Select the previously created network, e.g., internal.
15. Click **Launch** to deploy the instance.

STEP 4: Create and Configure a Router

1. To enable external internet connectivity for internal network instances, a router is required to bridge the internal network with the external (public) network.
2. Navigate to **Network > Routers** and click **Create Router**.
3. Specify a **Router Name**, e.g., router1, and click **Create Router**.
4. Access the router's **Details** page by clicking its name.
5. Click **Set Gateway**:
a. Select the **External Network**, typically named external.
b. Click **OK**.
6. Click **Add Interface**:
a. Select the **Subnet** created in Step 1.
b. Click **Add Interface**.
7. Navigate to **Network > Network Topology** to verify that the internal and external networks are correctly bridged via the router and instances are connected to the internal network.

STEP 5 : Allocate and Associate a Floating IP Address

Floating IPs are public IPs that enable access to instances from outside the cloud environment.
1. Navigate to **Compute > Instances**.
2. For the target instance, select **More > Associate Floating IP**.
3. In the **IP Address** field, click the + icon.
4. Set the **Pool** to external and click **Allocate IP**.
5. Click **Associate** to bind the floating IP to the selected instance.

Example: The instance may be assigned a public IP such as 8.21.28.120

STEP 6 : Configure Access and Security Rules

Security Groups in OpenStack control inbound and outbound traffic to instances, similar to firewall rules.
1. Navigate to **Compute > Access & Security > Security Groups** tab.
2. Locate the **default** group and click **Manage Rules**.

3. Add the following rules:
   - **ALL ICMP** to allow ping requests.
   - **HTTP** to allow web traffic (port 80).
   - **SSH** to allow secure shell access (port 22).
4. Additional ports may be opened by creating corresponding rules.

STEP 7 : Connect to the Instance via SSH

Access to the instance is now available via SSH using the floating IP address allocated in Step 4.

- Use a terminal (Linux/macOS) or PuTTY (Windows).

- Connect using the syntax:

```
ssh -i <private-key-file> ubuntu@<floating-ip>
```

- Example:

```
ssh -i
Client-Key.pem ubuntu@8.21.28.120
```

Note: The default user for Ubuntu images is ubuntu.

RESULT:

The lab exercise was successfully completed by following the standard procedures provided by OpenStack. It involved creating an isolated network, launching instances, configuring routers, associating floating IPs, and establishing secure access. The TryStack platform effectively demonstrated the capabilities of OpenStack in delivering a scalable and flexible IaaS solution suitable for enterprise-grade cloud computing.

Ex. No: 8	**INSTALL AND CONFIGURE A HADOOP SINGLE-NODE CLUSTER AND EXECUTE A SAMPLE APPLICATION (WORDCOUNT) TO VALIDATE THE SETUP.**
Date:	

AIM:

To install and configure a Hadoop single-node cluster and execute a sample application (WordCount) to validate the setup.

SOFTWARE/ TOOLS USED:

- **Operating System:** Ubuntu 20.04 LTS (or compatible Linux distribution)
- **Java Development Kit (JDK):** Java 8 (OpenJDK 1.8.0)
- **Hadoop Version:** Apache Hadoop 3.x (stable release)
- **Terminal/Command Line Interface**
- **Text Editor:** gedit, nano, or vim
- **SSH (Secure Shell):** Enabled for localhost communication

APPLICATION:

- **Big Data Processing:** Facilitates distributed processing of large datasets using the MapReduce programming model.
- **Data Analysis and Mining:** Suitable for batch processing tasks such as word frequency analysis, log parsing, and ETL operations.
- **Educational and Research Purpose:** Provides a simulated Hadoop environment for learning, testing, and prototyping in academic settings.

PROCEDURE:

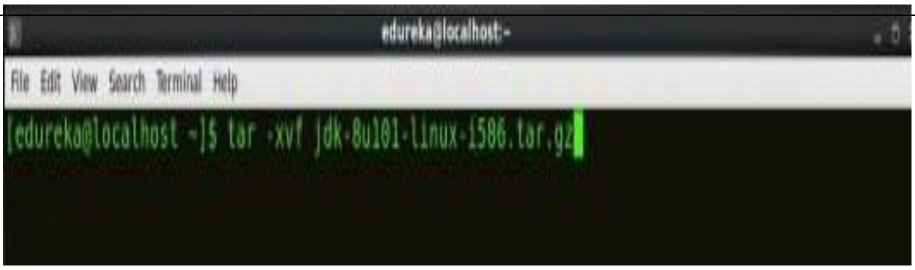
Hadoop Installation Procedure

Step 1: Download Java 8 Package

Download the required Java 8 Development Kit (JDK) and save the archive file in the home directory.
(Ensure compatibility with Hadoop by using Java 1.8.x only)

Step 2: Extract Java Tar File

Execute the following command to extract the Java archive:
bash
```
tar -xvf jdk-8u101-linux-i586.tar.gz
```
This unpacks the Java installation directory, which will later be configured in the environment path.

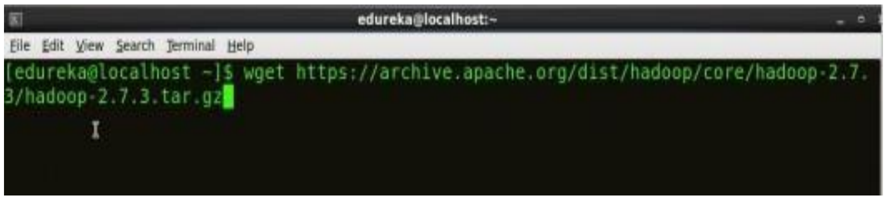


```
[edureka@localhost ~]$ tar -xvf jdk-8u101-linux-i586.tar.gz
```

Figure: Hadoop Installation – Extracting Java Files

Step 3: Download Hadoop 2.7.3 Package

Use the following command to download Hadoop 2.7.3 from the Apache archives:
bash
`wget` https://archive.apache.org/dist/hadoop/core/hadoop-2.7.3/hadoop-2.7.3.tar.gz



```
[edureka@localhost ~]$ wget https://archive.apache.org/dist/hadoop/core/hadoop-2.7.
3/hadoop-2.7.3.tar.gz
```

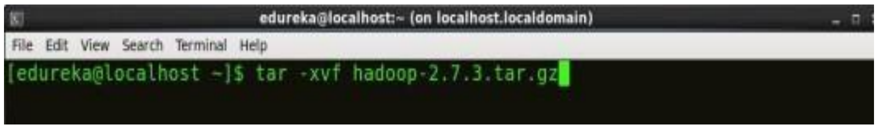
Figure: Hado op Installation – Downloading Hadoop

Step 4: Extract Hadoop Tar File

Unpack the Hadoop archive using the following command:
bash

```
tar -xvf hadoop-2.7.3.tar.gz
```

This will extract the Hadoop directory (`hadoop-2.7.3`) in the current location, which will be used to configure and launch the Hadoop environment.



```
[edureka@localhost ~]$ tar -xvf hadoop-2.7.3.tar.gz
```

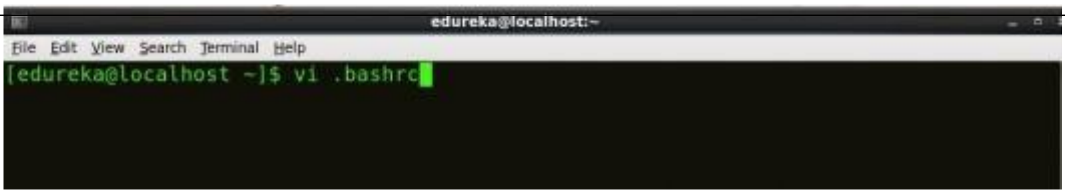
Fig: Hadoop Installation – Extracting Hadoop Files **Step**

Step 5: Configure Environment Variables

Update the system's environment variables to include the paths to the installed **Java** and **Hadoop** directories by modifying the `.bashrc` file.

Command to open .bashrc file:

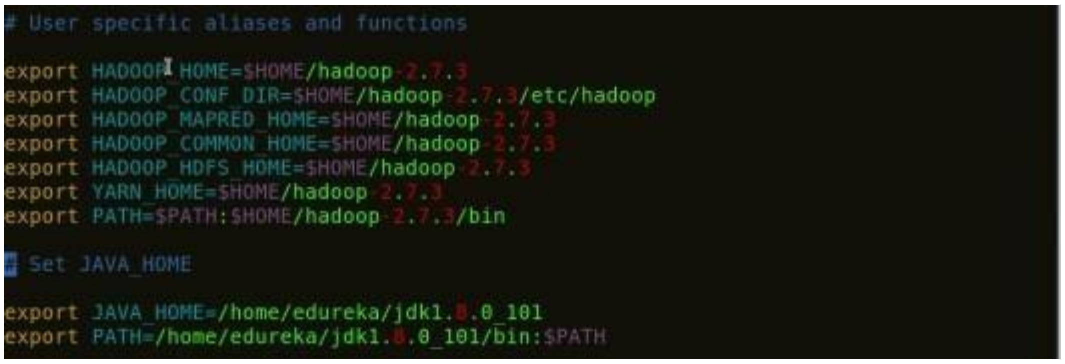
vi ~/.bashrc



```
[edureka@localhost ~]$ vi .bashrc
```

Add the following lines at the end of the .bashrc file. Update the paths according to the actual installation directories:

```
# Set JAVA environment
export JAVA_HOME=~/jdk1.8.0_101
export PATH=$JAVA_HOME/bin:$PATH

# Set Hadoop environment
export HADOOP_HOME=~/hadoop-2.7.3
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
```

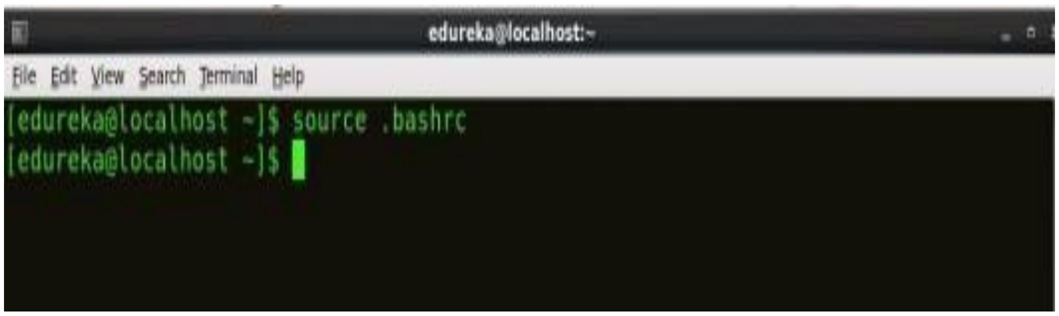


```
# User specific aliases and functions

export HADOOP_HOME=$HOME/hadoop-2.7.3
export HADOOP_CONF_DIR=$HOME/hadoop-2.7.3/etc/hadoop
export HADOOP_MAPRED_HOME=$HOME/hadoop-2.7.3
export HADOOP_COMMON_HOME=$HOME/hadoop-2.7.3
export HADOOP_HDFS_HOME=$HOME/hadoop-2.7.3
export YARN_HOME=$HOME/hadoop-2.7.3
export PATH=$PATH:$HOME/hadoop-2.7.3/bin

# Set JAVA_HOME

export JAVA_HOME=/home/edureka/jdk1.8.0_101
export PATH=/home/edureka/jdk1.8.0_101/bin:$PATH
```

Save the .bashrc file and close the editor.

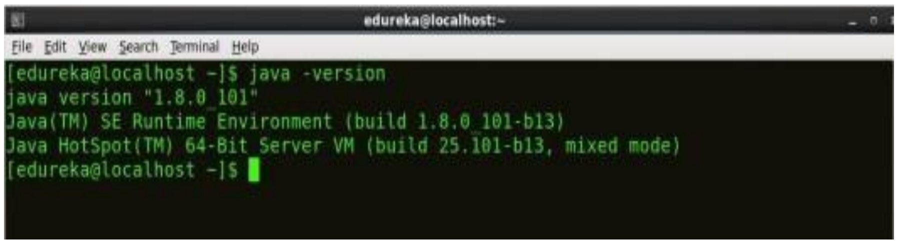
Step 6: Apply Environment Variable Changes
To apply the changes made to the .bashrc file in the current terminal session, use the source command:

Command:
source ~/.bashrc



```
[edureka@localhost ~]$ source .bashrc
[edureka@localhost ~]$ 
```

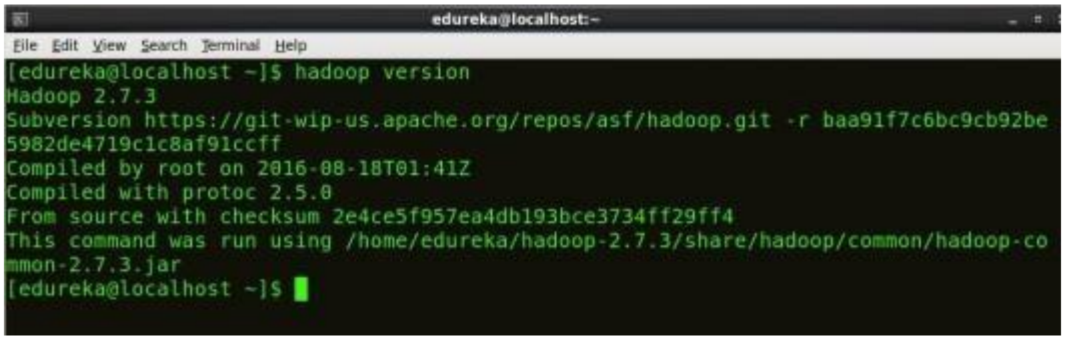
Step 7: Verify Java and Hadoop Installation
After setting the environment variables, verify whether **Java** and **Hadoop** are correctly installed and accessible through the terminal.
Check Java version:
bash
```
java -version
```



```
[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

Check Hadoop version:
hadoop version



```
[edureka@localhost ~]$ hadoop version
Hadoop 2.7.3
Subversion https://git-wip-us.apache.org/repos/asf/hadoop.git -r baa91f7c6bc9cb92be
5982de4719c1c8af91ccff
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-co
mmon-2.7.3.jar
[edureka@localhost ~]$ 
```

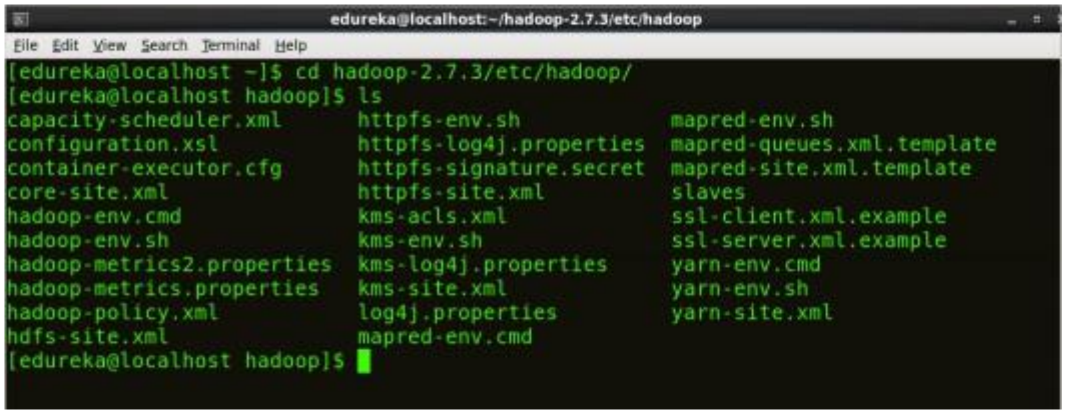
Fig: Hadoop Installation – Checking Java Version

Step 6: Edit the Hadoop Configuration Files
Navigate to the directory containing all configuration files necessary for setting up the Hadoop environment.

Command:
cd ~/hadoop-2.7.3/etc/hadoop/

To list all available configuration files within the Hadoop directory:

Command:
ls



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

Figure: Hadoop Installation – Hadoop Configuration Files Directory Listing

This directory contains key configuration files such as:

- core-site.xml
- hdfs-site.xml
- mapred-site.xml
- yarn-site.xml
- hadoop-env.sh

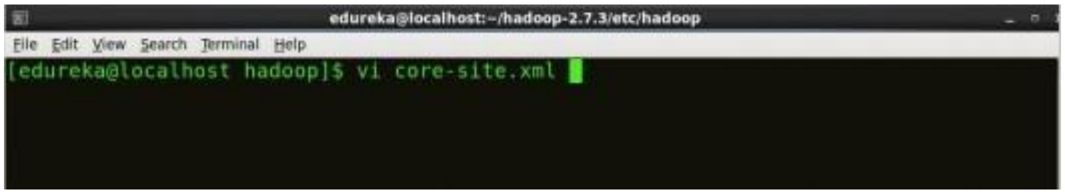
These configuration files must be edited to set up the Hadoop environment as a single-node cluster.

Step 7: Configure `core-site.xml`

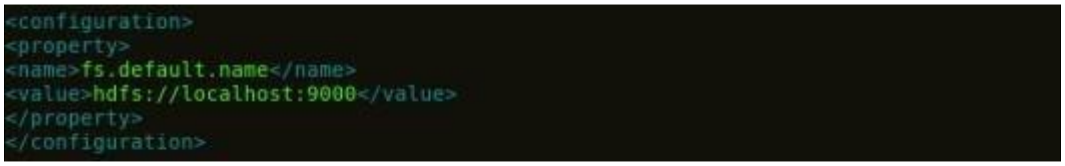
The `core-site.xml` file defines core Hadoop parameters and specifies the NameNode address within the cluster. It includes I/O settings shared by both HDFS and MapReduce frameworks.

Location:

~/hadoop-2.7.3/etc/hadoop/core-site.xml



```
[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

Command to Open:

vi core-site.xml

Within the `<configuration>` tag, insert the following property block to define the default file system:

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

</property>

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

</property>

Fig: Hadoop Installation – Configuring core-site.xml

- fs.defaultFS specifies the Hadoop Distributed File System (HDFS) URL.
- localhost refers to the single-node setup.
- Port 9000 is the default listening port for the NameNode.

SStep 8: Configure `hdfs-site.xml`

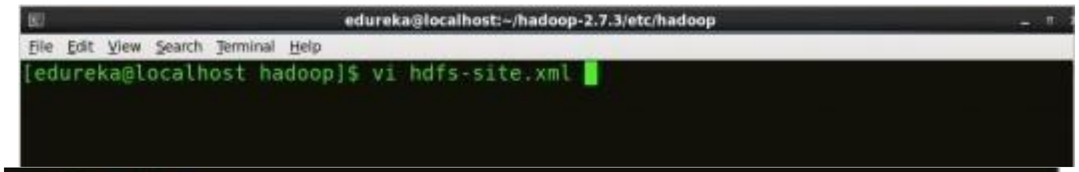
The `hdfs-site.xml` file defines configuration parameters for HDFS components such as the NameNode, DataNode, and Secondary NameNode. It includes settings for block replication, storage directories, and other filesystem properties.

Location:

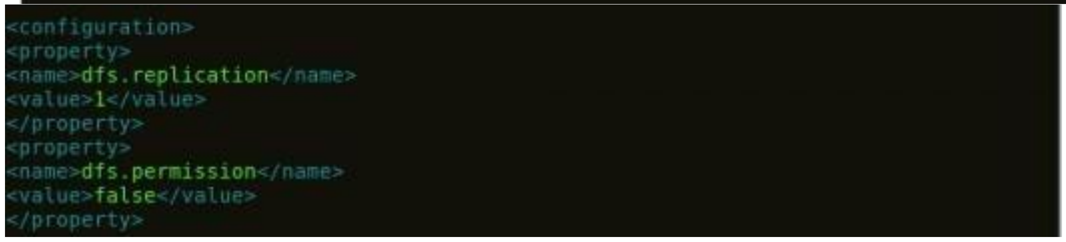
bash
```
~/hadoop-2.7.3/etc/hadoop/hdfs-site.xml
```

Command to Open:

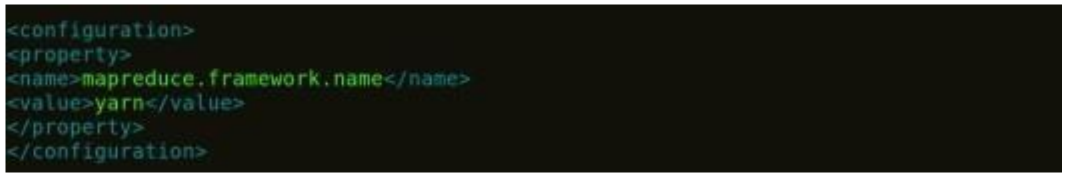
bash
```
vi hdfs-site.xml
```


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>
```



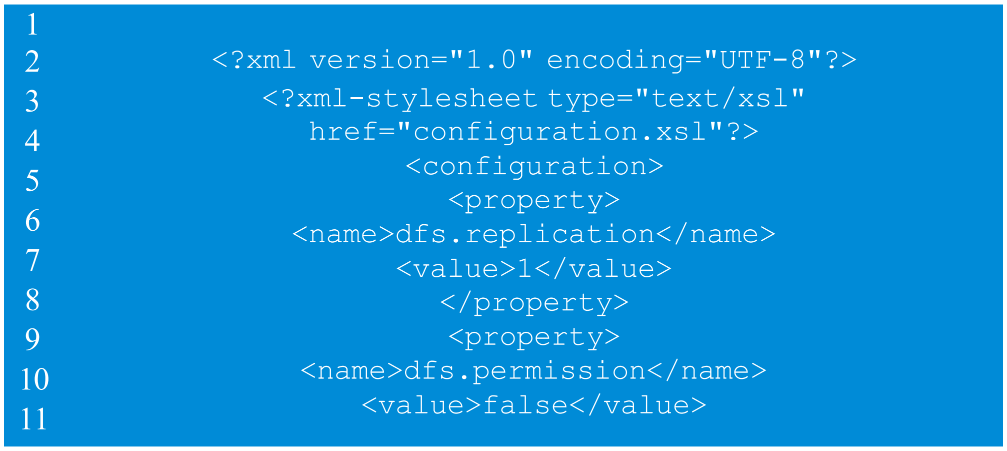
```
<configuration>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
</configuration>
```

Within the `<configuration>` tag, insert the following property blocks to set the replication factor and define storage directories:

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

   <property>
```

```
        <name>dfs.namenode.name.dir</name>
        <value>file:///home/hduser/hadoopdata/hdfs/namenode</value>
    </property>

    <property>
        <name>dfs.datanode.data.dir</name>
        <value>file:///home/hduser/hadoopdata/hdfs/datanode</value>
    </property>
</configuration>
```

- dfs.replication defines the number of data copies stored in HDFS. A value of 1 is optimal for a single-node cluster.
- dfs.namenode.name.dir and dfs.datanode.data.dir specify local directories for storing metadata and block data, respectively. Ensure these directories exist or are created before formatting HDFS.



```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl"
href="configuration.xsl"?>
<configuration>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
<property>
<name>dfs.permission</name>
<value>false</value>
```

Fig: Hadoop Installation – Configuring hdfs-site.xml

Step 9: Configure `mapred-site.xml`
The `mapred-site.xml` file specifies the configuration for MapReduce applications, including execution mode, resource allocation, and job scheduling. This configuration is critical for defining how MapReduce jobs are executed within the Hadoop ecosystem.
Note: In Hadoop 2.x distributions, the `mapred-site.xml` file may not be available by default. Instead, a template file named `mapred-site.xml.template` is provided. It must be copied and renamed before editing.

Commands:

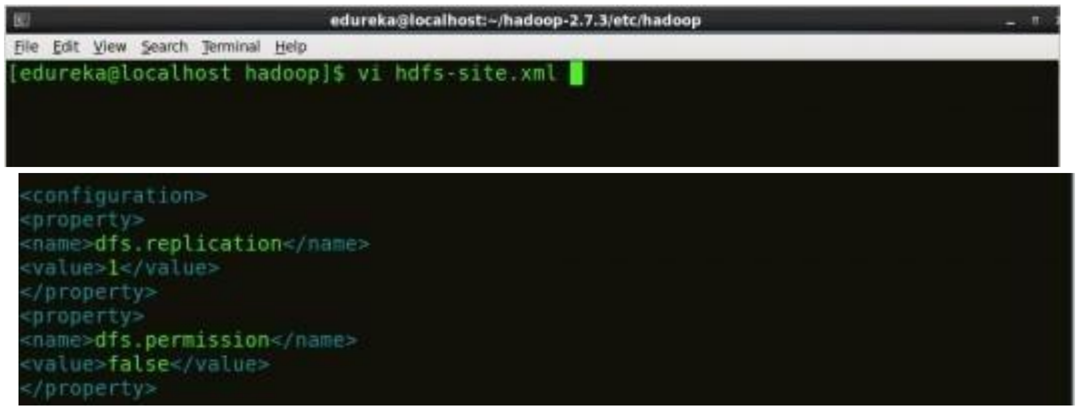
```
cd ~/hadoop-2.7.3/etc/hadoop/
cp mapred-site.xml.template mapred-site.xml
```

vi mapred-site.xml

Add the following content inside the <configuration> tag:

xml

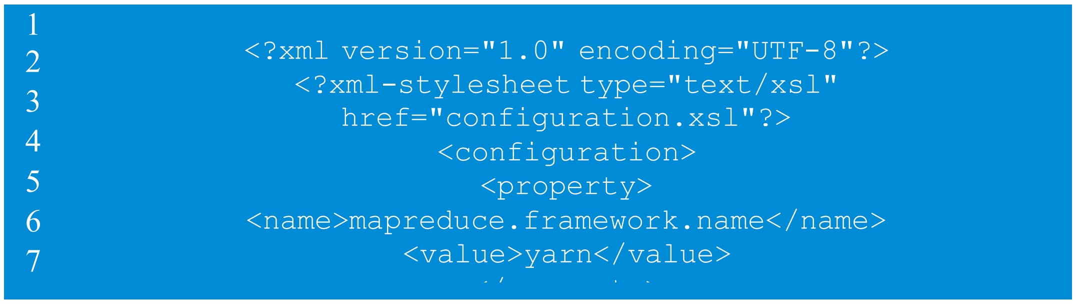
```
<configuration>
   <property>
      <name>mapreduce.framework.name</name>
      <value>yarn</value>
   </property>
</configuration>
```


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 yarn-site.xml



```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl"
href="configuration.xsl"?>
<configuration>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
```

Step 11: Configure `hadoop-env.sh`

The `hadoop-env.sh` file contains essential environment variable declarations required for the execution of Hadoop services. One of the most critical variables is the `JAVA_HOME` path, which must be set to the directory of the installed Java Development Kit (JDK).

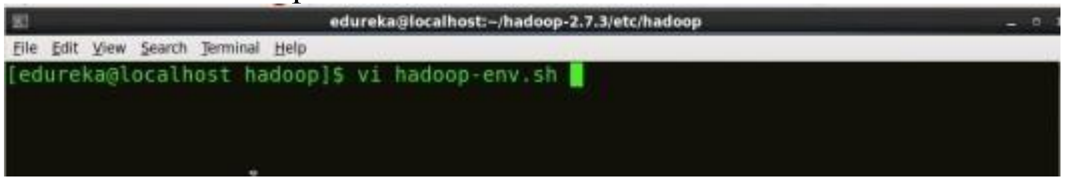
Purpose: Defines the Java environment required by Hadoop daemons during runtime.

Command: vi hadoop-env.sh
Edit or append the following line: export
JAVA_HOME=/home/username/jdk1.8.0_101
Replace /home/username/jdk1.8.0_101 with the actual path where Java is extracted.
Save and exit the file after modification.
Figure: Editing hadoop-env.sh to set JAVA_HOME

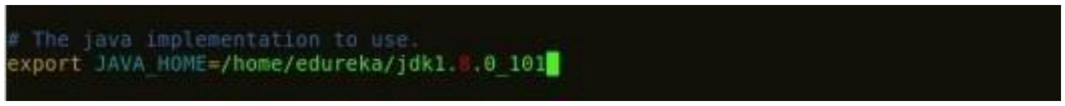
This step ensures that all Hadoop components reference the correct Java environment, preventing errors during the execution of daemons such as NameNode, DataNode, and ResourceManager.

```
<?xml version="1.0">
<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>
```

Command: vi hadoop–env.sh


edureka@localhost:~/hadoop-2.7.3/etc/hadoop
File Edit View Search Terminal Help

```
[edureka@localhost hadoop]$ vi hadoop-env.sh
```



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

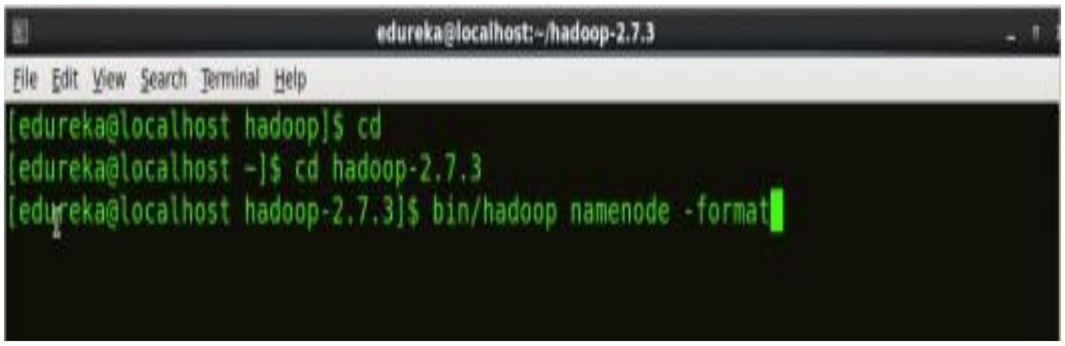
Fig: Hadoop Installation – Configuring hadoop-env.sh

Step 12: Format the NameNode
Formatting the NameNode initializes the Hadoop Distributed File System (HDFS) by setting up the required filesystem structure within the directory defined by the `dfs.name.dir` configuration property. This step prepares the NameNode for managing metadata of the distributed filesystem.
Note: Formatting should be performed **only once during initial setup**. Reformatting an active HDFS will result in **irrecoverable data loss**.

Navigate to the Hadoop home directory: cd
cd hadoop-2.7.3

Format the NameNode: bin/hadoop namenode -format

Once successfully formatted, a new file system structure will be created, and Hadoop is ready to start its core daemons.


edureka@localhost:~/hadoop-2.7.3
File Edit View Search Terminal Help

```
[edureka@localhost hadoop]$ cd
[edureka@localhost ~]$ cd hadoop-2.7.3
[edureka@localhost hadoop-2.7.3]$ bin/hadoop namenode -format
```

Fig: Hadoop Installation – Formatting NameNode

Step 13: Start Hadoop Daemons
After formatting the NameNode, the essential Hadoop daemons must be started to initiate the Hadoop Distributed File System (HDFS), YARN resource manager, and the MapReduce history server. These daemons coordinate storage, processing, and resource management across the cluster.
Navigate to the `sbin` directory:
bash
```
cd hadoop-2.7.3/sbin
```

Option A: Start All Daemons at Once
Use the following command to start all core Hadoop services simultaneously:
bash
```
./start-all.sh
```
This command internally triggers the following individual scripts:

- `start-dfs.sh` – Starts HDFS daemons (NameNode, DataNode)
- `start-yarn.sh` – Starts YARN daemons (ResourceManager, NodeManager)
- `mr-jobhistory-daemon.sh` – Starts MapReduce JobHistory Server

Option B: Start Services Individually

Start NameNode:

bash
CopyEdit

```
./hadoop-daemon.sh start namenode
```

Figure: Starting the HDFS NameNode daemon

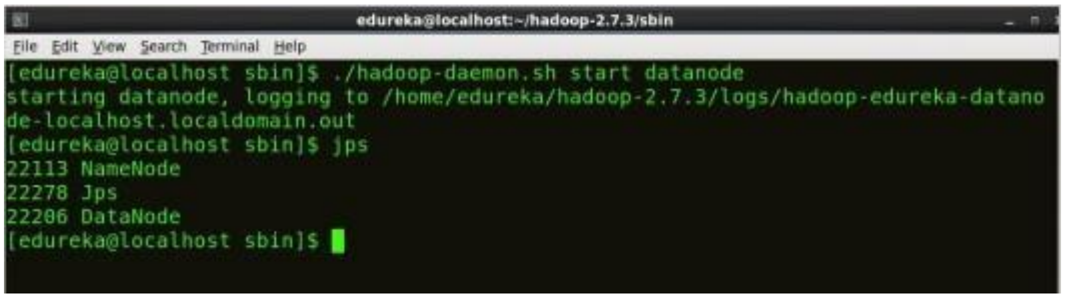
The **NameNode** is the master server of HDFS. It manages the metadata of the file system and tracks the location of blocks across DataNodes.

Other services like **DataNode**, **ResourceManager**, **NodeManager**, and **JobHistoryServer** can also be started individually using their respective commands.

Step 14 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



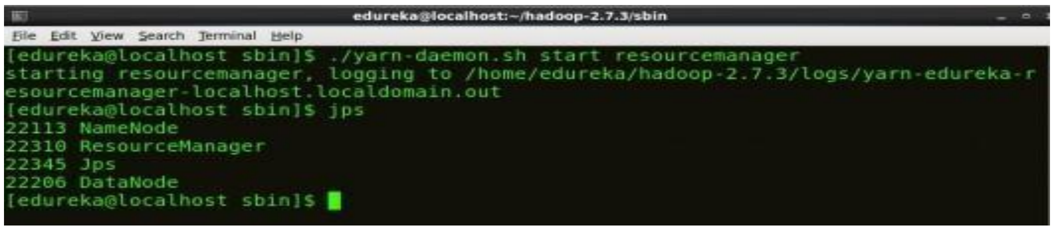
```
[edureka@localhost sbin]$ ./hadoop-daemon.sh start datanode
starting datanode, logging to /home/edureka/hadoop-2.7.3/logs/hadoop-edureka-datano
de-localhost.localdomain.out
[edureka@localhost sbin]$ jps
22113 NameNode
22278 Jps
22206 DataNode
[edureka@localhost sbin]$
```

Fig: Hadoop Installation – Starting DataNode

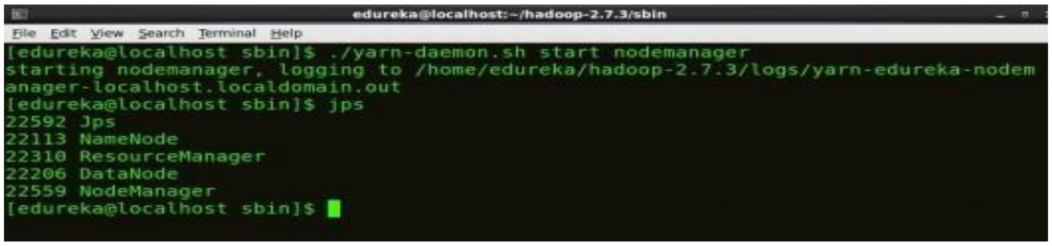
Start ResourceManager:

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 sbin]$ ./yarn-daemon.sh start resourcemanager
starting resourcemanager, logging to /home/edureka/hadoop-2.7.3/logs/yarn-edureka-r
esourcemanager-localhost.localdomain.out
[edureka@localhost sbin]$ jps
22113 NameNode
22310 ResourceManager
22345 Jps
22206 DataNode
[edureka@localhost sbin]$
```

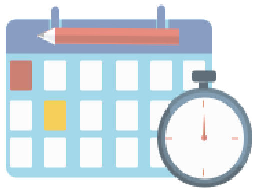

edureka@localhost:~/hadoop-2.7.3/sbin
File Edit View Search Terminal Help

```
[edureka@localhost sbin]$ ./yarn-daemon.sh start nodemanager
starting nodemanager, logging to /home/edureka/hadoop-2.7.3/logs/yarn-edureka-nodem
anager-localhost.localdomain.out
[edureka@localhost sbin]$ jps
22592 Jps
22113 NameNode
22310 ResourceManager
22206 DataNode
22559 NodeManager
[edureka@localhost sbin]$
```

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



See Batch Details

Fig: Hadoop Installation – Starting NodeManager

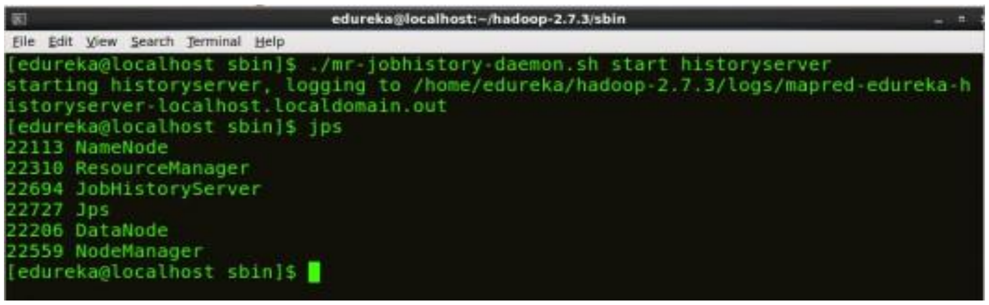
Start JobHistoryServer:

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.**

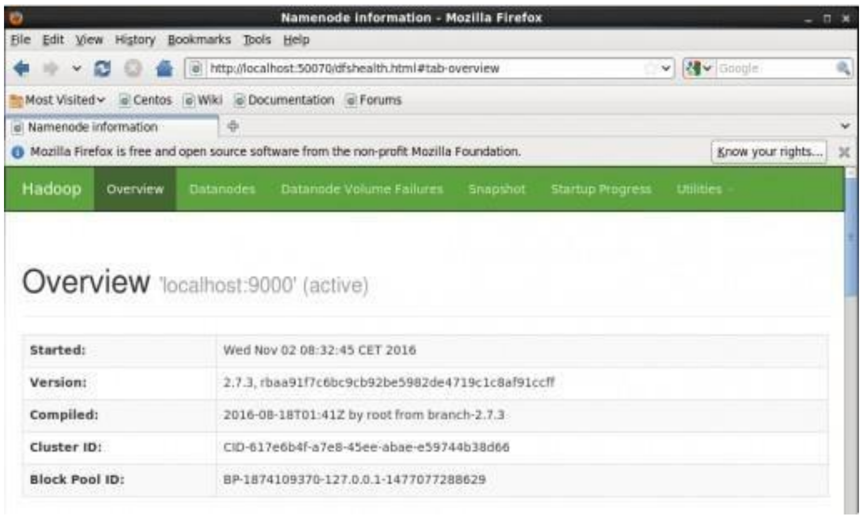
Command: jps


edureka@localhost:~/hadoop-2.7.3/sbin
File Edit View Search Terminal Help

```
[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


Namenode information - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:50070/dfshealth.html#tab-overview
Google
Most Visited Centos Wiki Documentation Forums
Namenode information
Mozilla Firefox is free and open source software from the non-profit Mozilla Foundation.
Know your rights...
Hadoop Overview Datanodes Datanode Volume Failures Snapshot Startup Progress Utilities

Overview 'localhost:9000' (active)

Started:	Wed Nov 02 08:32:45 CET 2016
Version:	2.7.3, rbaa91f7c6bc9cb92be5982de4719c1c8af91ccff
Compiled:	2016-08-18T01:41Z by root from branch-2.7.3
Cluster ID:	CID-617e6b4f-a7e8-45ee-abae-e59744b38d66
Block Pool ID:	BP-1874109370-127.0.0.1-1477077288629

check the NameNode interface.

Fig: Hadoop Installation – Starting WebUI

Congratulations, you have successfully installed a single node Hadoop cluster

RESULT:

The single-node Hadoop cluster setup was completed successfully. The MapReduce-based WordCount application was executed, demonstrating the capability of Hadoop to perform distributed processing of data, even in a minimal node environment.

Ex. No: 9	**CREATING AND RUNNING DOCKER CONTAINERS**
Date:	

AIM:

To create and execute a Docker container using Docker Desktop, demonstrating containerization and basic Docker operations on a Windows system.

SOFTWARE/ TOOLS USED:

- Docker Desktop for Windows
- Windows 10/11 with WSL 2 enabled
- PowerShell / Command Prompt / Docker CLI

  Internet connection for Docker Hub access

APPLICATIONS:

- Demonstrates containerization technology and its real-time applications.
- Helps in isolating environments for application deployment.
- Enables developers to build, ship, and run applications efficiently using containers.

Useful for microservices, DevOps pipelines, and CI/CD automation

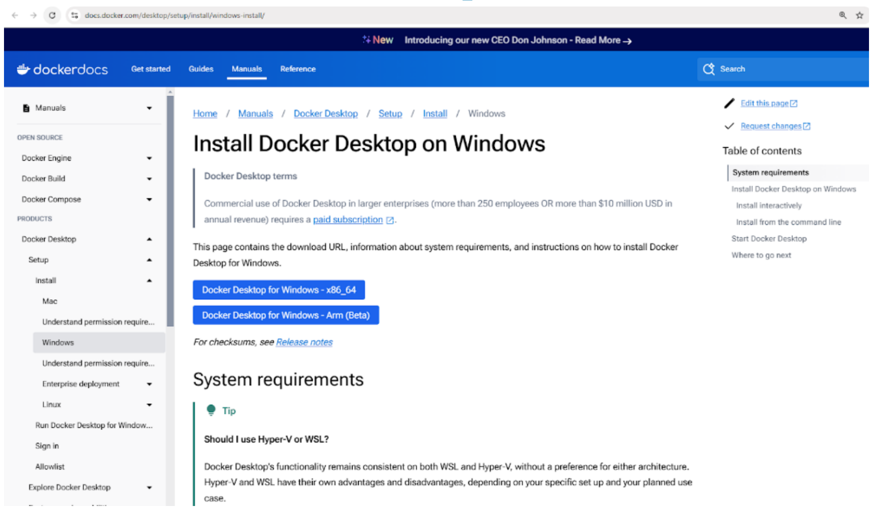
PROCEDURE:

Installing Docker

Docker can be installed on various operating systems, including Windows, macOS, and Linux. While the core functionality remains the same across all platforms, the installation process differs slightly depending on the system. Below, you'll find step-by-step instructions for installing Docker on your preferred operating system.

Installing Docker on Windows

1. Download **Docker Desktop for Windows**.



docs.docker.com/desktop/setup/install/windows-install/

Home / Manuals / Docker Desktop / Setup / Install / Windows

Install Docker Desktop on Windows

Docker Desktop terms

Commercial use of Docker Desktop in larger enterprises (more than 250 employees OR more than $10 million USD in annual revenue) requires a paid subscription.

This page contains the download URL, information about system requirements, and instructions on how to install Docker Desktop for Windows.

Docker Desktop for Windows - x86_64

Docker Desktop for Windows - Arm (Beta)

For checksums, see Release notes

System requirements

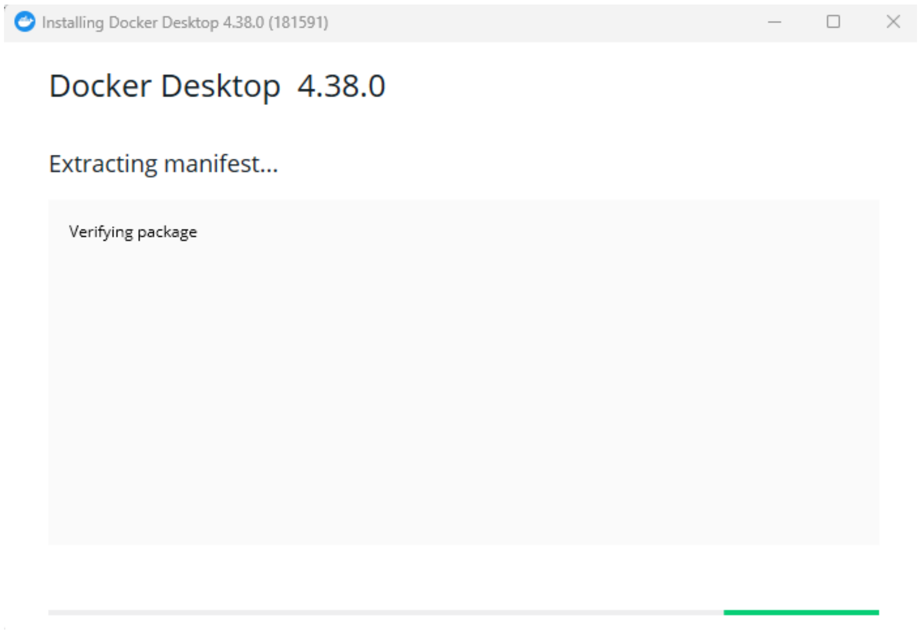
Tip

Should I use Hyper-V or WSL?

Docker Desktop's functionality remains consistent on both WSL and Hyper-V, without a preference for either architecture. Hyper-V and WSL have their own advantages and disadvantages, depending on your specific set up and your planned use case.

Download Docker Desktop Installer for Windows

2. Run the installer and follow the setup instructions.



Installing Docker Desktop 4.38.0 (181591)

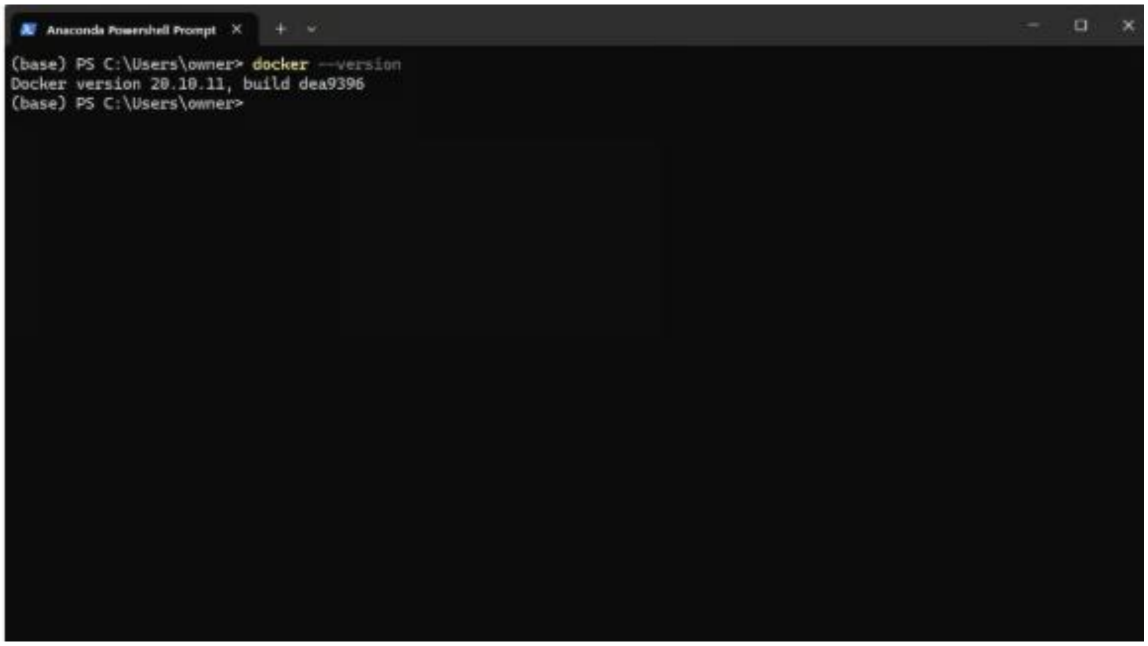
Docker Desktop 4.38.0

Extracting manifest...

Verifying package

Installing Docker Desktop for Windows

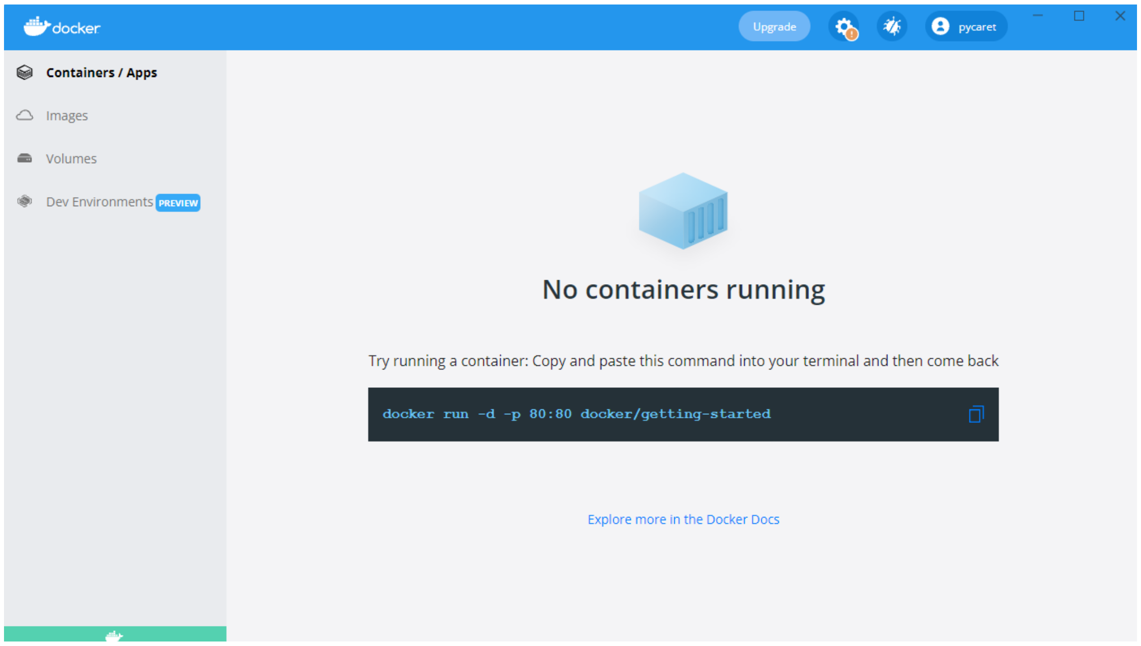
3. Enable WSL 2 integration if prompted.
4. Verify installation by running `docker –version` in PowerShell.



```
(base) PS C:\Users\owner> docker --version
Docker version 20.10.11, build dea9396
(base) PS C:\Users\owner>
```

Checking Docker version after installation through Powershell

5. Start the Docker Desktop app from your run menu.



No containers running

Try running a container: Copy and paste this command into your terminal and then come back

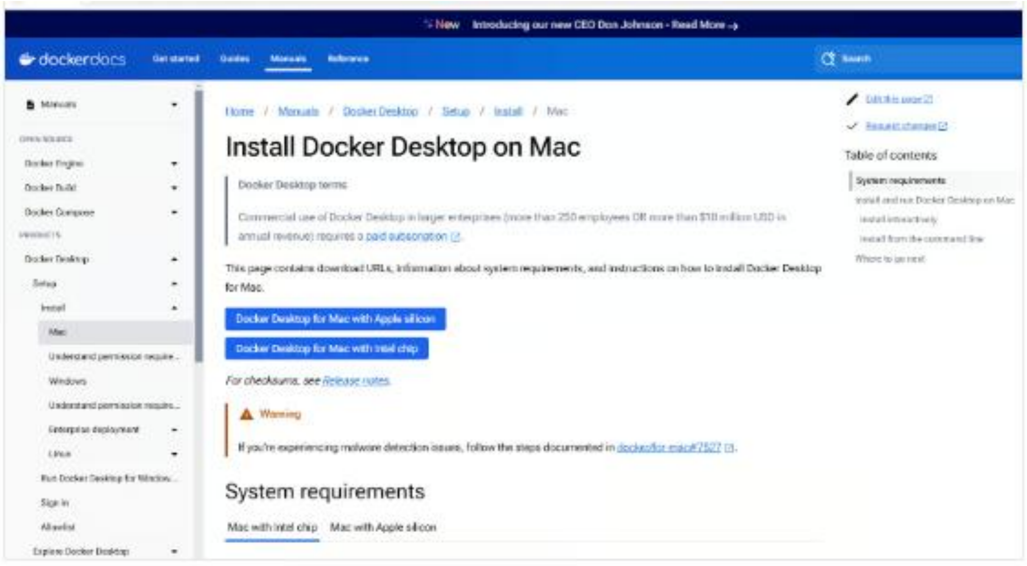
```
docker run -d -p 80:80 docker/getting-started
```

Explore more in the Docker Docs

Launching Docker Desktop Application on Windows

Installing Docker on macOS

1. Download **Docker Desktop for Mac**.



Home / Manuals / Docker Desktop / Setup / Install / Mac

Install Docker Desktop on Mac

Docker Desktop terms

Commercial use of Docker Desktop in larger enterprises (more than 250 employees OR more than $10 million USD in annual revenue) requires a paid subscription.

This page contains download URLs, information about system requirements, and instructions on how to install Docker Desktop for Mac.

Docker Desktop for Mac with Apple silicon

Docker Desktop for Mac with Intel chip

For checksums, see Release notes.

Warning

If you're experiencing malware detection issues, follow the steps documented in docker/for-mac#7527.

System requirements

Mac with Intel chip | Mac with Apple silicon

Download Docker Desktop installer for Mac

2. Open the downloaded `.dmg` file and drag Docker to the Applications folder.
3. Launch Docker and complete the setup.
4. Verify installation using `docker -version` in the terminal.

Installing Docker on Linux (Ubuntu)

1. Update package lists: `sudo apt update`

2. Install dependencies: `sudo apt install apt-transport-https ca-certificates curl software-properties-common`
3. Add Docker's official GPG key: `curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -`
4. Add Docker's repository: `sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"`
5. Install Docker: `sudo apt install docker-ce`
6. Verify installation: `docker –version`

Basic Docker Concepts

At the heart of Docker are **images**, which serve as blueprints for containers; **containers**, which are the running instances of these images; and **Docker Hub**, a centralized repository for sharing and managing images.

Docker images

Docker images are the fundamental building blocks of containers. They are immutable, read-only templates containing everything needed to run an application, including the operating system, application code, runtime, and dependencies.

Images are built using a `Dockerfile`, which defines the instructions for creating an image layer by layer.

Images can be stored in and retrieved from container registries such as Docker Hub.

Here are some example commands for working with images:

- `docker pull nginx`: Fetch the latest Nginx image from Docker Hub.
- `docker images`: List all available images on the local machine.
- `docker rmi nginx`: Remove an image from the local machine.

Docker containers

A Docker container is a running instance of a Docker image. Containers provide an isolated runtime environment where applications can run without interfering with each other or the host system.Each container has its own filesystem, networking, and process space but shares the host kernel.

Containers follow a simple lifecycle involving creation, starting, stopping, and deletion. Here's a breakdown of common container management commands:

1. Creating a container: `docker create` or `docker run`
2. Starting a container: `docker start`
3. Stopping a container: `docker stop`
4. Restarting a container: `docker restart`
5. Deleting a container: `docker rm`

The following command runs an Nginx container in detached mode (running in the background), mapping port 80 inside the container to port 8080 on the host machine:

```
docker run -d -p 8080:80 nginx
```

POWERED BY

After running this command, Docker will pull the Nginx image (if not already available), create a container, and start it.

To check all running and stopped containers:

```
docker ps -a
```

POWERED BY

This will display a list of all containers and details like their status and assigned ports.

Docker Hub

Docker Hub is a cloud-based registry service for finding, storing, and distributing container images. Users can push custom images to Docker Hub and share them publicly or privately.

Here are some commands for interacting with Docker Hub:

- `docker login`: Authenticate with Docker Hub.
- `docker push my-image`: Upload a custom-built image to Docker Hub.
- `docker search ubuntu`: Search for official and community images.
- `docker pull ubuntu`: Download an Ubuntu image from Docker Hub.

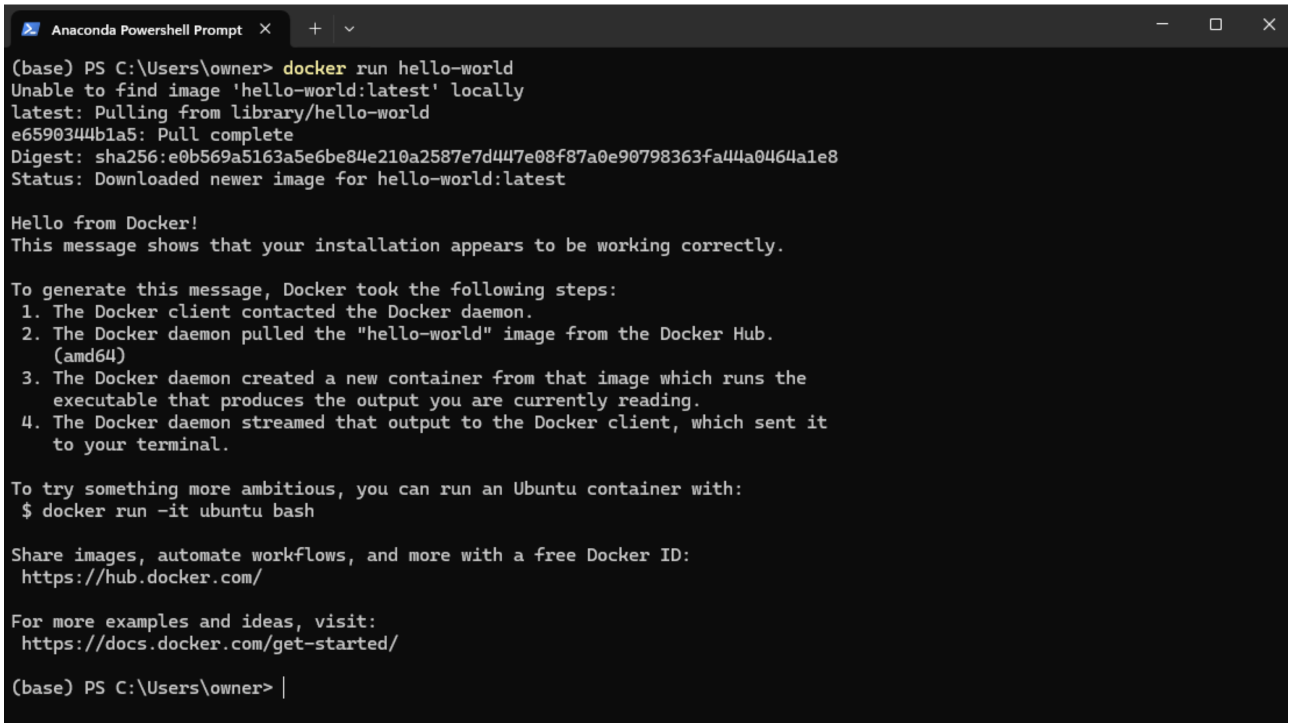
Running Your First Docker Container

To test your Docker installation, open PowerShell (Windows) or Terminal (Mac and Linux) and run:

```
docker run hello-world
```

POWERED BY

This pulls the `hello-world` image from DockerHub and runs it in a container.



Anaconda Powershell Prompt

```
(base) PS C:\Users\owner> docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
e6590344b1a5: Pull complete
Digest: sha256:e0b569a5163a5e6be84e210a2587e7d447e08f87a0e90798363fa44a0464a1e8
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

(base) PS C:\Users\owner> |
```

Run a real-world application—an Nginx web server. Execute the following command:

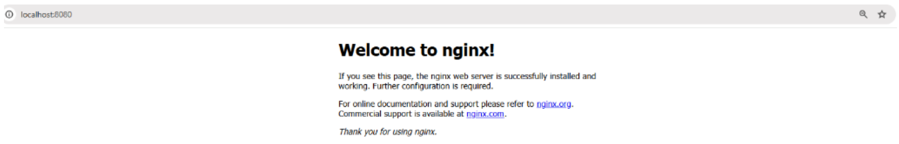
```
docker run -d -p 8080:80 nginx
```

POWERED BY

The above command does the following:

- The `-d` flag runs the container in detached mode, meaning it runs in the background.
- The `-p 8080:80` flag maps port 80 inside the container to port 8080 on your local machine, allowing you to access the web server.

Once the command runs successfully, open a browser and visit: `http://localhost:8080`



localhost:8080

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

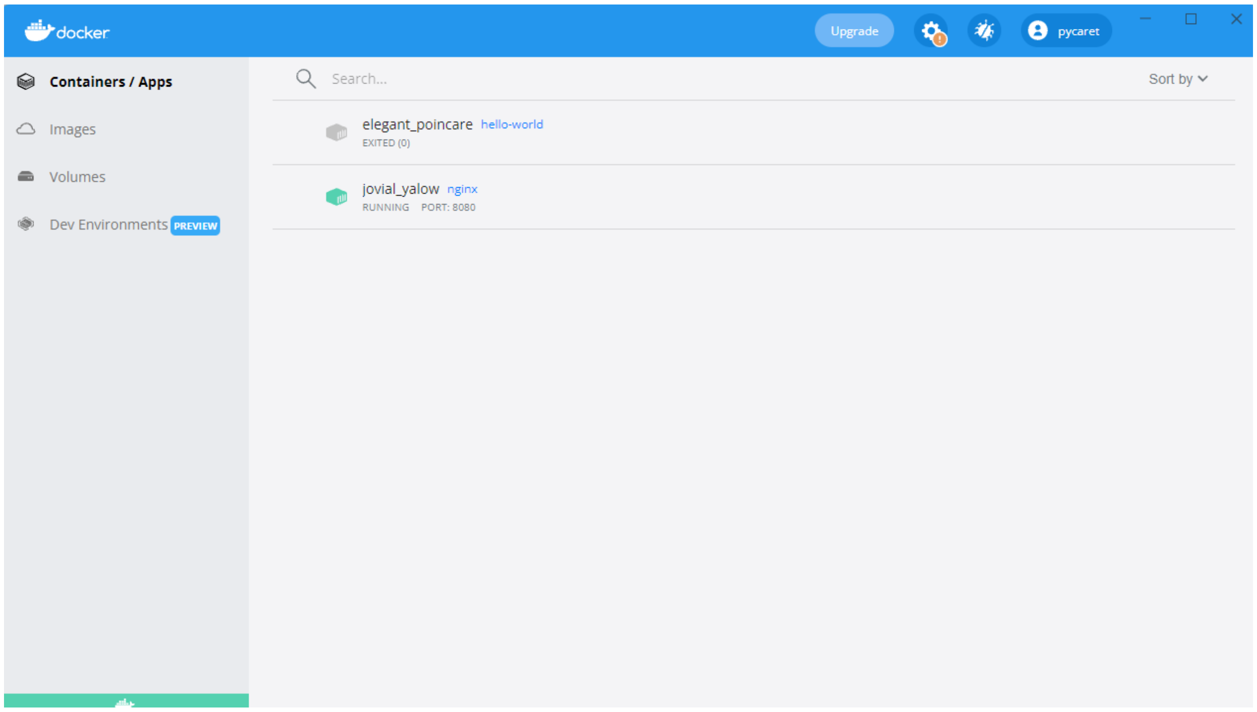
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Accessing web server at localhost:8080

The Default Nginx welcome page appears, confirming that web server is running inside a container!

We can see a container running in your Docker Desktop:



elegant_poincare hello-world
EXITED (0)

jovial_yalow nginx
RUNNING PORT: 8080

Nginx container running on port 8080

Building Your First Docker Image

Creating a Docker image involves writing a `Dockerfile`, a script that automates image-building. This ensures consistency and portability across different environments. Once an image is built, it can be run as a container to execute applications in an isolated environment.

Dockerfile basics

A `Dockerfile` is a script containing a series of instructions that define how a Docker image is built. It automates the image creation process, ensuring consistency across environments. Each instruction in a `Dockerfile` creates a new layer in the image. Here's a breakdown of an example Dockerfile for a simple Python Flask app:

```
# Base image containing Python runtime
FROM python:3.9

# Set the working directory inside the container
WORKDIR /app

# Copy the application files from the host to the container
```

```
COPY . /app

# Install the dependencies listed in requirements.txt
RUN pip install -r requirements.txt

# Define the command to run the Flask app when the container starts
CMD ["python", "app.py"]
```

POWERED BY

In the above command:

- `-v my-volume:/app/data` mounts the `my-volume` storage to the `/app/data` directory inside the container.
- Any data stored in `/app/data` will persist even if the container stops or is removed.

Breaking down the Dockerfile above:

- `FROM python:3.9`: Specifies the base image with Python 3.9 pre-installed.
- `WORKDIR /app`: Sets `/app` as the working directory inside the container.
- `COPY . /app`: Copies all files from the host's current directory to `/app` in the container.
- `RUN pip install -r requirements.txt`: Installs all required dependencies inside the container.
- `CMD ["python", "app.py"]`: Defines the command to execute when the container starts.

Building and running the image

Once the Dockerfile is defined, you can build and run the image using the following commands:

Step 1: Build the image

```
docker build -t my-flask-app .
```

POWERED BY

The above command:

- Uses the current directory (`.`) as the build context.
- Reads the `Dockerfile` and executes its instructions.
- Tags (`-t`) the resulting image as `my-flask-app`.

Step 2: Run the image as a container

```
docker run -d -p 5000:5000 my-flask-app
```

POWERED BY

The above command:

- Runs the container in detached mode (`-d`).
- Maps port 5000 inside the container to port 5000 on the host (`-p 5000:5000`).

Once running, you can access the Flask application by navigating to `http://localhost:5000` in a browser.

Docker Volumes and Persistence

By default, data inside a Docker container is **temporary**—once the container stops or is removed, the data disappears. To persist data across container restarts and share it between multiple containers, Docker provides volumes, a built-in mechanism for managing persistent storage efficiently.

Unlike storing data inside the container's filesystem, volumes are managed separately by Docker, making them more efficient, flexible, and easier to back up.

creating and using Docker volumes to ensure data persistence in your containers.

Creating and using Docker volumes

Step 1: Create a volume

Before using a volume, we need to create one. Run the following command:

```
docker volume create my-volume
```

POWERED BY

This creates a named volume called `my-volume`, which Docker will manage separately from any specific container.**Step 2: Use the volume in a container**

Now, let's start a container and mount the volume inside it:

```
docker run -d -v my-volume:/app/data my-app
```

POWERED BY

In the above command:

- `-v my-volume:/app/data` mounts the `my-volume` storage to the `/app/data` directory inside the container.
- Any data stored in `/app/data` will persist even if the container stops or is removed.

Docker Compose for Multi-Container Applications

So far, we've been working with single-container applications, but many real-world applications require multiple containers to work together. For example, a web application might need a backend server, a database, and a caching layer—each running in its own container. Managing these containers manually with separate `docker run` commands can quickly become tedious.

That's where Docker Compose comes in.

Docker Compose is a tool that simplifies the management of multi-container applications. Instead of running multiple `docker run` commands, you can define an entire application stack using a `docker-compose.yml` file and deploy it with a single command.

Writing a Docker Compose file

Now, let's create a real-world example—a simple Node.js application that connects to a **MongoDB database**. Instead of managing the two containers separately, we'll define them in a `docker-compose.yml` file.

Here's how we define our multi-container setup in Docker Compose:

```
version: '3'
services:
  web:
    build: .
    ports:
      - "3000:3000"
    depends_on:
      - database
  database:
    image: mongo
    volumes:
      - db-data:/data/db
volumes:
  db-data:
```

POWERED BY

Breaking down the file above:

- `version: '3'`: Specifies the Docker Compose version.
- `services:`: Defines individual services (containers).
- `web:`: Defines the Node.js web application.
- `database:`: Defines the MongoDB database container.
- `volumes:`: Creates a named volume (`db-data`) for MongoDB data persistence.

Running multi-container applications

Once the `docker-compose.yml` file is ready, we can launch the entire application stack with a single command:

```
docker-compose up -d
```

POWERED BY

The previous command starts both the web and database containers in detached mode (`-d`).

To stop all services, use:

```
docker-compose down
```

POWERED BY

This stops and removes all containers while preserving volumes and network settings.

RESULT:

This experiment demonstrated how to install Docker, download a container image, and run a simple container to validate the installation and execution process.

Ex. No: 10	**RUN A CONTAINER FROM DOCKER HUB**
Date:	

AIM:

To pull and run a Docker container image from Docker Hub, demonstrating basic image deployment using Docker CLI.

SOFTWARE/ TOOLS USED:

- Docker Desktop for Windows
- PowerShell / Command Prompt
- Docker Engine
- Internet connection to access Docker Hub

APPLICATIONS:

- Facilitates access to pre-built application environments.
- Simplifies deployment of web servers, databases, and development environments.
- Enables consistent software behavior across platforms.

Useful in DevOps pipelines for testing and deployment automation

PROCEDURE:

Step 1: Launch Docker Desktop

- Ensure Docker Desktop is running and the Docker Engine is active.
- Confirm Docker is installed by running:

```
docker –version
```

Step 2: Pull a Container Image from Docker Hub

- Use the docker pull command to download an official image from Docker Hub. For example:

```
docker pull nginx
```

This pulls the latest version of the Nginx web server image.

Step 3: Verify Downloaded Image

- List all available Docker images:

```
docker images
```

Step 4: Run the Container

- Create and start a container from the pulled image:

```
docker run --name mynginx -d -p 8080:80 nginx
```

  - --name mynginx: Names the container.
  - -d: Runs the container in detached mode.
  - -p 8080:80: Maps port 8080 on the host to port 80 in the container.

Step 5: Verify Container Execution

- View running containers:

```
docker ps
```

Step 6: Access the Web Application

- Open a web browser and navigate to:

  http://localhost:8080

  - The default Nginx welcome page will be displayed if the container is running correctly.

Step 7: Stop and Remove the Container (Optional)

- To stop the container:

```
docker stop mynginx
```

- To remove the container:

```
docker rm mynginx
```

RESULT:

The Docker image nginx was successfully pulled from Docker Hub. A container was created and executed from the image. The default Nginx web server page was accessible at http://localhost:8080, confirming successful container deployment and operation.