ARUNAI ENGINEERING COLLEGE

(Affiliated to Anna University)
Velu Nagar,Thiruvannamalai-606 603
www.arunai.org

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

BACHELOR OF ENGINEERING

2023 - 2024

FIFTH SEMESTER

CCS335-CLOUD COMPUTING LABORATORY

ARUNAI ENGINEERING COLLEGE

TIRUVANNAMALAI – 606 603

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

Certified that this is a bonafide record of work done by

Name :

University Reg.No :

Semester :

Branch :

Year :

Staff-in-Charge **Head of the Department**

Submitted for the _____

Practical Examination held on _____

Internal Examiner **External Examiner**

TABLE OF CONTENTS

EXP. NO.	DATE	NAME OF THE EXPERIMENT	PAGE NO.	SIGNATURE
1		Install Virtual box/VMware/ Equivalent open source cloud Workstation with different flavours of Linux or Windows OS on top of windows 8 and above		
2		Install a C compiler in the virtual machine created using a virtual box and execute Simple Programs		
3		Install Google App Engine. Create a hello world app and other simple web applications using python/java		
4		Use the GAE launcher to launch the web applications		
5		Simulate a cloud scenario using CloudSim and run a scheduling algorithm that is not present in CloudSim		
6		Find a procedure to transfer the files from one virtual machine to another virtual machine		
7		Install Hadoop single node cluster and run simple applications like word count		
8		Creating and Executing Your First Container Using Docker		
9		Run a Container from Docker Hub		

EX NO. : 1

DATE:

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

Aim:

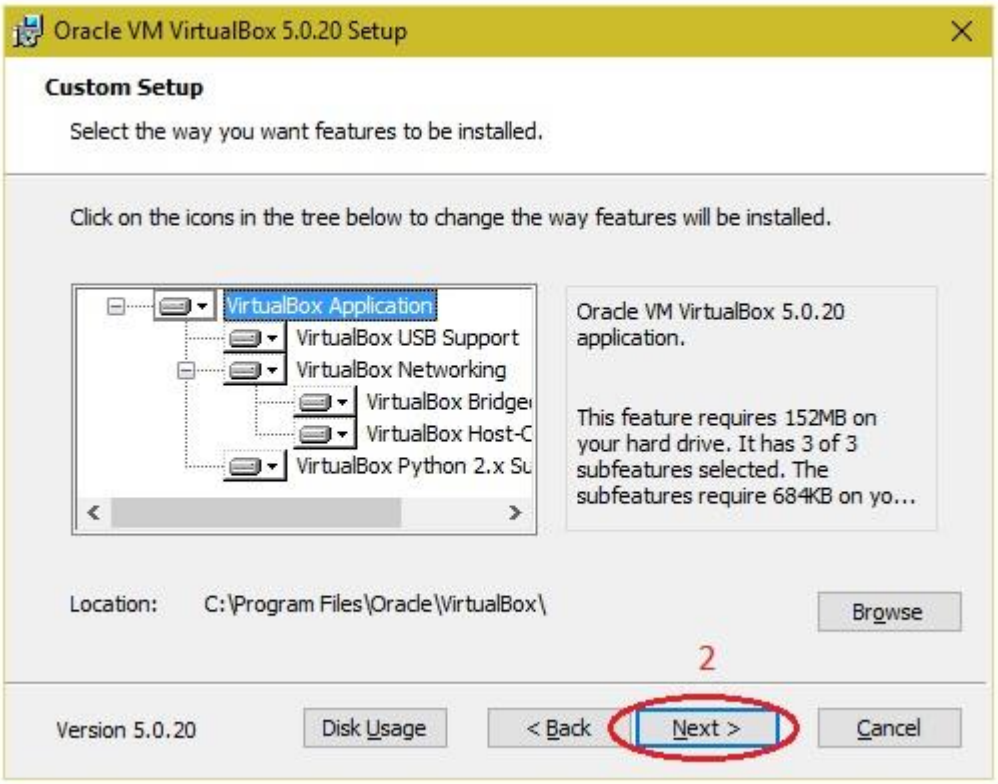
To Install Virtualbox / VMware Workstation with different flavours of linux or windows OS on top of windows7 or 8.

PROCEDURE:

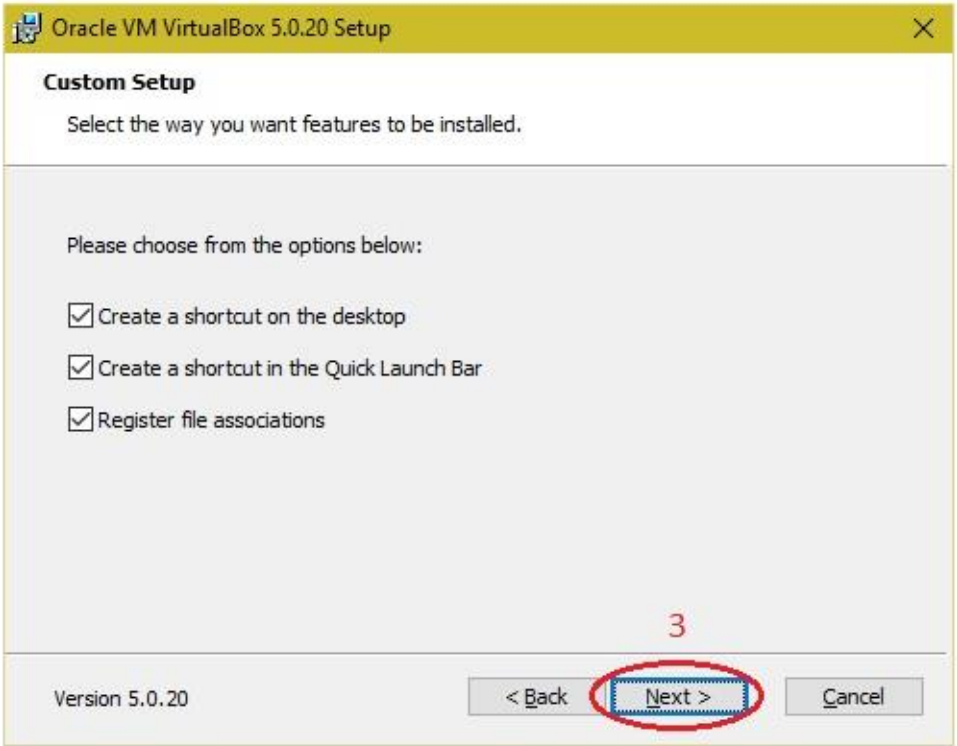
Steps to install Virtual Box:

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

2. Click the next button..

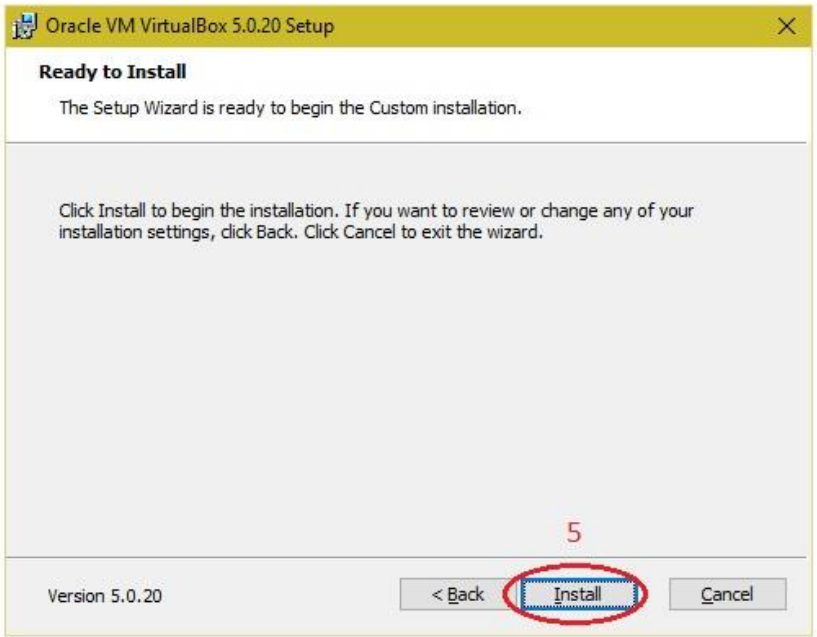


3. Click the next button



4. Click the YES button..

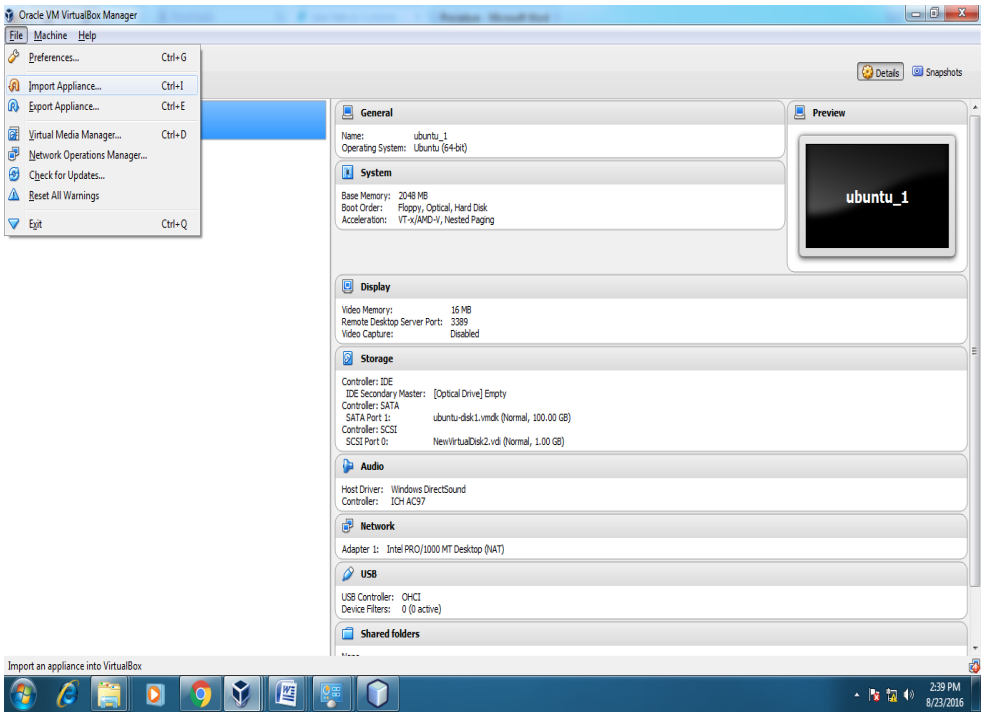
5. **Click the install button…**

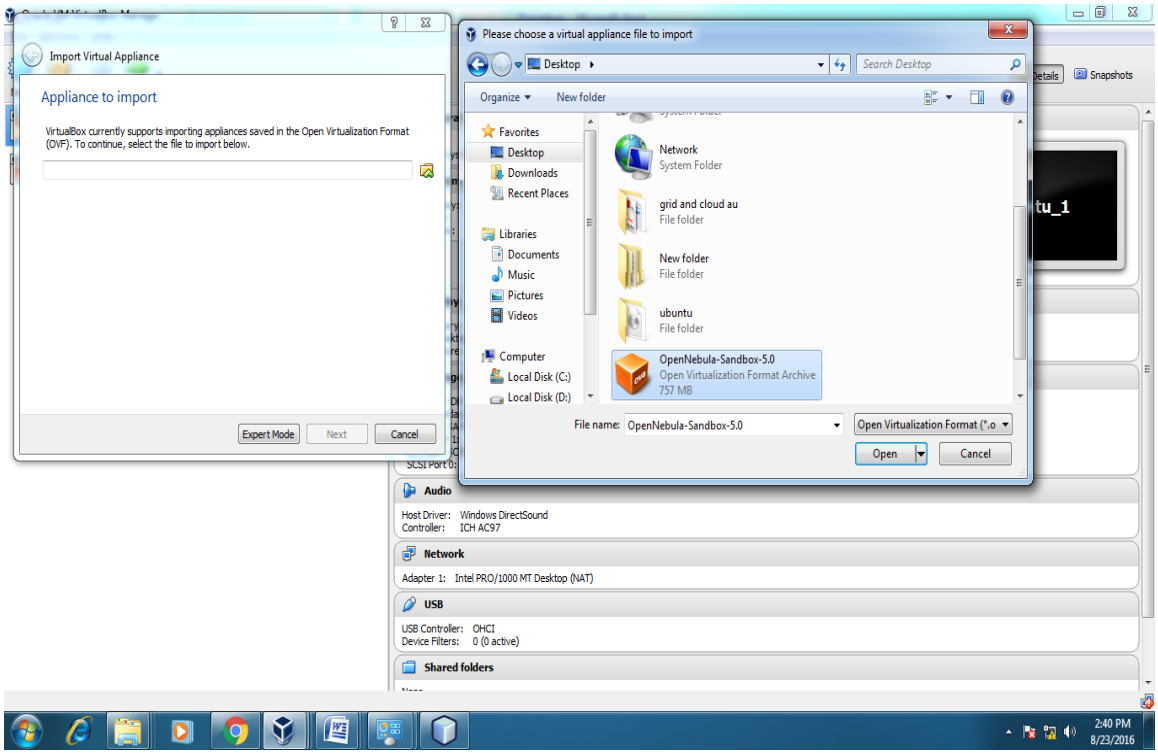


6. Then installation was completed..the show virtual box icon on desktop screen….

Steps to import Open nebula sandbox:

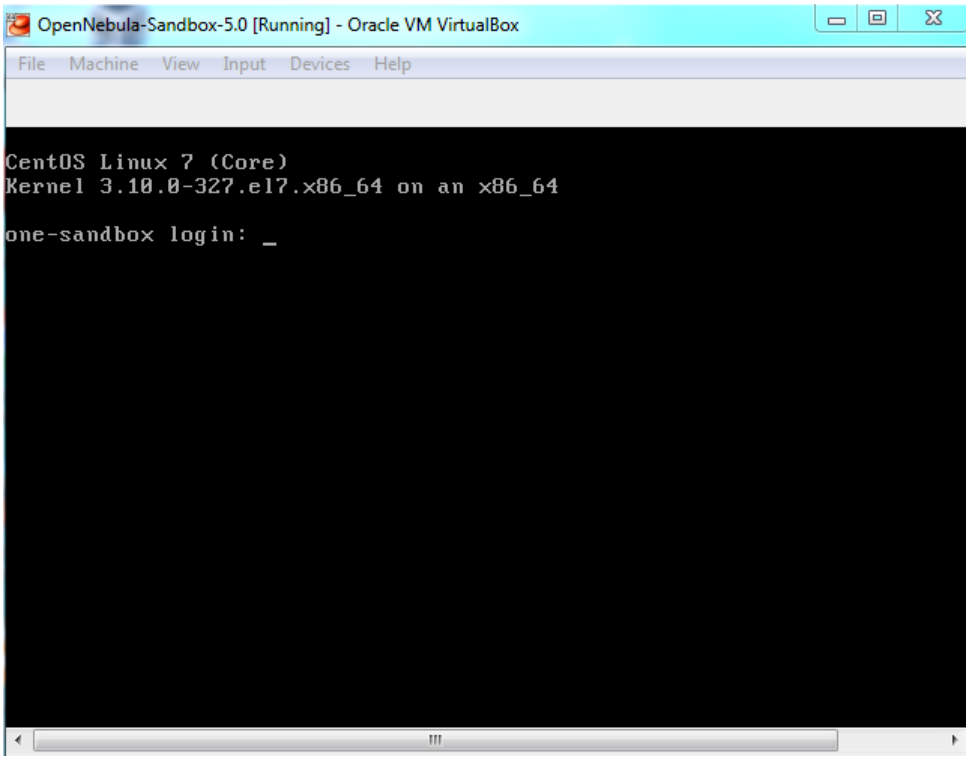
1. Open Virtual box
2. File → import Appliance
3. Browse OpenNebula-Sandbox-5.0.ova file
4. Then go to setting, select Usb and choose USB 1.1
5. Then Start the Open Nebula
6. Login using username: root, password:opennebula

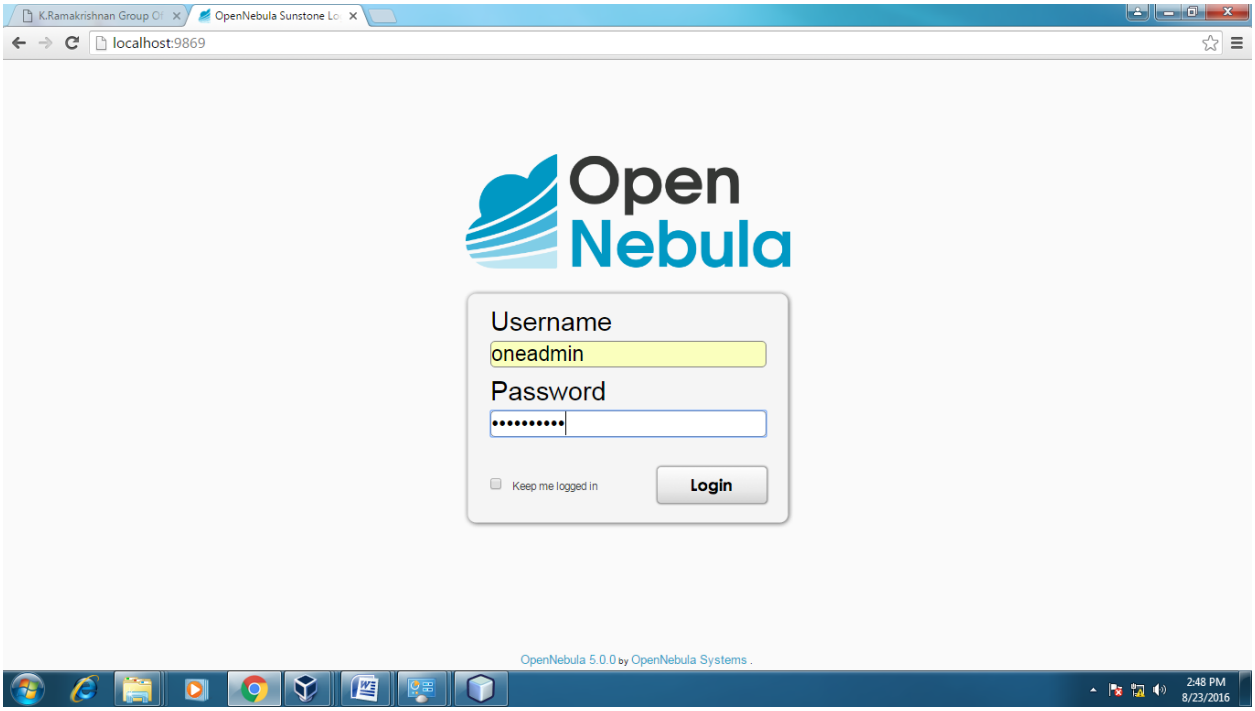


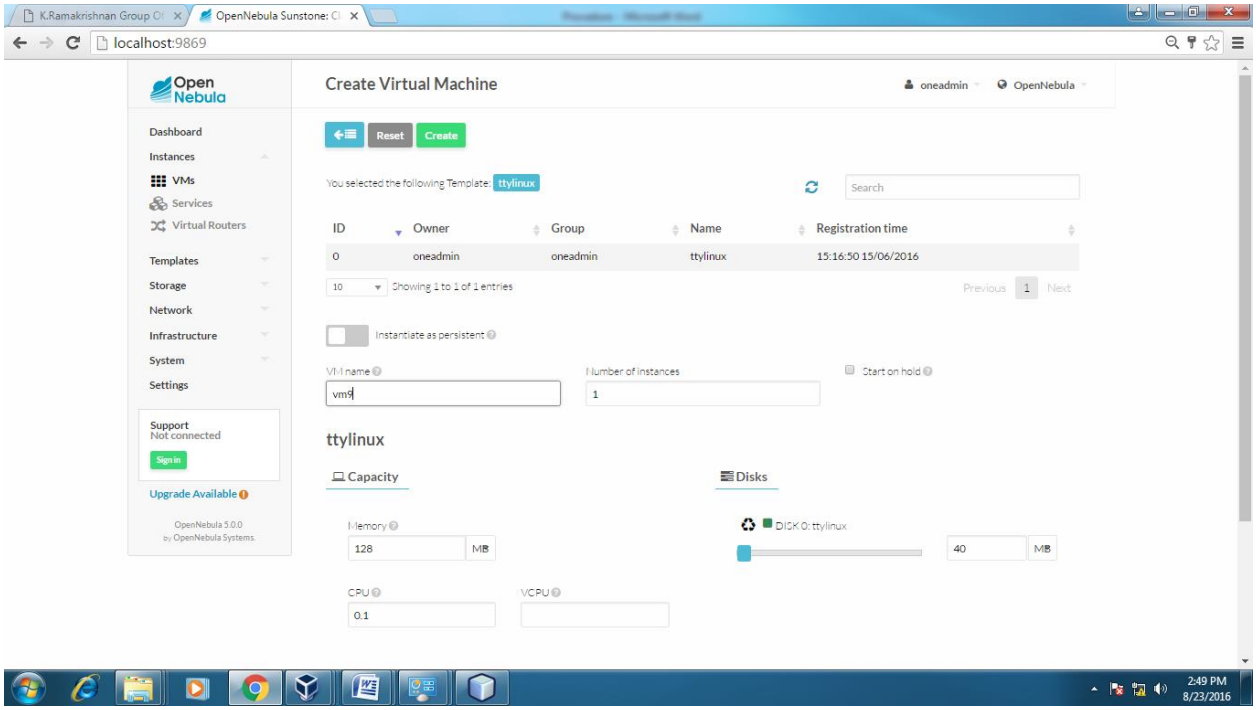


Steps to create Virtual Machine through opennebula

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


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


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



APPLICATIONS:

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

RESULT:

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

EX.NO.:2
DATE:

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

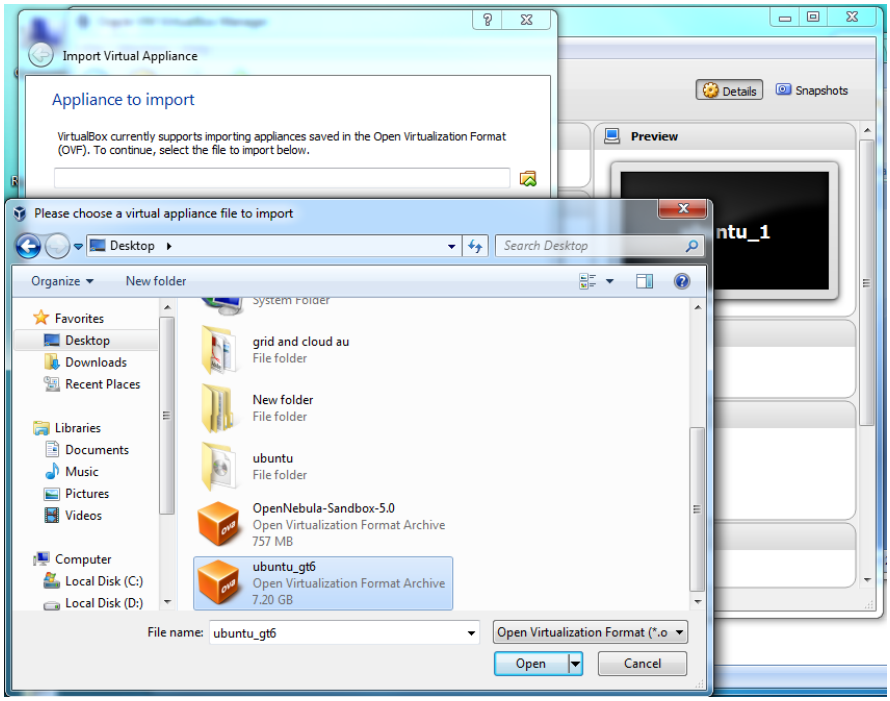
Aim:

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

PROCEDURE:

Steps to import .ova file:

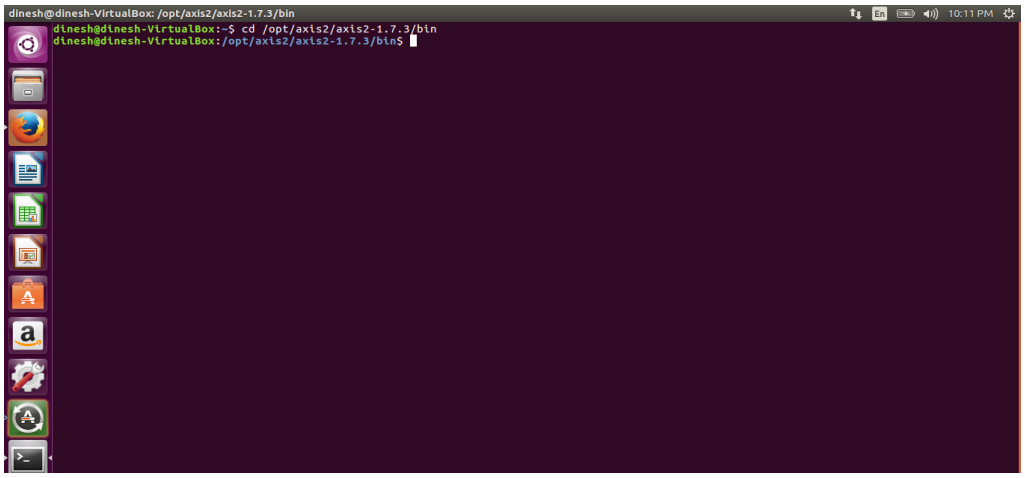
1. Open Virtual box
2. File →import Appliance
3. Browse ubuntu_gt6.ova file
4. Then go to setting, select Usb and choose USB 1.1
5. Then Start the ubuntu_gt6
6. Login using username: dinesh, password:99425.



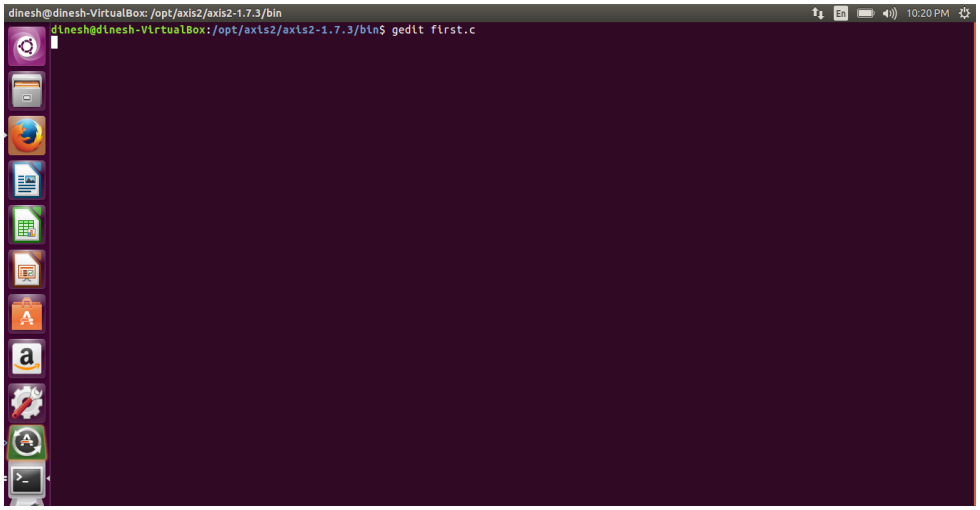
Steps to run c program:

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

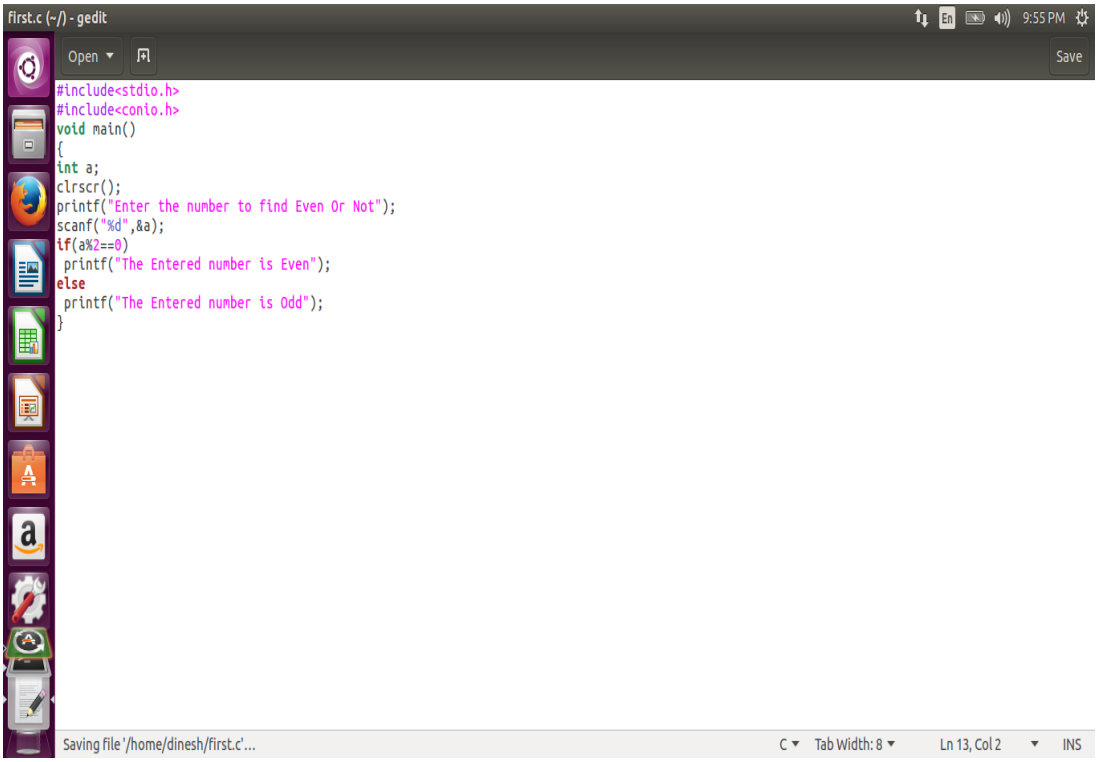
1. **Type cd /opt/axis2/axis2-1.7.3/bin then press enter**



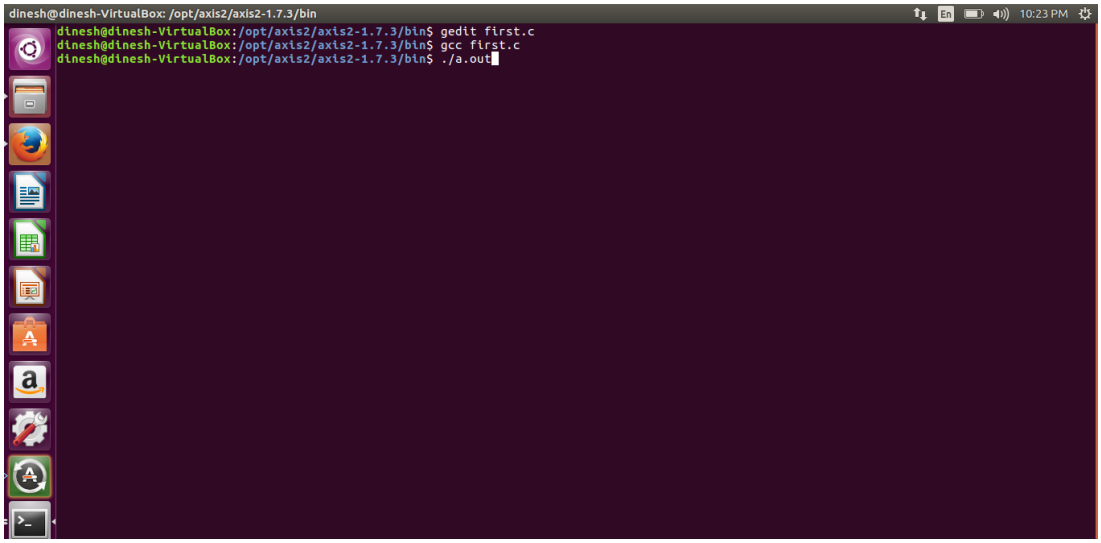
2. **Type gedit first.c**



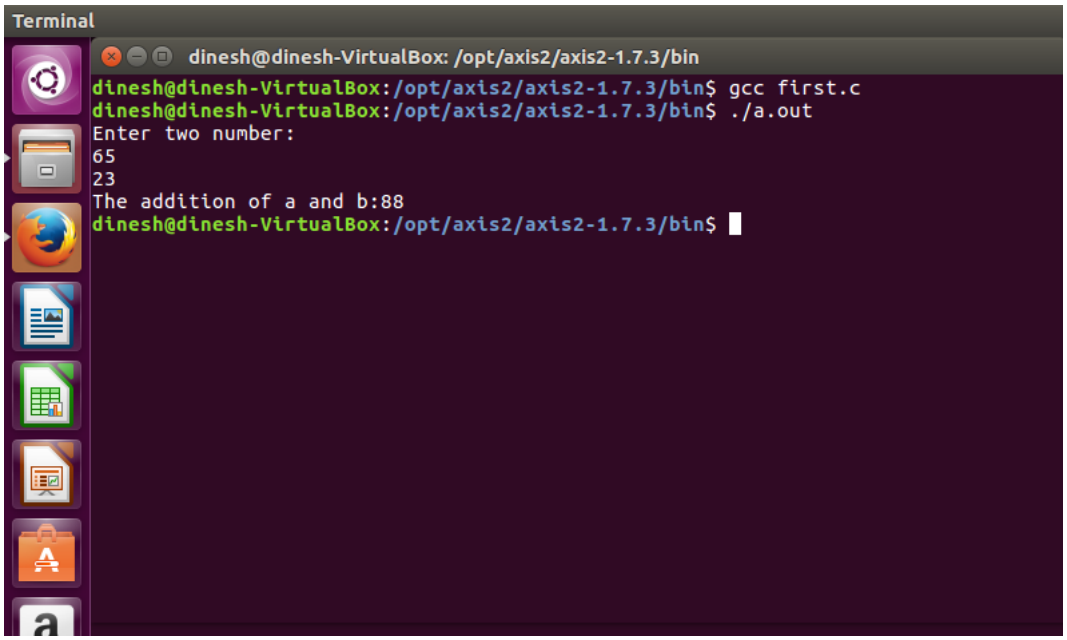
3. Type the c program



4. Running the C program



5. **Display the output:**



APPLICATIONS:

Simply running all programs in grid environment.

RESULT:

Thus the simple C programs executed successfully.

EX NO.:3
DATE:

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

Aim:

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

Procedure:

1. Install Google Plugin for Eclipse

Read this guide – how to install Google Plugin for Eclipse. If you install 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

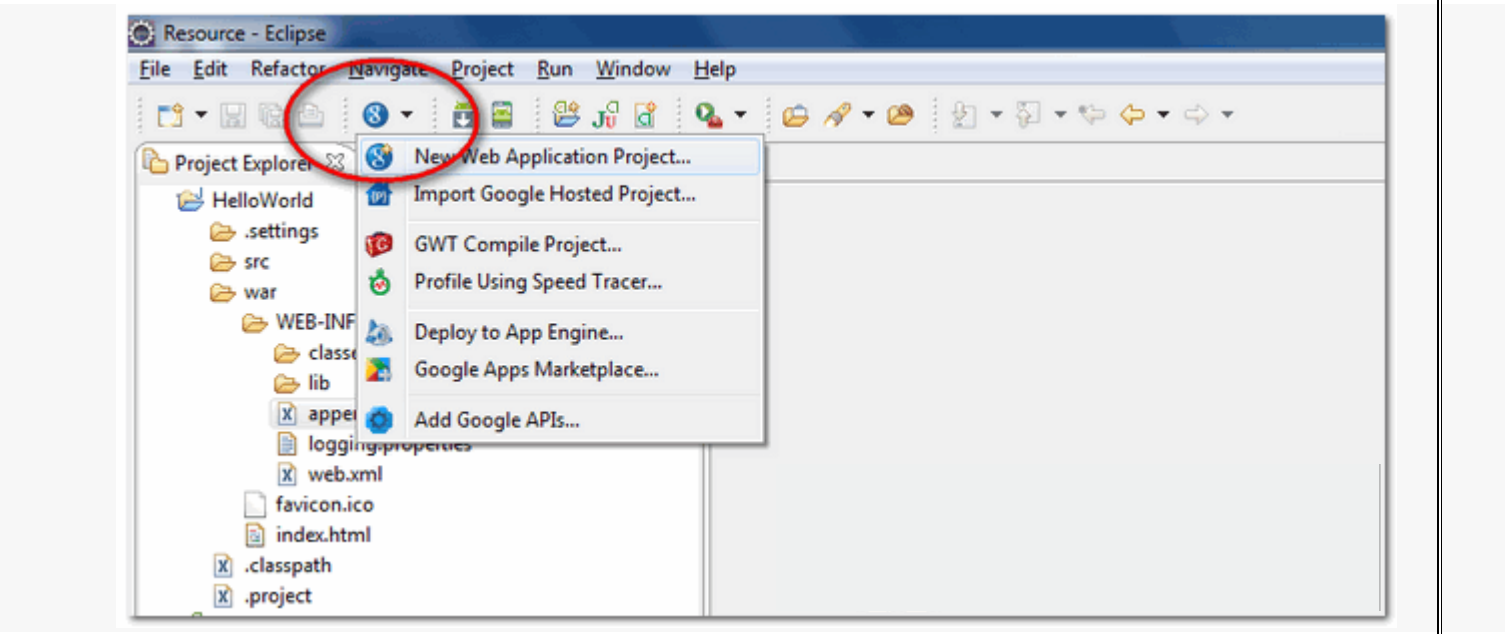
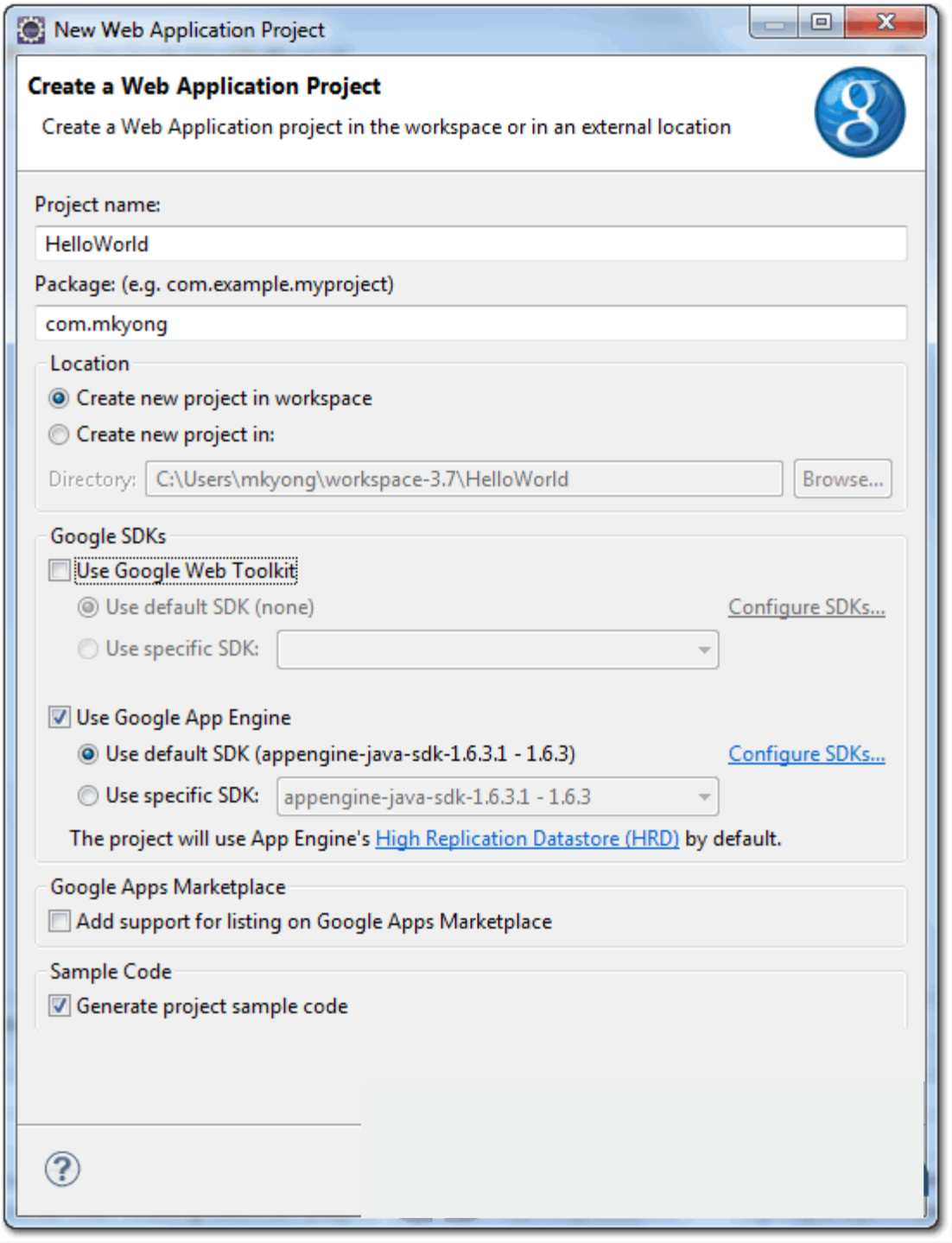


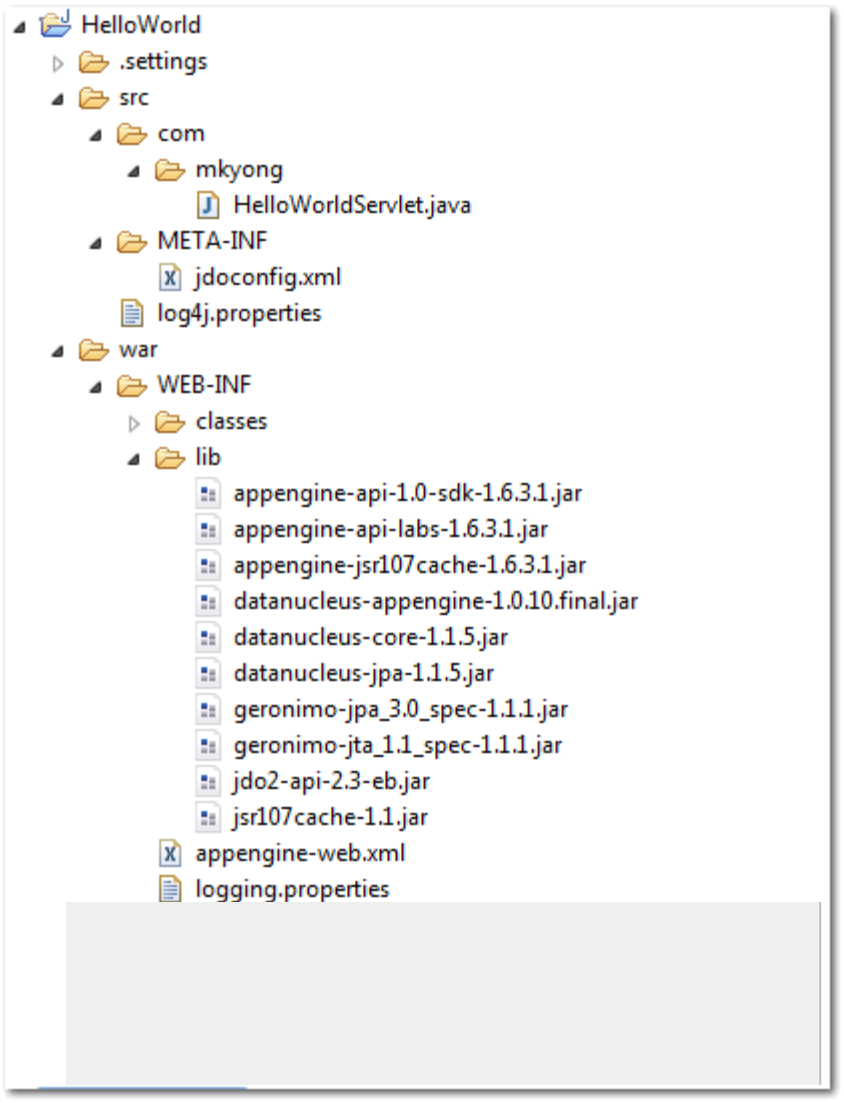
*Figure – Deselect the "**Google Web ToolKit**", and link your GAE Java SDK via the "**configure SDK**" link.*



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

3. Hello World

Review the generated project directory.



Nothing special, a standard Java web project structure.

```
HelloWorld/
  src/
    ...Java source code...
    META-INF/
      ...other configuration...
  war/
    ...JSPs, images, data files...
    WEB-INF/
      ...app configuration...
      lib/
        ...JARs for libraries...
      classes/
        ...compiled classes...
```

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

File : appengine-web.xml

```
<?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>
```

Copy

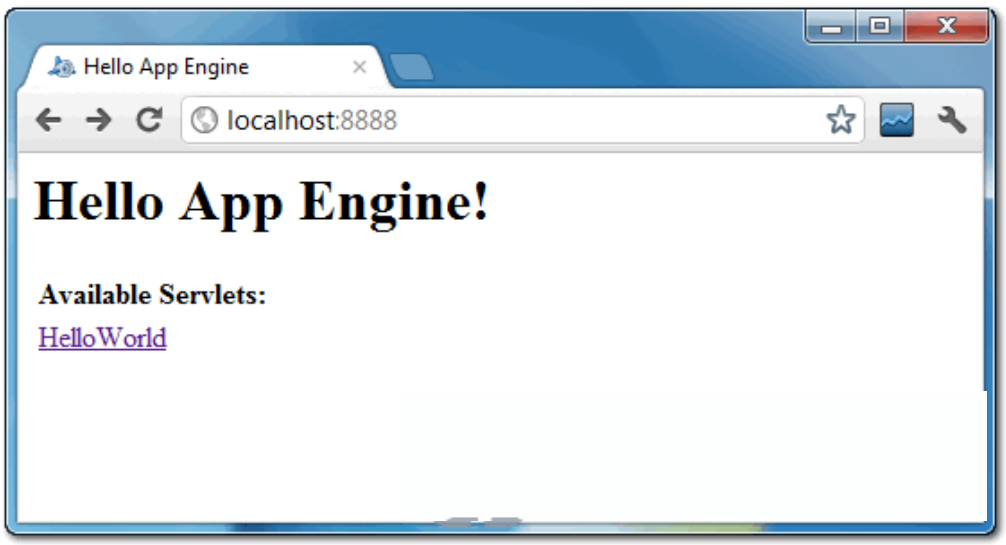
4. Run it local

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

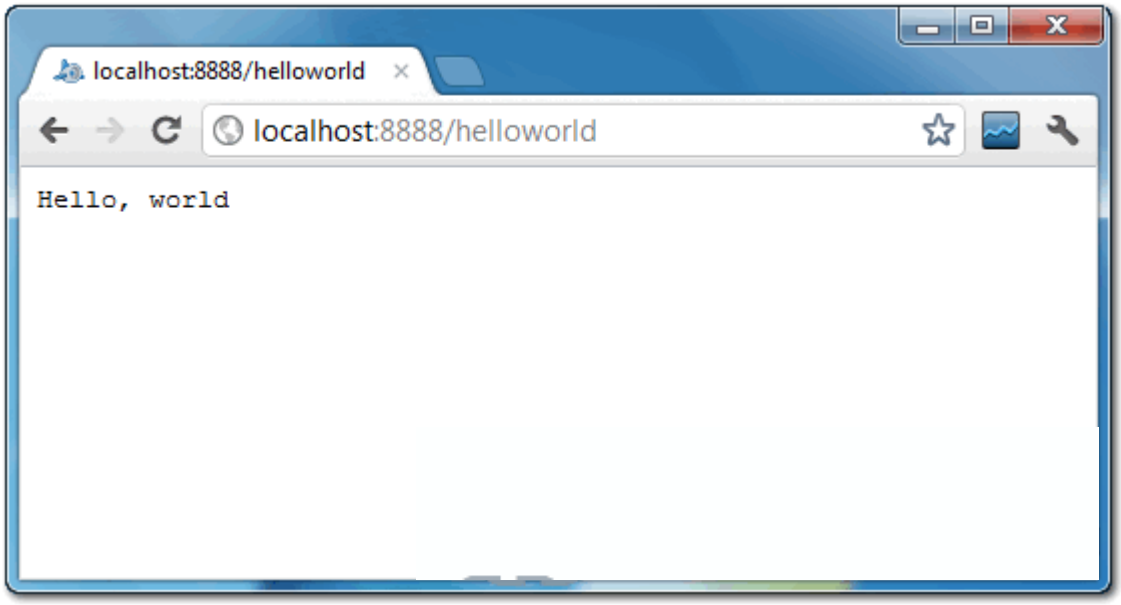
Eclipse console :

```
//...
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/, see output



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



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.

File : appengine-web.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>
```

Copy

To deploy, see following steps:

Figure 1.1 – Click on GAE deploy button on the toolbar.

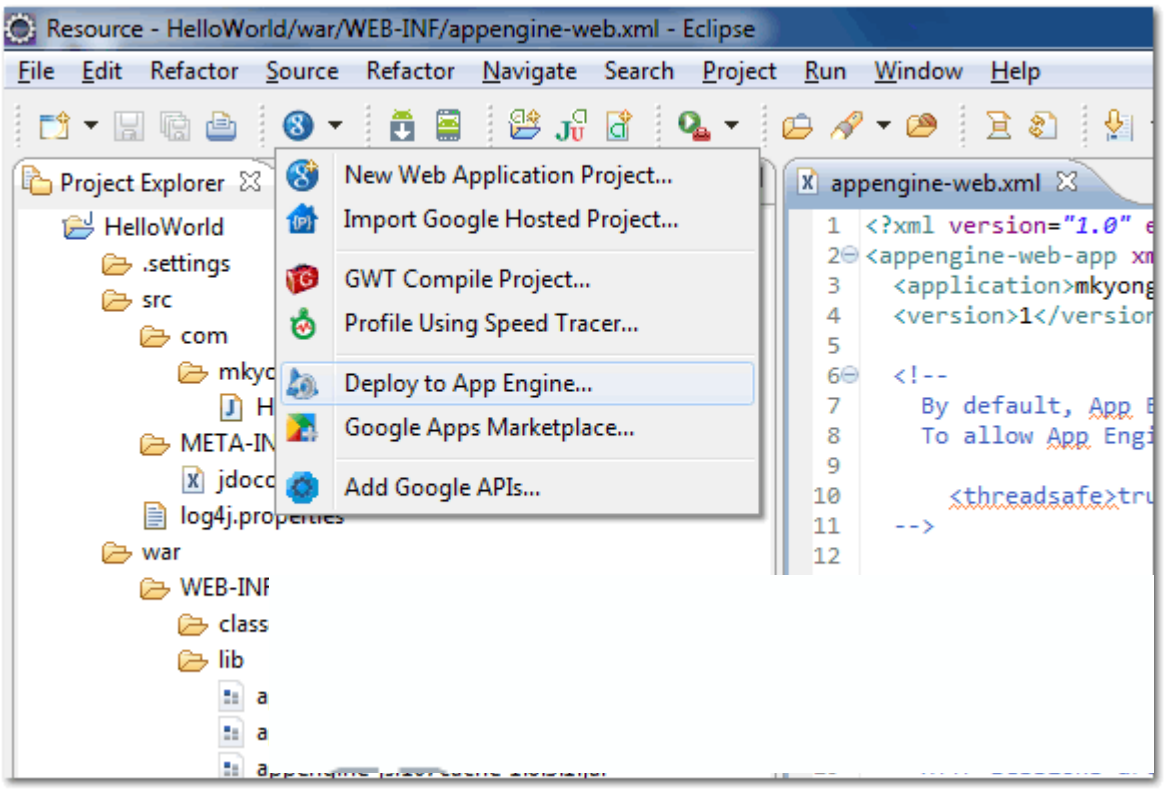


Figure 1.2 – Sign in with your Google account and click on the Deploy button.

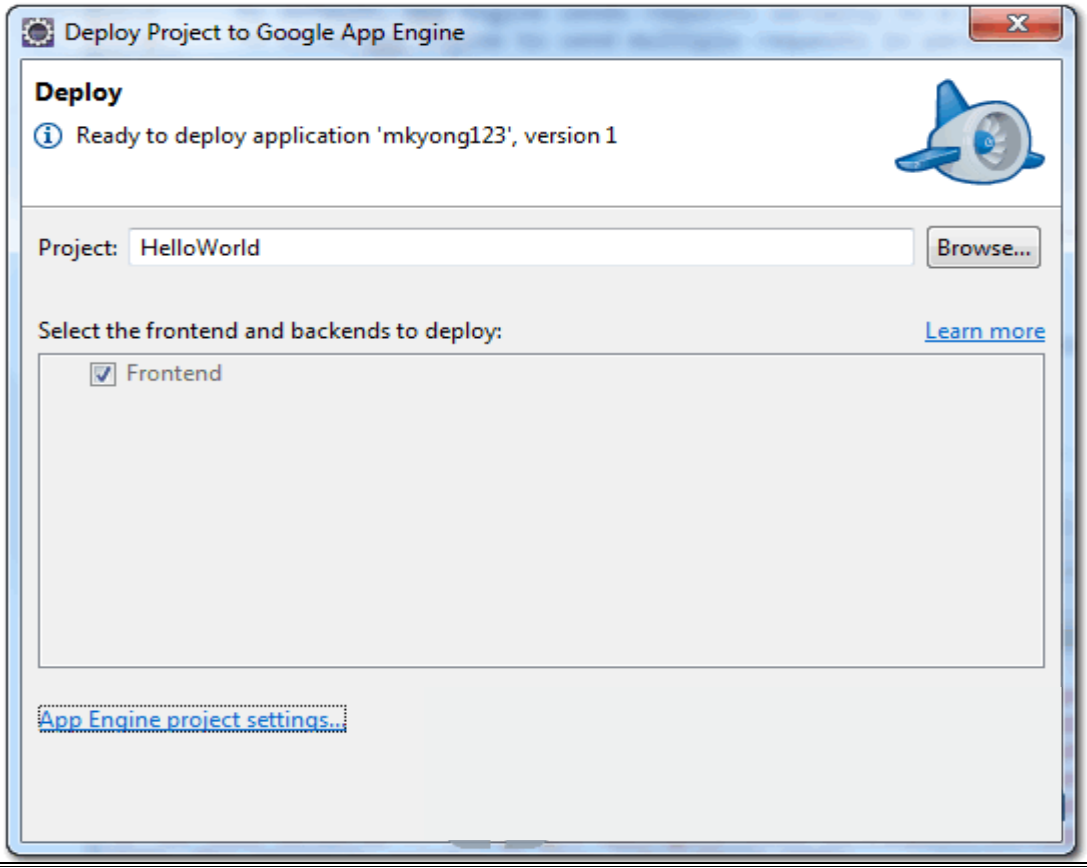
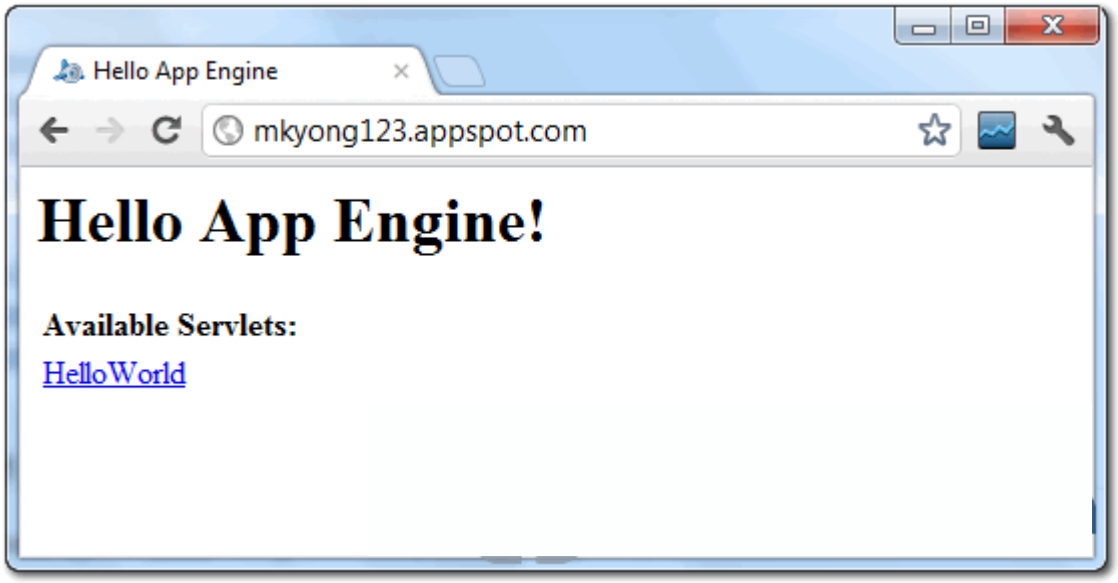


Figure 1.3 – If everything is fine, the hello world web application will be deployed to this URL – http://mkyong123.appspot.com/



Result:

Thus the simple application was created successfully.

EX.NO.:4
DATE:

Use GAE launcher to launch the web applications.

Aim:

To Use GAE launcher to launch the web applications.

Steps:

Making your First Application

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

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

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

And then make a sub-folder in within apps called "ae-01-trivial" — the path to this folder would be:

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

Using a text editor such as JEdit (www.jedit.org), create a file called app.yaml in the ae-01-trivial folder with the following contents:

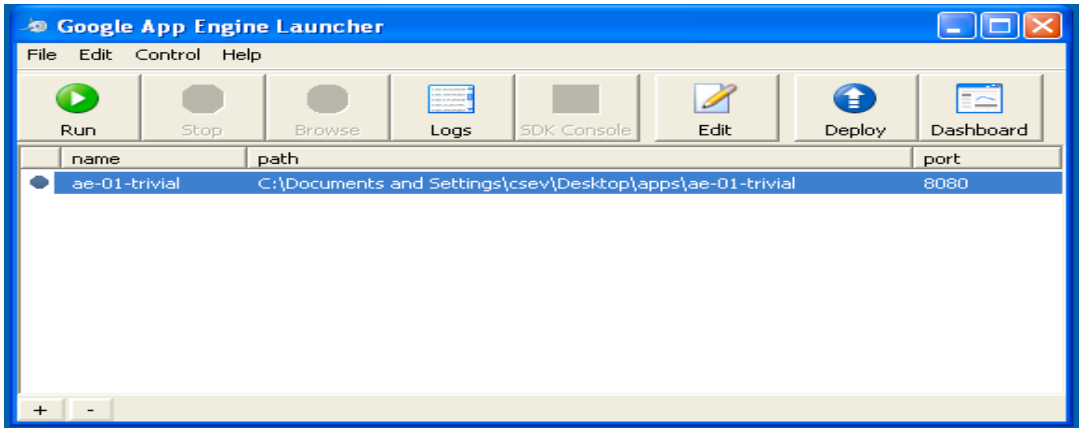
```
application: ae-01-trivial
    version: 1
runtime: python api_version: 1
handlers:- url: /.*
script: index.py
```

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

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

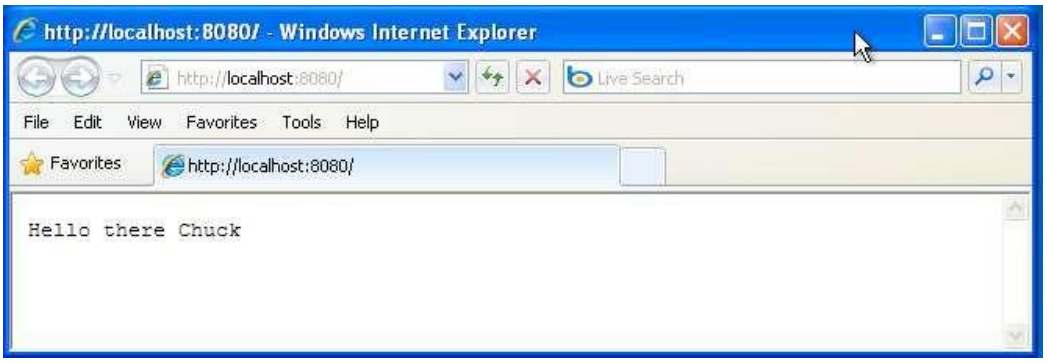
```
print 'Content-Type: text/plain'
  print ' '
print 'Hello there Chuck'
```

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



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

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

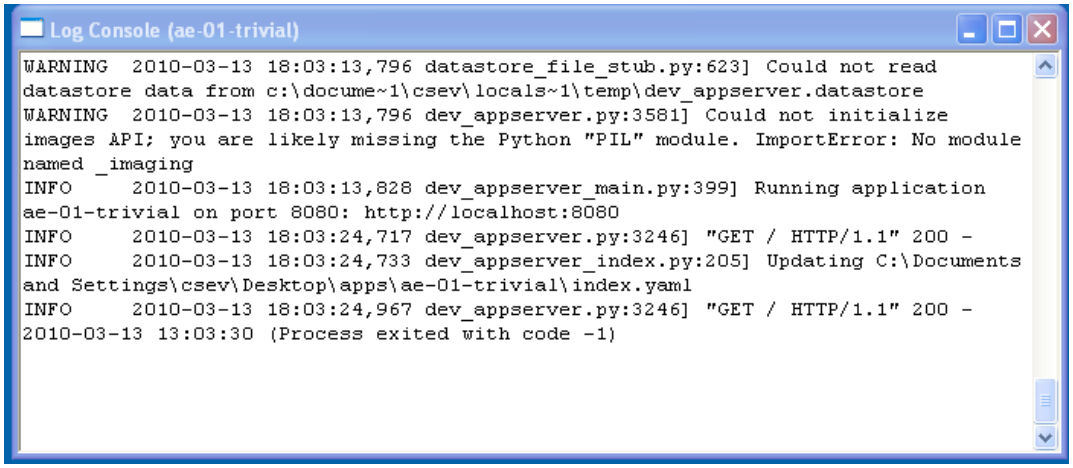


Just for fun, edit the index.py to change the name "Chuck" to your own name and press Refresh in the browser to verify your updates.

Watching the Log

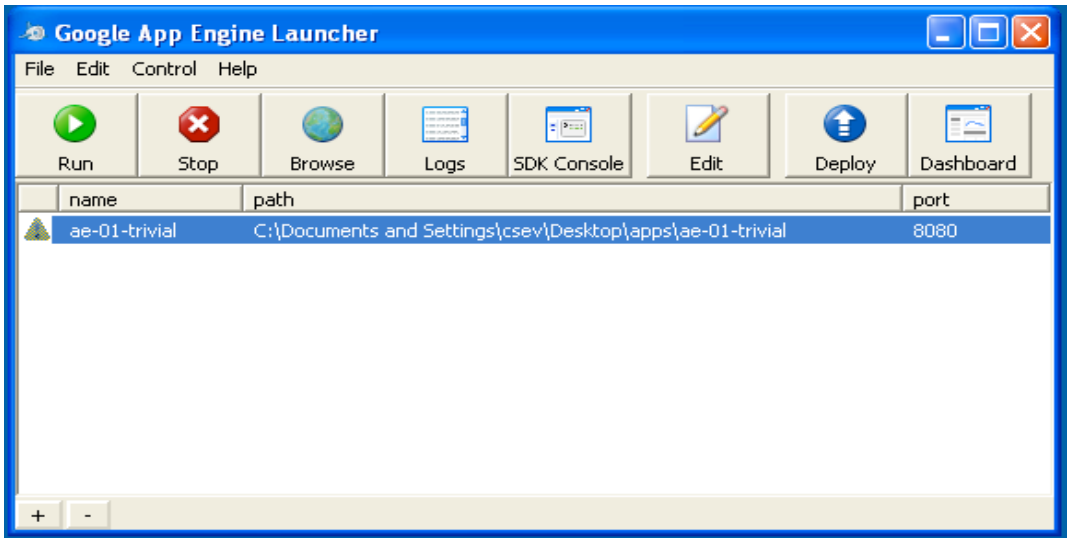
You can watch the internal log of the actions that the web server is performing when you are interacting with your application in the browser. Select your application in the Launcher and press the Logs button to bring up a log window:

Each time you press Refresh in your browser – you can see it retrieving the output with a GET request.

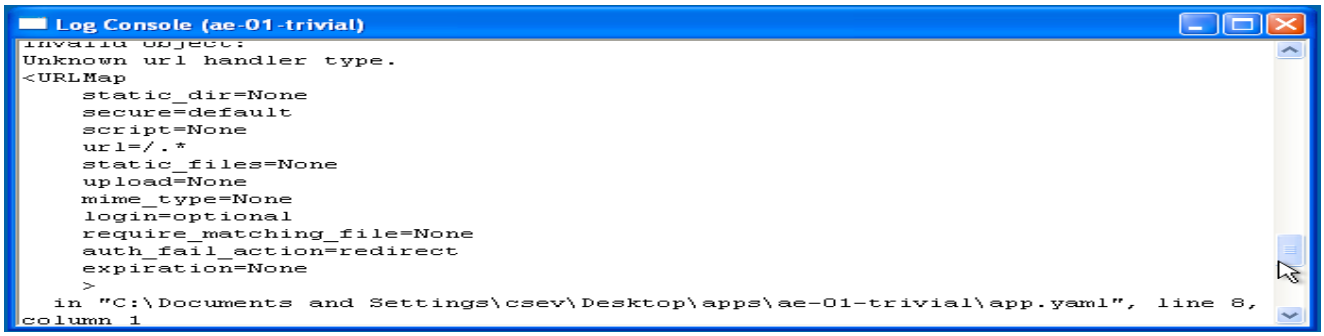


Dealing With Errors

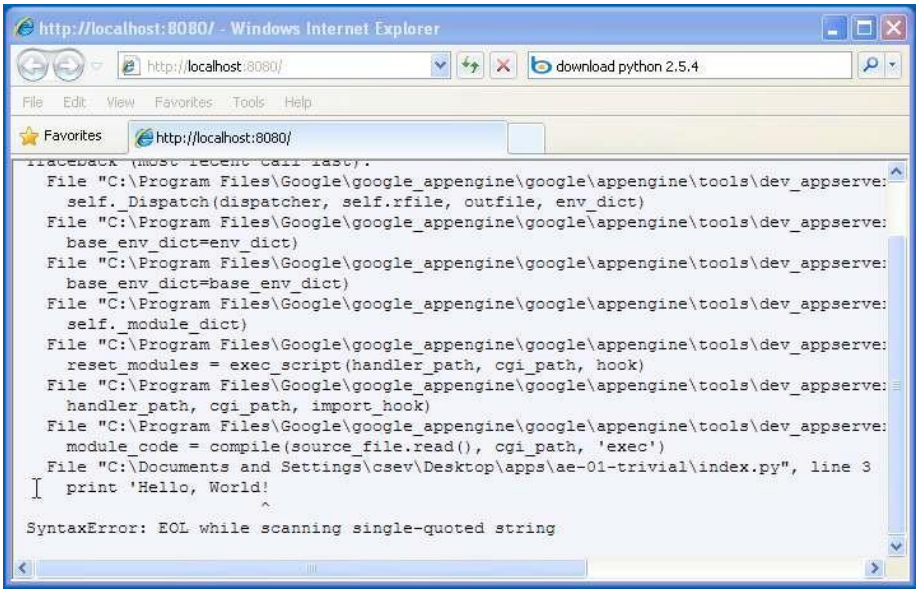
With two files to edit, there are two general categories of errors that you may encounter. If you make a mistake on the app.yaml file, the App Engine will not start and your launcher will show a yellow icon near your application:



To get more detail on what is going wrong, take a look at the log for the application:



In this instance – the mistake is mis-indenting the last line in the app.yaml (line 8).
If you make a syntax error in the index.py file, a Python trace back error will appear in your browser.



The error you need to see is likely to be the last few lines of the output – in this case I made a Python syntax error on line one of our one-line application.
Reference: http://en.wikipedia.org/wiki/Stack_trace

When you make a mistake in the app.yaml file – you must the fix the mistake and attempt to start the application again.

If you make a mistake in a file like index.py, you can simply fix the file and press refresh in your browser – there is no need to restart the server.

Shutting Down the Server

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

Result:

Thus the GAE web applications was created.

EX. NO.:5
DATE:

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

Aim:

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

Steps:

How to use CloudSim in Eclipse

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

To use CloudSim in Eclipse:

1. Download CloudSim installable files from *https://code.google.com/p/cloudsim/downloads/list and unzip*
2. Open Eclipse
3. Create a new Java Project: File -> New
4. Import an unpacked CloudSim project into the new Java Project

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

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

5. Data centres are the resource providers in CloudSim; hence, creation of data centres is a second step. To create Datacenter, you need the DatacenterCharacteristics object that stores the properties of a data centre such as architecture, OS, list of machines, allocation policy that covers the time or spaceshared, the time zone and its price:

```
Datacenter datacenter9883 = new Datacenter(name, characteristics, new
VmAllocationPolicySimple(hostList), s
```

6. The third step is to create a broker:

```
DatacenterBroker broker = createBroker();
```

7. The fourth step is to create one virtual machine unique ID of the VM, userId ID of the VM's owner, mips, number Of Pes amount of CPUs, amount of RAM, amount of bandwidth, amount of storage, virtual machine monitor, and cloudletScheduler policy for cloudlets:

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

8. Submit the VM list to the broker:
broker.submitVmList(vmlist)

9. Create a cloudlet with length, file size, output size, and utilisation model:
Cloudlet cloudlet = new Cloudlet(id, length, pesNumber, fileSize, outputSize, utilizationModel, utilizationMode

10. Submit the cloudlet list to the broker:
broker.submitCloudletList(cloudletList)

Sample Output from the Existing Example:

Starting
CloudSimExample1...
Initialising...
Starting CloudSim
version 3.0 Datacenter_0
is starting...
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>null
Broker is
starting...
Entities started.
: Broker: Cloud Resource List received with 1
resource(s) 0.0: Broker: Trying to Create VM #0
in Datacenter_0
: Broker: VM #0 has been created in Datacenter #2, Host #0
0.1: Broker: Sending cloudlet 0 to VM #0
400.1: Broker: Cloudlet 0 received
: Broker: All Cloudlets executed.
Finishing... 400.1: Broker: Destroying
VM #0
Broker is shutting down...
Simulation: No more future
events
CloudInformationService: Notify all CloudSim entities for
shutting down. Datacenter_0 is shutting down...
Broker is shutting
down... Simulation
completed.
Simulation completed.

```
========== OUTPUT ==========
Cloudlet ID  STATUS  Data center ID   VM ID   Time   Start Time
    Finish Time 0         SUCCESS    2                 0              400
              0.1             400.1
*****Datacenter:
Datacenter_0***** User id
            Debt
   3        35.6

CloudSimExample1 finished!
```

RESULT:

The simulation was successfully executed.

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.

Steps:

1. You can copy few (or more) lines with *copy & paste* mechanism.
   For this you need to share clipboard between host OS and guest OS, installing Guest Addition on both the virtual machines (probably setting *bidirectional* and restarting them). You *copy* from *guest OS* in the clipboard that is shared with the *host OS*.
   Then you *paste* from the *host OS* to the second *guest OS*.
2. You can enable drag and drop too with the same method (Click on the machine, settings, general, advanced, drag and drop: set to *bidirectional*)
3. You can have common *Shared Folders* on both virtual machines and use one of the directory shared as buffer to copy.
   Installing Guest Additions you have the possibility to set Shared Folders too. As you put a file in a shared folder from *host OS* or from *guest OS*, is immediately visible to the other. (Keep in mind that can arise some problems for date/time of the files when there are different clock settings on the different virtual machines).
   If you use the same folder shared on more machines you can exchange files directly copying them in this folder.
4. You can use usual method to copy files between 2 different computer with client-server application. (e.g. scp with sshd active for linux, winscp... you can get some info about SSH servers e.g. here)
   You need an active server (sshd) on the receiving machine and a client on the sending machine. Of course you need to have the authorization setted (via password or, better, via an automatic authentication method).
   Note: many Linux/Ubuntu distribution install sshd by default: you can see if it is running with pgrep sshd from a shell. You can install with sudo apt-get install openssh-server.
5. You can mount part of the file system of a virtual machine via NFS or SSHFS on the other, or you can share file and directory with Samba.
   You may find interesting the article Sharing files between guest and host without VirtualBox shared folders with detailed step by step instructions.

You should remember that you are dialling with a little network of machines with different operative systems, and in particular:

- Each virtual machine has its own operative system running on and acts as a physical machine.
- Each virtual machine is an instance of a program *owned* by an *user* in the hosting operative system and should undergo the restrictions of the *user* in the *hosting OS*.
  E.g Let we say that Hastur and Meow are users of the hosting machine, but they did not allow each other to see their directories (no read/write/execute authorization). When each of them run a virtual machine, for the hosting OS those virtual machine are two normal programs owned by Hastur and Meow and cannot see the private directory of the other user. This is a restriction due to the *hosting OS*. It's easy to overcame it: it's enough to give authorization to read/write/execute to a directory or to chose a different directory in which both users can read/write/execute.
- Windows likes mouse and Linux fingers. :-)
  I mean I suggest you to enable *Drag & drop* to be cosy with the Windows machines and the *Shared folders* or to be cosy with Linux.
  When you will need to be fast with Linux you will feel the need of ssh-keygen and to Generate once SSH Keys to copy files on/from a remote machine without writing password anymore. In this way it functions bash auto-completion remotely too!

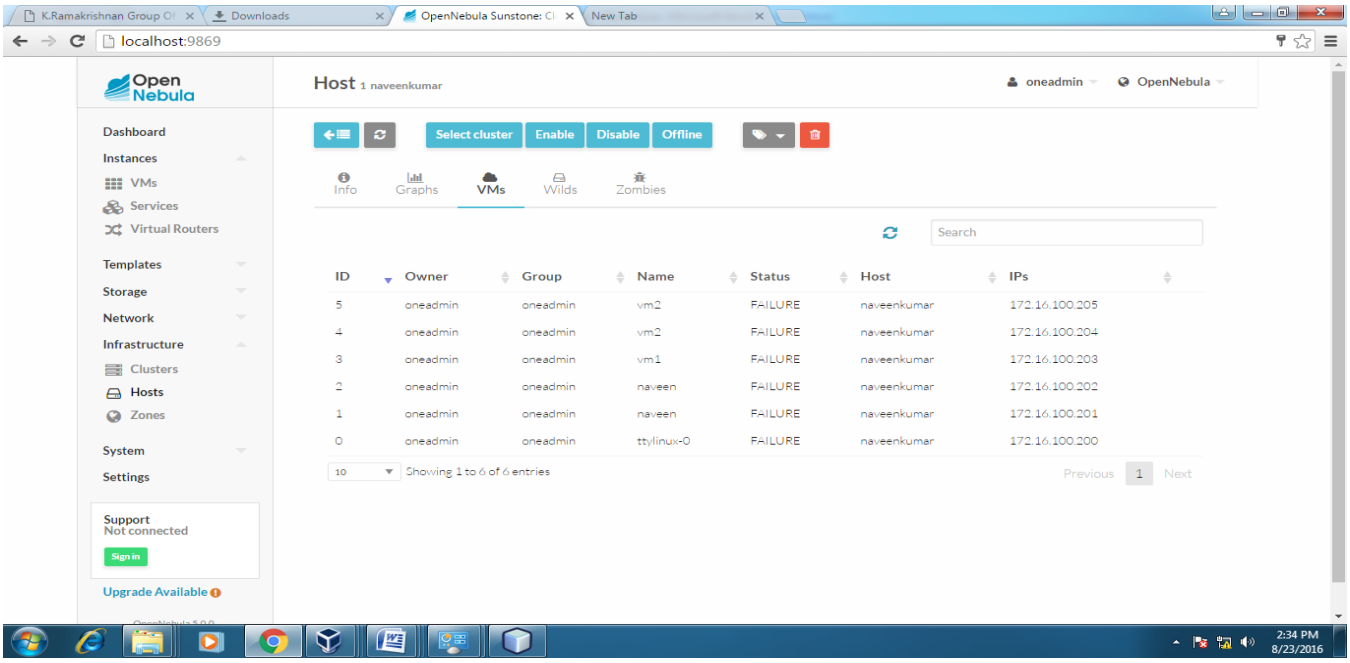
PROCEDURE:

Steps:

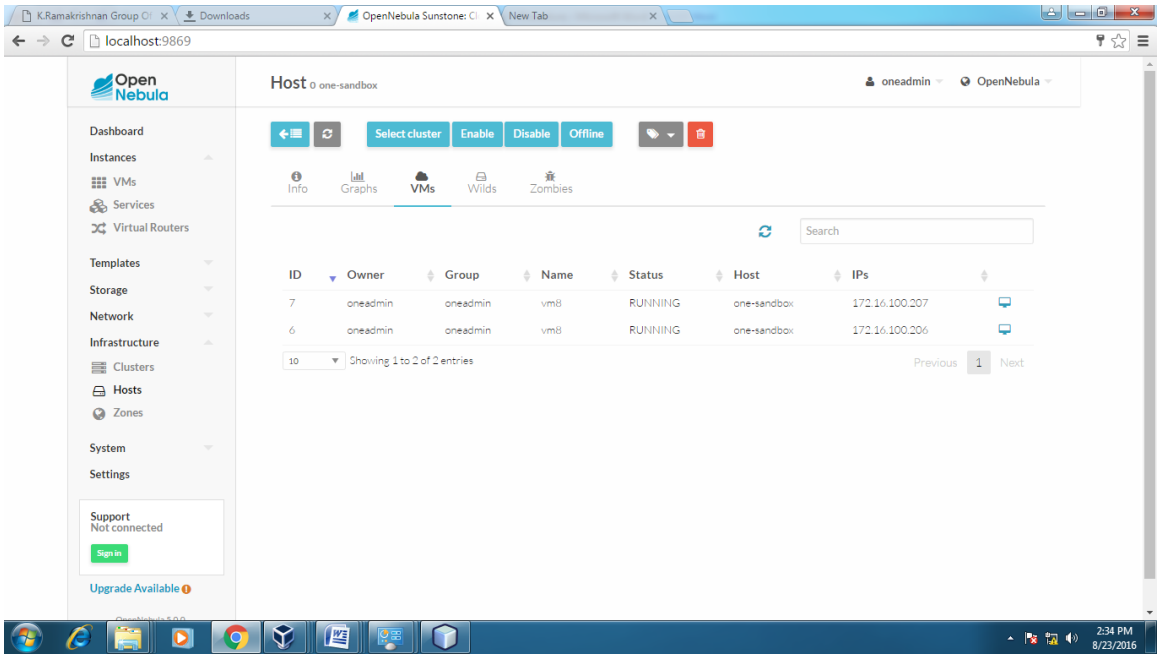
1. Open Browser, type localhost:9869
2. Login using username: oneadmin, password: opennebula
3. Then follow the steps to migrate VMs
   a. Click on infrastructure
   b. Select clusters and enter the cluster name
   c. Then select host tab, and select all host
   d. Then select Vnets tab, and select all vnet
   e. Then select datastores tab, and select all datastores
   f. And then choose host under infrastructure tab
   g. Click on + symbol to add new host, name the host then click on create.
4. on instances, select VMs to migrate then follow the stpes
   a. Click on 8th icon ,the drop down list display
   b. Select migrate on that ,the popup window display
   c. On that select the target host to migrate then click on migrate.

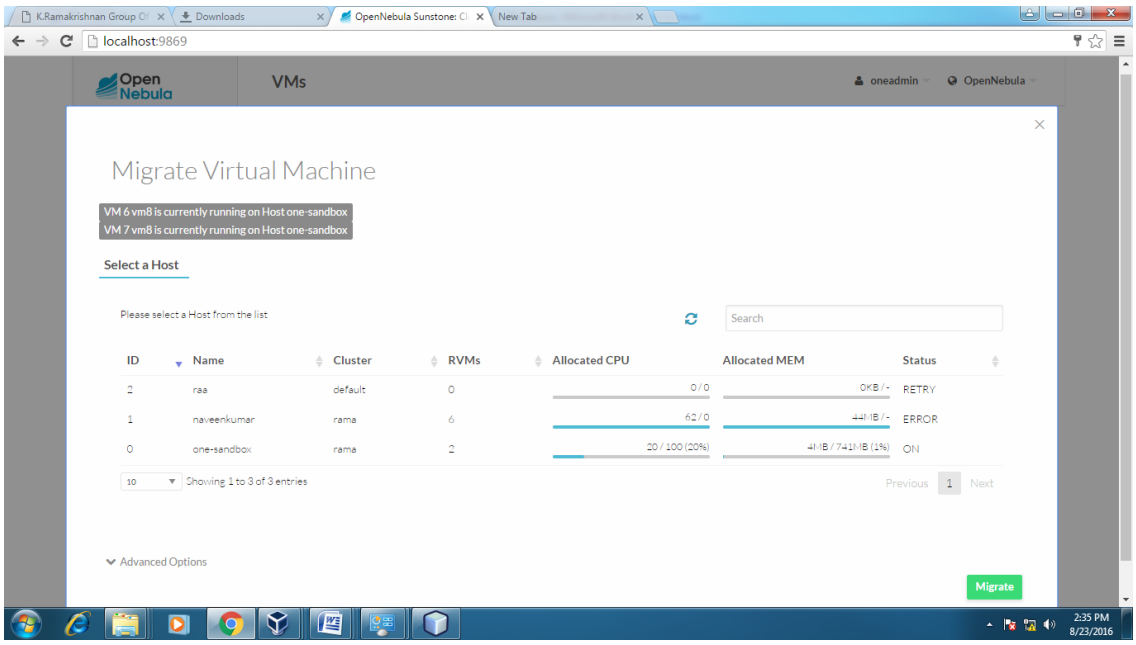
Before migration

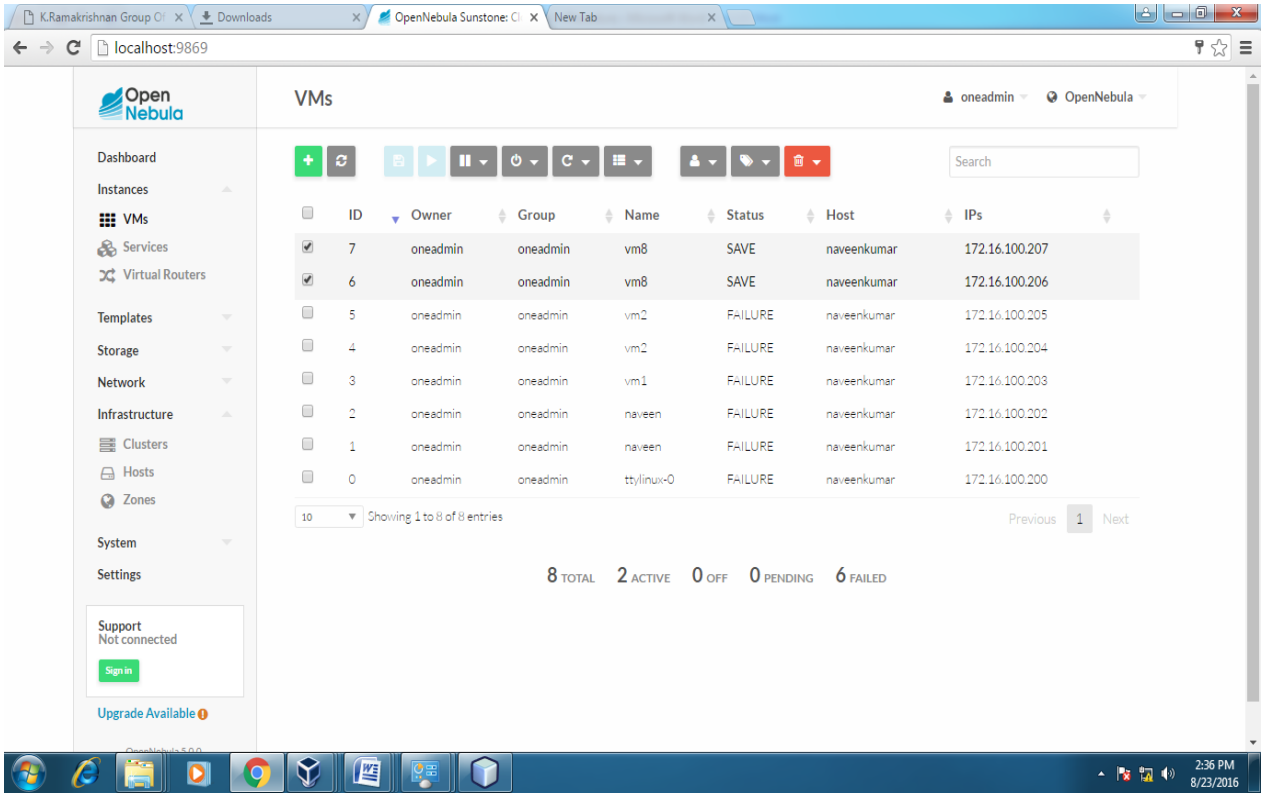
Host:SACET



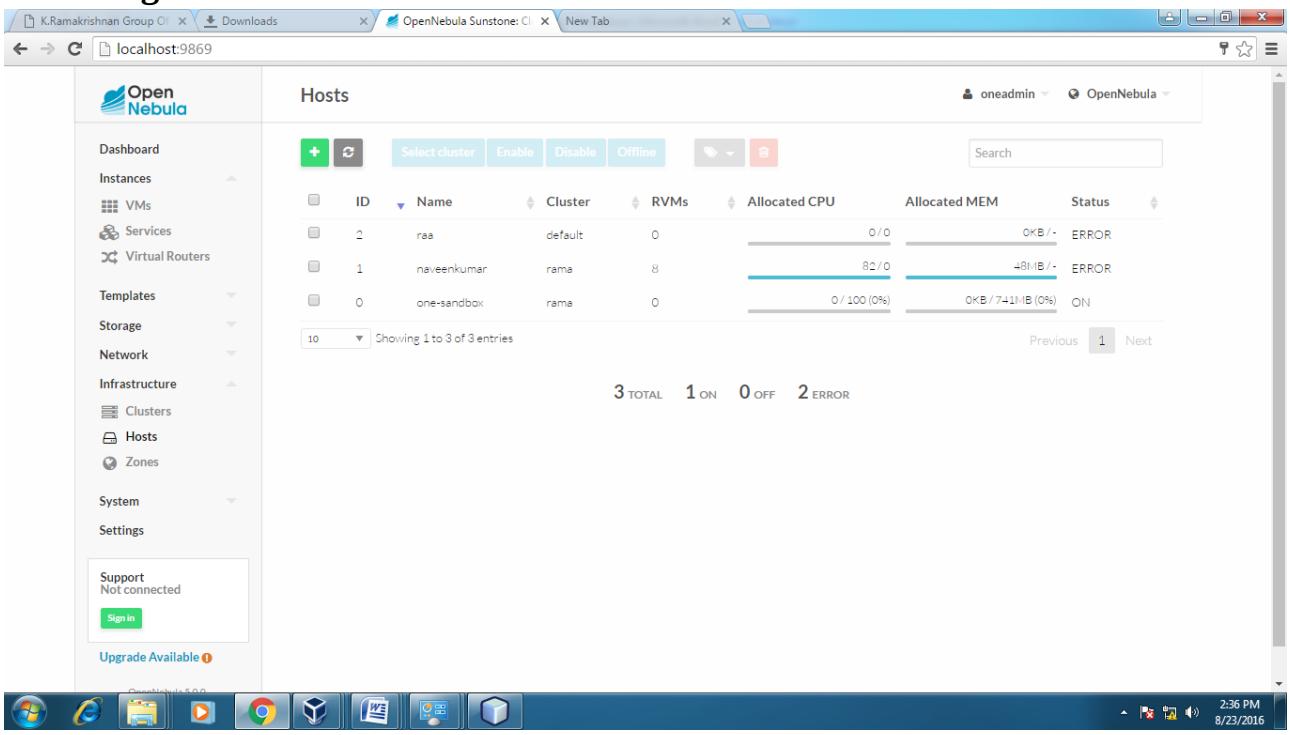
Host:one-sandbox



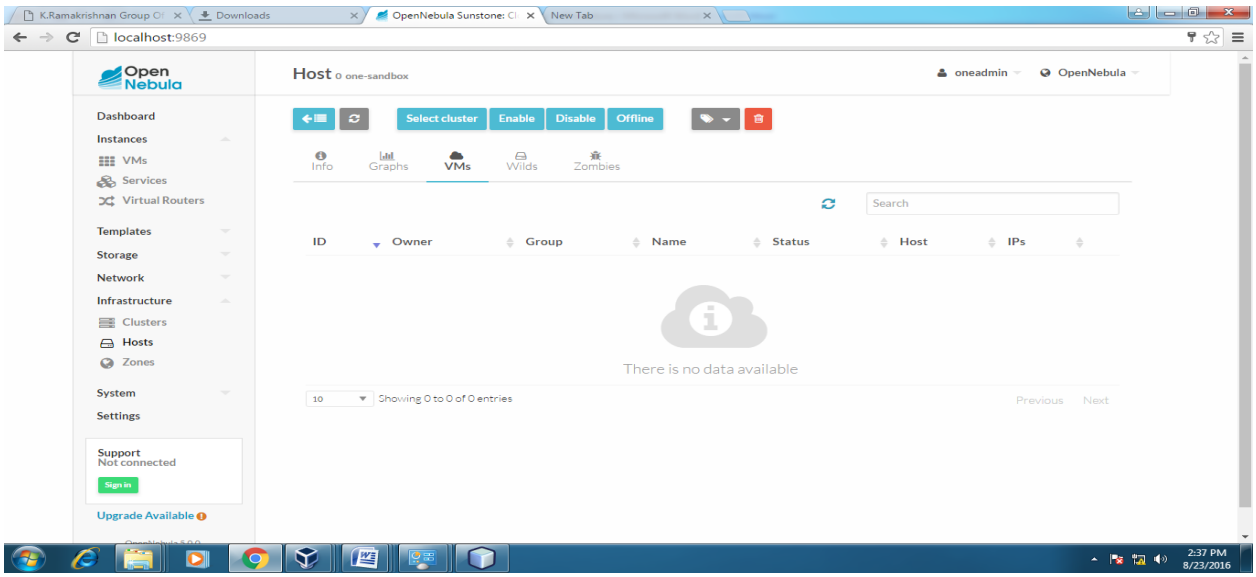

localhost:9869
Open Nebula
VMs
oneadmin
OpenNebula
Migrate Virtual Machine
VM 6 vm8 is currently running on Host one-sandbox
VM 7 vm8 is currently running on Host one-sandbox
Select a Host
Please select a Host from the list
Search
ID
Name
Cluster
RVMs
Allocated CPU
Allocated MEM
Status
2 raa default 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
10
Showing 1 to 3 of 3 entries
Previous
1
Next
Advanced Options
Migrate
2:35 PM
8/23/2016


localhost:9869
Open Nebula
VMs
oneadmin
OpenNebula
Dashboard
Instances
VMs
Services
Virtual Routers
Templates
Storage
Network
Infrastructure
Clusters
Hosts
Zones
System
Settings
Support
Not connected
Sign in
Upgrade Available
Search
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
10
Showing 1 to 8 of 8 entries
Previous
1
Next
8 TOTAL 2 ACTIVE 0 OFF 0 PENDING 6 FAILED
2:36 PM
8/23/2016

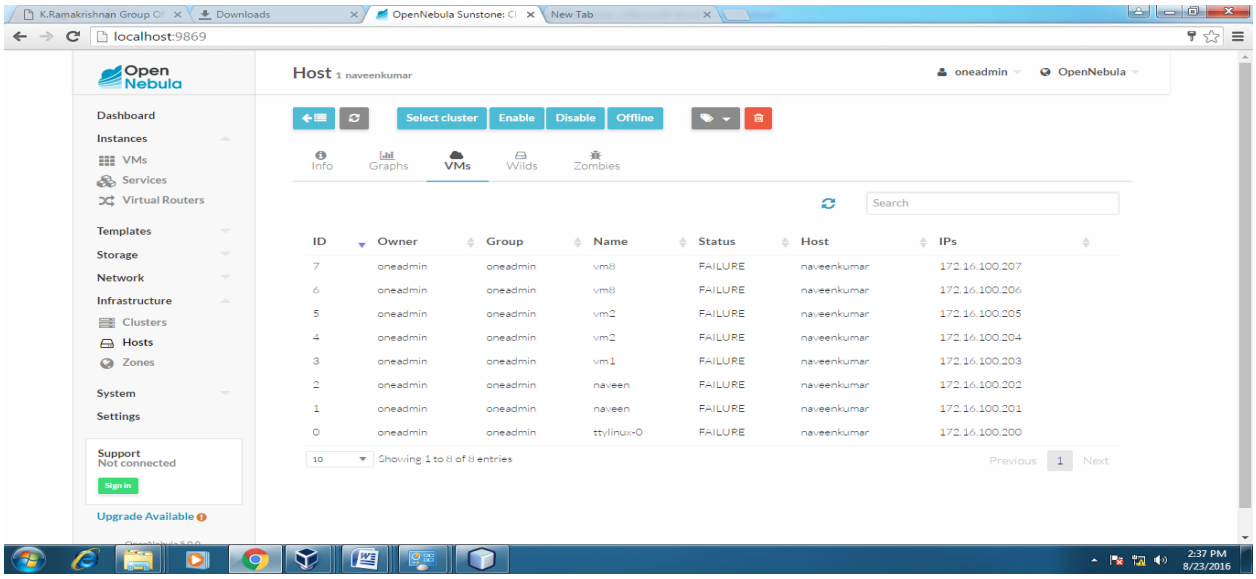
After Migration:



Host:one-sandbox



Host:SACET



APPLICATIONS:

Easily migrate your virtual machine from one pc to another.

Result:

Thus the file transfer between VM was successfully completed…..

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

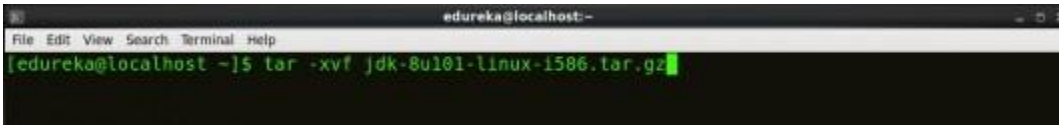


Fig: Hadoop Installation – Extracting Java Files

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

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

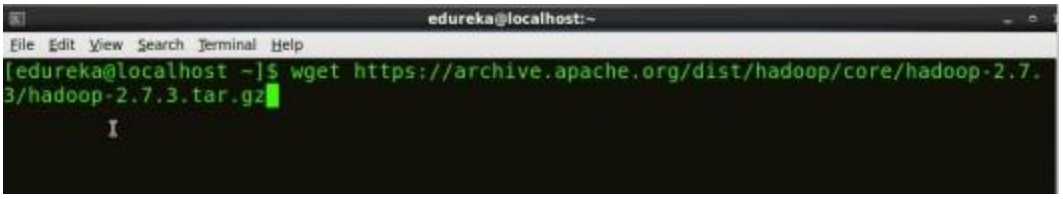


Fig: Hadoop Installation – Downloading Hadoop

Step 4: Extract the Hadoop tar File.

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

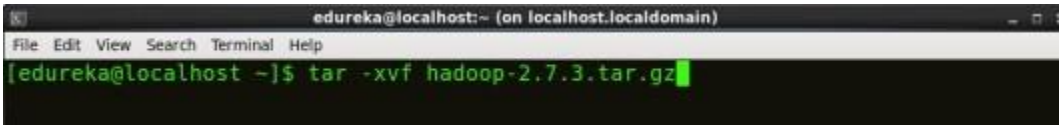
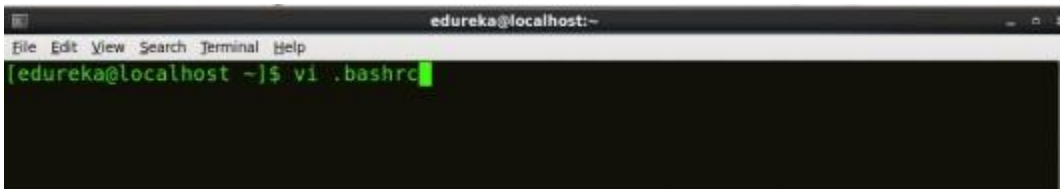


Fig: Hadoop Installation – Extracting Hadoop Files **Step**

5: Add the Hadoop and Java paths in the bash file (.bashrc). Open**. bashrc**

file. Now, add Hadoop and Java Path as shown below.

***Command*:** vi .bashrc



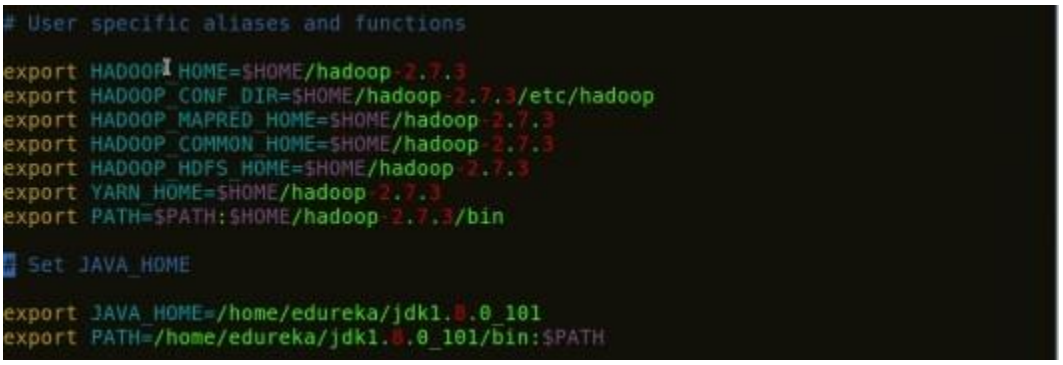


Fig: Hadoop Installation – Setting Environment Variable

Then, save the bash file and close it.

For applying all these changes to the current Terminal, execute the source command.
***Command*:** source .bashrc

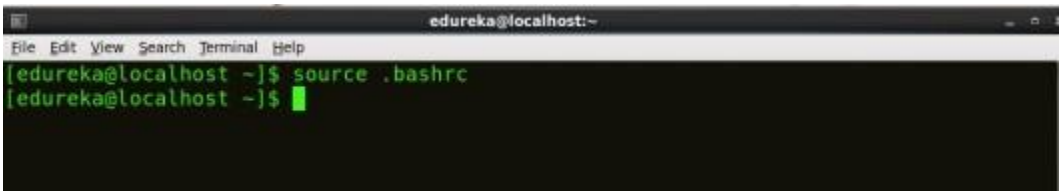
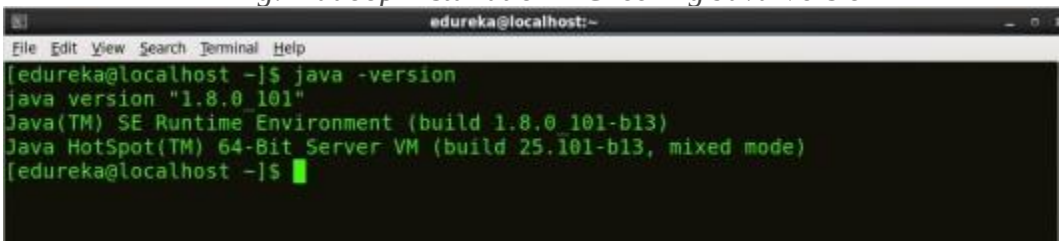


Fig: Hadoop Installation – Refreshing environment variables

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

***Command*:** java -version

Fig: Hadoop Installation – Checking Java Version



***Command*:** hadoop version

```
edureka@localhost:~
File  Edit  View  Search  Terminal  Help
[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 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.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]$
```

Fig: Hadoop Installation – Hadoop Configuration Files

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

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

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

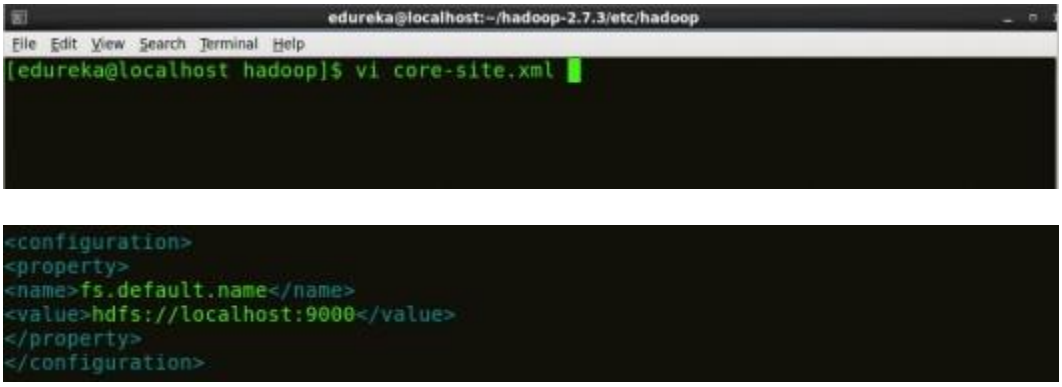


Fig: Hadoop Installation – Configuring core-site.xml

```
<?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>
</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

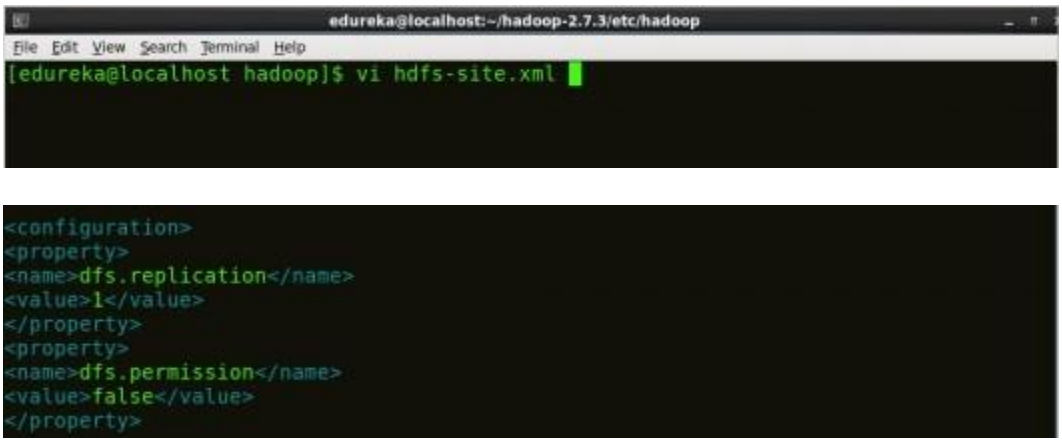


Fig: Hadoop Installation – Configuring hdfs-site.xml

```
<?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>
</property>
</configuration>
```

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

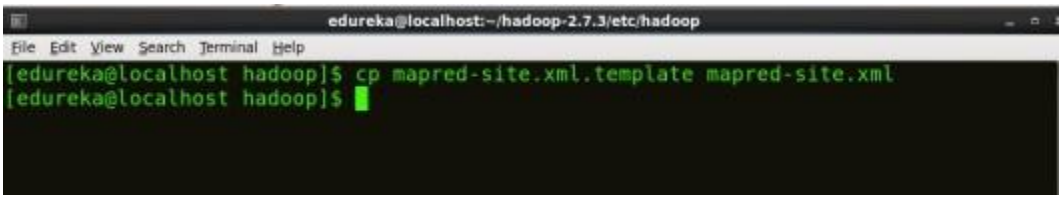
inside configuration tag:

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.



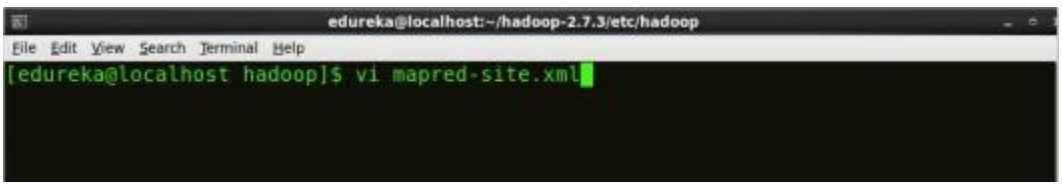


Fig: Hadoop Installation – Configuring mapred-site.xml

```
<?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>
</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

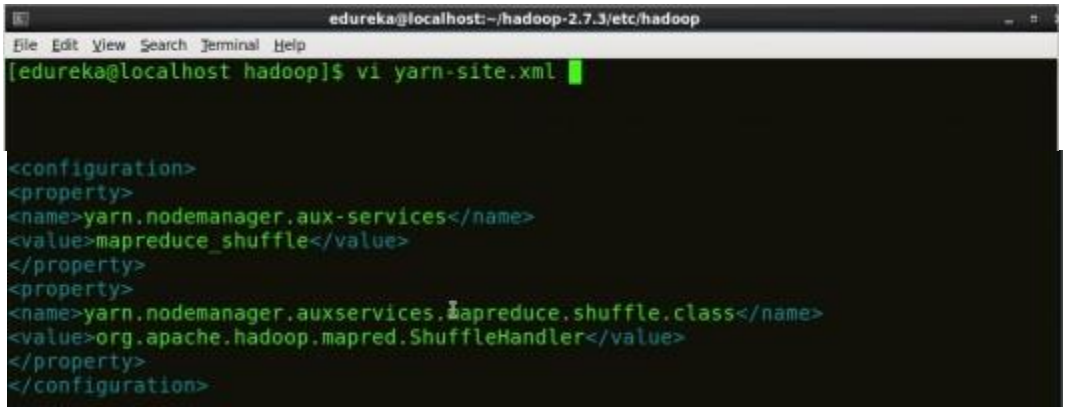


Fig: Hadoop Installation – Configuring yarn-site.xml

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

```
<?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>
</property>
</configuration>
```

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

***Command*:** vi hadoop–env.sh

Fig: Hadoop Installation – Configuring hadoop-env.sh **Step**

12: Go to Hadoop home directory and format the NameNode.

***Command*:** cd

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

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

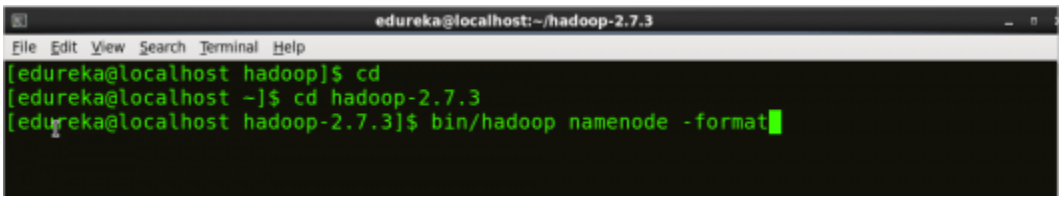


Fig: Hadoop Installation – Formatting NameNode

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

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

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

Command: cd hadoop-2.7.3/sbin

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

Command: ./start-all.sh

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

Or you can run all the services individually as below:

Start NameNode:

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

Command: ./hadoop-daemon.sh start namenode

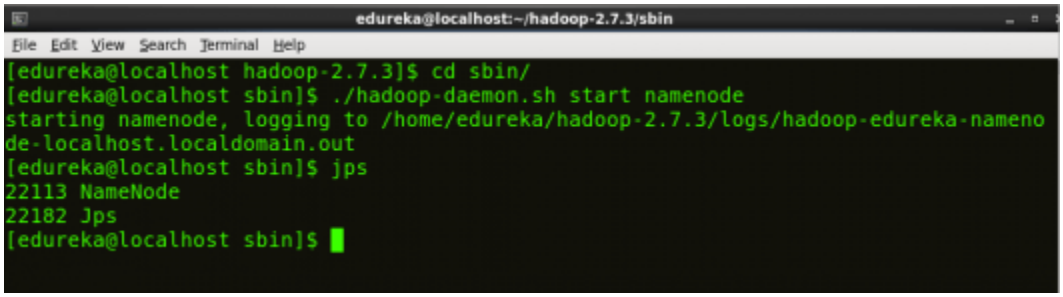


Fig: Hadoop Installation – Starting NameNode

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

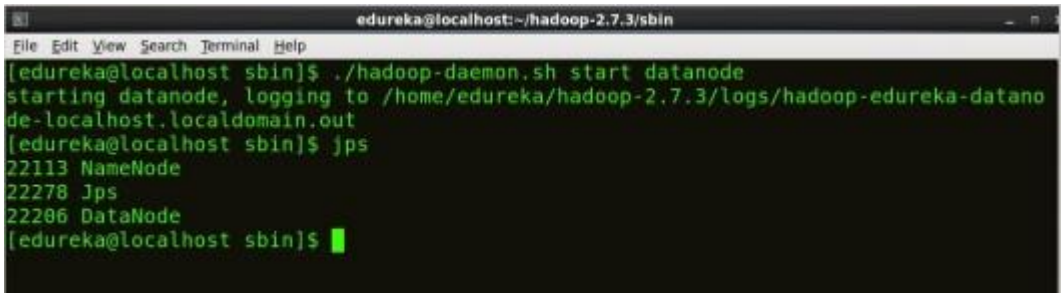


Fig: Hadoop Installation – Starting DataNode

Start ResourceManager:

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

Command: ./yarn-daemon.sh start resourcemanager

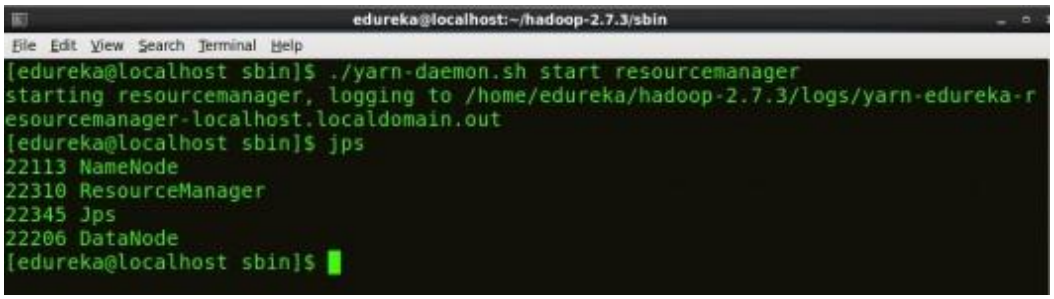


Fig: Hadoop Installation – Starting ResourceManager

Start NodeManager:

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

Command: ./yarn-daemon.sh start nodemanager

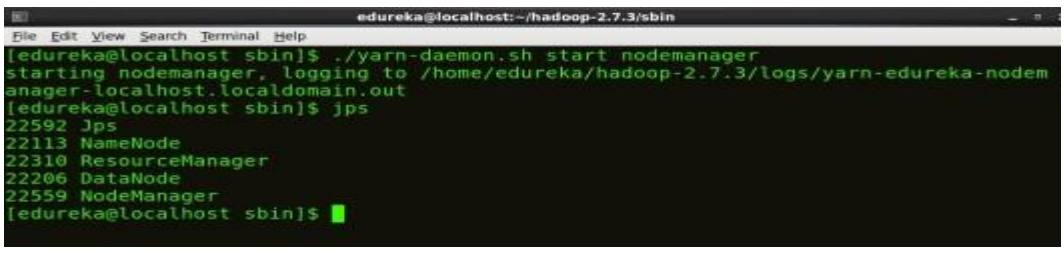


Fig: Hadoop Installation – Starting NodeManager

Start JobHistoryServer:

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

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

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

Command: jps

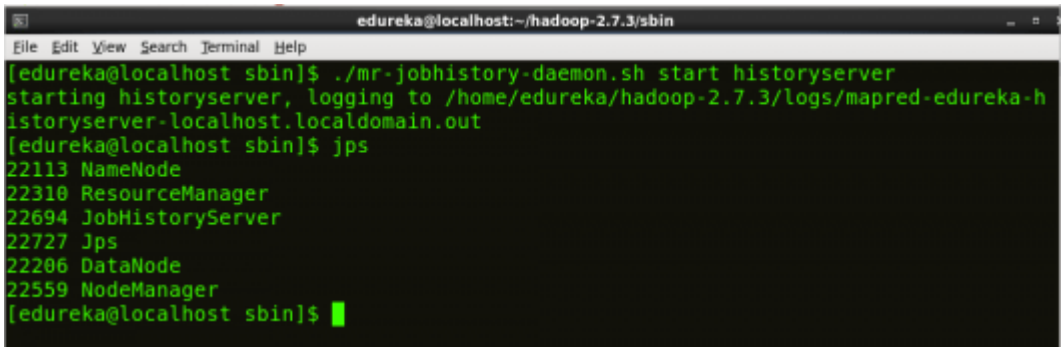


Fig: Hadoop Installation – Checking Daemons

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

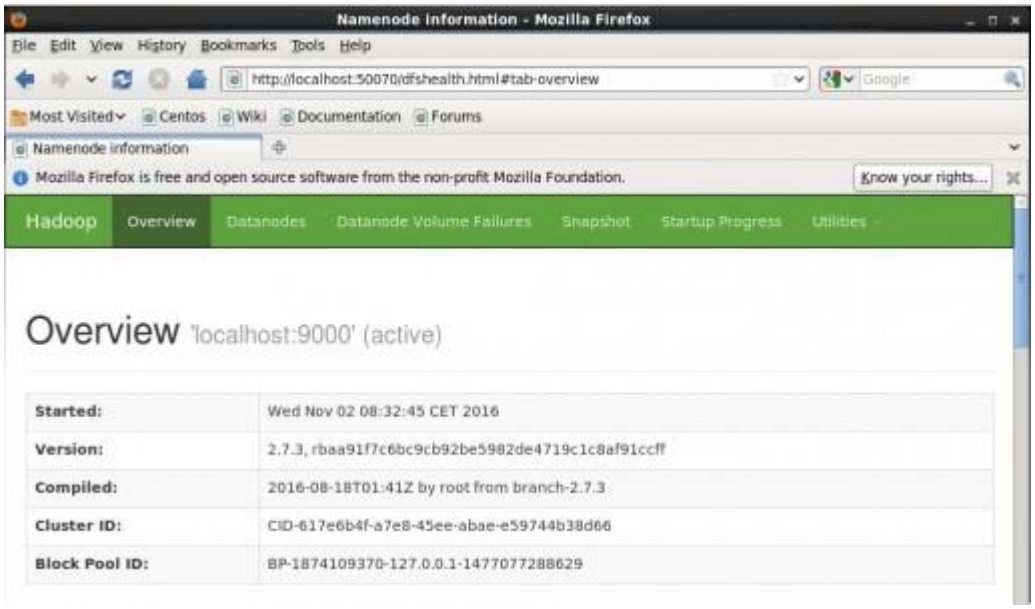


Fig: Hadoop Installation – Starting WebUI

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 Find a procedure to Creating and Executing the First Container Using Docker

Steps:

Prerequisites

Its must have access to a docker client, either on localhost, use a terminal from Theia - Cloud IDE at https://labs.cognitiveclass.ai/tools/theiadocker or be using Play with Docker for example.

Get Started

Run docker -h,

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

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

...

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

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

If podman is installed, you can run the alternative command for comparison.

sudo podman –h

You can additionally review the version of your Docker installation,

docker version

```
Client:
 Version:      19.03.6
 ...

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

You note that Docker installs both a Client and a Server: Docker Engine. For instance, if you run the same command for podman, you will see only a CLI version, because podman runs daemonless and relies on an OCI compliant container runtime (runc, crun, runv etc) to interface with the OS to create the running containers.

```
sudo podman version --events-backend=none
Version:      2.1.1
API Version:  2.0.0
Go Version:   go1.15.2
Built:        Thu Jan  1 00:00:00 1970
OS/Arch:      linux/amd64
```

Step 1: Run your first container

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

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

Use the docker container run command to run a container with the ubuntu image using the top command. The -t flags allocate a pseudo-TTY which we need for the top to work correctly

```
$ docker container run -it ubuntu top
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
aafe6b5e13de: Pull complete
0a2b43a72660: Pull complete
18bdd1e546d2: Pull complete
8198342c3e05: Pull complete
f56970a44fd4: Pull complete
Digest: sha256:f3a61450ae43896c4332bda5e78b453f4a93179045f20c8181043b26b5e79028

Status: Downloaded newer image for ubuntu:latest
```

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

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

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

top is a linux utility that prints the processes on a system and orders them by resource consumption. Notice that there is only a single process in this output: it is the top process itself. We don't see other processes from our host in this list because of the PID namespace isolation.

Containers use linux namespaces to provide isolation of system resources from other containers or the host. The PID namespace provides isolation for process IDs. If you run top while inside the container, you will notice that it shows the processes within the PID namespace of the container, which is much different than what you can see if you ran top on the host.

Even though we are using the ubuntu image, it is important to note that our container does not have its own kernel. Its uses the kernel of the host and the ubuntu image is used only to provide the file system and tools available on an ubuntu system.

 Inspect the container with docker container exec

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

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

Using play-with-docker.com, to open a new terminal connected to node1, click "Add New Instance" on the lefthand side, then ssh from node2 into node1 using the IP that is listed by 'node1 '.

For example:
[node2] (local) root@192.168.0.17 ~
$ ssh 192.168.0.18
[node1] (local) root@192.168.0.18 ~
$
In the new terminal, use the docker container ls command to get the ID of the running container you just created.

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

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

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

And Voila! We just used the docker container exec command to "enter" our container's namespaces with our bash process. Using docker container exec with bash is a common pattern to inspect a docker container.

Notice the change in the prefix of your terminal. e.g. root@b3ad2a23fab3:/. This is an indication that we are running bash "inside" of our container.

Note: This is not the same as ssh'ing into a separate host or a VM. We don't need an ssh server to connect with a bash process. Remember that containers use kernel-level features to achieve isolation and that containers run on top of the kernel. Our container is just a group of processes running in isolation on the same host, and we can use docker container exec to enter that isolation with the bash process. After running docker container exec, the group of processes running in isolation (i.e. our container) include top and bash.

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

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

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

For comparison, exit the container, and run ps -ef or top on the host. These commands will work on linux or mac. For windows, you can inspect the running processes using tasklist.

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

Technical Deep Dive PID is just one of the linux namespaces that provides containers with isolation to system resources. Other linux namespaces include: - MNT - Mount and unmount directories without affecting other namespaces - NET - Containers have their own network stack - IPC - Isolated interprocess communication mechanisms such as message queues. - User - Isolated view of users on the system - UTC - Set hostname and domain name per container
These namespaces together provide the isolation for containers that allow them to run together securely and without conflict with other containers running on the same system. Next, we will demonstrate different uses of containers. and the benefit of isolation as we run multiple containers on the same host.
Note: Namespaces are a feature of the **linux** kernel. But Docker allows you to run containers on Windows and Mac... how does that work? The secret is that embedded in the Docker product or Docker engine is a linux subsystem. Docker open-sourced this linux subsystem to a new project: LinuxKit. Being able to run containers on many different platforms is one advantage of using the Docker tooling with containers.
In addition to running linux containers on Windows using a linux subsystem, native Windows containers are now possible due the creation of container primitives on the Windows OS. Native Windows containers can be run on Windows 10 or Windows Server 2016 or newer.
Note: if you run this exercise in a containerized terminal and execute the ps -ef command in the terminal, e.g. in https://labs.cognitiveclass.ai, you will still see a limited set of processes after exiting the exec command. You can try to run the ps -ef command in a terminal on your local machine to see all processes.
 Clean up the container running the top processes by typing: <ctrl>-c, list all containers and remove the containers by their id.

4. docker ps -a
5.
6. docker rm <CONTAINER ID>
7.

Step 2: Run Multiple Containers

1. Explore the Docker Hub
   The Docker Hub is the public central registry for Docker images, which contains community and official images.
   When searching for images you will find filters for "Docker Certified", "Verified Publisher" and "Official Images" images. Select the "Docker Certified" filter, to find images that are deemed enterprise-ready and are tested with Docker Enterprise Edition product. It is important to avoid using unverified content from the Docker Store when developing your own images that are intended to be deployed into the production environment. These unverified images may contain security vulnerabilities or possibly even malicious software.

In Step 2 of this lab, we will start a couple of containers using some verified images from the Docker Hub: nginx web server, and mongo database

2. Run an Nginx server

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

 $ docker container run --detach --publish 8080:80 --name nginx nginx

Unable to find image 'nginx:latest' locally

latest: Pulling from library/nginx

36a46ebd5019: Pull complete

57168433389f: Pull complete

332ec8285c50: Pull complete

Digest: sha256:c15f1fb8fd55c60c72f940a76da76a5fccce2fefa0dd9b17967b9e40b0355316

Status: Downloaded newer image for nginx:latest

5e1bf0e6b926bd73a66f98b3cbe23d04189c16a43d55dd46b8486359f6fdf048

We are using a couple of new flags here. The --detach flag will run this container in the background. The publish flag publishes port 80 in the container (the default port for nginx), via port 8080 on our host. Remember that the NET namespace gives processes of the container their own network stack. The --publish flag is a feature that allows us to expose networking through the container onto the host.

How do you know port 80 is the default port for nginx? Because it is listed in the documentation on the Docker Hub. In general, the documentation for the verified images is very good, and you will want to refer to them when running containers using those images.

We are also specifying the --name flag, which names the container. Every container has a name, if you don't specify one, Docker will randomly assign one for you. Specifying your own name makes it easier to run subsequent commands on your container since you can reference the name instead of the id of the container. For example: docker container inspect nginx instead of docker container inspect 5e1.

Since this is the first time you are running the nginx container, it will pull down the nginx image from the Docker Store. Subsequent containers created from the Nginx image will use the existing image located on your host.

Nginx is a lightweight web server. You can access it on port 8080 on your localhost.

 Access the nginx server on localhost:8080.

curl localhost:8080

will return the HTML home page of Nginx,

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

</head>
<body>
<h1>Welcome to nginx!</h1>

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

1. Run an Nginx server
   Let's run a container using the official Nginx image from the Docker Hub.

 $ docker container run --detach --publish 8080:80 --name nginx nginx
Unable to find image 'nginx:latest' locally
    latest: Pulling from library/nginx

We are using a couple of new flags here. The --detach flag will run this container in the background. The publish flag publishes port 80 in the container (the default port for nginx), via port 8080 on our host. Remember that the NET namespace gives processes of the container their own network stack. The --publish flag is a feature that allows us to expose networking through the container onto the host.
How do you know port 80 is the default port for nginx? Because it is listed in the documentation on the Docker Hub. In general, the documentation for the verified images is very good, and you will want to refer to them when running containers using those images.
We are also specifying the --name flag, which names the container. Every container has a name, if you don't specify one, Docker will randomly assign one for you. Specifying your own name makes it easier to run subsequent commands on your container since you can reference the name instead of the id of the container. For example: docker container inspect nginx instead of docker container inspect 5e1.
Since this is the first time you are running the nginx container, it will pull down the nginx image from the Docker Store. Subsequent containers created from the Nginx image will use the existing image located on your host.
Nginx is a lightweight web server. You can access it on port 8080 on your localhost.
 Access the nginx server on localhost:8080.
curl localhost:8080
will return the HTML home page of Nginx,
 <!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
  body {
    width: 35em;
    margin: 0 auto;
    font-family: Tahoma, Verdana, Arial, sans-serif;
  }
</style>
</head>
<body>
      <h1>Welcome to nginx!</h1>

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.

- Run a mongo DB server

Now, run a mongoDB server. We will use the official mongoDB image from the Docker Hub. Instead of using the latest tag (which is the default if no tag is specified), we will use a specific version of the mongo image: 4.4.

- $ docker container run --detach --publish 8081:27017 --name mongo mongo:4.4

Unable to find image mongo:4.4 locally
4.4: Pulling from library/mongo
Status: Downloaded newer image for mongo:4.4

Again, since this is the first time we are running a mongo container, we will pull down the mongo image from the Docker Store. We are using the --publish flag to expose the 27017 mongo port on our host. We have to use a port other than 8080 for the host mapping, since that port is already exposed on our host. Again refer to the official docs on the Docker Hub to get more details about using the mongo image.

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

curl localhost:8081

which will return a warning from MongoDB,

- It looks like you are trying to access MongoDB over HTTP on the native driver port.
    - If you are using play-with-docker, look for the 8080 link near the top of the page.

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

- Check your running containers with docker container ls

8. $ docker container ls
9. CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
10. d6777df89fea nginx "nginx -g 'daemon ..." Less than a second ago Up 2 seconds 0.0.0.0:8080->80/tcp nginx
11. ead80a0db505 mongo "docker-entrypoint..." 17 seconds ago Up 19 seconds 0.0.0.0:8081->27017/tcp mongo
12. af549dccd5cf ubuntu "top" 5 minutes ago Up 5 minutes priceless_kepler
13. You should see that you have an Nginx web server container, and a MongoDB container running on your host. Note that we have not configured these containers to talk to each other.
    You can see the "nginx" and "mongo" names that we gave to our containers, and the random name (in my case "priceless_kepler") that was generated for the ubuntu container. You can also see that the port mappings that we specified with the --publish flag. For more details information on these running containers you can use the docker container inspect [container id command.
    One thing you might notice is that the mongo container is running the docker-entrypoint command.

    This is the name of the executable that is run when the container is started. The mongo image requires some prior configuration before kicking off the DB process. You can see exactly what the script does by looking at it on github. Typically, you can find the link to the github source from the image description page on the Docker Store website.
    Containers are self-contained and isolated, which means we can avoid potential conflicts between containers with different system or runtime dependencies. For example: deploying an app that uses Java 7 and another app that uses Java 8 on the same host. Or running multiple nginx containers that all have port 80 as their default listening ports (if exposing on the host using the --publish flag, the ports selected for the host will need to be unique). Isolation benefits are possible because of Linux Namespaces.
    Note: You didn't have to install anything on your host (other than Docker) to run these processes! Each container includes the dependencies that it needs within the container, so you don't need to install anything on your host directly.
    Running multiple containers on the same host gives us the ability to fully utilize the resources (cpu, memory, etc) available on single host. This can result in huge cost savings for an enterprise.
    While running images directly from the Docker Hub can be useful at times, it is more useful to create custom images, and refer to official images as the starting point for these images. We will dive into building our own custom images in Lab 2.

Step 3: Clean Up

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

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

- $ docker container ls

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
d6777df89fea nginx "nginx -g 'daemon ..." 3 minutes ago Up 3 minutes 0.0.0.0:8080->80/tcp nginx
ead80a0db505 mongo "docker-entrypoint..." 3 minutes ago Up 3 minutes 0.0.0.0:8081->27017/tcp mongo

af549dccd5cf ubuntu "top" 8 minutes ago Up 8 minutes priceless_kepler

 Next, run docker container stop [container id] for each container in the list. You can also use the names of the containers that you specified before.

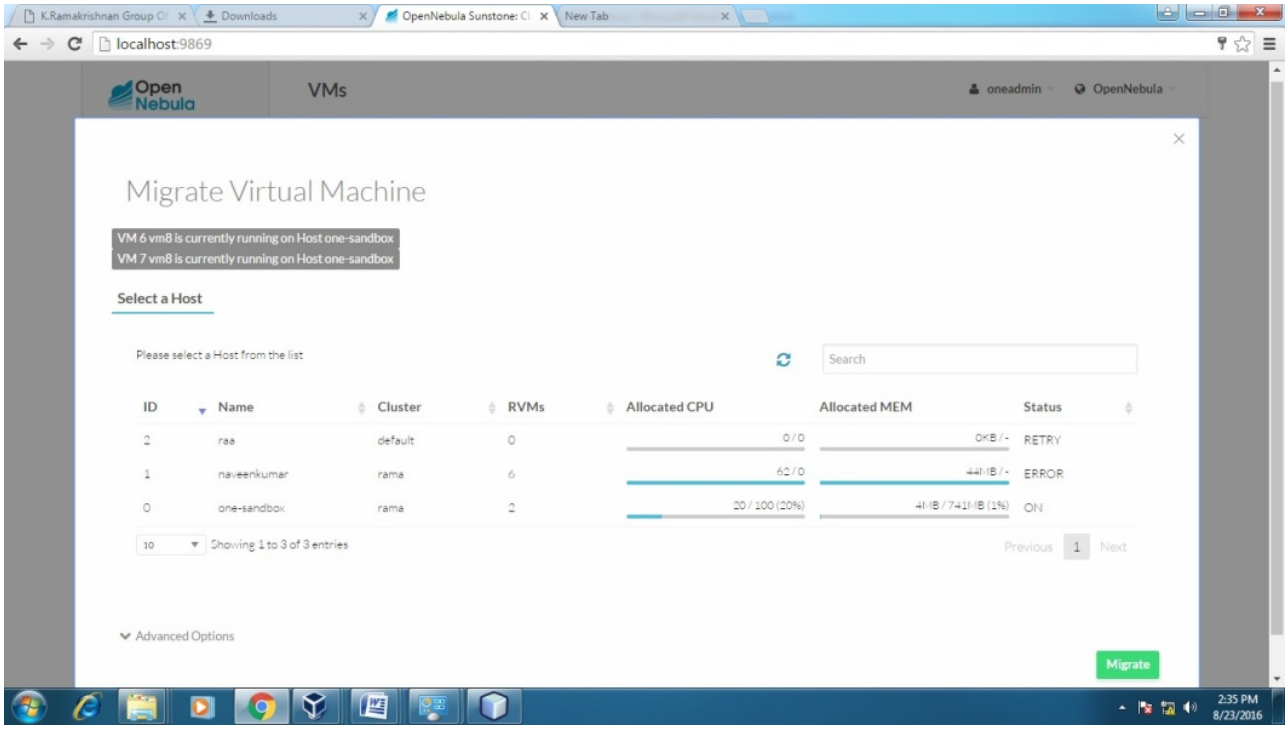
 $ docker container stop d67 ead af5 d67 ead af5

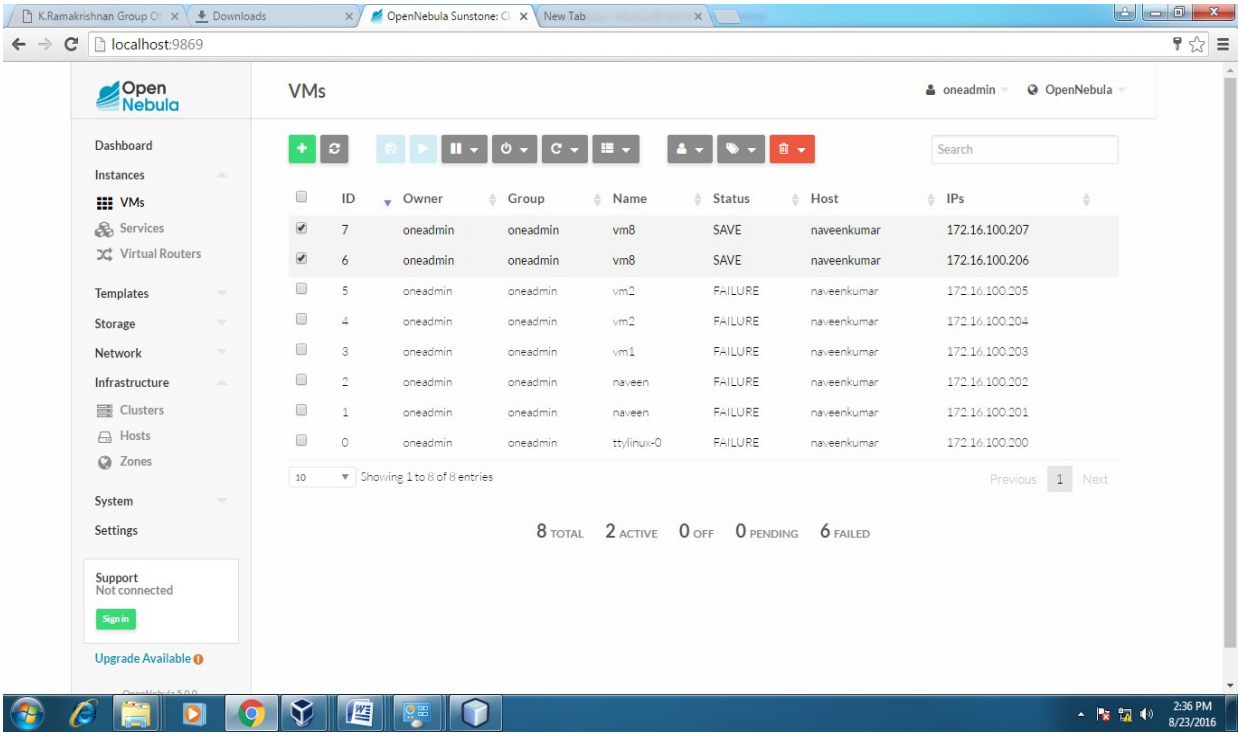
Note: You only have to reference enough digits of the ID to be unique. Three digits is almost always enough.

 Remove the stopped containers

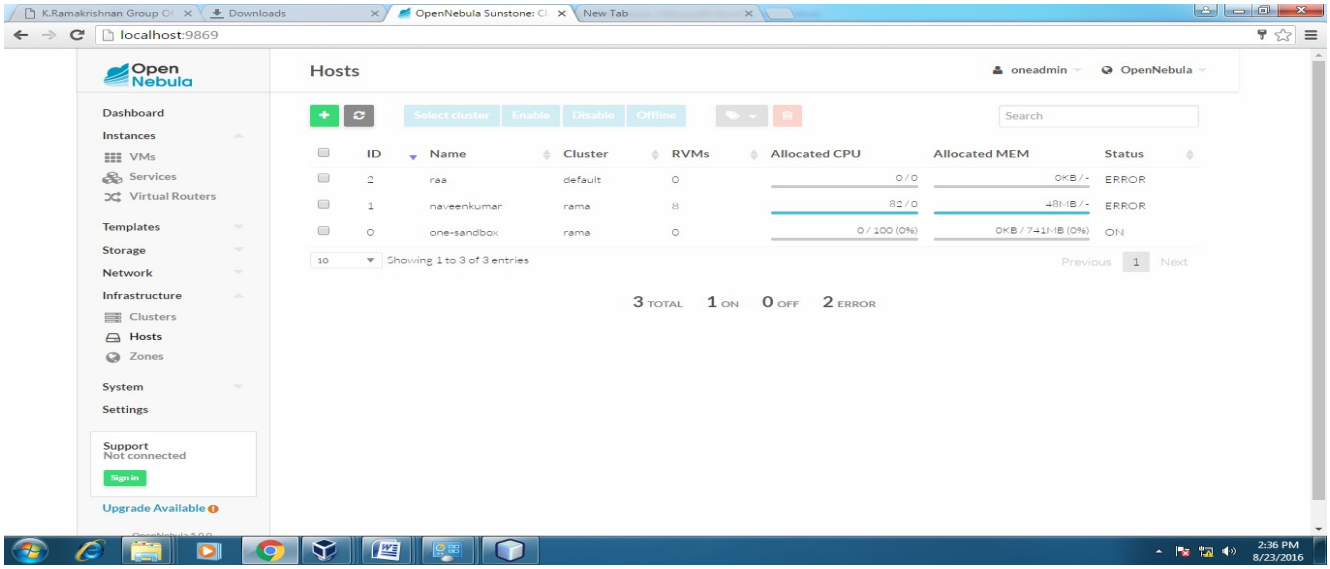
docker system prune is a really handy command to clean up your system. It will remove any stopped containers, unused volumes and networks, and dangling images.

3. $ docker system prune
4. WARNING! This will remove:
5. - all stopped containers
6. - all volumes not used by at least one container
7. - all networks not used by at least one container
8. - all dangling images
9. Are you sure you want to continue? [y/N] y
10. Deleted Containers:
11. Total reclaimed space: 12B
12.

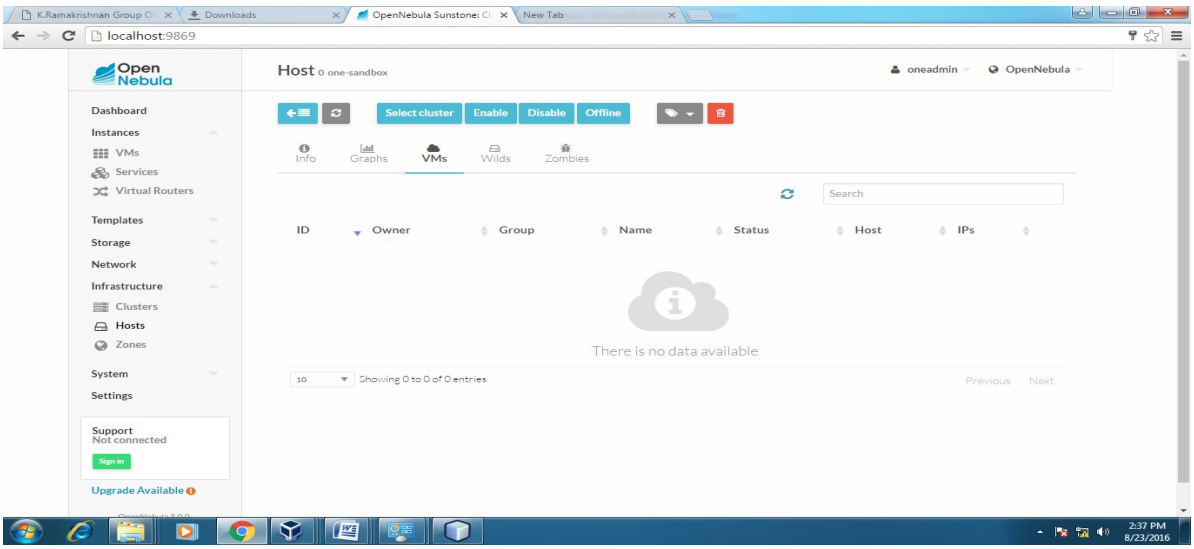



K.Ramakrishnan Group Of
Downloads
OpenNebula Sunstone: Cl
New Tab
localhost:9869
Open Nebula
VMs
oneadmin
OpenNebula
Dashboard
Instances
VMs
Services
Virtual Routers
Templates
Storage
Network
Infrastructure
Clusters
Hosts
Zones
System
Settings
Support
Not connected
Sign in
Upgrade Available
Search
ID
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
10
Showing 1 to 8 of 8 entries
Previous
1
Next
8 TOTAL
2 ACTIVE
0 OFF
0 PENDING
6 FAILED
2:36 PM
8/23/2016

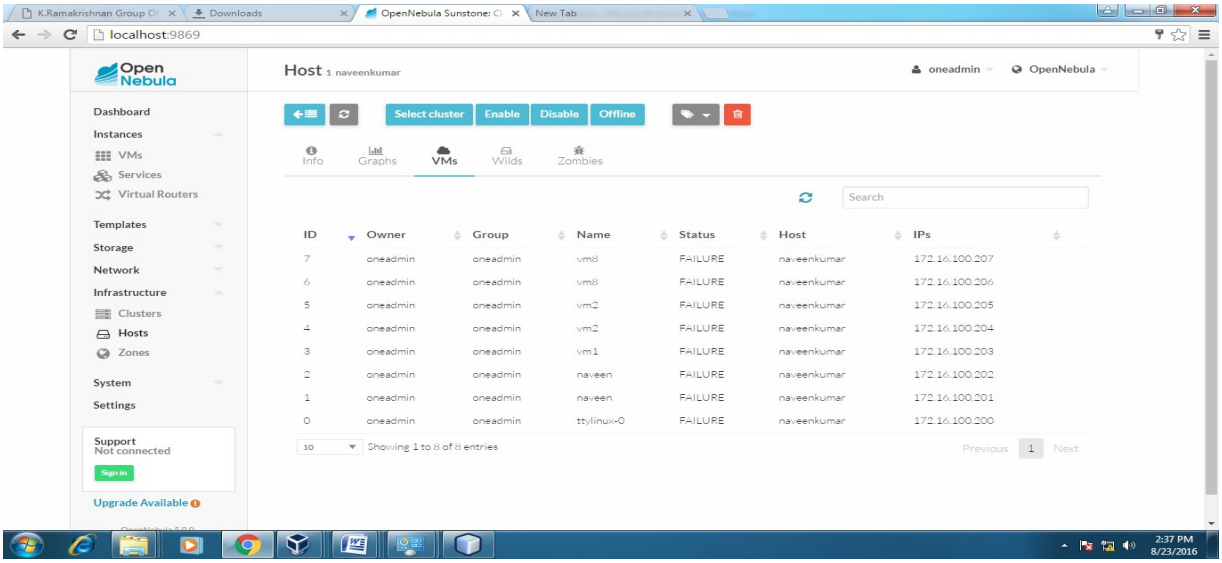
After Migration:



Host:one-sandbox



Host:SACET



APPLICATIONS:

Easily migrate your virtual machine from one pc to another.

Result:

Thus the Creating and Executing the First Container Using Docker is completed successfully.

EX.NO:9
DATE:

Run a Container from Dockers Hub

Aim:

To find a procedure to run a Container from Dockers Hub

Steps:

The following section contains step-by-step instructions on how to get started with Docker Hub.

Step 1: Sign up for a Docker account

Start by creating a Docker ID

.A Docker ID grants you access to Docker Hub repositories and lets you explore available images from the community and verified publishers. You also need a Docker ID to share images on Docker Hub.

Step 2: Create your first repository

To create a repository:

1. Sign in to Docker Hub.
2. Select **Create a Repository** on the Docker Hub welcome page.
3. Name it **<your-username>/my-private-repo**.
4. Set the visibility to **Private**.
5. Select **Create**.

You've created your first repository.

Step 3: Download and install Docker Desktop

You need to download Docker Desktop to build, push, and pull container images.

1. Download and install Docker Desktop.
2. Sign in to Docker Desktop using the Docker ID you created in step one.

Step 4: Pull and run a container image from Docker Hub

1. In your terminal, run docker pull hello-world to pull the image from Docker Hub. You should see output similar to:
2. $ docker pull hello-world
3. Using default tag: latest
4. latest: Pulling from library/hello-world

5. Status: Downloaded newer image for hello- world: latest
6. docker.io/library/hello-world:latest
7. Run docker run hello-world to run the image locally. You should see output similar to:
8. $ docker run hello-world
9. Hello from Docker!
10. To generate this message, Docker took the following steps:
11. 1. The Docker client contacted the Docker daemon.
12. 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
13. (amd64)
14. 3. The Docker daemon created a new container from that image which runs the
15. executable that produces the output you are currently reading.
16. 4. The Docker daemon streamed that output to the Docker client, which sent
17. it to your terminal.
18.
19. To try something more ambitious, you can run an Ubuntu container with:
20. $ docker run -it ubuntu bash
21.
22. Share images, automate workflows, and more with a free Docker ID:
23. https://hub.docker.com/
24.
25. For more examples and ideas, visit:
26. https://docs.docker.com/get-started/

Step 5: Build and push a container image to Docker Hub from your computer

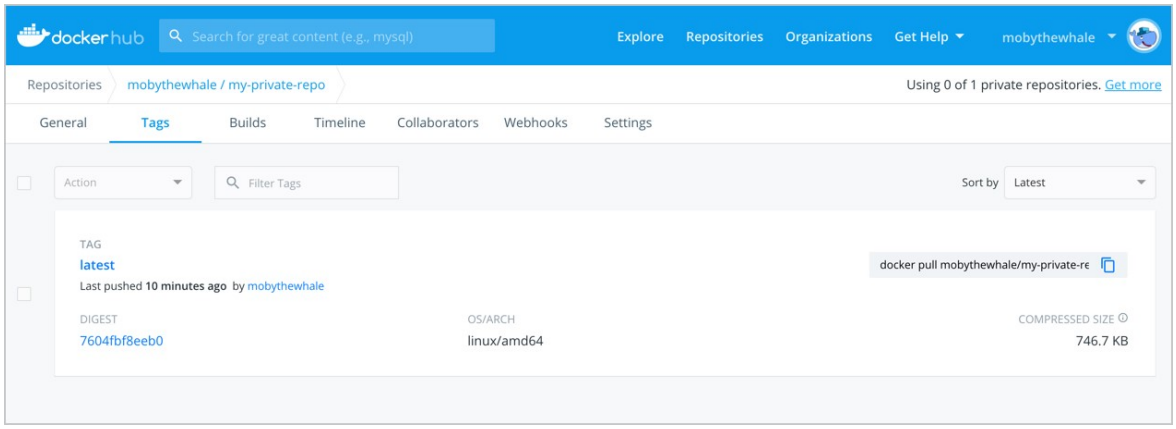
1. Start by creating a Dockerfile to specify your application as shown below:
2. # syntax=docker/dockerfile:1
3. FROM busybox
   CMD echo "Hello world! This is my first Docker image."
4. Run docker build -t <your_username>/my-private-repo . to build your Docker image.
5. Run docker run <your_username>/my-private-repo to test your Docker image locally.
   Run docker push <your_username>/my-private-repo to push your Docker image to Docker Hub. You should see output similar to:

```
[redacted] cat > Dockerfile <<EOF
FROM busybox
CMD echo "Hello world! This is my first Docker image."
EOF
[redacted] docker build -t mobythewhale/my-private-repo .
[+] Building 1.2s (5/5) FINISHED
 => [internal] load build definition from Dockerfile                    0.0s
 => => transferring dockerfile: 110B                                    0.0s
 => [internal] load .dockerignore                                       0.0s
 => => transferring context: 2B                                         0.0s
 => [internal] load metadata for docker.io/library/busybox:latest       1.2s
 => CACHED [1/1] FROM docker.io/library/busybox@sha256:a9286defaba7b3a519  0.0s
 => exporting to image                                                  0.0s
 => => exporting layers                                                 0.0s
 => => writing image sha256:dcdb1fd928bfb257bfc0122ea47accd911a3a386ce618  0.0s
 => => naming to docker.io/mobythewhale/my-private-repo                 0.0s
[redacted] docker run mobythewhale/my-private-repo
Hello world! This is my first Docker image.
[redacted] docker push mobythewhale/my-private-repo
The push refers to repository [docker.io/mobythewhale/my-private-repo]
d2421964bad1: Layer already exists
latest: digest: sha256:7604fbf8eeb03d866fd005fa95cdbb802274bf9fa51f7dafba6658294
efa9baa size: 526
[redacted]
```

Note

You must be signed in to Docker Hub through Docker Desktop or the command line, and you must also name your images correctly, as per the above steps.

Your repository in Docker Hub should now display a new latest tag under **Tags**



You've successfully:

- Signed up for a Docker account
- Created your first repository
- Pulled an existing container image from Docker Hub
- Built your own container image on your computer

  Pushed it successfully to Docker Hub

Result:

Thus the Running a Container from Docker Hub is completed successfully