LABORATORY

1. Install Virtualbox/VMware Workstation with different flavours of linux or windows OS on top of windows7 or 8.
2. Install a C compiler in the virtual machine created using virtual box and execute Simple Programs
3. Install Google App Engine. Create *hello world* app and other simple web applications using python/java.
4. Use GAE launcher to launch the web applications.
5. Simulate a cloud scenario using CloudSim and run a scheduling algorithm that is not present in CloudSim.
6. Find a procedure to transfer the files from one virtual machine to another virtual machine.
7. Install Hadoop single node cluster and run simple applications like wordcount.
8. Creating and Executing Your First Container Using Docker.
9. Run a Container from Docker Hub

Ex. No:1 Install Virtualbox / VMware Workstation with different flavours of linux or windows OS on top of windows 7 or 8.

Aim : To Install Virtualbox / VMware Workstation with different flavours of linux

Procedure:

How to Install VMware Workstation

While you have downloaded VMware on your system then the file will be on download folder by default, and the file name should be like this VMware-workstation-full-15.0.2-10952284.exe. so now let's start the installation, read each step carefully and follow the same step.

#1. After clicking on the setup so click on the **Yes** button.

#2. **Setup Wizard:** then the setup wizard will install the VMware Workstation Pro on your system so click **Next** to go ahead and click **Exit** to cancel the installation.

#3. **End-User License Agreement:** in this step accept the terms in the license agreement and click on the **Next** button.

#4. **Custom Setup:** select the folder in which you would like to install the VMware application. and besides that select enhanced keyboard driver.

#5. **User Experience Setting:** select both options but you can uncheck it, so I leave it as default and click on the **Next** button.

#6. **Application Shortcuts preference:** here select the place you want the shortcut icons to be placed on your system to launch the application so I recommend you select both options and click on the **Next** button.

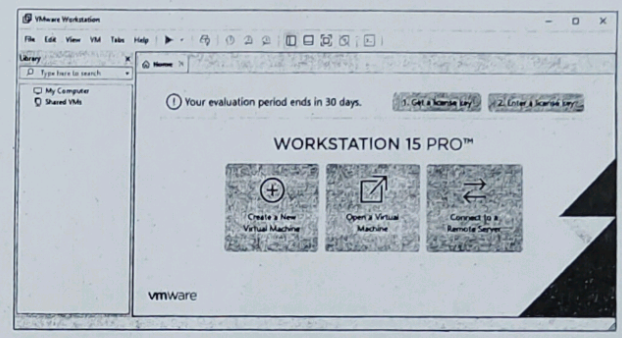
#7. **Install VM Workstation:** in this step, the installation is ready to go, so click on **the Install** button to begin the installation.

#8. **Installing VM Workstation On Windows: The** below screenshot shows the installation is in progress, so wait for this to complete.

#9. **Complete the Setup Wizard:** In the end, you will see the installation complete dialog box. so, click on **Finish** and you are done with installation progress. while you click on Finish then it will ask you to restart your computer so you can do it by clicking on **Yes** or if you want to do it later then click on **No.**

Step 4. Launch VMware Workstation on Windows

Ok, now you have successfully installed VM Workstation on your system, so while you want to launch the application then it will ask you to provide the license key if you don't have so you have another option too. select the next option that you have 15 to 30-day trial and you will like this page.



Result :

Thus the Installation of Virtualbox/VMware Workstation with different flavours of linux or windows OS on top of windows7 successfully.

Ex. No:2 Install a C compiler in the virtual machine and execute a sample program.

Aim :

To find procedure to attach virtual block to the virtual machine and check whether it holds the data even after the release of the virtual machine.

Procedure:

1. Install a C compiler in the virtual machine and execute a sample program.

   Through Openstack portal create virtual machine. Through the portal connect to virtual machines. Login to VMs and install c compiler using commands.

   Eg : apt-get install gcc

2. Show the virtual machine migration based on the certain condition from one node to the other.

   To demonstrate virtual machine migration, two machines must be configured in one cloud. Take snapshot of running virtual machine and copy the snapshot file to the other destination machine and restore the snapshot. On restoring the snapshot, VM running in source will be migrated to destination machine.

   1. List the VMs you want to migrate, run:

      $ nova list

   2. After selecting a VM from the list, run this command where *VM_ID* is set to the ID in the list returned in the previous step:

      $ nova show VM_ID

   3. Use the **nova migrate** command.

      $ nova migrate VM_ID

   4. To migrate an instance and watch the status, use this example script:

```
#!/bin/bash
# Provide usage
usage() {
echo "Usage: $0 VM_ID"
exit 1  }
[[ $# -eq 0 ]] && usage
# Migrate the VM to an alternate hypervisor
echo -n "Migrating instance to alternate host"
VM_ID=$1
nova migrate $VM_ID
VM_OUTPUT=`nova show $VM_ID`
VM_STATUS=`echo "$VM_OUTPUT" | grep status | awk '{print $4}'`
while [[ "$VM_STATUS" != "VERIFY_RESIZE" ]]; do
echo -n "."
sleep 2
VM_OUTPUT=`nova show $VM_ID`
VM_STATUS=`echo "$VM_OUTPUT" | grep status | awk '{print $4}'`
done
nova resize-confirm $VM_ID
echo " instance migrated and resized."
echo;
# Show the details for the VM
echo "Updated instance details:"
nova show $VM_ID
# Pause to allow users to examine VM details
read -p "Pausing, press <enter> to exit."
```


Result : Thus the Install a C compiler in the virtual machine and execute a sample program created successfully.

Ex. No:3a Install Google App Engine. Create *hello world* app and other simple web applications using java.

Aim :

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

Procedure :

Tools used :

1. JDK 1.6
2. Eclipse 3.7 + Google Plugin for Eclipse
3. Google App Engine Java SDK 1.6.3.1

Note
GAE supports Java 1.5 and 1.6.

P.S Assume JDK1.6 and Eclipse 3.7 are installed.

1. 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...**"

   *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

   Expand and 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...
```

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

File : appengine-web.xml

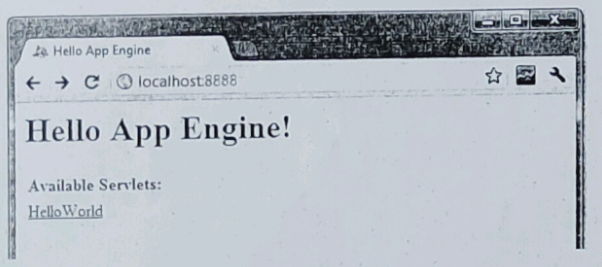
```
<?xml version="1.0" encoding="utf-8"?>
<appengine-web-app xmlns="http://appengine.google.com/ns/1.0">
 <application></application>
 <version>1</version>

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

</appengine-web-app>
```

4. Run it local

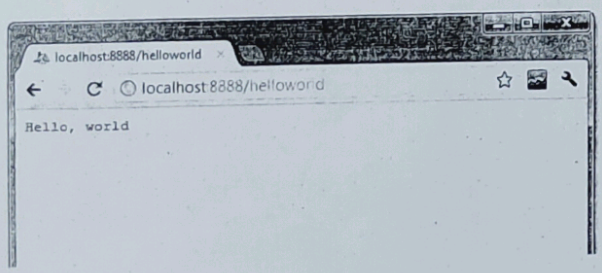
   Right click on the project and run as "**Web Application**".

   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
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.

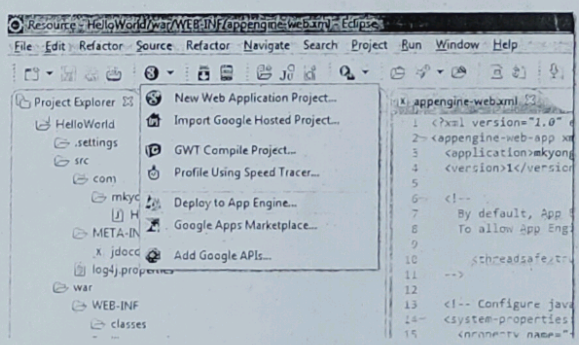
In this demonstration, I created an application ID, named "myapp", and put it in appengine-web.xml.

File : appengine-web.xml

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

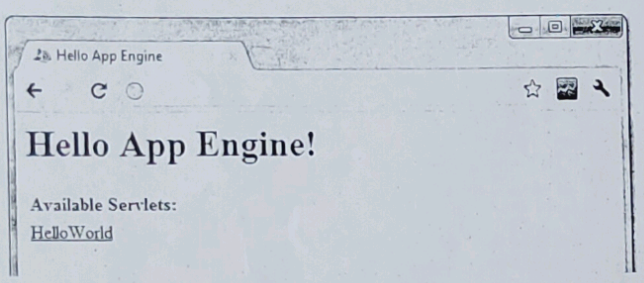
To deploy, see following steps:

Figure 1 – Click on GAE deploy button on the toolbar.



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

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



Ex. No:3b **Install Google App Engine. Create *hello world* app and other simple web applications using python.**

Aim:

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

Procedure:

Tools used :

1. Python 2.7
2. Eclipse 3.7 + PyDev plugin
3. Google App Engine SDK for Python 1.6.4

P.S Assume Python 2.7 and Eclipse 3.7 are installed.

1. Install PyDev plugin for Eclipse

   Use following URL to install PyDev as Eclipse plugin.

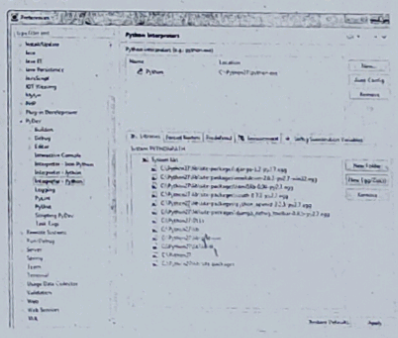
   http://pydev.org/updates

   In Eclipse , menu, "Help –> Install New Software.." and put above URL. Select "**PyDev for Eclipse**" option, follow steps, and restart Eclipse once completed.

2. Verify PyDev

   After Eclipse is restarted, make sure **PyDev's interpreter** is pointed to your "python.exe".

   Eclipse -> Windows –> Preferences, make sure "**Interpreter – Python**" is configured properly.



3. Google App Engine SDK Python

   Download and install Google App Engine SDK for Python.

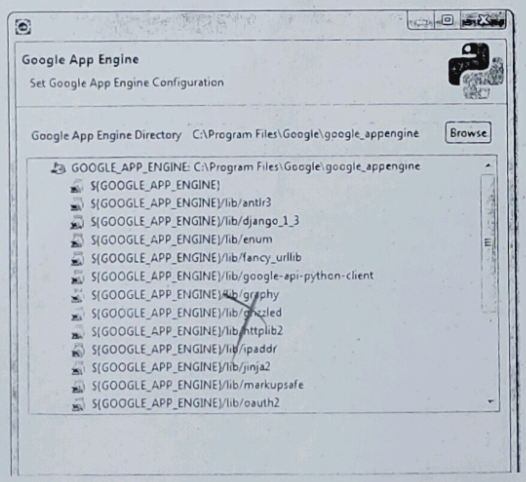
4. Python Hello World in Eclipse

   Following steps to show you how to create a GAE project via Pydev plugin.

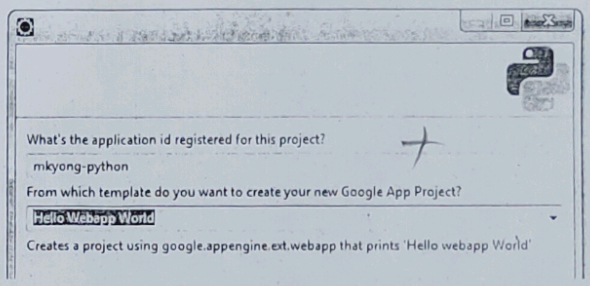
   Eclipse menu, File -> New -> Other… , PyDev folder, choose "**PyDev Google App Engine Project**".

Type project name, if the interpreter is not configure yet (in step 2), you can do it now. And select this option – **"Create 'src' folder and add it to PYTHONPATH"**.

Click "Browse" button and point it to the Google App Engine installed directory (in step 3).



Name your application id in GAE, type anything, you can change it later. And choose "**Hello Webapp World**" template to generate the sample files.



Done, 4 files are generated, Both ".pydevproject" and ".project" are Eclipse project files, ignore it.

Review the generated Python's files :

File : helloworld.py – Just output a hello world.

```
from google.appengine.ext import webapp
from google.appengine.ext.webapp.util import run_wsgi_app

class MainPage(webapp.RequestHandler):

    def get(self):
        self.response.headers['Content-Type'] = 'text/plain'
        self.response.out.write('Hello, webapp World!')

application = webapp.WSGIApplication([('/', MainPage)], debug=True)

def main():
    run_wsgi_app(application)

if __name__ == "__main__":
    main()
```

File : app.yaml – GAE need this file to run and deploy your Python project, it's quite self-explanatory, for detail syntax and configuration, visit yaml and app.yaml reference.

```
application: mkyong-python
version: 1
runtime: python
api_version: 1

handlers:
- url: /.*
  script: helloworld.py
```

5. Run it locally

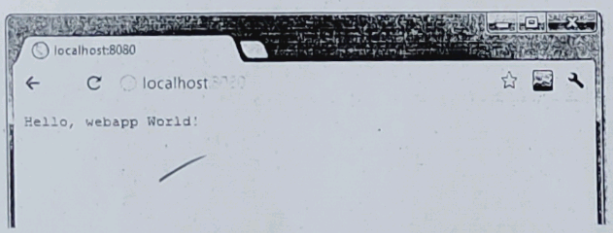
To run it locally, right click on the helloworld.py, choose "Run As" –> "Run Configuration", create a new "**PyDev Google App Run**".

In Main tab -> Main module, manually type the directory path of "**dev_appserver.py**". "Browse" button is not able to help you, type manually.

In Arguments tab -> Program arguments, put "**${project_loc}/src**".

Run it. By default, it will deploy to *http://localhost:8080*.

Done.



Ex. No: 4 Use GAE launcher to launch the web applications.

Aim:

To use **GAE** launcher to launch the web applications

Procedure :

Deploy to Google App Engine

Register an account on https://appengine.google.com/, and create an application ID for your web application. Review "app.yaml" again, this web app will be deployed to GAE with application ID "**mkyong-python**".

File : app.yaml

```
application: mkyong-python
version: 1
runtime: python
api_version: 1

handlers:
- url: /.*
  script: helloworld.py
```

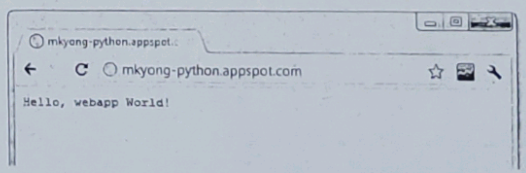
To deploy to GAE, see following steps :

Create another new "PyDev Google App Run", In Main tab -> Main module, manually type the directory path of "**appcfg.py**".

In Arguments tab -> Program arguments, put "**update ${project_loc}/src**".

During deploying process, you need to type your GAE email and password for authentication.

If success, the web app will be deployed to – *http://mkyong-python.appspot.com/*.



Result

Thus, GAE launcher is used to launch the web applications.

Ex. No:5 Simulate a cloud scenario using CloudSim and run a scheduling algorithm that is not present in CloudSim.

Aim:

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

Procedure:

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

   CloudSim.init(num_user, calendar, trace_flag)

6. 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), storageList, 0);

7. The third step is to create a broker:

   DatacenterBroker broker = createBroker();

8. 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())

9. Submit the VM list to the broker:

   broker.submitVmList(vmlist)

10. Create a cloudlet with length, file size, output size, and utilisation model:

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

11. Submit the cloudlet list to the broker:

   broker.submitCloudletList(cloudletList)

12. Start the simulation:

```
CloudSim.startSimulation()
Sample Output from the Existing Example:
Starting CloudSimExample1...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>null
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.1: Broker: VM #0 has been created in Datacenter #2, Host #0
0.1: Broker: Sending cloudlet 0 to VM #0
400.1: Broker: Cloudlet 0 received
```

```
400.1: Broker: All Cloudlets executed. Finishing...
400.1: Broker: Destroying VM #0
Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.
```

========== OUTPUT ==========

Cloudlet ID	STATUS	Data center ID	VM ID	Time	Start Time	Finish Time
0	SUCCESS	2	0	400	0.1	400.1

*****Datacenter: Datacenter_0*****

User id	Debt
3	35.6

CloudSimExample1 finished!

Result:

Thus the scheduling algorithm executed successfully using CloudSim.

Ex. No:6 Find a procedure to transfer the files from one virtual machine to another virtual machine.

Aim:

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

Prerequisites

- Verify that the virtual machines use a guest operating system that supports shared folders.
- Verify that the latest version of VMware Tools is installed in the guest operating system.
- Verify that permission settings on the host system allow access to files in the shared folders. For example, if you are running Workstation Player as a user named User, the virtual machine can read and write files in the shared folder only if User has permission to read and write them.

Procedure

1. Select the virtual machine and select **Player > Manage > Virtual Machine Settings**.
2. On the **Options** tab, select **Shared Folders**.
3. Select a folder sharing option.

Option	Description
Always enabled	Keep folder sharing enabled, even when the virtual machine is shut down, suspended, or powered off.
Enabled until next power off or suspend	Enable folder sharing temporarily, until you power off, suspend, or shut down the virtual machine. If you restart the virtual machine, shared folders remain enabled. This setting is available only when the virtual machine is powered on.

4. **(Optional)** To map a drive to the Shared Folders directory, select **Map as a network drive in Windows guests**.

This directory contains all of the shared folders that you enable. Workstation Player selects the drive letter.

5. Click **Add** to add a shared folder.

   On Windows hosts, the **Add Shared Folder** wizard starts.

6. Type the path on the host system to the directory to share.

   If you specify a directory on a network share, such as D:\share, Workstation Player always attempts to use that path. If the directory is later connected to the host on a different drive letter, Workstation Player cannot locate the shared folder.

7. Specify the name of the shared folder as it should appear inside the virtual machine.

   Characters that the guest operating system considers illegal in a share name appear differently when viewed inside the guest. For example, if you use an asterisk in a share name, you see %002A instead of * in the share name on the guest. Illegal characters are converted to their ASCII hexadecimal value.

8. Select shared folder attributes.

Option	Description
Enable this share	Enable the shared folder. Deselect this option to disable a shared folder without deleting it from the virtual machine configuration.
Read-only	Make the shared folder read-only. When this property is selected, the virtual machine can view and copy files from the shared folder, but it cannot add, change, or remove files. Access to files in the shared folder is also governed by permission settings on the host computer.

9. Click **Finish** to add the shared folder.

   The shared folder appears in the Folders list. The check box next to folder name indicates that the folder is being shared. You can deselect this check box to disable sharing for the folder.

10. Click **OK** to save your changes.

View the shared folder. On Linux guests, shared folders appear under /mnt/hgfs. On Solaris guests, shared folders appear under /hgfs.

Creating a Virtual Machine in OpenStack via OpenStack CLI

Source the credential file and then execute the nova-boot command,

```
# nova boot —flavor m1.small —image centos7 —nic net-id={private_network_id}
—security-group norprod_sec_grp  —key-name my_key stack_testvm
```

Once nova boot command is executed then following are steps are executed behind the scene,

Step:1) The Horizon Dashboard or OpenStack CLI gets user credentials and authenticates with identity service via REST API

- The identity service (Keystone) authenticate the user with the user credentials and then generates and send back an auth-token, that auth-token which will be used for sending the request to other components through REST-Call

Step:2) The Dashboard or OpenStack CLI converts new instance request specified in launch instance or nova boot command to a REST API request and sent it to nova-api

Step:3) Then nova-api service gets the request and send that request to the identity service (Keystone) for validation of auth-token and access permission,

- Keystone service validates the token and send the updated authentication headers with roles along with the permissions

Step:4) After getting the repsonse from keystone, then nova-api checks for conflicts with nova-database and then it creates initial database entry for new instance or VM.

Step:5) nova-api sends the rpc.call request to nova-scheduler expecting to get updated instance entry with host id specified

Step:6) Now nova-scheduler picks the request from the queue

Step:7) nova-scheduler talks to nova-database to locate an appropriate host using filtering and weighing mechanism,

- nova-scheduler returns the updated instance entry with the appropriate host ID after filtering and weighing

- nova-scheduler sends the rpc.cast request to nova compute for launching an instance on the appropriate host

Step:8) nova-compute picks the request from the queue and it sends the rpc.call request to nova-conductor to get the VM or instance info such as host id and flavor (RAM,CPU and Disk)

Step:9) nova-conductor takes the request from queue and communicate with nova-database,

- nova-conductor gets the instance information
- now nova-compute picks the instance information from the queue

Step:10) nova-compute connects to glance-api by making a REST Call using auth-token and then nova-compute uses the image id to get the image URI from image service and loads the image from image storage

Step:11) glance-api validates the auth-token with keystone and after that nova-compute gets the image metadata

Step:12) Nova-compute make the REST-call by passing the auth-token to Network API (Neutron) to allocate and configure network so that vm gets the IP address

Step:13) Neutron-server validates the auth-token with keystone and after that nova-compute retrieves the network information.

Step:14) Nova-Compute makes the REST-call by passing the auth-token to Volume API to attach the volume to the instance or VM.

Step:15) cinder-api validates the auth-token with keystone and then nova-compute gets the block storage information.

Step:16) nova-compute generates data for the hypervisor driver and executes the request on the hypervisor using libvirt or API and then finally a VM is created on the hypervior. We can see that VM in Dashboard and also using "nova list" command.

Result:

Thus, the procedure to transfer the files from one virtual machine to another virtual machine completed successfully.

Ex. No:7 Install Hadoop single node cluster and run simple applications like wordcount.

Prerequisites

- *VIRTUAL BOX*: it is used for installing the operating system on it.
- *OPERATING SYSTEM*: You can install Hadoop on Linux based operating systems. Ubuntu and CentOS are very commonly used. In this tutorial, we are using CentOS.
- *JAVA*: You need to install the Java 8 package on your system.
- *HADOOP*: You require Hadoop 2.7.3 package.

Install Hadoop

Step 1: Click here to download the Java 8 Package. Save this file in your home directory.

Step 2: Extract the Java Tar File.

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

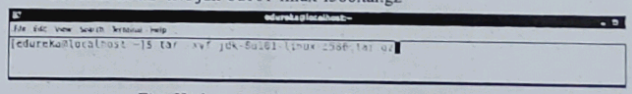


Fig: Hadoop Installation – Extracting Java Files

Step 3: Download the Hadoop 2.7.3 Package.

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

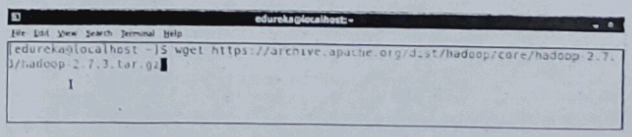


Fig: Hadoop Installation – Downloading Hadoop

Step 4: Extract the Hadoop tar File.

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

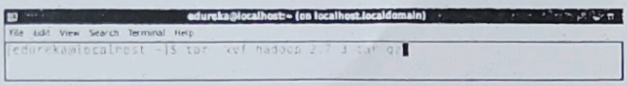


Fig: Hadoop Installation – Extracting Hadoop Files

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

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

Command: vi .bashrc

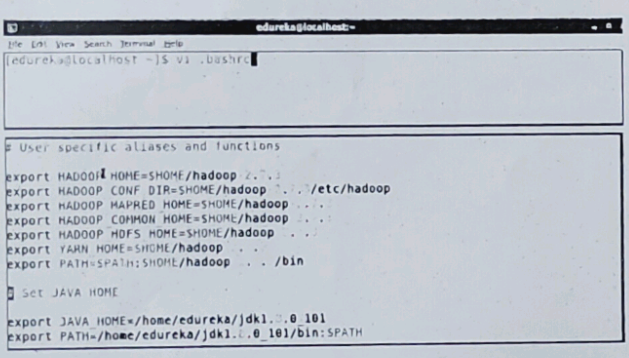


Fig: Hadoop Installation – Setting Environment Variable

Then, save the bash file and close it.

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

Command: source .bashrc

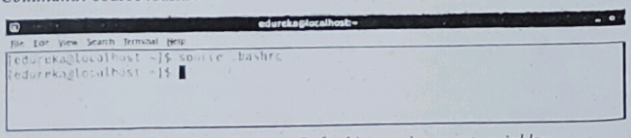


Fig: Hadoop Installation – Refreshing environment variables

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

Command: java -version

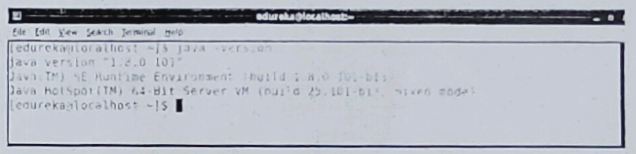


Fig: Hadoop Installation – Checking Java Version

Command: hadoop version

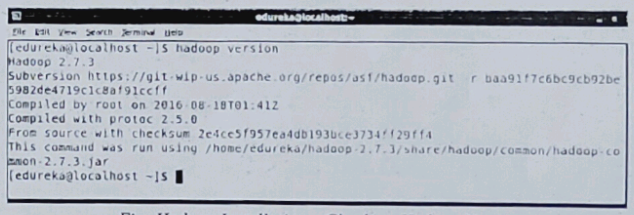


Fig: Hadoop Installation – Checking Hadoop Version

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

Command: cd hadoop-2.7.3/etc/hadoop/

Command: ls

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

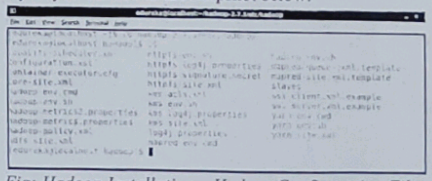


Fig: Hadoop Installation – Hadoop Configuration Files

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

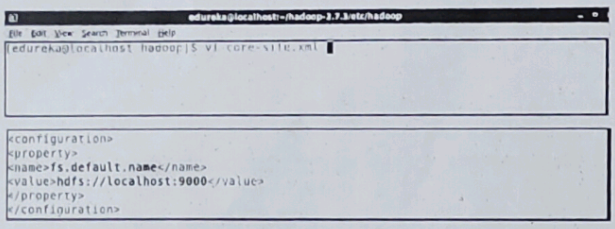


Fig: Hadoop Installation – Configuring core-site.xml

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

Step 8: Edit *hdfs-site.xml* and edit the property mentioned below inside configuration tag:

hdfs-site.xml contains configuration settings of HDFS daemons (i,e. NameNode, DataNode, Secondary NameNode). It also includes the replication factor and block size of HDFS.

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

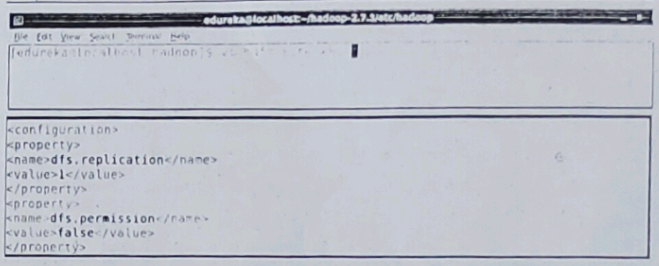


Fig: Hadoop Installation – Configuring hdfs-site.xml

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

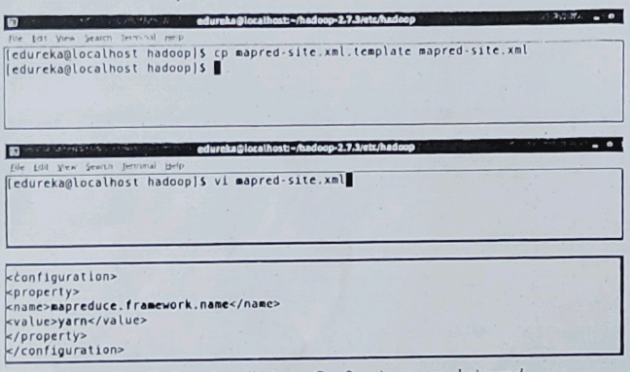


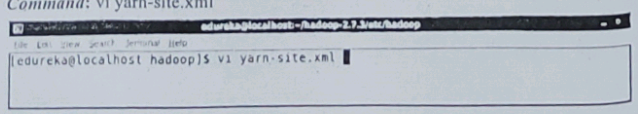
Fig: Hadoop Installation – Configuring mapred-site.xml

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



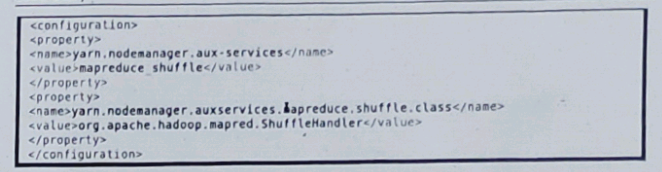


Fig: Hadoop Installation – Configuring yarn-site.xml

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

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

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

***Command*:** vi hadoop–env.sh

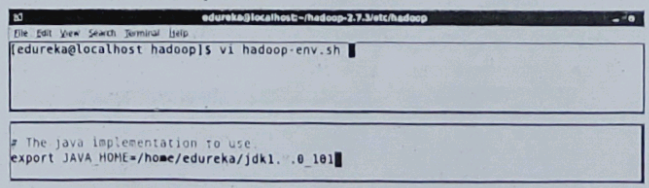


Fig: Hadoop Installation – Configuring hadoop-env.sh

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

Command: cd

Command: cd hadoop-2.7.3

Command: bin/hadoop namenode -format

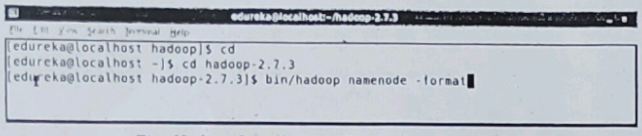


Fig: Hadoop Installation – Formatting NameNode

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

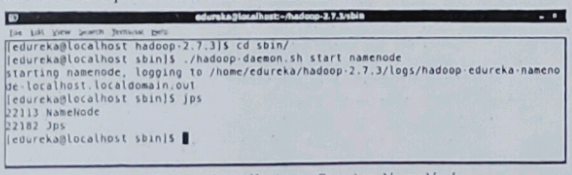


Fig: Hadoop Installation – Starting NameNode

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

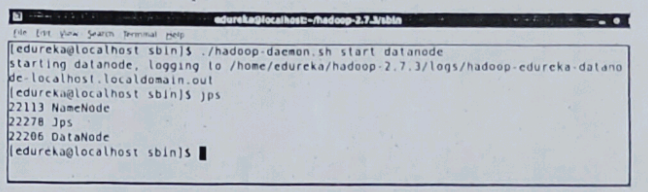


Fig: Hadoop Installation – Starting DataNode

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

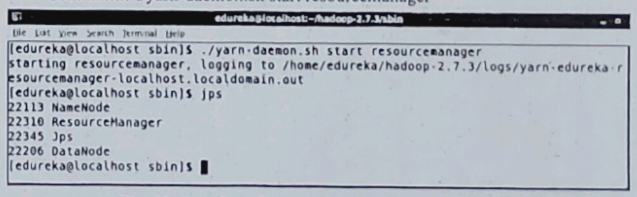


Fig: Hadoop Installation – Starting ResourceManager

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

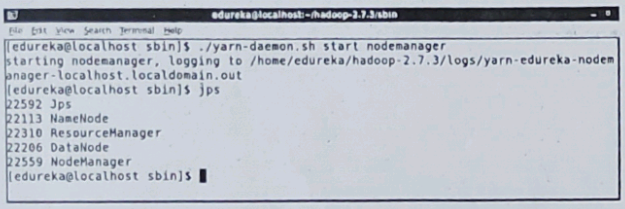


Fig: Hadoop Installation – Starting NodeManager

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

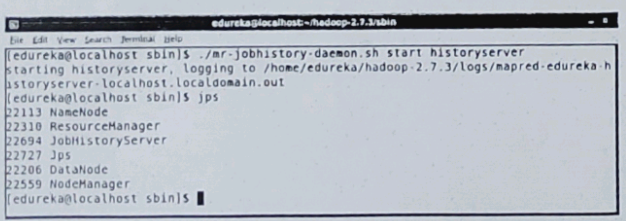


Fig: Hadoop Installation – Checking Daemons

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

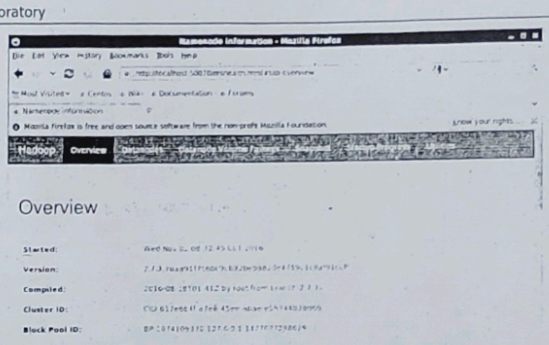


Fig: Hadoop Installation – Starting WebUI

Tools and Technologies used in this article

1. Apache Hadoop 2.2.0
2. Windows 7 OS
3. JDK 1.6

1. Install Apache Hadoop 2.2.0 in Microsoft Windows OS

If **Apache Hadoop 2.2.0** is not already installed then follow the post Build, Install, Configure and Run Apache Hadoop 2.2.0 in Microsoft Windows OS.

2. Start HDFS (Namenode and Datanode) and YARN (Resource Manager and Node Manager)

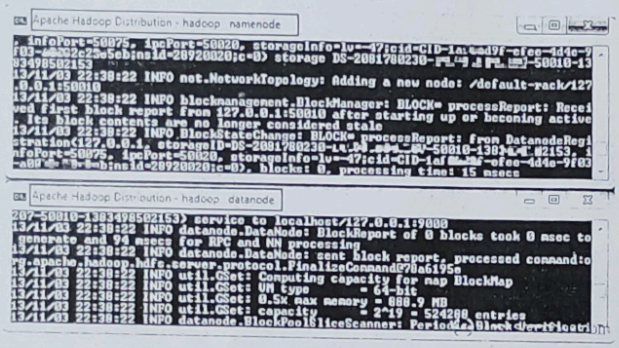
Run following commands.

Command Prompt

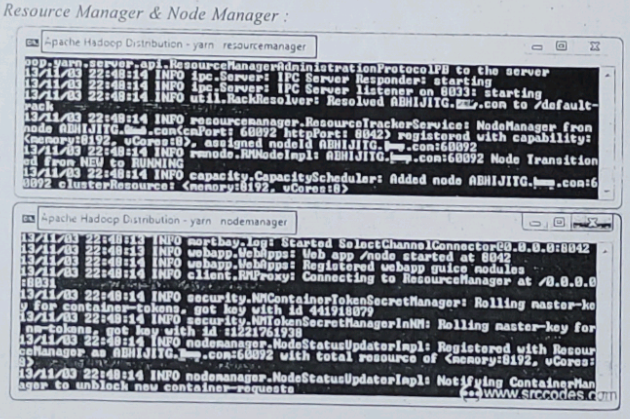
```
C:\Users\abhijitg>cd c:\hadoop
c:\hadoop>sbin\start-dfs
c:\hadoop>sbin\start-yarn
starting yarn daemons
```

Namenode, Datanode, Resource Manager and **Node Manager** will be started in few minutes and ready to execute Hadoop **MapReduce** job in the Single Node (pseudo-distributed mode) cluster.

Namenode & Datanode :



Resource Manager & Node Manager :



3. Run wordcount MapReduce job

   Now we'll run **wordcount** MapReduce job available in %HADOOP_HOME%\share\hadoop\mapreduce\hadoop-mapreduce-examples-2.2.0.jar

1. Create a text file with some content. We'll pass this file as input to the **wordcount** MapReduce job for counting words.

   C:\file1.txt

2. Install Hadoop

3. Run Hadoop Wordcount Mapreduce Example

4. Create a directory (say 'input') in HDFS to keep all the text files (say 'file1.txt') to be used for counting words.

5. C:\Users\abhijitg>cd c:\hadoop

   C:\hadoop>bin\hdfs dfs -mkdir input

6. Copy the text file(say 'file1.txt') from local disk to the newly created 'input' directory in HDFS.

   C:\hadoop>bin\hdfs dfs -copyFromLocal c:/file1.txt input

7. Check content of the copied file.

   C:\hadoop>hdfs dfs -ls input

   Found 1 items

   -rw-r--r-- 1 ABHIJITG supergroup 55 2014-02-03 13:19 input/file1.txt

   C:\hadoop>bin\hdfs dfs -cat input/file1.txt

   Install Hadoop

   Run Hadoop Wordcount Mapreduce Example

8. Run the **wordcount** MapReduce job provided in **%HADOOP_HOME%\share\hadoop\mapreduce\hadoop-mapreduce-examples-2.2.0.jar**

   C:\hadoop>bin\yarn jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.2.0.jar wordcount input output

   14/02/03 13:22:02 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032

   14/02/03 13:22:03 INFO input.FileInputFormat: Total input paths to process : 1

   14/02/03 13:22:03 INFO mapreduce.JobSubmitter: number of splits:1

:

:

14/02/03 13:22:04 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1391412385921_0002

14/02/03 13:22:04 INFO impl.YarnClientImpl: Submitted application application_1391412385921_0002 to ResourceManager at /0.0.0.0:8032

14/02/03 13:22:04 INFO mapreduce.Job: The url to track the job: http://ABHIJITG:8088/proxy/application_1391412385921_0002/

14/02/03 13:22:04 INFO mapreduce.Job: Running job: job_1391412385921_0002

14/02/03 13:22:14 INFO mapreduce.Job: Job job_1391412385921_0002 running in uber mode : false

14/02/03 13:22:14 INFO mapreduce.Job: map 0% reduce 0%

14/02/03 13:22:22 INFO mapreduce.Job: map 100% reduce 0%

14/02/03 13:22:30 INFO mapreduce.Job: map 100% reduce 100%

14/02/03 13:22:30 INFO mapreduce.Job: Job job_1391412385921_0002 completed successfully

14/02/03 13:22:31 INFO mapreduce.Job: Counters: 43

File System Counters

FILE: Number of bytes read=89

FILE: Number of bytes written=160142

FILE: Number of read operations=0

FILE: Number of large read operations=0

FILE: Number of write operations=0

HDFS: Number of bytes read=171

HDFS: Number of bytes written=59

HDFS: Number of read operations=6

HDFS: Number of large read operations=0

HDFS: Number of write operations=2

Job Counters

Launched map tasks=1

Launched reduce tasks=1

Data-local map tasks=1

Total time spent by all maps in occupied slots (ms)=5657

Total time spent by all reduces in occupied slots (ms)=6128

Map-Reduce Framework

Map input records=2

Map output records=7

Map output bytes=82

Map output materialized bytes=89

Input split bytes=116

Combine input records=7

Combine output records=6

Reduce input groups=6

Reduce shuffle bytes=89

Reduce input records=6

Reduce output records=6

Spilled Records=12

Shuffled Maps =1

Failed Shuffles=0

Merged Map outputs=1

GC time elapsed (ms)=145

CPU time spent (ms)=1418

Physical memory (bytes) snapshot=368246784

Virtual memory (bytes) snapshot=513716224

Total committed heap usage (bytes)=307757056

Shuffle Errors

BAD_ID=0

CONNECTION=0

IO_ERROR=0

WRONG_LENGTH=0

WRONG_MAP=0

WRONG_REDUCE=0

File Input Format Counters

Bytes Read=55

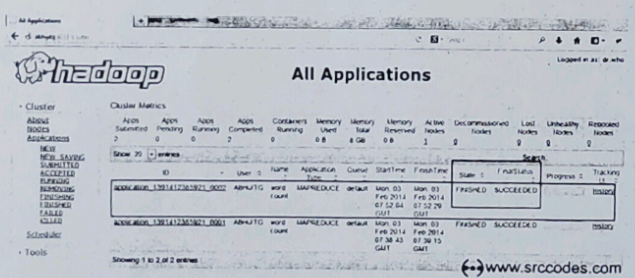
File Output Format Counters

Bytes Written=59

9. Check output.
10. C:\hadoop>bin\hdfs dfs -cat output/*
11. Example 1
12. Hadoop 2
13. Install 1
14. Mapreduce 1
15. Run 1

Wordcount 1

http://abhijitg:8088/cluster



Result: Thus the Installation of Hadoop single node cluster and word count program executed successfully.

Ex. No:8 Creating and Executing Your First Container Using Docker.

Aim:

To create and execute the container using Docker.

Prerequisites

Install Docker on your machine

For Ubuntu:

First, update your packages:

$ sudo apt update

Next, install docker with apt-get:

$ sudo apt install docker.io

Finally, verify that Docker is installed correctly:

$ sudo docker run hello-world

1. ***Create project***

In order to create first Docker application, create a folder on computer. It must contain the following two files:

- A '*main.py*' file (python file that will contain the code to be executed).
- A '*Dockerfile*' file (Docker file that will contain the necessary instructions to create the environment).

Normally the folder architecture is:

Dockerfile

main.py

0 directories, 2 files

2. ***Edit the Python file***

You can add the following code to the '*main.py*' file:

#!/usr/bin/env python3

print("Docker is magic!")

Nothing exceptional, but once you see "*Docker is magic!*" displayed in your terminal you will know that your Docker is working.

3. *Edit the Docker file*

The first step to take when you create a Docker file is to access the DockerHub website. This site contains many pre-designed images to save your time (for example: all images for linux or code languages).

```
# A dockerfile must always start by importing the base image.
# We use the keyword 'FROM' to do that.
# In our example, we want import the python image.
# So we write 'python' for the image name and 'latest' for the version.
FROM python:latest

# In order to launch our python code, we must import it into our image.
# We use the keyword 'COPY' to do that.
# The first parameter 'main.py' is the name of the file on the host.
# The second parameter '/' is the path where to put the file on the image.
# Here we put the file at the image root folder.
COPY main.py /

# We need to define the command to launch when we are going to run the image.
# We use the keyword 'CMD' to do that.
# The following command will execute "python ./main.py".
CMD [ "python", "./main.py" ]
```

4. *Create the Docker image*

Once your code is ready and the Dockerfile is written, all you have to do is create your image to contain your application.

```
$ docker build -t python-test .
```

The '*-t*' option allows you to define the name of your image. In our case we have chosen '*python-test*' but you can put what you want.

5. *Run the Docker image*

Once the image is created, your code is ready to be launched.

```
$ docker run python-test
```

You need to put the name of your image after '*docker run*'.

"Docker is magic!" displayed in your terminal.

Result: Thus the creation and Execution of Container program using Docker executed successfully.

Ex. No:9 Run a Container from Docker Hub

Aim

To run a Container from Docker Hub.

Procedures

Step-1: Verify Docker version and also login to Docker Hub

```
docker version
docker login
```

Step-2: Pull Image from Docker Hub

```
docker pull stacksimplify/dockerintro-springboot-helloworld-rest-api:1.0.0-RELEASE
```

Step-3: Run the downloaded Docker Image & Access the Application

- Copy the docker image name from Docker Hub
- docker run —name app1 -p 80:8080 -d stacksimplify/dockerintro-springboot-helloworld-rest-api:1.0.0-RELEASE
- http://localhost/hello

Step-4: List Running Containers

```
docker ps
docker ps -a
docker ps -a -q
```

Step-5: Connect to Container Terminal

```
docker exec -it <container-name> /bin/sh
```

Step-6: Container Stop, Start

```
docker stop <container-name>
docker start  <container-name>
```

Step-7: Remove Container

```
docker stop <container-name>
docker rm <container-name>
```

Step-8: Remove Image

```
docker images
docker rmi  <image-id>
```

Result : Thus the Container from Docker Hub executed successfully.