

**EX.NO:1**

## **Hadoop Installation and Configuration**

**AIM:**

### **PROCEDURE:**

**Step 1:** [Click here](#) to download the Java 8 Package. Save this file in your home .

**Step 2:** Extract the Java Tar File.

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

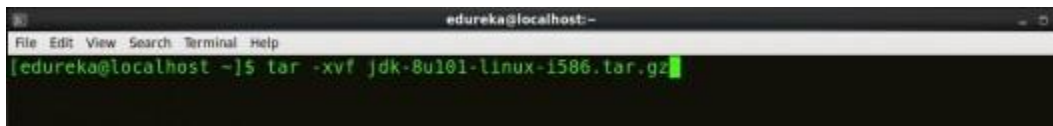
A terminal window titled 'edureka@localhost:~' with a menu bar (File, Edit, View, Search, Terminal, Help). The command '[edureka@localhost ~]\$ tar -xvf jdk-8u101-linux-i586.tar.gz' is entered and highlighted with a green cursor.

Fig: Hadoop Installation – Extracting Java Files

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

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

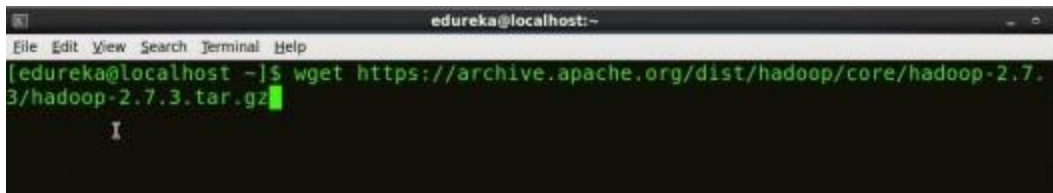
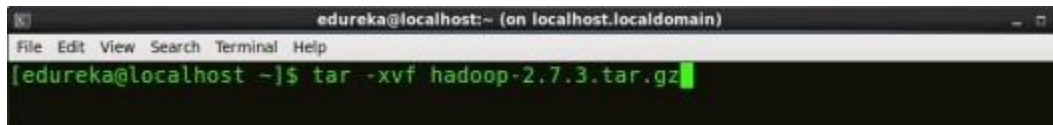
A terminal window titled 'edureka@localhost:~' with a menu bar (File, Edit, View, Search, Terminal, Help). The command '[edureka@localhost ~]\$ wget https://archive.apache.org/dist/hadoop/core/hadoop-2.7.3/hadoop-2.7.3.tar.gz' is entered and highlighted with a green cursor.

Fig: Hadoop Installation – Downloading Hadoop

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

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

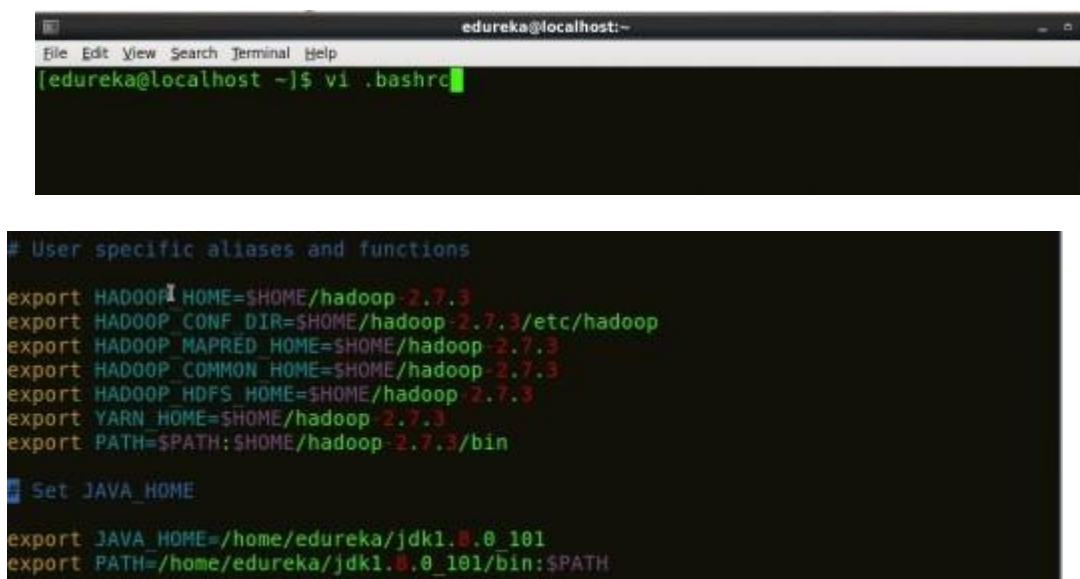


```
edureka@localhost:~ (on localhost.localdomain)
File Edit View Search Terminal Help
[edureka@localhost ~]$ tar -xvf hadoop-2.7.3.tar.gz
```

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



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

# User specific aliases and functions

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

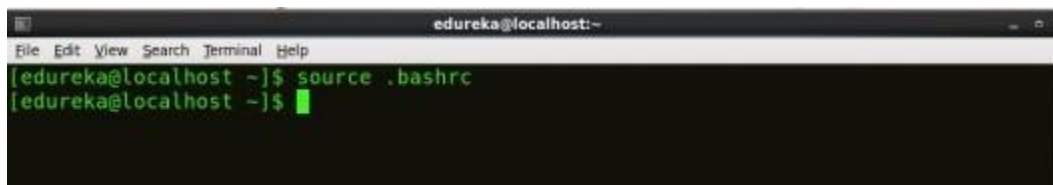
# Set JAVA_HOME
export JAVA_HOME=/home/edureka/jdk1.8.0_101
export PATH=/home/edureka/jdk1.8.0_101/bin:$PATH
```

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



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

Fig: Hadoop Installation – Refreshing environment variables

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

**Command:** `java -version`

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

Fig: Hadoop Installation – Checking Java Version

**Command:** `hadoop version`

```
edureka@localhost:~  
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**.

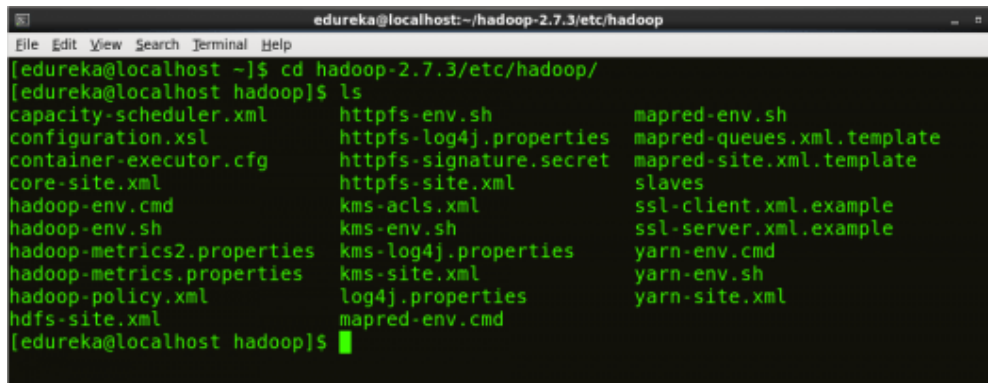
**Command:** `cd hadoop-2.7.3/etc/hadoop/`



**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:



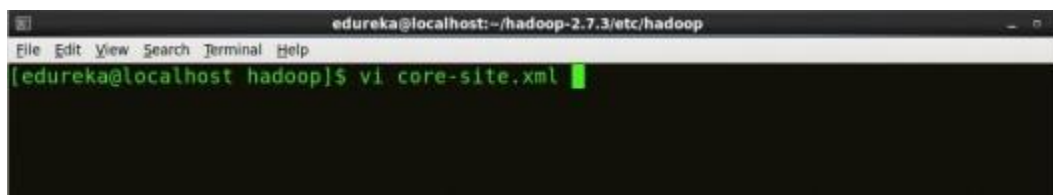
```
edureka@localhost: ~/hadoop-2.7.3/etc/hadoop
[edureka@localhost ~]$ cd /hadoop-2.7.3/etc/hadoop/
[edureka@localhost hadoop]$ ls
capacity-scheduler.xml      httpfs-env.sh              mapred-env.sh
configuration.xml           httpfs-log4j.properties   mapred-queues.xml.template
container-executor.cfg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

Fig: Hadoop Installation – Hadoop Configuration Files


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

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

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



```
edureka@localhost: ~/hadoop-2.7.3/etc/hadoop
[edureka@localhost hadoop]$ vi core-site.xml
```



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

Fig: Hadoop Installation – Configuring core-site.xml

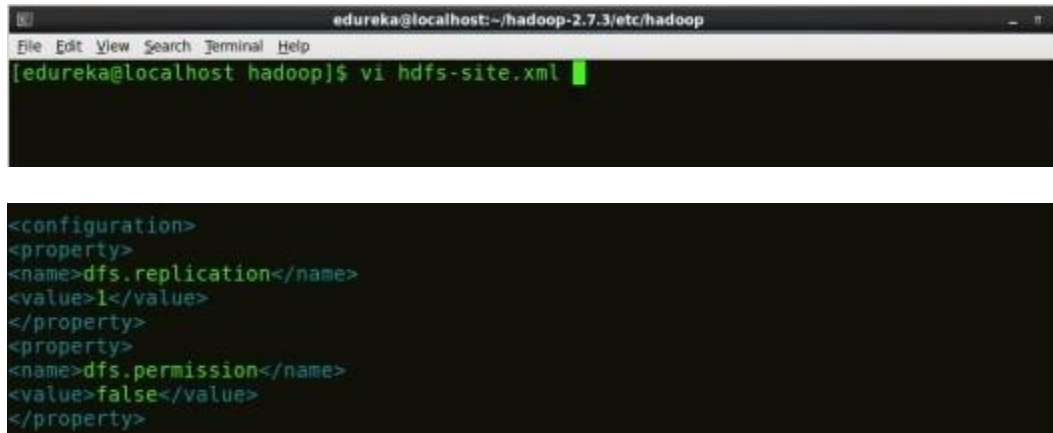
```
1
2      <?xml version="1.0" encoding="UTF-8"?>
3      <?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
4          <configuration>
5              <property>
6                  <name>fs.default.name</name>
7                  <value>hdfs://localhost:9000</value>
8                  </property>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

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

**configuration tag:**

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

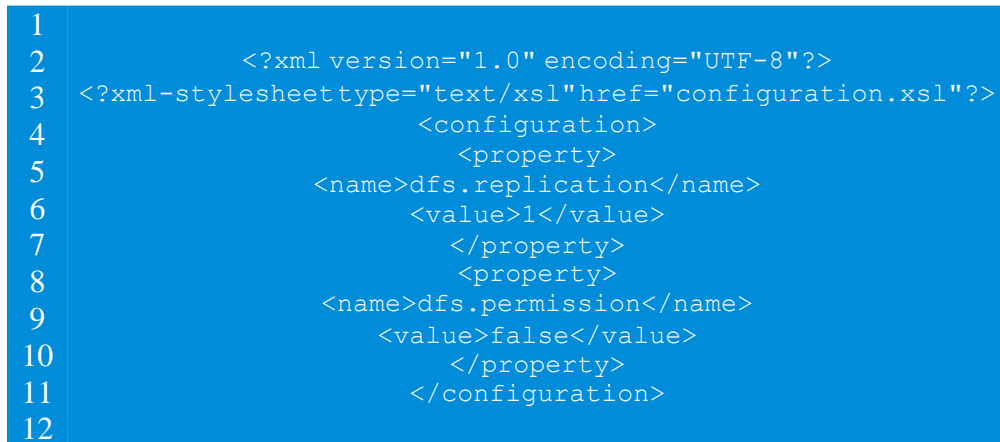
**Command:** vi hdfs-site.xml



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

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

Fig: Hadoop Installation – Configuring hdfs-site.xml



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

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

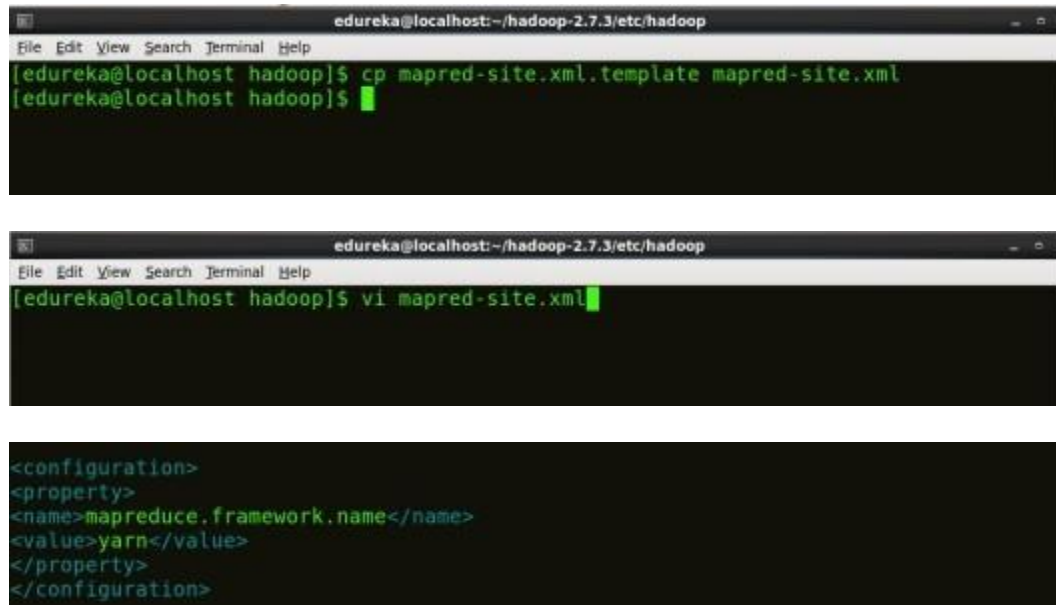
**inside configuration tag:**

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

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

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

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



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

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

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

Fig: Hadoop Installation – Configuring mapred-site.xml

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

**Step 10:** Edit *yarn-site.xml* and edit the property mentioned below inside configuration tag:

*yarn-site.xml* contains configuration settings of ResourceManager and NodeManager like application memory management size, the operation needed on program & algorithm, etc.

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

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

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

Fig: Hadoop Installation – Configuring yarn-site.xml

**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

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

Hadoop like Java home path, etc.

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

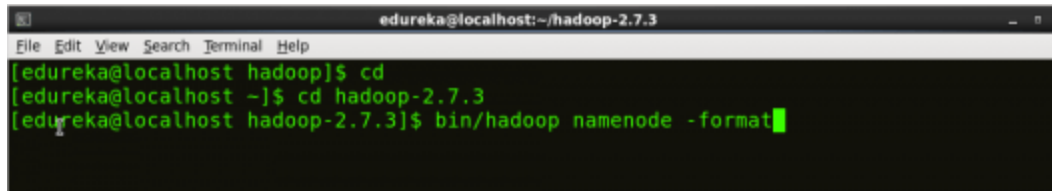
```
# The java implementation to use.
export JAVA_HOME=/home/edureka/jdk1.8.0_101
```

**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

A terminal window titled 'edureka@localhost:~/hadoop-2.7.3' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows three commands being executed: '[edureka@localhost hadoop]\$ cd', '[edureka@localhost ~]\$ cd hadoop-2.7.3', and '[edureka@localhost hadoop-2.7.3]\$ bin/hadoop namenode -format'. Each command is followed by a green cursor.

```
edureka@localhost:~/hadoop-2.7.3
File Edit View Search Terminal Help
[edureka@localhost hadoop]$ cd
[edureka@localhost ~]$ cd hadoop-2.7.3
[edureka@localhost hadoop-2.7.3]$ bin/hadoop namenode -format
```

Fig: Hadoop Installation – Formatting NameNode

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`

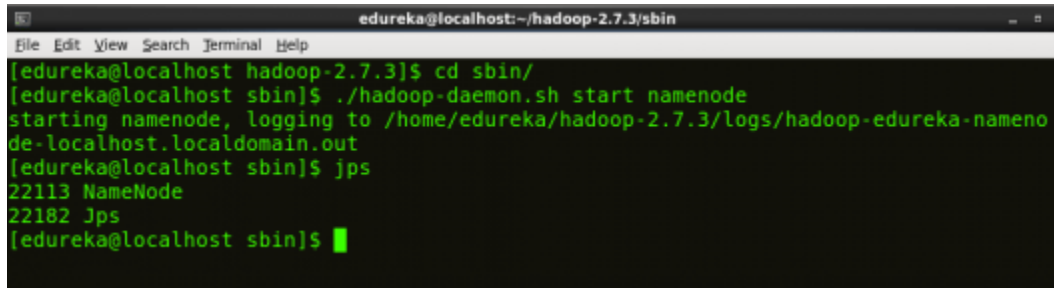
A terminal window titled 'edureka@localhost:~/hadoop-2.7.3/sbin' showing the execution of the command to start the Hadoop NameNode. The user navigates to the 'sbin' directory and runs './hadoop-daemon.sh start namenode'. The output shows the NameNode starting and logging to a specific file. Then, the user runs 'jps' which shows '22113 NameNode' and '22182 Jps'.

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`

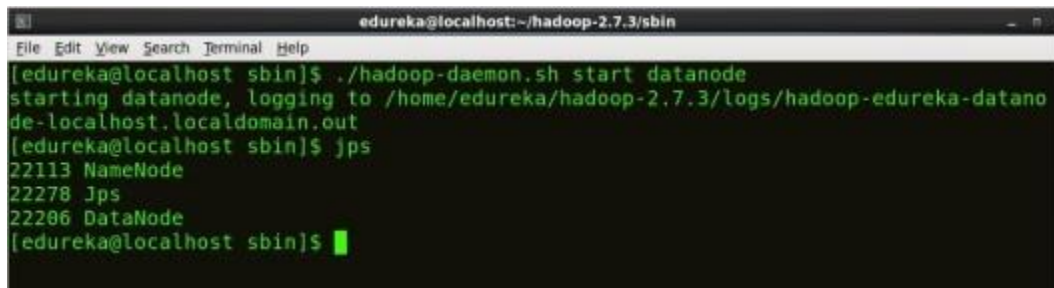
A terminal window titled 'edureka@localhost:~/hadoop-2.7.3/sbin' showing the execution of the command to start the Hadoop DataNode. The user runs './hadoop-daemon.sh start datanode'. The output shows the DataNode starting and logging to a specific file. Then, the user runs 'jps' which shows '22113 NameNode', '22278 Jps', and '22286 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`

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

Fig: Hadoop Installation – Starting ResourceManager

### Start NodeManager:

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

**Command:** `./yarn-daemon.sh start nodemanager`

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



[See Batch Details](#)

Fig: Hadoop Installation – Starting NodeManager

### Start JobHistoryServer:

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

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

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

**Command:** jps

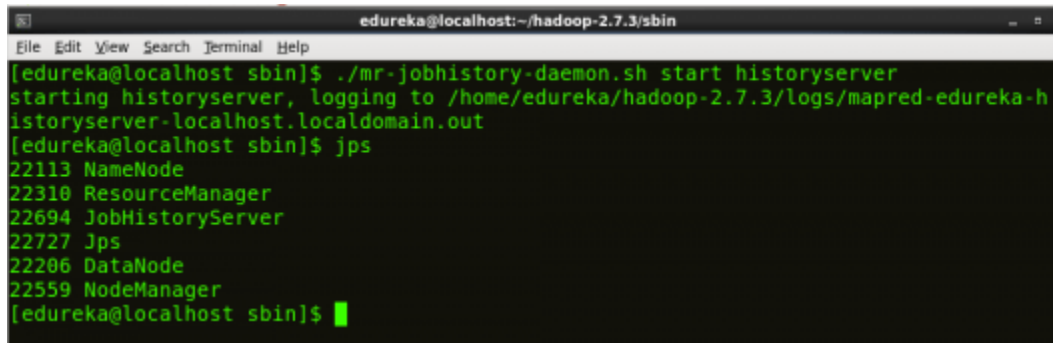
A terminal window titled 'edureka@localhost: ~/hadoop-2.7.3/sbin' shows the execution of the 'jps' command. The output lists several processes: '22113 NameNode', '22310 ResourceManager', '22694 JobHistoryServer', '22727 Jps', '22206 DataNode', and '22559 NodeManager'. The prompt returns to '[edureka@localhost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

Fig: Hadoop Installation – Checking Daemons

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

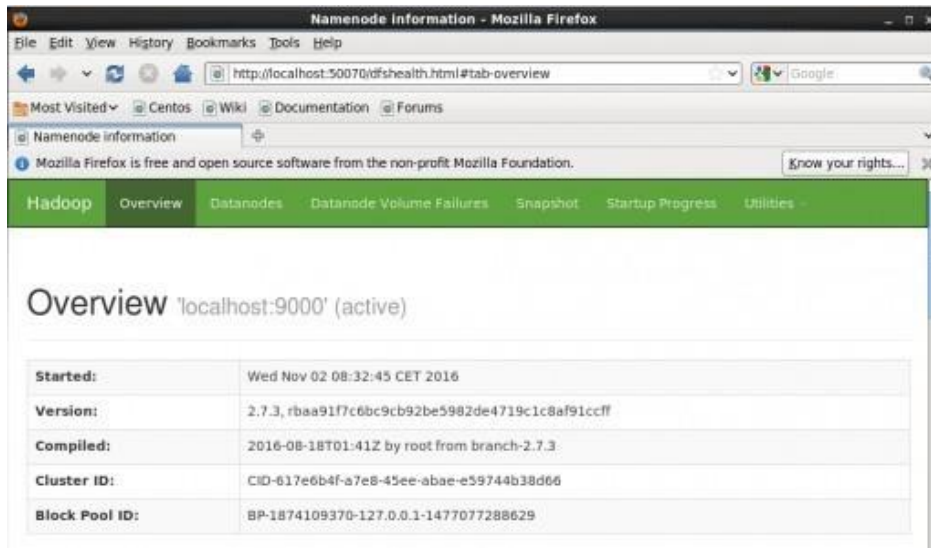


Fig: Hadoop Installation – Starting WebUI

Congratulations, you have successfully installed a single node Hadoop cluster

**RESULT:**

**EX NO: 2**

## **FILE MANAGEMENT IN HADOOP**

**AIM:**

### **PROCEDURE:**

Hint: A typical Hadoop workflow creates data files (such as log files) elsewhere and copies them into HDFS using one of the above command line utilities.

### **RESOURCES:**

VMWare stack, 4 GB RAM, Hard Disk 80 GB.

### **PROGRAM LOGIC:**

#### **Adding Files and Directories to HDFS**

Before you can run Hadoop programs on data stored in HDFS, you will need to put the data into HDFS first. Let us create a directory and put a file in it. HDFS has a default working directory of /user/\$USER, where \$USER is your login user name. This directory is not automatically created for you, though, so let us create it with the mkdir command. For the purpose of illustration, we use chuck. You should substitute your user name in the example commands.

```
hadoop fs -mkdir /user/chuck
```

```
hadoop fs -put
```

```
hadoop fs -put example.txt /user/chuck
```

#### **Retrieving Files from HDFS**

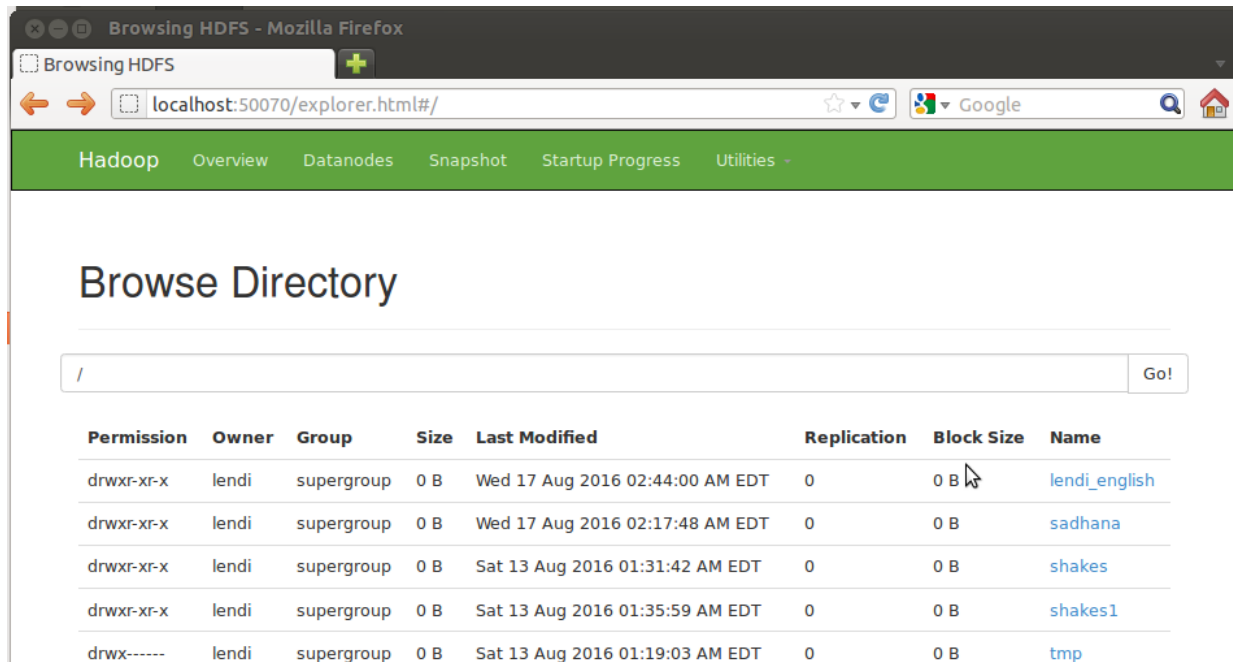
The Hadoop command get copies files from HDFS back to the local filesystem. To retrieve example.txt, we can run the following command:

```
hadoop fs -cat example.txt
```

#### **Deleting files from HDFS**

```
hadoop fs -rm example.txt
```

## OUTPUT:



The screenshot displays the Hadoop Distributed File System (HDFS) web interface. The browser window is titled "Browsing HDFS - Mozilla Firefox" and shows the URL "localhost:50070/explorer.html#/" in the address bar. A green navigation bar contains links for "Hadoop", "Overview", "Datanodes", "Snapshot", "Startup Progress", and "Utilities". The main content area is titled "Browse Directory" and features a search bar with the "/" symbol and a "Go!" button. Below the search bar is a table listing files in the directory.

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	lendi	supergroup	0 B	Wed 17 Aug 2016 02:44:00 AM EDT	0	0 B	<a href="#">lendi_english</a>
drwxr-xr-x	lendi	supergroup	0 B	Wed 17 Aug 2016 02:17:48 AM EDT	0	0 B	<a href="#">sadhana</a>
drwxr-xr-x	lendi	supergroup	0 B	Sat 13 Aug 2016 01:31:42 AM EDT	0	0 B	<a href="#">shakes</a>
drwxr-xr-x	lendi	supergroup	0 B	Sat 13 Aug 2016 01:35:59 AM EDT	0	0 B	<a href="#">shakes1</a>
drwx-----	lendi	supergroup	0 B	Sat 13 Aug 2016 01:19:03 AM EDT	0	0 B	<a href="#">tmp</a>

## RESULT:

**EX NO: 3**

## **A WORD COUNT PROGRAM USING MAP-REDUCE TASKS**

**AIM:**

**PROCEDURE:**

### **STEP: 1**

Download Hadoop-core-1.2.1.jar, which is used to compile and execute the MapReduce program. Visit the following link <http://mvnrepository.com/artifact/org.apache.hadoop/hadoop-core/1.2.1> to download the jar. Let us assume the downloaded folder is /home/hadoop/.

### **STEP: 2**

The following commands are used for compiling the **WordCount.java** program.

```
javac -classpath hadoop-core-1.2.1.jar -d . WordCount.java
```

### **STEP: 3**

Create a jar for the program.

```
jar -cvf sample1.jar sample1/
```

### **STEP: 4**

```
cd $HADOOP_PREFIX
bin/hadoop namenode -format
sbin/start-dfs.sh
sbin/start-yarn.sh
jps
```

### **STEP: 5**

The following command is used to create an input directory in HDFS.

```
bin/hdfs dfs -mkdir /input
```

### **STEP: 6**

The following command is used to copy the input file named **sal.txt** in the input directory of HDFS.

```
bin/hdfs dfs -put /home/it08/Downloads/sal.txt /input
```

### **STEP: 7**

The following command is used to run the application by taking the input files from the input directory.

```
bin/hadoop jar /home/it08/Downloads/sample1.jar sample1.WordCount /input /output
```

## PROGRAM:

### WordCount.java

```
package sample;

import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {

    public static class TokenizerMapper
        extends Mapper<Object, Text, Text, IntWritable>{

        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();
        public void map(Object key, Text value, Context context
            ) throws IOException, InterruptedException {
            StringTokenizer itr = new StringTokenizer(value.toString());
            while (itr.hasMoreTokens()) {
                word.set(itr.nextToken());
                context.write(word, one);
            }
        }
    }

    public static class IntSumReducer
        extends Reducer<Text,IntWritable,Text,IntWritable> {
        private IntWritable result = new IntWritable();
        public void reduce(Text key, Iterable<IntWritable> values,
```

Context context

```
        ) throws IOException, InterruptedException {
    int sum = 0;
    for (IntWritable val : values) {
        sum += val.get();
    }
    result.set(sum);
    context.write(key, result);
}
}
```

```
public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
}
}
```

**INPUT FILE: (Sample.txt)**

```
1949 12 12 34 23 45 34 12 23
1987 13 11 32 34 45 56 12 34
1997 12 12 12 12 12 11 34 12
1998 23 34 23 34 45 56 23 34
2000 10 11 12 23 14 13 15 16
```



OUTPUT:

Search for groups, artifacts, categories

Search

Home » org.apache.hadoop » hadoop-core » 1.2.1



Hadoop Core » 1.2.1

Hadoop Core

License	Apache
Categories	Distributed Computing
Date	(Jul 24, 2013)
Repository	central
Usages	395

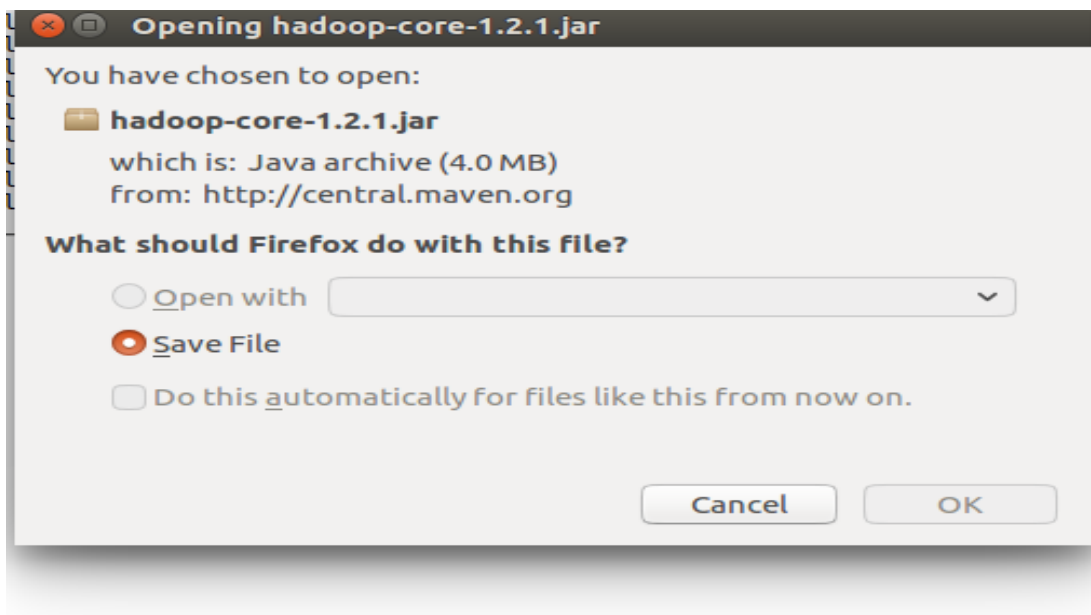
central.maven.org/maven2/org/apache/hadoop/hadoop-core/1.2.1/

Search

Index of /maven2/org/apache/hadoop/hadoop-core/1.2.1/

./		
<a href="#">hadoop-core-1.2.1.jar</a>	24-Jul-2013 20:41	4203713
<a href="#">hadoop-core-1.2.1.jar.asc</a>	24-Jul-2013 20:41	827
<a href="#">hadoop-core-1.2.1.jar.asc.md5</a>	24-Jul-2013 20:41	32
<a href="#">hadoop-core-1.2.1.jar.asc.sha1</a>	24-Jul-2013 20:41	40
<a href="#">hadoop-core-1.2.1.jar.md5</a>	24-Jul-2013 20:41	32
<a href="#">hadoop-core-1.2.1.jar.sha1</a>	24-Jul-2013 20:41	40
<a href="#">hadoop-core-1.2.1.pom</a>	24-Jul-2013 20:41	4658
<a href="#">hadoop-core-1.2.1.pom.asc</a>	24-Jul-2013 20:41	827
<a href="#">hadoop-core-1.2.1.pom.asc.md5</a>	24-Jul-2013 20:41	32
<a href="#">hadoop-core-1.2.1.pom.asc.sha1</a>	24-Jul-2013 20:41	40
<a href="#">hadoop-core-1.2.1.pom.md5</a>	24-Jul-2013 20:41	32
<a href="#">hadoop-core-1.2.1.pom.sha1</a>	24-Jul-2013 20:41	40

```
it53@it53-HP-280-G1-MT: ~/Downloads
it53@it53-HP-280-G1-MT:~$ cd /home/it53/Downloads
it53@it53-HP-280-G1-MT:~/Downloads$ mkdir mapreduce
mkdir: cannot create directory 'mapreduce': File exists
it53@it53-HP-280-G1-MT:~/Downloads$ javac -classpath hadoop-core-1.2.1.jar ./map
red/WordCount.java
it53@it53-HP-280-G1-MT:~/Downloads$
```



```
it53@it53-HP-280-G1-MT: ~/Downloads
it53@it53-HP-280-G1-MT:~$ cd /home/it53/Downloads
it53@it53-HP-280-G1-MT:~/Downloads$ mkdir mapreduce
mkdir: cannot create directory 'mapreduce': File exists
it53@it53-HP-280-G1-MT:~/Downloads$ javac -classpath hadoop-core-1.2.1.jar ./map
red/WordCount.java
it53@it53-HP-280-G1-MT:~/Downloads$ jar -cvf mapred.jar -C mapred/ .
added manifest
adding: sample/(in = 0) (out= 0)(stored 0%)
adding: sample/WordCount.class(in = 1512) (out= 818)(deflated 45%)
adding: sample/WordCount$IntSumReducer.class(in = 1753) (out= 745)(deflated 57%)
adding: sample/WordCount$TokenizerMapper.class(in = 1750) (out= 759)(deflated 56
%)
adding: WordCount.class(in = 1512) (out= 818)(deflated 45%)
adding: WordCount.java(in = 2105) (out= 713)(deflated 66%)
adding: WordCount$IntSumReducer.class(in = 1753) (out= 745)(deflated 57%)
adding: WordCount$TokenizerMapper.class(in = 1750) (out= 759)(deflated 56%)
it53@it53-HP-280-G1-MT:~/Downloads$
```

```
it53@it53-HP-280-G1-MT: ~/Downloads/hadoop-2.7.0
377 mkdir mapreduce
378 mkdir mapred
379 cd /home/it53/Downloads
380 mkdir WordCount
381 source /etc/profile
382 cd HADOOP_PREFIX
383 cd HADOOP_PREFIX
384 cd $HADOOP_PREFIX
385 bin/hadoop namenode -format
386 sbin/start-all.sh
387 jps
388 cd
389 sudo dpkg -i cdh5-repository-1.0-all.deb
390 http://archive.cloudera.com/cdh5/one-click-install/trusty/amd64/cdh5-repo
sitory_1.0_all.deb
391 cd /home/it53/Downloads
392 mkdir mapreduce
393 javac -classpath hadoop-core-1.2.1.jar ./mapred/WordCount.java
394 jar -cvf mapred.jar -C mapred/ .
395 bin/hadoop namenode -format
396 sbin/start-dfs.sh
397 history
it53@it53-HP-280-G1-MT:~/Downloads/hadoop-2.7.0$ bin/hdfs dfs -mkdir /input
it53@it53-HP-280-G1-MT:~/Downloads/hadoop-2.7.0$
```

```

it53@it53-HP-280-G1-MT: ~/Downloads/hadoop-2.7.0
3/Downloads/input/sample.txt /input
it53@it53-HP-280-G1-MT:~/Downloads/hadoop-2.7.0$ bin/hadoop jar /home/it53/Downloads/mapred.jar sample.WordCount /input /output7
16/08/25 09:32:07 INFO client.RMPProxy: Connecting to ResourceManager at /0.0.0.0:8032
16/08/25 09:32:07 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
16/08/25 09:32:08 INFO input.FileInputFormat: Total input paths to process : 1
16/08/25 09:32:08 INFO mapreduce.JobSubmitter: number of splits:1
16/08/25 09:32:08 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1472097423643_0001
16/08/25 09:32:08 INFO impl.YarnClientImpl: Submitted application application_1472097423643_0001
16/08/25 09:32:08 INFO mapreduce.Job: The url to track the job: http://it53-HP-280-G1-MT:8088/proxy/application_1472097423643_0001/
16/08/25 09:32:08 INFO mapreduce.Job: Running job: job_1472097423643_0001
16/08/25 09:32:13 INFO mapreduce.Job: Job job_1472097423643_0001 running in uber mode : false
16/08/25 09:32:13 INFO mapreduce.Job:  map 0% reduce 0%
16/08/25 09:32:16 INFO mapreduce.Job:  map 100% reduce 0%
16/08/25 09:32:21 INFO mapreduce.Job:  map 100% reduce 100%
16/08/25 09:32:21 INFO mapreduce.Job: Job job_1472097423643_0001 completed successfully
16/08/25 09:32:21 INFO mapreduce.Job: Counters: 49
    File System Counters
        FILE: Number of bytes read=169
        FILE: Number of bytes written=229117
        FILE: Number of read operations=0
        FILE: Number of large read operations=0
        FILE: Number of write operations=0
        HDFS: Number of bytes read=250
        HDFS: Number of bytes written=96
        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)=1357

```

```

it53@it53-HP-280-G1-MT: ~/Downloads/hadoop-2.7.0
379 cd /home/it53/Downloads
380 mkdir WordCount
381 source /etc/profile
382 cd HADOOP_PREFIX
383 cd HADOOP_PREFIX
384 cd $HADOOP_PREFIX
385 bin/hadoop namenode -format
386 sbin/start-all.sh
387 jps
388 cd
389 sudo dpkg -i cdh5-repository-1.0-all.deb
390 http://archive.cloudera.com/cdh5/one-click-install/trusty/amd64/cdh5-repository_1.0_all.deb
391 cd /home/it53/Downloads
392 mkdir mapreduce
393 javac -classpath hadoop-core-1.2.1.jar ./mapred/WordCount.java
394 jar -cvf mapred.jar -C mapred/ .
395 bin/hadoop namenode -format
396 sbin/start-dfs.sh
397 history
it53@it53-HP-280-G1-MT:~/Downloads/hadoop-2.7.0$ bin/hdfs dfs -mkdir /input
it53@it53-HP-280-G1-MT:~/Downloads/hadoop-2.7.0$ bin/hdfs dfs -put /home/it53/Downloads/input/sample.txt /input
it53@it53-HP-280-G1-MT:~/Downloads/hadoop-2.7.0$

```

Hadoop Overview Datanodes Snapshot Startup Progress Utilities

# Browse Directory

Permission	Owner	Group
-rw-r--r--	it53	sup
-rw-r--r--	it53	sup

Hadoop, 2014.

File information - part-r-00000

[Download](#)

Block information -- Block 0

Block ID: 1073741832

Block Pool ID: BP-981847713-127.0.1.1-1472097390460

Generation Stamp: 1008

Size: 96

Availability:

- it53-HP-280-G1-MT

Close

Block Size	Name
128 MB	<a href="#">_SUCCESS</a>
128 MB	<a href="#">part-r-00000</a>

localhost:50070/explorer.html#/output7

Hadoop Overview Datanodes Snapshot Startup Progress Utilities

# Browse Directory

Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	it53	supergroup	0 B	25/8/2016, 9:32:20 AM	1	128 MB	<a href="#">_SUCCESS</a>
-rw-r--r--	it53	supergroup	96 B	25/8/2016, 9:32:19 AM	1	128 MB	<a href="#">part-r-00000</a>

Hadoop, 2014.

Block Size

Name

MB

[sample.txt](#)



The screenshot shows a gedit text editor window titled "part-r-00000(2) (~/.Downloads) - gedit". The window has a menu bar with "Open" and "Save" options. The main text area contains a list of numbers and their corresponding values:

10	1
11	3
12	11
13	2
14	1
15	1
16	1
1949	1
1987	1
1997	1
1998	1
2000	1
23	6
32	1
34	8
45	3
56	2

The status bar at the bottom indicates "Plain Text", "Tab Width: 8", "Ln 1, Col 1", and "INS".

**RESULT:**

**EX.No:4**

## **Install Virtual box/VMware Workstation**

**AIM:**

**PROCEDURE:**

Step 1- Download Link

Link for downloading the software is <https://www.vmware.com/products/workstation-pro/workstation-pro-evaluation.html>. Download the software for windows. Good thing is that there is no sign up process. Click and download begins. Software is around 541 MB.

Step 2- Download the installer file

It should probably be in the download folder by default, if you have not changed the settings in your browser. File name should be something like VMware-workstation-full-15.5.1-15018445.exe. This file name can change depending on the version of the software currently available for download. But for now, till the next version is available, they will all be VMware Workstation 15 Pro.

Step 3- Locate the downloaded installer file

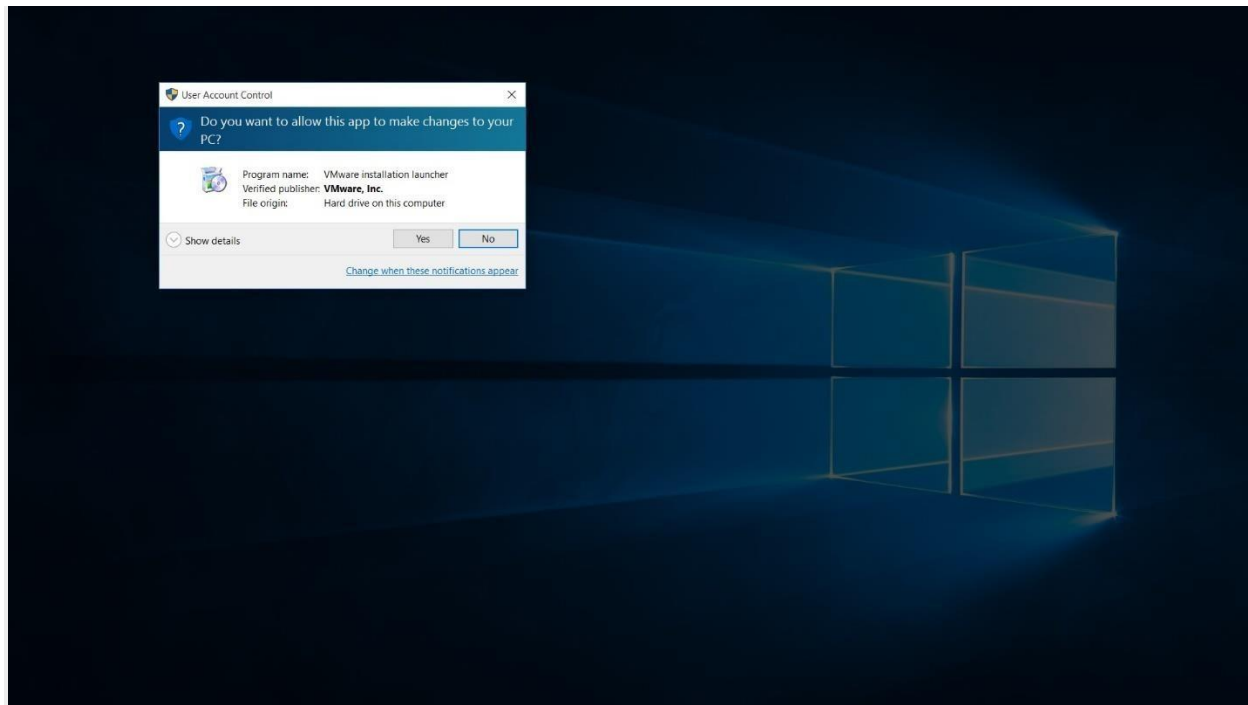
For demonstration purpose, I have placed the downloaded installer on my desktop. Find the installer on your system and double click to launch the application.



VMware workstation 15 pro for windows 10 installer file screenshot.

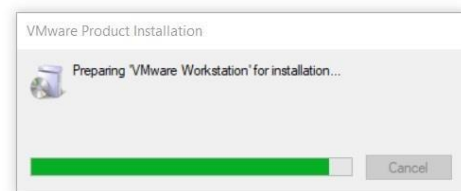
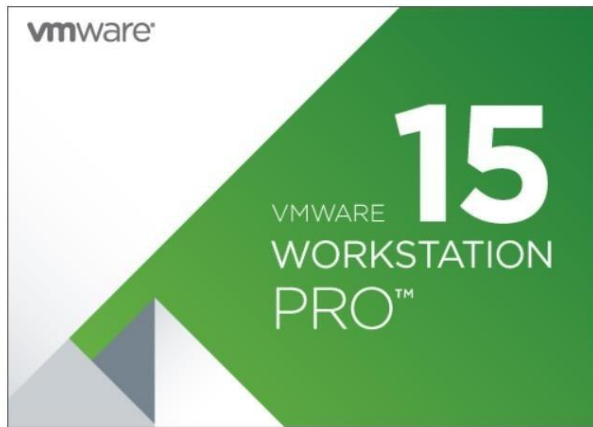
#### Step 4- User Access Control (UAC) Warning

Now you should see User Access Control (UAC) dialog box. Click yes to continue.



VMware Workstation 12 Pro installer windows 10 UAC screenshot

Initial Splash screen will appear. Wait for the process to complete.



VMware Workstation 15 Installation Splash Screen

#### Step 5- VMware Workstation Setup wizard

Now you will see VMware Workstation setup wizard dialog box. Click next to continue.



#### Step 6- End User Licence Agreement

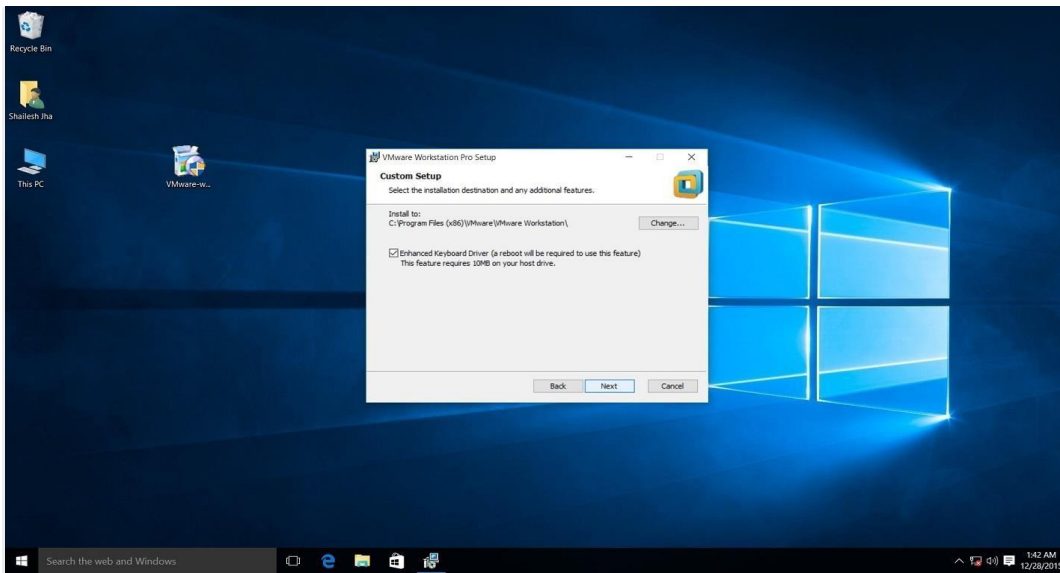


This time you should see End User Licence Agreement dialog box. Check “I accept the terms in the Licence Agreement” box and press next to continue.



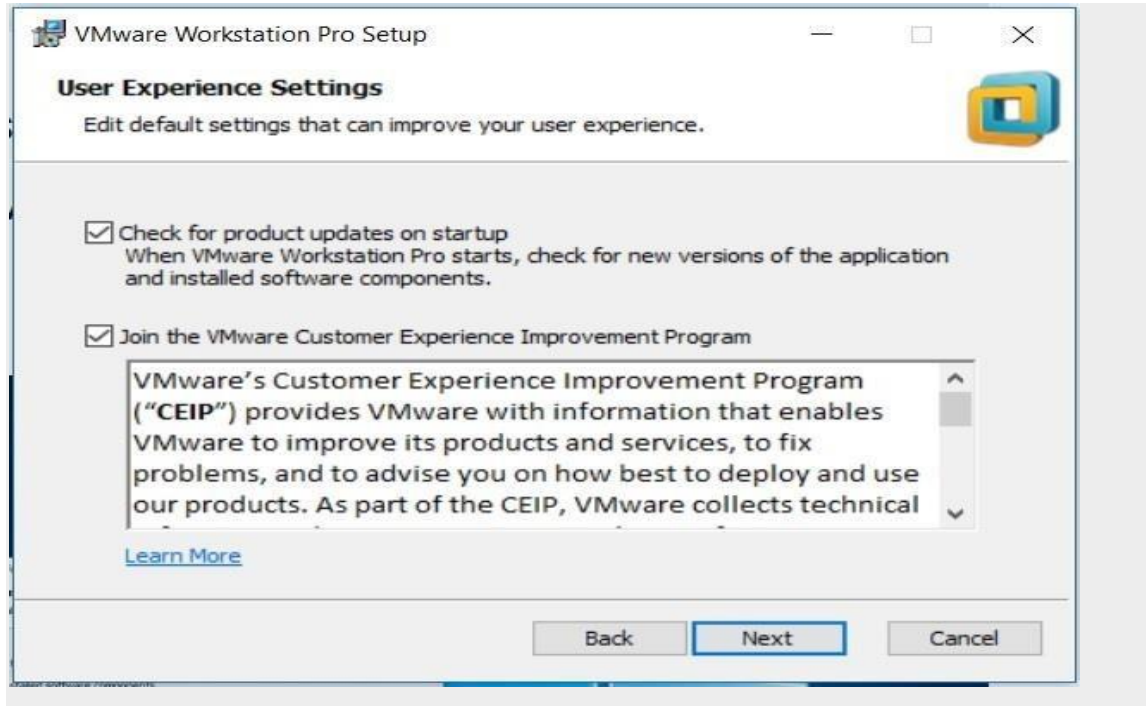
#### Step 7- Custom Setup options

Select the folder in which you would like to install the application. There is no harm in leaving the defaults as it is. Also select Enhanced Keyboard Driver check box.



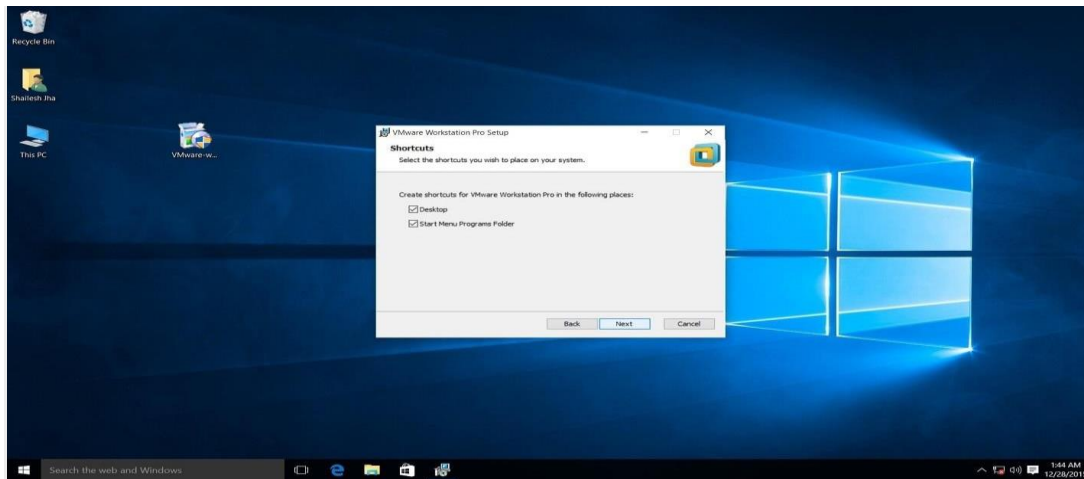
## Step 8- User Experience Settings

Next you are asked to select “Check for Updates” and “Help improve VMware Workstation Pro”. Do as you wish. I normally leave it to defaults that is unchecked.



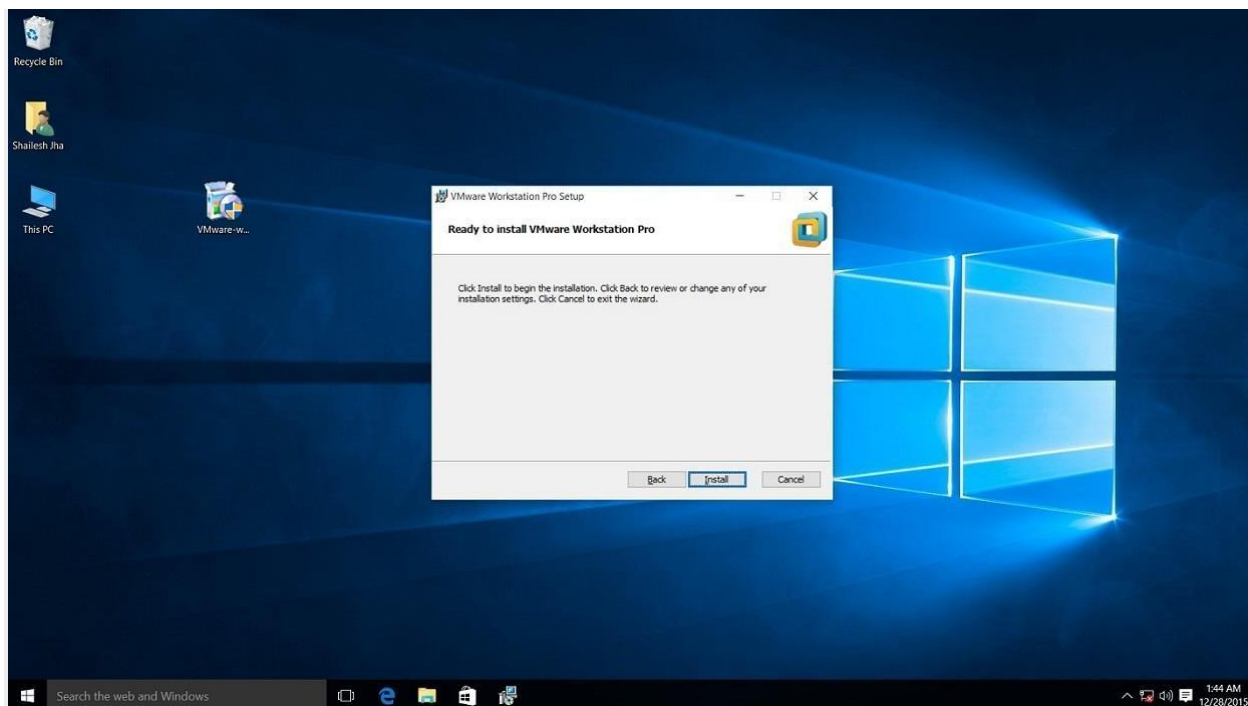
### Step 9- Application Shortcuts preference

Next step is to select the place you want the shortcut icons to be placed on your system to launch the application. Please select both the options, desktop and start menu and click next.



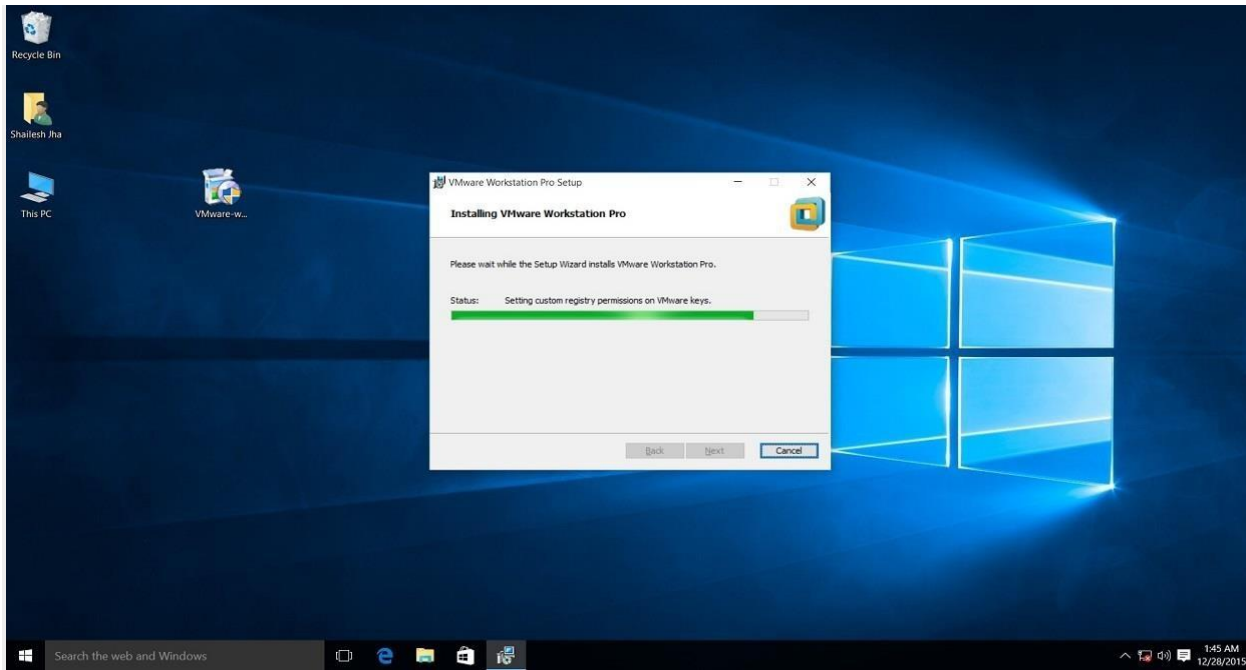
### Step 10- Installation begins

Now you see the begin installation dialog box. Click install to start the installation process.



Screenshot for VMware Workstation 15 pro installation begin confirmation dialog box on windows 10.

Below screenshot shows Installation in progress. Wait for this to complete.

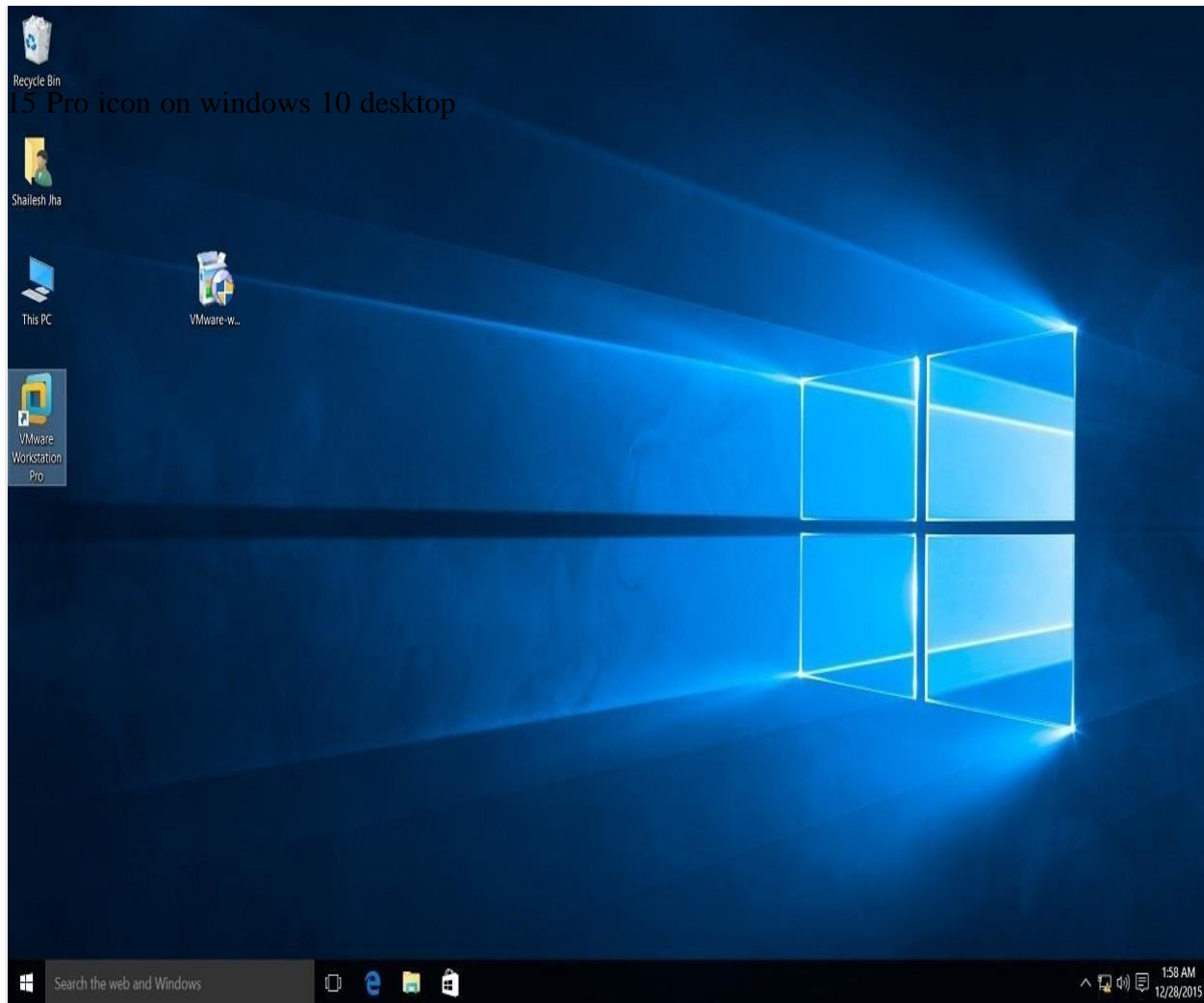


At the end you will see installation complete dialog box. Click finish and you are done with the installation process. You may be asked to restart your computer. Click on Yes to restart.



After the installation completes, you should see VMware Workstation icon on the desktop. Double click on it to launch the application.

## OUTPUT:



## RESULT:

**EX.NO : 5****Installing and Running the Google App Engine****AIM:****PROCEDURE:**

The App Engine SDK allows you to run Google App Engine Applications on your local computer. It simulates the run-time environment of the Google App Engine infrastructure.

**Pre-Requisites: Python 2.5.4**

If you don't already have Python 2.5.4 installed in your computer, download and install Python 2.5.4 from:

<http://www.python.org/download/releases/2.5.4/>

**Download and Install**

You can download the Google App Engine SDK by going to:

<http://code.google.com/appengine/downloads.html>

**Download the Google App Engine SDK**

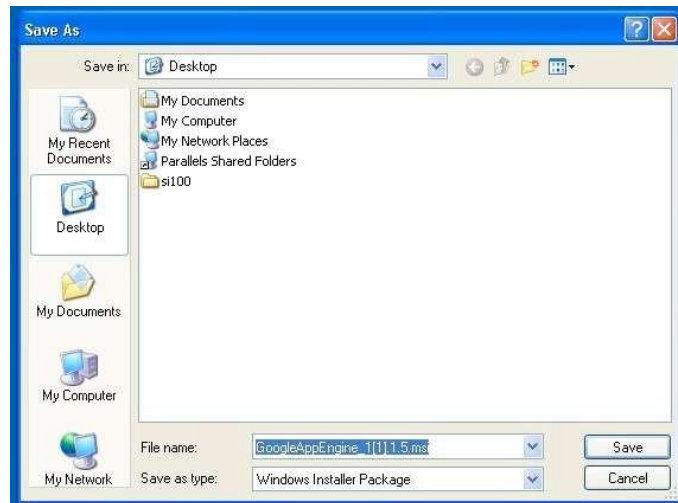
Before downloading, please read the [Terms](#) that govern your use of the App Engine SDK.

Please note: The App Engine SDK is under **active development**, please keep this in mind as you explore its capabilities. See the [SDK Release Notes](#) for the information on the most recent changes to the App Engine SDK. If you discover any issues, please feel free to notify us via our [Issue Tracker](#).

Platform	Version	Package	Size	SHA1 Checksum
Windows	1.1.5 - 10/03/08	<a href="#">GoogleAppEngine_1.1.5.msi</a>	2.5 MB	e974312b4aefc0b3873ff0d93eb4c525d5e88c30
Mac OS X	1.1.5 - 10/03/08	<a href="#">GoogleAppEngineLauncher-1.1.5.dmg</a>	3.6 MB	f62208ac01c1b3e39796e58100d5f1b2f052d3e7
Linux/Other Platforms	1.1.5 - 10/03/08	<a href="#">google_appengine_1.1.5.zip</a>	2.6 MB	cbb9ce817bdabf1c4f181d9544864e55ee253de1

Download the Windows installer – the simplest thing is to download it to your Desktop or another folder that you remember.





Double Click on the **GoogleApplicationEngine** installer.



Click through the installation wizard, and it should install the App Engine. If you don't have Python 2.5, it will install Python 2.5 as well.

Once the install is complete you can discard the downloaded installer





## 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](http://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.

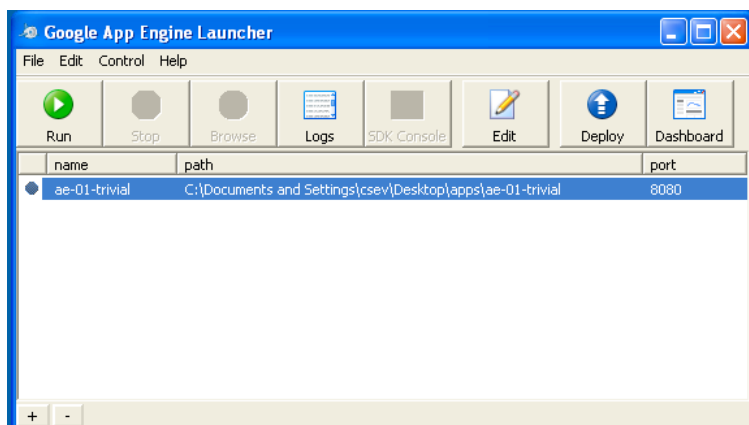
Then create a file in the **ae--01--trivial** folder called **index.py** with three lines in it:

```
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.

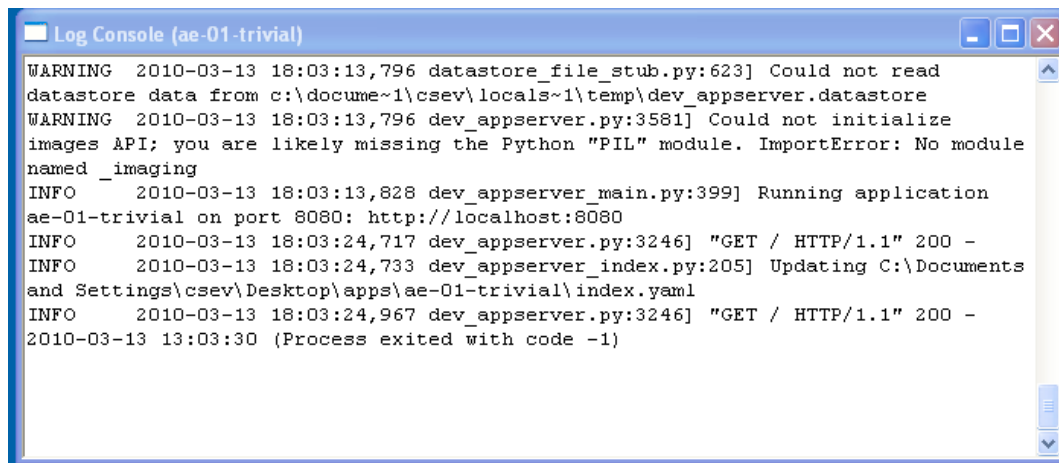
Then press **Browse** to open a browser pointing at your application which is running at **http://localhost:8080/**

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



## Watching the Log

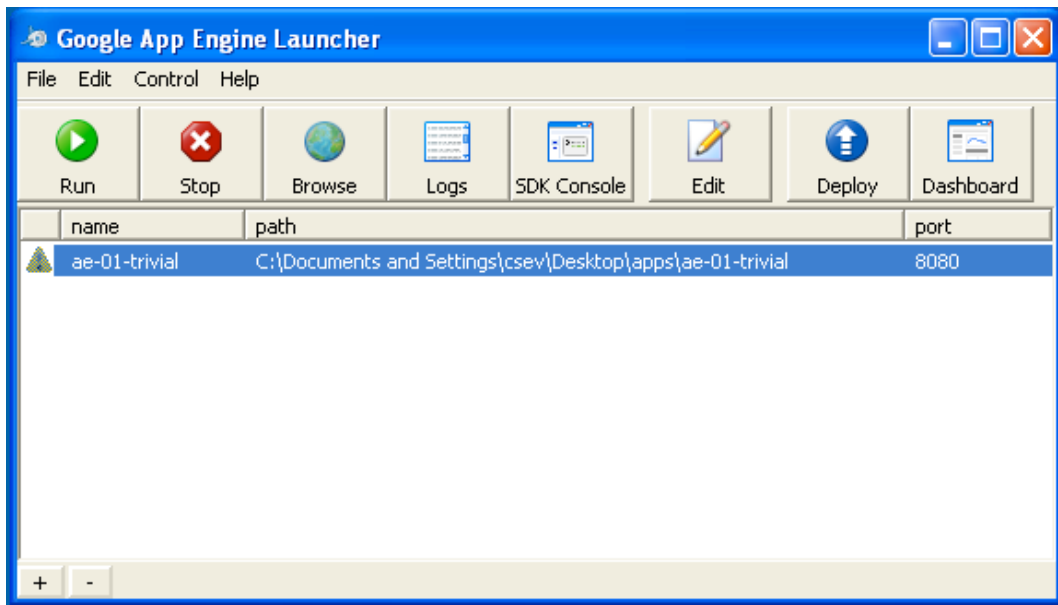
You can watch the internal log of the actions that the web server is performing when you are interacting with your application in the browser. Select your application in the Launcher and press the **Logs** button to bring up a log window:



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:

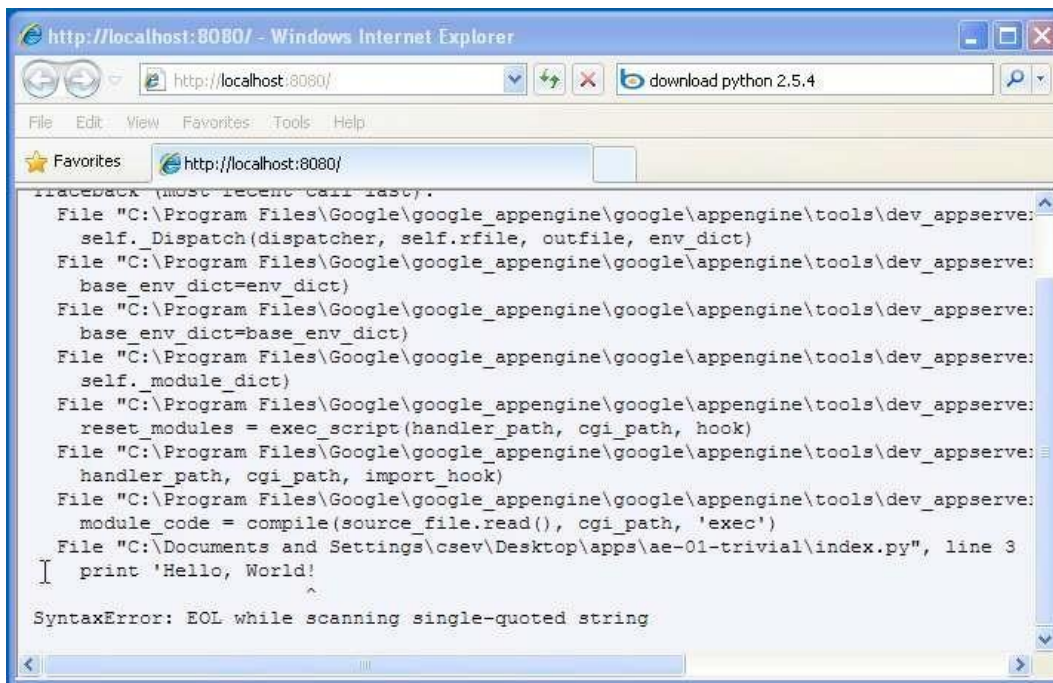


To get more detail on what is going wrong, take a look at the log for the application:



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](http://en.wikipedia.org/wiki/Stack_trace)

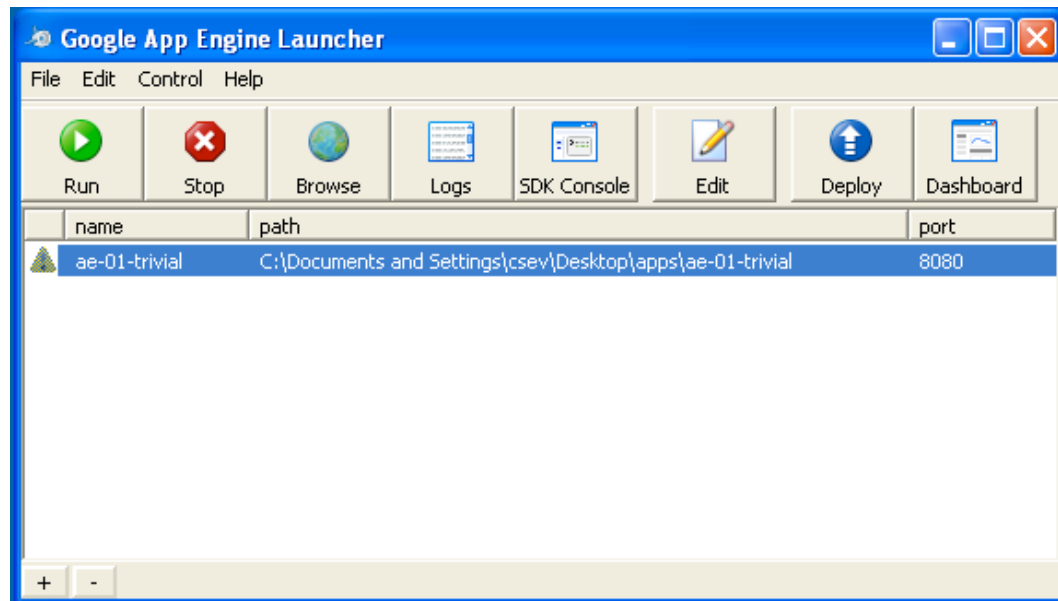
When you make a mistake in the **app.yaml** file – you must fix the mistake and attempt to start the application again.

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.

## OUTPUT:



## RESULT:

**EX NO: 6                      INSTALL A C COMPILER IN THE VIRTUAL MACHINE AND  
EXECUTE A SAMPLE PROGRAM**

**AIM:**

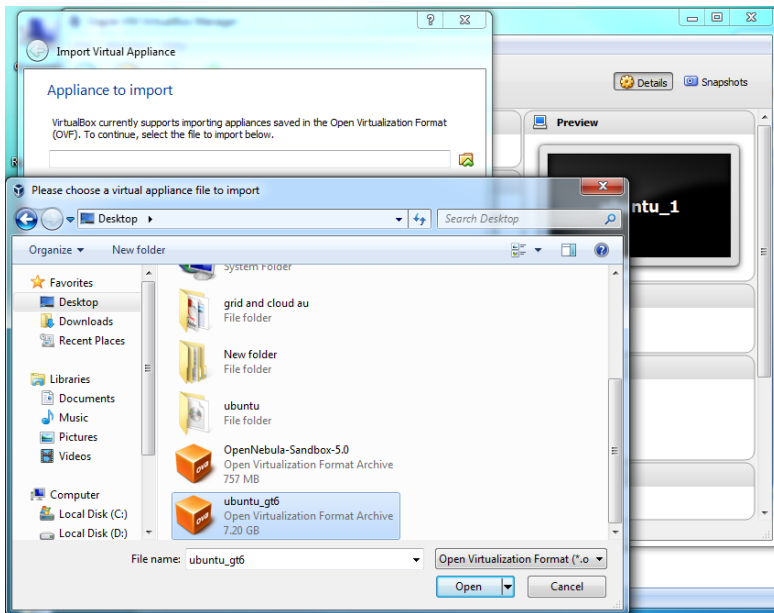
**PROCEDURE:**

**REQUIREMENTS:**

1. ORACLE VIRTUAL BOX
2. OPEN NEBULA SANDBOX
3. UBUNTU Gt6.Ova

**Steps to import .ova file:**

- Open Virtual box
- File →import Appliance
- Browse ubuntu\_gt6.ova file
- Then go to setting, select Usb and choose USB 1.1
- Then Start the ubuntu\_gt6
- Login using username: dinesh, password:99425.



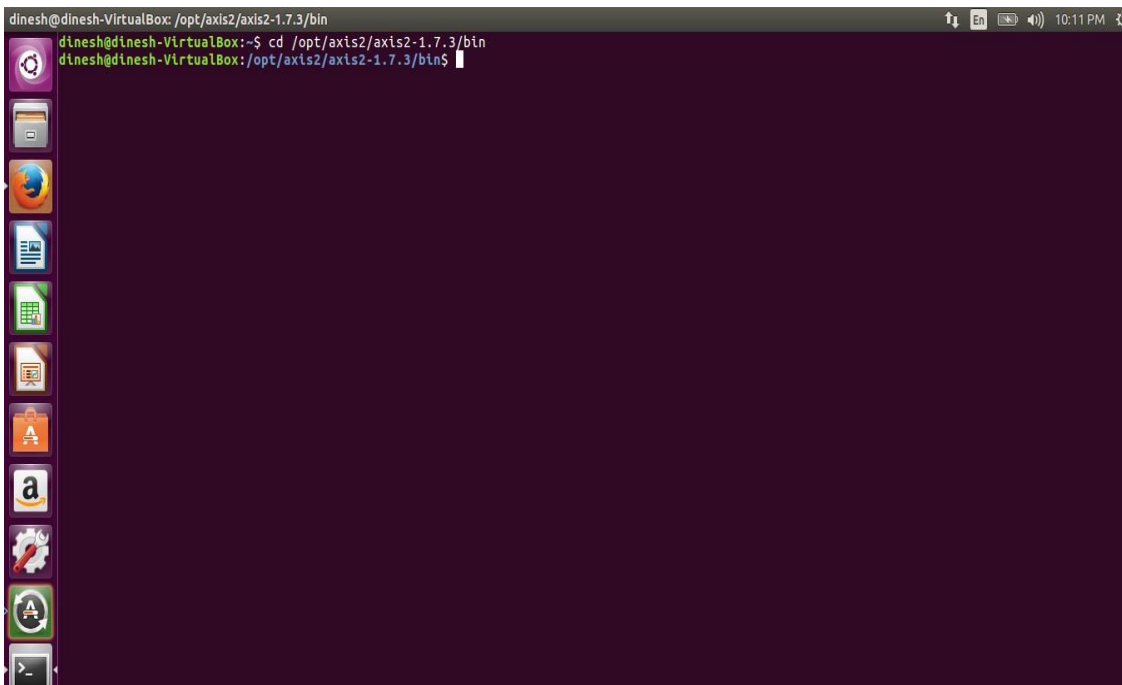
## Steps to run c program:

### STEP 1:

Open the terminal

### STEP 2:

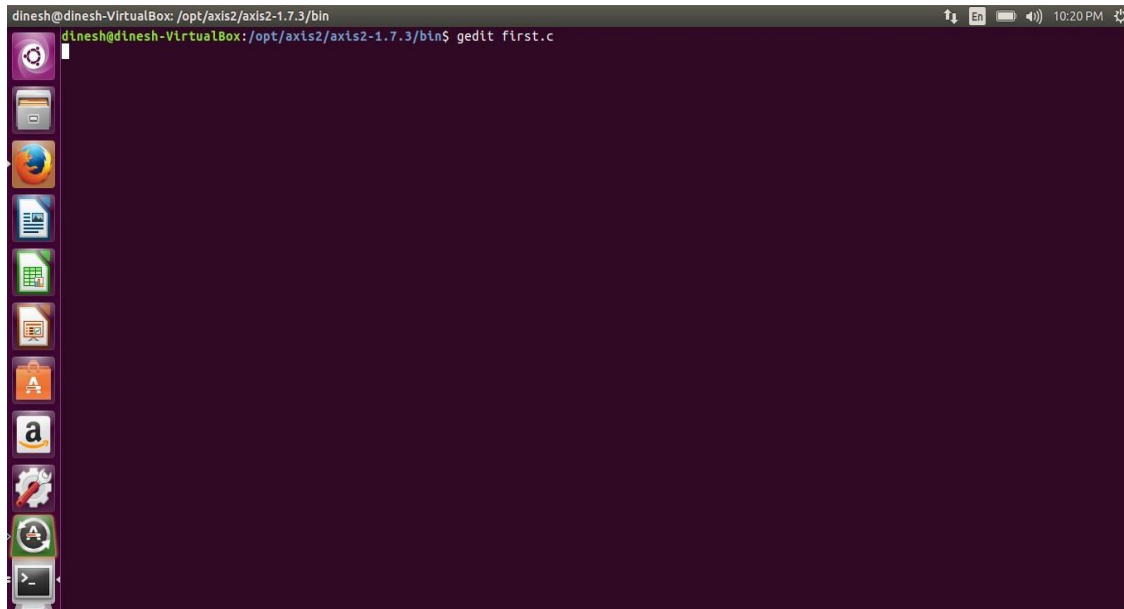
Type `cd /opt/axis2/axis2-1.7.3/bin` then press enter



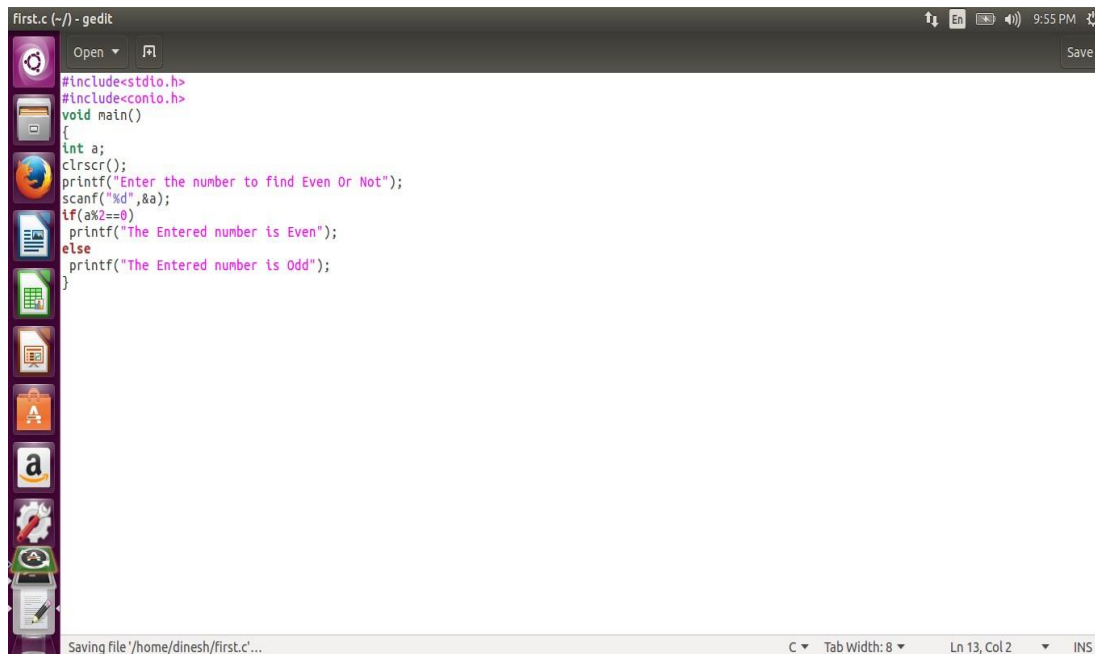


### STEP 3:

To type a sample c program and save it  
gedit hello.c



A terminal window titled 'dinesh@dinesh-VirtualBox: /opt/axis2/axis2-1.7.3/bin' with a dark purple background. The command prompt shows 'dinesh@dinesh-VirtualBox: /opt/axis2/axis2-1.7.3/bin\$ gedit first.c'. The command has been executed, and the cursor is on a new line. On the left side of the terminal, there is a vertical dock with various application icons including a terminal, file manager, web browser, and others.



A Gedit editor window titled 'first.c (~/) - gedit' with a dark theme. The window contains a C program for checking if a number is even or odd. The code is as follows:

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

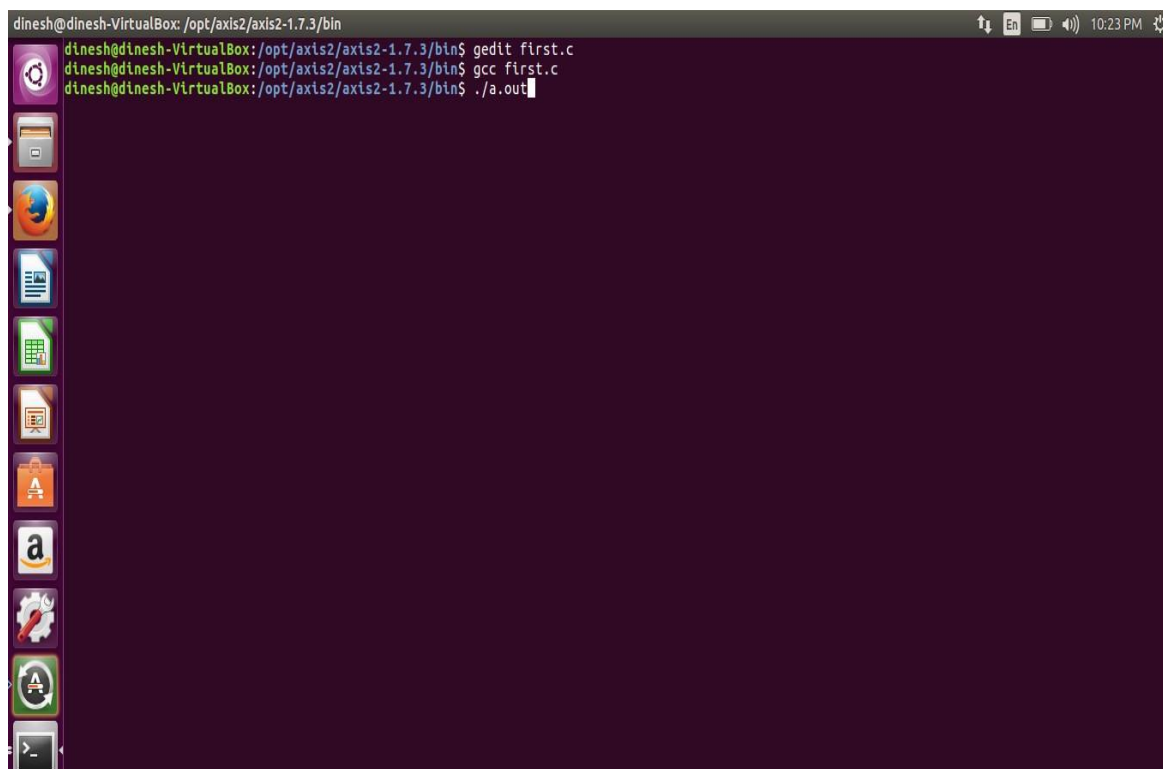
The window has a menu bar with 'Open' and 'Save' options. At the bottom, a status bar shows 'Saving file '/home/dinesh/first.c'...', 'C', 'Tab Width: 8', 'Ln 13, Col 2', and 'INS'.

#### STEP 4:

To compile and run a sample c program

```
gcc hello.c
```

```
./a.out
```

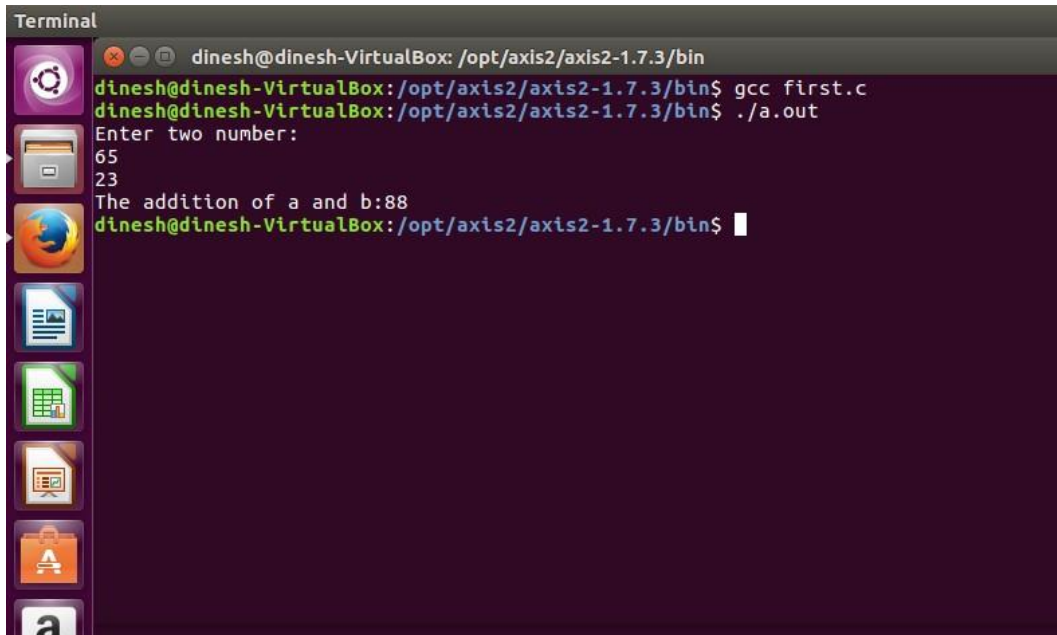


The screenshot shows a terminal window titled "dinesh@dinesh-VirtualBox: /opt/axis2/axis2-1.7.3/bin". The terminal displays the following commands and their outputs:

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

The terminal window has a dark purple background and a light green text color. The command prompt is "\$". The output of the compilation command is empty, and the output of the execution command is also empty. The terminal window is part of a virtual machine interface, with a taskbar visible on the left side showing various application icons.

## OUTPUT:

A screenshot of a Linux terminal window titled "Terminal". The window shows a user named "dinesh" at a machine named "dinesh-VirtualBox". The current directory is "/opt/axis2/axis2-1.7.3/bin". The user has compiled a C program named "first.c" using "gcc" and then executed the resulting binary "a.out". The program prompts the user to "Enter two number:", and the user has entered "65" and "23". The program then outputs "The addition of a and b:88". The terminal prompt is now "dinesh@dinesh-VirtualBox:/opt/axis2/axis2-1.7.3/bin\$". On the left side of the terminal window, there is a vertical dock with several application icons, including a file manager, a web browser, and a terminal icon.

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

## RESULT:

**Aim:** AWS Case Study: Amazon.com.



**Theory: About AWS**

Launched in 2006, Amazon Web Services (AWS) began exposing key infrastructure services to businesses in the form of web services -- now widely known as cloud computing.

The ultimate benefit of cloud computing, and AWS, is the ability to leverage a new business model and turn capital infrastructure expenses into variable costs.

Businesses no longer need to plan and procure servers and other IT resources weeks or months in advance.

Using AWS, businesses can take advantage of Amazon's expertise and economies of scale to access resources when their business needs them, delivering results faster and at a lower cost.

Today, Amazon Web Services provides a highly reliable, scalable, low-cost infrastructure platform in the cloud that powers hundreds of thousands of businesses in 190 countries around the world.

Amazon.com is the world's largest online retailer. In 2011, Amazon.com switched from tape backup to using Amazon Simple Storage Service (Amazon S3) for backing up the majority of its

- With data center locations in the U.S., Europe, Singapore, and Japan, customers across all
  - Industries are taking advantage of the following benefits:

- Low Cos
- Agility and Instant Elasticity
- Open and Flexible
- Secure

### **Why Amazon Web Services?**

Amazon.com initiated the evaluation of Amazon S3 for economic and performance improvements related to data backup. As part of that evaluation, they considered security, availability, and performance aspects of Amazon S3 backups. Amazon.com also executed a cost-benefit analysis to ensure that a migration to Amazon S3 would be financially worthwhile. That cost benefit analysis included the following elements:

- Greater durability and availability. Amazon S3 is designed to provide 99.999999999% durability and 99.99% availability of objects over a given year. Amazon.com compared these figures with those observed from their tape infrastructure, and determined that Amazon S3 offered significant improvement.
- Less operational friction. Amazon.com DBAs had to evaluate whether Amazon S3 backups would be viable for their database backups. They determined that using Amazon S3 for backups was easy to implement because it worked seamlessly with Oracle RMAN.
- Strong data security. Amazon.com found that AWS met all of their requirements for physical security, security accreditations, and security processes, protecting data in flight, data at rest, and utilizing suitable encryption standards.

**Benefits:**

With the migration to Amazon S3 well along the way to completion, Amazon.com has realized several benefits, including:

- Reduced capital expenditures. Amazon.com no longer needs to acquire tape robots, tape drives, tape inventory, data center space, networking gear, enterprise backup software, or predict future tape consumption. This eliminates the burden of budgeting for capital equipment well in advance as well as the capital expense.
- Easy implementation of Oracle RMAN backups to Amazon S3. The DBAs found it easy to start backing up their databases to Amazon S3. Directing Oracle RMAN backups to Amazon S3 requires only a configuration of the Oracle Secure Backup Cloud (SBC) module. The effort required to configure the Oracle SBC module amounted to an hour or less per database. After this one-time setup, the database backups were transparently redirected to Amazon S3.
- Elimination of physical tape transport to off-site location. Any company that has been storing Oracle backup data offsite should take a hard look at the costs involved in transporting, securing and storing

their tapes offsite – these costs can be reduced or possibly eliminated by storing the data in Amazon S3.

**Products & Services:**

- Database
- Deployment & Management
- E-Commerce
- Messaging
- Monitoring
- Networking
- Payments & Billing
- Storage
- Support
- Web Traffic

- Workforce
- Content delivery

## **Database:**

### **Amazon SimpleDB**

Amazon SimpleDB works in conjunction with Amazon S3 and AmazonEC2 to run queries on structured data in real time.

### **Amazon Relational Database Service (RDS)**

Amazon Relational Database Service is a web service that makes it easy to set up, operate, and scale a relational database in the cloud.

## **E-Commerce:**

### **Amazon Fulfillment Web Service (FWS)**

Amazon Fulfillment Web Service allows merchants to deliver products using Amazon.com's worldwide fulfillment capabilities.

## **Deployment & Management:**

### **Amazon Elasti Cache**

Amazon Elasti Cache is a web service that makes it easy to deploy, operate, and scale an in-memory cache in the cloud.

### **AWS Elastic Beanstalk**

AWS Elastic Beanstalk is an even easier way to quickly deploy and manage applications in the AWS cloud. We simply upload our application, and Elastic Beanstalk automatically handles the deployment details of capacity provisioning, load balancing, auto-scaling, and application health monitoring.

### **AWS Cloud Formation**

AWS Cloud Formation is a service that gives developers and businesses an easy way to create a collection of related AWS resources and provision them in an orderly and predictable fashion.

### **Monitoring:**

#### **Amazon CloudWatch**

Amazon CloudWatch is a web service that provides monitoring for AWS cloud resources, starting with Amazon EC2

### **Messaging:**

#### **Amazon Simple Queue Service (SQS)**

Amazon Simple Queue Service provides a hosted queue for storing messages as they travel between computers, making it easy to build automated workflow between Web service

#### **Amazon Simple Notification Service (SNS)**

Amazon Simple Notification Service is a web service that makes it easy to set up, operate, and send notifications from the cloud.

#### **Amazon Simple Email Service (SES)**

Aazon Simple Email Service is a highly scalable and cost-effective bulk and transactional email-sending service for the cloud.

### **Workforce:**

#### **Amazon Mechanical Turk**

Amazon Mechanical Turk enables companies to access thousands of global workers on demand and programmatically integrate their work into various business processes.

### **Networking:**

#### **Amazon Route 53**

Amazon Route 53 is a highly available and scalable Domain Name System (DNS) web service.



## **Amazon Virtual Private Cloud (VPC)**

Amazon Virtual Private Cloud (Amazon VPC) lets you provision a private, isolated section of the Amazon Web Services (AWS) Cloud where we can launch AWS resources in a virtual network that you define. With Amazon VPC, we can define a virtual network topology that closely resembles a traditional network that you might operate in your own datacenter.

## **AWS Direct Connect**

AWS Direct Connect makes it easy to establish a dedicated network connection from your premise to AWS, which in many cases can reduce our network costs, increase bandwidth throughput, and provide a more consistent network experience than Internet-based connections.

## **Elastic Load Balancing**

Elastic Load Balancing automatically distributes incoming application traffic across multiple Amazon EC2 instances.

## **Payments & Billing:**

### **Amazon Flexible Payments Service (FPS)**

Amazon Flexible Payments Service facilitates the digital transfer of money between any two entities, humans or computers.

### **Amazon DevPay**

Amazon DevPay is a billing and account management service which enables developers to collect payment for their AWS applications.

## **Storage:**

### **Amazon Simple Storage Service (S3)**

Amazon Simple Storage Service provides a fully redundant data storage infrastructure for storing and retrieving any amount of data, at any time, from anywhere on the Web.

### **Amazon Elastic Block Store (EBS)**

Amazon Elastic Block Store provides block level storage volumes for use with Amazon EC2 instances. Amazon EBS volumes are off-instance storage that persists independently from the life of an instance.

**Conclusion:**

Thus we have studied a case study on amazon web services.

**EX NO: 8**

## **Creating a Guestbook Application**

**AIM:**

### **PROCEDURE:**

Objectives

- Build and test an App Engine app using Python.
- Integrate your application with Google Accounts for user authentication.
- Use the webapp2 framework.
- Use Jinja2 templates.
- Store data in Cloud Datastore.
- Deploy your app to App Engine.
- Costs

App Engine has generous free quotas that will cover your testing this tutorial in a liveproduction environment.

Before you begin

1. Create a new GCP Console project or retrieve the project ID of an existing projectfrom the Google Cloud Platform Console:

GO TO THE PROJECTS PAGE

Tip: Retrieve a list of your existing project IDs with gcloud.

2. Install the Google Cloud SDK and then initialize the gcloud tool:

DOWNLOAD THE SDK

### **Cloning the project from GitHub**

1. Clone the Guestbook application repository to your local machine:

git clone <https://github.com/GoogleCloudPlatform/appengine-guestbook-python.git>

Go to the directory that contains the sample code:

cd appengine-guestbook-pythonBuilding and running locally

### **To build and run the sample locally:**

1. Start the local development web server by running the following command fromthe appengine-guestbook-python directory: dev\_appserver.py ./

The development web server runs and listens for requests on port 8080.

**Note:** The dev\_appserver.py tool is installed globally with the App EngineSDK whether you

installed with the Cloud SDK or the standalone SDK.

2. Visit <http://localhost:8080/> in your web browser to view the app.

Click **Login**, then sign in with any email address. The development server accepts any email you supply, valid or not. This same code requires a valid Google Account and email when deployed to production.

3. Stop the development server by pressing Control+C.

### Authenticating Users

This part of the Python Guestbook code walkthrough shows how to authenticate users and display a customized greeting for the signed-in user.

This page is part of a multi-page tutorial. To start from the beginning and see instructions for setting up, go to [Creating a Guestbook](#).

### Signing in users

The `MainPage` class defines a handler for HTTP GET requests to the root path `/`. The handler checks to see whether a user is signed in:

`guestbook.py`

```
class MainPage(webapp2.RequestHandler):
    def get(self):
        guestbook_name = self.request.get('guestbook_name',
            DEFAULT_GUESTBOOK_NAME)
        greetings_query = Greeting.query(ancestor=guestbook_key(guestbook_name)).order(
            -Greeting.date)
        greetings = greetings_query.fetch(10)
        user = users.get_current_user()
        if user:
            url = users.create_logout_url(self.request.uri)
            url_linktext = 'Logout'
        else:
            url = users.create_login_url(self.request.uri)
            url_linktext = 'Login'
        template_values = {
            'user': user,
            'greetings': greetings,
            'guestbook_name': urllib.quote_plus(guestbook_name),
            'url': url,
            'url_linktext': url_linktext,
        }
        template = JINJA_ENVIRONMENT.get_template('index.html')
        self.response.write(template.render(template_values))
```

If the user is already signed in to your application, the `get_current_user()` method returns a `User` object, and the app displays the user's nickname. If the user has not signed in, the code redirects the user's browser to the Google account sign-in screen. The Google account sign-in mechanism sends the user back to the app after the user has signed in.

For more information, see the Users API.

## Handling User Input in a Form

This part of the Python Guestbook code walkthrough shows how to handle user input. This page is part of a multi-page tutorial. To start from the beginning and see instructions for setting up, go to [Creating a Guestbook](#).

### ***Configuring the app to use webapp2***

The Guestbook sample uses the webapp2 framework, which is included in the AppEngine environment and the App Engine Python SDK. You don't need to bundle webapp2 with your application code to use it.

The app.yaml file specifies that the app uses the webapp2 framework:

app.yaml

```
libraries:
  name: webapp2
  version: latest
  latest name: jinja2
  version: latest
```

A webapp2 application has two parts:

- One or more RequestHandler classes that process requests and build responses.
- A WSGIApplication instance that routes incoming requests to handlers based on the URL.

The app.yaml file specifies the app object in guestbook.py as the handler for all URLs:

app.yaml

#### **handlers:**

```
url: /favicon.ico static_files:
  favicon.ico upload: favicon.ico
url: /bootstrap static_dir:
  bootstrap url: /*
script: guestbook.app
```

## Defining a handler for form submission

The app object in guestbook.py is a WSGIApplication that defines which scripts handle requests for given URLs.

guestbook.py

```
app = webapp2.WSGIApplication([('/', MainPage), ('/sign', Guestbook)], debug=True)
```

The debug=True parameter tells webapp2 to print stack traces to the browser output if a handler

encounters an error or raises an uncaught exception. This option should be removed before deploying the final version of your application, otherwise you will inadvertently expose the internals of your application.

The Guestbook handler has a `post()` method instead of a `get()` method. This is because the form displayed by Main Page uses the HTTP POST method to submit the form data.

guestbook.py

```
class Guestbook(webapp2.RequestHandler):
    def post(self):
        # We set the same parent key on the 'Greeting' to ensure each
        # Greeting is in the same entity group. Queries across the
        # single entity group will be consistent. However, the write
        # rate to a single entity group should be limited to ~1/second.
        guestbook_name = self.request.get('guestbook_name',
            DEFAULT_GUESTBOOK_NAME)
        greeting = Greeting(parent=guestbook_key(guestbook_name))
        if users.get_current_user():
            greeting.author = Author(
                identity=users.get_current_user().user_id(),
                email=users.get_current_user().email())
        greeting.content = self.request.get('content')
        greeting.put()
        query_params = {'guestbook_name': guestbook_name}
        self.redirect('/?' + urllib.urlencode(query_params))
```

The `post()` method gets the form data from `self.request`.

### Generating Dynamic Content from Templates

This part of the Python Guestbook code walkthrough shows how to use Jinja templates to generate dynamic web content.

This page is part of a multi-page tutorial. To start from the beginning and see instructions for setting up, go to [Creating a Guestbook](#).

HTML embedded in code is messy and difficult to maintain. It's better to use a templating system, where the HTML is kept in a separate file with special syntax to indicate where the data from the application appears. There are many templating systems for Python: EZT, Cheetah, ClearSilver, Quixote, Django, and Jinja2 are just a few. You can use your template engine of choice by bundling it with your application code.

For your convenience, App Engine includes the Django and Jinja2 templating engines.

### Using Jinja2 Templates

The app.yaml file lists the latest version of jinja2 as a required library. Production applications should use an actual version number rather than version: latest. app.yaml

libraries:

```
name: webapp2version: latest
```

```
name: jinja2 version: latest
```

The app imports jinja2 and creates a jinja2.Environment object.guestbook.py

```
import os import urllib
```

```
from google.appengine.api import usersfrom
```

```
google.appengine.ext import ndb import jinja2
```

```
import webapp2
```

```
    JINJA_ENVIRONMENT=jinja2.Environment(loader=jinja2.FileSystemLoader(os.path.dirname(_file_)),
```

```
    extensions= ['jinja2.ext.autoescape'],auto escape=True)
```

The get method for the Main Page request handler forms a dictionary of key/value pairs and passes it to template. Render.

```
    guestbook.py
```

```
    class MainPage(webapp2.RequestHandler):def get(self):
```

```
        guestbook_name = self.request.get('guestbook_name',DEFAULT_GUESTBOOK_NAME)
```

```
        greetings_query = Greeting. Query( ancestor=guestbook_key(guestbook_name)).order(-
```

```
Greeting. Date)
```

```
        greetings = greetings_query.fetch (10)user = users.get_current_user()
```

```
        if user:
```

```
            url = users.create_logout_url(self.request.uri)url_linktext = 'Logout'
```

```
        else:
```

```
            url = users.create_login_url(self.request.uri)url_linktext = 'Login'
```

```
        template_values = { 'user': user, 'greetings': greetings,
```

```
        'guestbook_name': urllib.quote_plus(guestbook_name),'url': url,
```

```
        'url_linktext
```

```
        ': url_linktext,
```

```
        }
```

```
        template = JINJA_ENVIRONMENT.get_template('index.html')
```

```
        self.response.write(template.render(template_values))
```

The page is rendered according to the index.html template, which receives the dictionary

as input.

```
index.html
{% for greeting in greetings %}
<div class="row">
{% if greeting.author %}
<b>{{ greeting.author.email }}
{% if user and user.user_id() == greeting.author.identity %}(You)
{% endif %}
</b> wrote:
{% else %}
An anonymous person wrote:
{% endif %}
<blockquote>{{ greeting.content }}</blockquote>
</div>
{% endfor %}
```

The `JINJA_ENVIRONMENT.get_template(name)` method takes the name of a template file and returns a template object. The `template.render(template_values)` call takes a dictionary of values, and returns the rendered text. The template uses Jinja2 templating syntax to access and iterate over the values, and can refer to properties of those values.

< PREVIOUS >

## Storing Data in Cloud Datastore

This part of the Python Guestbook code walkthrough shows how to store structured data in Cloud Datastore. With App Engine and Cloud Datastore, you don't have to worry about distribution, replication, and load balancing of data. That is done for you behind a simple API—and you get a powerful query engine and transactions as well. **Note:** Cloud Datastore is only one option you have for storing application data. If you prefer to use relational data and SQL instead of data structures, try walking through the instructions for using Google Cloud SQL, which also use the Guestbook example application.

This page is part of a multi-page tutorial. To start from the beginning and see instructions for setting up, go to [Creating a Guestbook](#).

## Storing the submitted greetings

Data is written to Cloud Datastore in objects known as entities. Each entity has a key that uniquely identifies it. An entity can optionally designate another entity as its parent; the first entity is a child of the parent entity. The entities in the data store thus form a hierarchically-structured space similar to the



directory structure of a file system. For detailed information, see Structuring Data for Strong Consistency. App Engine includes a data modeling API for Python. To use the data modeling API, the sample app imports the `google.appengine.ext.ndb` module. Each greeting includes the author's name, the message content, and the date and time the message was posted. The app displays messages in chronological order. The following code defines the data model:

```
guestbook.py
class Author(ndb.Model):
    """Sub model for representing an author.""" identity =
    ndb.StringProperty(indexed=False) email =
    ndb.StringProperty(indexed=False) class
    Greeting(ndb.Model):
    """A main model for representing an individual Guestbook entry.""" author =
    ndb.StructuredProperty(Author)
    content = ndb.StringProperty(indexed=False)
    date = ndb.DateTimeProperty(auto_now_add=True)
```

- The code defines a Greeting model with three properties: author whose value is an Author object with the email address and the author's identity, content whose value is a string, and date whose value is a datetime.datetime.
- Some property constructors take parameters to further configure their behavior. Passing the `ndb.StringProperty` constructor the `indexed=False` parameter says that values for this property will not be indexed. This saves writes which aren't needed because the app never uses that property in a query.
- Passing the `ndb.DateTimeProperty` constructor an `auto_now_add=True` parameter configures the model to automatically give new objects a datetime stamp of the time the object is created, if the application doesn't otherwise provide a value. For a complete list of property types and their options, see NDB Properties.

The application uses the data model to create new Greeting objects and put them into Cloud Datastore. The Guestbook handler creates new greetings and saves them to the data store:

```
guestbook.py
class Guestbook(webapp2.RequestHandler):
    def post(self):
        # We set the same parent key on the 'Greeting' to ensure each #
```

```

Greeting is in the same entity group. Queries across the
# single entity group will be consistent. However, the write# rate to a
single entity group should be limited to
# ~1/second.
guestbook_name = self.request.get('guestbook_name', greeting =
Greeting(parent=guestbook_key(guestbook_name))if
users.get_current_user():
    greeting.author = Author(
        identity=users.get_current_user().user_id(),
        email=users.get_current_user().email())
    greeting.content = self.request.get('content')
greeting.put()
query_params = {'guestbook_name': guestbook_name} self.redirect('/?' +
urllib.urlencode(query_params))

```

This Guestbook handler creates a new Greeting object, then sets its author and content properties with the data posted by the user. The parent of Greeting is a Guestbook entity. There's no need to create the Guestbook entity before setting it to be the parent of another entity. In this example, the parent is used as a placeholder for transaction and consistency purposes. See the Transactions page for more information. Objects that share a common ancestor belong to the same entity group. The code does not set the date property, so date is automatically set to the present, using `auto_now_add=True`.

Finally, `greeting.put()` saves the new object to the data store. If we had acquired this object from a query, `put()` would have updated the existing object. Because we created this object with the model constructor, `put()` adds the new object to the data store.

Because querying in Cloud Datastore is strongly consistent only within entity groups, the code assigns all of one book's greetings to the same entity group by setting the same parent for each greeting. This means the user always sees a greeting immediately after it is written. However, the rate at which you can write to the same entity group is limited to one write to the entity group per second. When you design a real application you'll need to keep this fact in mind. Note that by using services such as Memcache, you can lower the chance that a user sees stale results when querying across entity groups after a write.

### **Retrieving submitted greetings**

Cloud Datastore has a sophisticated query engine for data models. Because Cloud Datastore is not a traditional relational database, queries are not specified using SQL. Instead, data is queried one of two ways: either by using Datastore queries, or by using an SQL-like query language called GQL. To access

the full range of Cloud Datastore's query capabilities, we recommend using queries over GQL. The `MainPage` handler retrieves and displays previously submitted greetings. The `greetings_query.fetch(10)` call performs the query.

#### More about Cloud Datastore indexes

Every query in Cloud Datastore is computed from one or more indexes—tables that map ordered property values to entity keys. This is how App Engine is able to serve results quickly regardless of the size of your application's data store. Many queries can be computed from the built-in indexes, but for queries that are more complex, Cloud Datastore requires a custom index. Without a custom index, Cloud Datastore can't execute these queries efficiently. For example, the Guestbook application filters by guestbook, and orders by date, using an ancestor query and a sort order. This requires a custom index to be specified in the application's `index.yaml` file. You can edit this file manually, or you can take care of it automatically by running the queries in the application locally. After the index is defined in `index.yaml`, deploying the application will also deploy the custom index information.

The definition for the query in `index.yaml` looks like this:

```
index.yaml
indexes:
  kind: Greeting
  ancestor:
  yes
  properties:
    name: date
    direction: desc
```

You can read all about Cloud Datastore indexes in the Datastore Indexes page. You can read about the proper specification for `index.yaml` files in Python Datastore `IndexConfiguration`.

#### Serving Static Files

This part of the Python Guestbook code walkthrough shows how to serve static files. App Engine does not serve files directly out of your application's source directory unless configured to do so. But there are many cases where you want to serve static files directly to the web browser. Images, CSS stylesheets, JavaScript code, movies, and Flash animations are all typically stored with a web application and served directly to the browser.

This page is part of a multi-page tutorial. To start from the beginning and see instructions for setting up, go to [Creating a Guestbook](#).

#### Configuring the app to use static files

The CSS files for the Guestbook app are in the `bootstrap/css` directory. The template for the app's web page, `index.html`, instructs the browser to load `bootstrap.css` and `bootstrap-responsive.css`, which are static files:

`index.html`

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

```
<link type="text/css" rel="stylesheet" href="/bootstrap/css/bootstrap-responsive.css">
```

The `app.yaml` file

specifies the bootstrap directory as the location for static files: app.yaml

handlers:

- url: /favicon\.

static\_files: favicon.ico

upload: favicon\.

- url: /bootstrap static\_dir:

bootstrap

- url: /.\*

script: guestbook.app

The handlers section defines two handlers for URLs. When App Engine receives a request for a URL beginning with /bootstrap, it maps the remainder of the path to files in the bootstrap directory, and if an appropriate file is found, the contents of the file are returned to the client. All other URLs match the /. \* pattern, and are handled by the app object in the guestbook module.

URL path patterns are tested in the order they appear in app.yaml. In this case, the /bootstrap pattern matches before the /. \* pattern for the appropriate paths. For more information on URL mapping and other options you can specify in app.yaml, see the app.yaml reference.

### **Serving Static Files**

This part of the Python Guestbook code walkthrough shows how to serve static files. App Engine does not serve files directly out of your application's source directory unless configured to do so. But there are many cases where you want to serve static files directly to the web browser. Images, CSS stylesheets, JavaScript code, movies, and Flash animations are all typically stored with a web application and served directly to the browser.

This page is part of a multi-page tutorial. To start from the beginning and see instructions for setting up, go to [Creating a Guestbook](#).

### **Configuring the app to use static files**

The CSS files for the Guestbook app are in the bootstrap/css directory. The template for the app's web page, index.html, instructs the browser to

load bootstrap.css and bootstrap-responsive.css, which are static files: index.html

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

```
<link type="text/css" rel="stylesheet" href="/bootstrap/css/bootstrap-responsive.css">
```

The app.yaml file specifies the bootstrap directory as the location for static files:

app.yaml

handlers:

- url: /favicon\.

static\_files: favicon\.

```
- url: /bootstrap static_dir:  
  bootstrap  
  
- url: /*  
  script: guestbook.app
```

The handlers section defines two handlers for URLs. When App Engine receives a request for a URL beginning with `/bootstrap`, it maps the remainder of the path to files in the `bootstrap` directory, and if an appropriate file is found, the contents of the file are returned to the client. All other URLs match the `/*` pattern, and are handled by the `app` object in the `guestbook` module.

URL path patterns are tested in the order they appear in `app.yaml`. In this case, the `/bootstrap` pattern matches before the `/*` pattern for the appropriate paths. For more information on URL mapping and other options you can specify in `app.yaml`, see the `app.yaml` reference.

**RESULT:**