Smt . Durgadevi Sharma Charitable Trust's

Chandrabhan Sharma College

of Arts, Commerce & Science

(Autonomous)

Hindi Linguistic Minority Institution

(Affiliated to University of Mumbai)

NAAC RE-ACCREDITED 'A' GRADE (CGPA 3.10)

Adi Shankaracharya Marg, Powai Vihar Complex, Powai, Mumbai - 400 076

MASTER OF SCIENCE (INFORMATION TECHNOLOGY)

Subject Name : Big Data Analytics & Modern Networking & Computer Vision

Name : YASH CHANDRAKANT SAUNDALKAR

Seat No : 1312229

Teaching Faculty : Mr. Arvind Singh & Mr. Sandeep Vishwakarma & Miss.Sailaja Tiwari

DEPARTMENT OF INFORMATION TECHNOLOGY

CHANDRABHAN SHARMA COLLEGE OF ARTS,

COMMERCE & SCIENCE

NAAC RE-ACCREDITED 'A' Grade (CGPA 3.10)

(Autonomous)

MUMBAI, 400076

MAHARASHTRA

2024-2025

Smt. Durgadevi Sharma Charitable Trust's

Chandrabhan Sharma College

of Arts, Commerce & Science

(Hindi Linguistic Minority Institution)

(Affiliated to the University of Mumbai)

NAAC Re-Accredited 'A' Grade (CGPA 3.10)

MASTER OF SCIENCE (INFORMATION TECHNOLOGY)

Subject Name : Big Data Analytics

Name : Yash Chandrakant Saundalkar

Seat No : 1312229

Teaching Faculty : Mr. Arvind Singh

DEPARTMENT OF INFORMATION TECHNOLOGY

CHANDRABHAN SHARMA COLLEGE OF ARTS,

COMMERCE & SCIENCE

NAAC RE-ACCREDITED 'A' Grade (CGPA 3.10)

(Affiliated to the University of Mumbai)

(Autonomous)

MUMBAI, 400076

MAHARASHTRA

2024-2025

Smt. Durgadevi Sharma Charitable Trust's
Chandrabhan Sharma College
of Arts, Commerce & Science
(Hindi Linguistic Minority Institution)
(Affiliated to the University of Mumbai)
NAAC Re-Accredited 'A' Grade (CGPA 3.10)

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that Mr./~~Miss~~ **Yash Chandrakant Saundalkar** having Exam Seat No. **1312229** of M.Sc.IT **(Semester II)** has completed the Practical work in the subject of "**Big Data Analytics**" during the **Academic year 2024-25** under the guidance of **Mr. Arvind Singh** being the Partial requirement for The fulfilment of the curriculum of Degree of Master of Science in Information Technology, University of Mumbai.

Internal Examiner **Co-ordinator**

Date: **College Seal**

External Examiner

INDEX

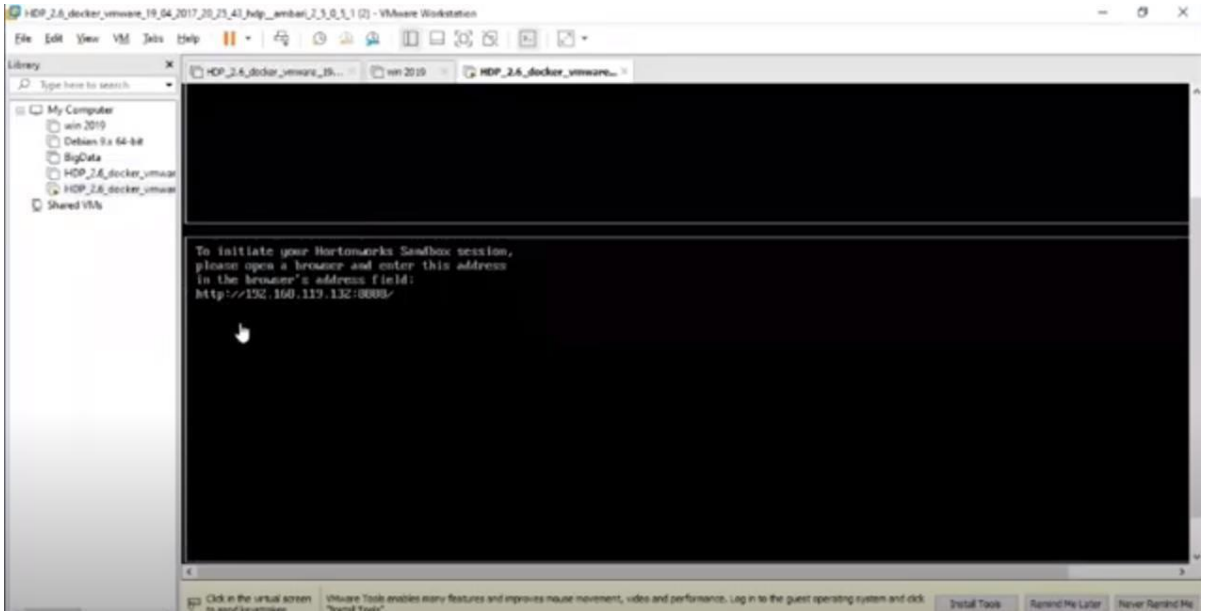
Sr. No	Practical	Date	Sign
1	Install, configure and run Hadoop and HDFS ad explore HDFS.		
2	Implement word count / frequency programs using MapReduce		
3	Implement an MapReduce program that processes a weather dataset.		
4	Implement the program using Pig.		
5	Implement the application in Hive.		
6	Implement an application that stores big data in Hbase/ Python		
7	Implement Decision tree classification techniques		
8	Implement SVM classification techniques		

Practical 1

Install, configure and run Hadoop and HDFS ad explore HDFS.

Download Virtual machine setup i.e. VMware setup (in which Hadoop is configured.

Step 1: Load the server on VM ware workstation



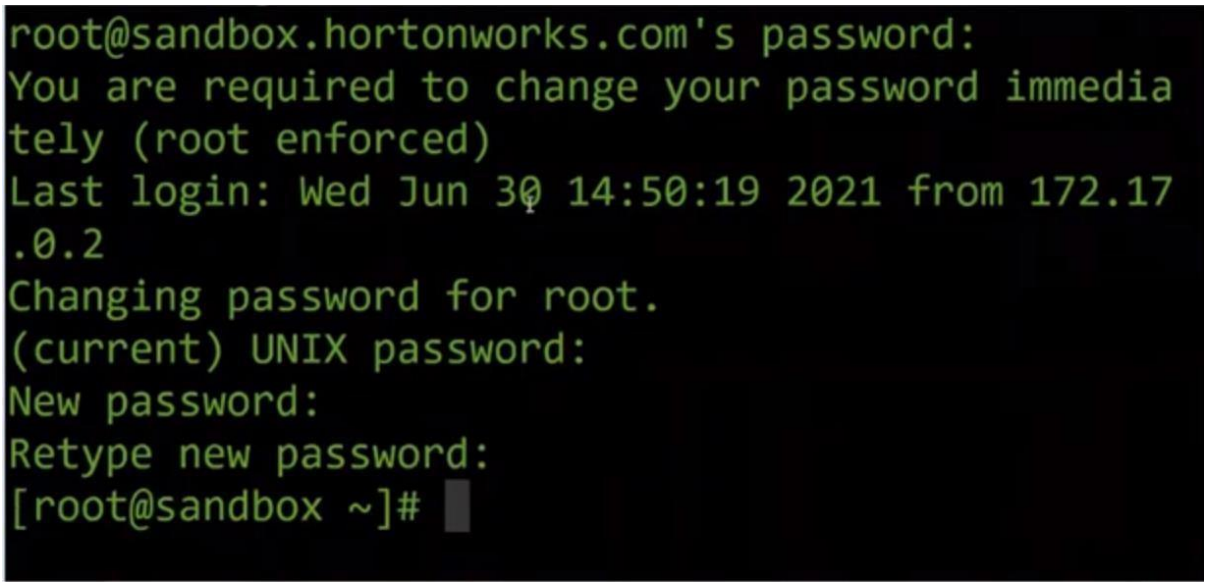
Step 2: To enable admin login open shell and reset root login.

Open Terminal 192.168.119.132:4200

In Sandbox login enter root

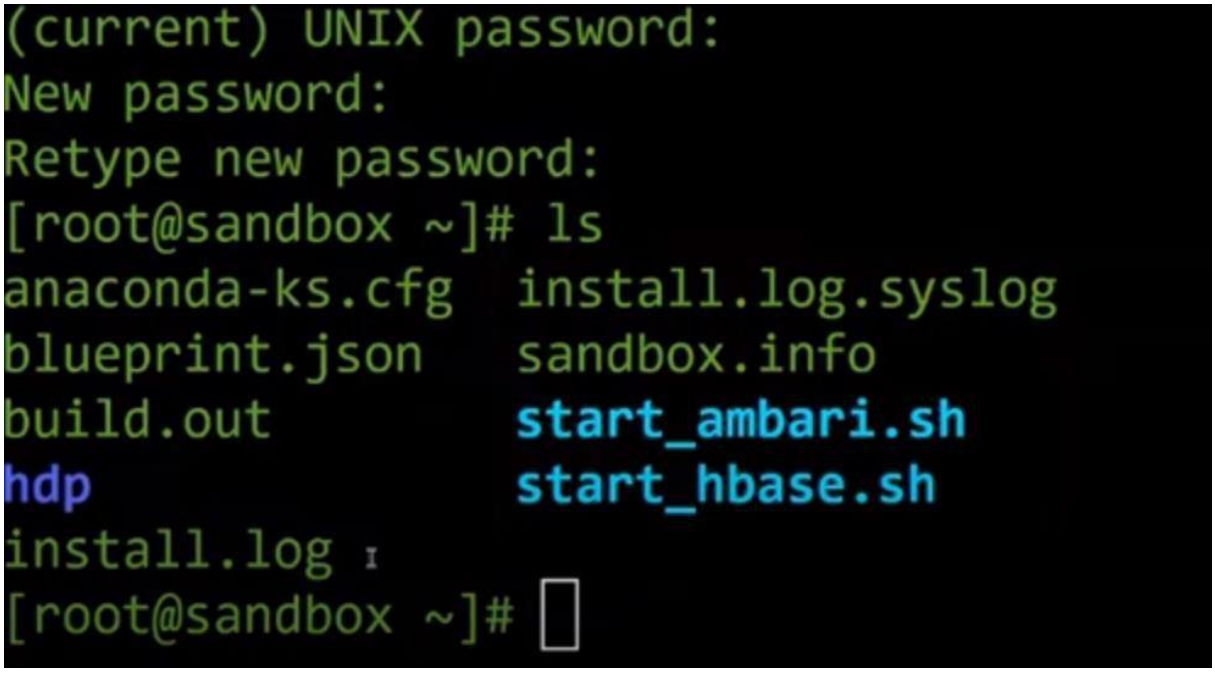
And Password is Hadoop

And reset the password



Windows linux system and Hadoop system are different

When we type **ls** command it is executed in local system



When we type **hdfs dfs -ls** it will execute in Hadoop system directory

```
New password:
Retype new password:
[root@sandbox ~]# ls
anaconda-ks.cfg  install.log.syslog
blueprint.json   sandbox.info
build.out        start_ambari.sh
hdp              start_hbase.sh
install.log
[root@sandbox ~]# hdfs dfs -ls /
```

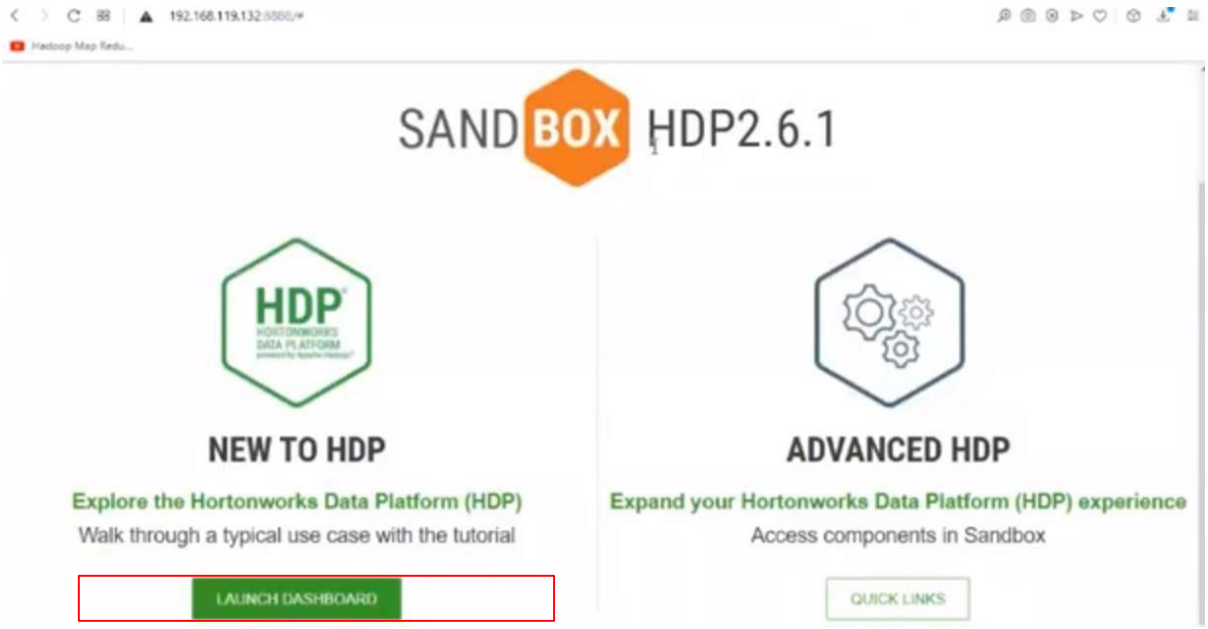
Step 3: Reset Admin account Password

```
[root@sandbox ~]# ambari-admin-password-reset
Please set the password for admin:
Please retype the password for admin:

The admin password has been set.
Restarting ambari-server to make the password change effective...

Using python  /usr/bin/python
Restarting ambari-server
Waiting for server stop...
Ambari Server stopped
Ambari Server running with administrator privileges.
Organizing resource files at /var/lib/ambari-server/resources...
Ambari database consistency check started...
Server PID at: /var/run/ambari-server/ambari-server.pid
Server out at: /var/log/ambari-server/ambari-server.out
Server log at: /var/log/ambari-server/ambari-server.log
Waiting for server start.....
```

Server listening on 8080 and shell login is complete.

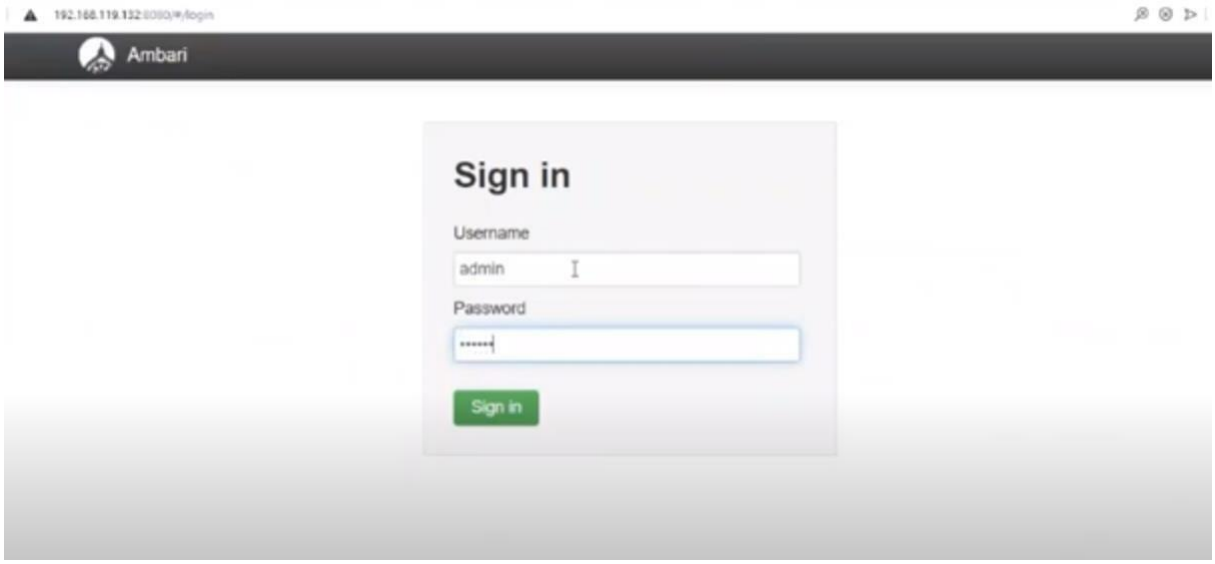
```
The admin password has been set.
Restarting ambari-server to make the password change effective...

Using python  /usr/bin/python
Restarting ambari-server
Waiting for server stop...
Ambari Server stopped
Ambari Server running with administrator privileges.
Organizing resource files at /var/lib/ambari-server/resources...
Ambari database consistency check started...
Server PID at: /var/run/ambari-server/ambari-server.pid
Server out at: /var/log/ambari-server/ambari-server.out
Server log at: /var/log/ambari-server/ambari-server.log
Waiting for server start.....................................................
Server started listening on 8080

DB configs consistency check: no errors and warnings were found.
[root@sandbox ~]#
```

To use graphical user interface login to 192.168.119.132:4200

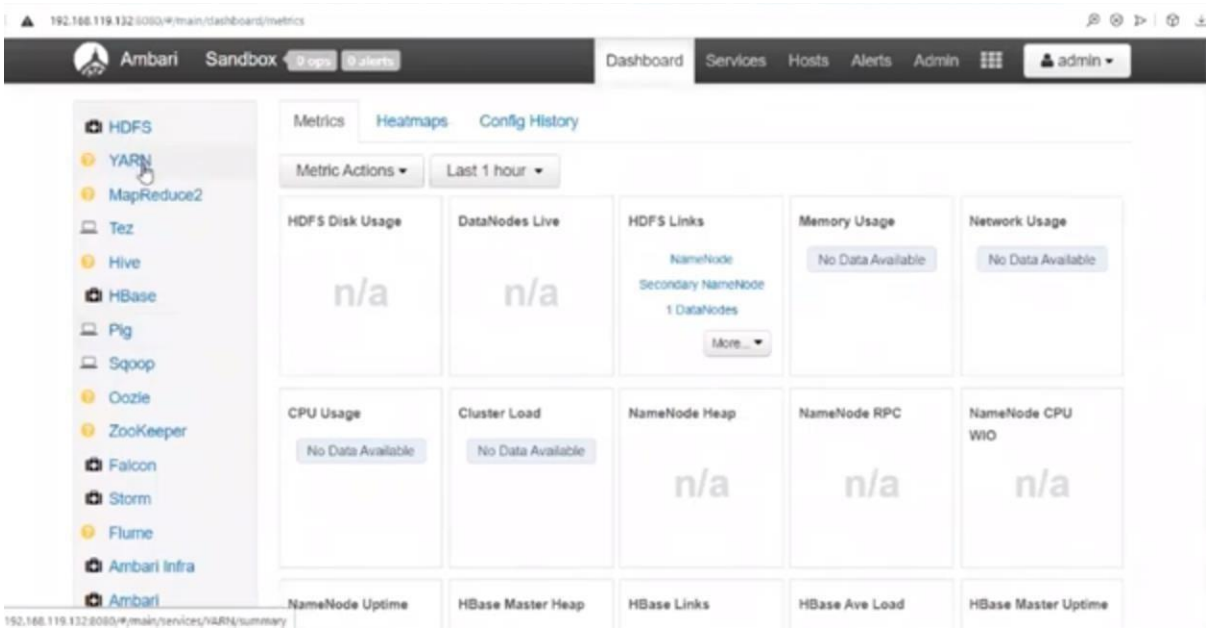


Click on Launch Dashboard

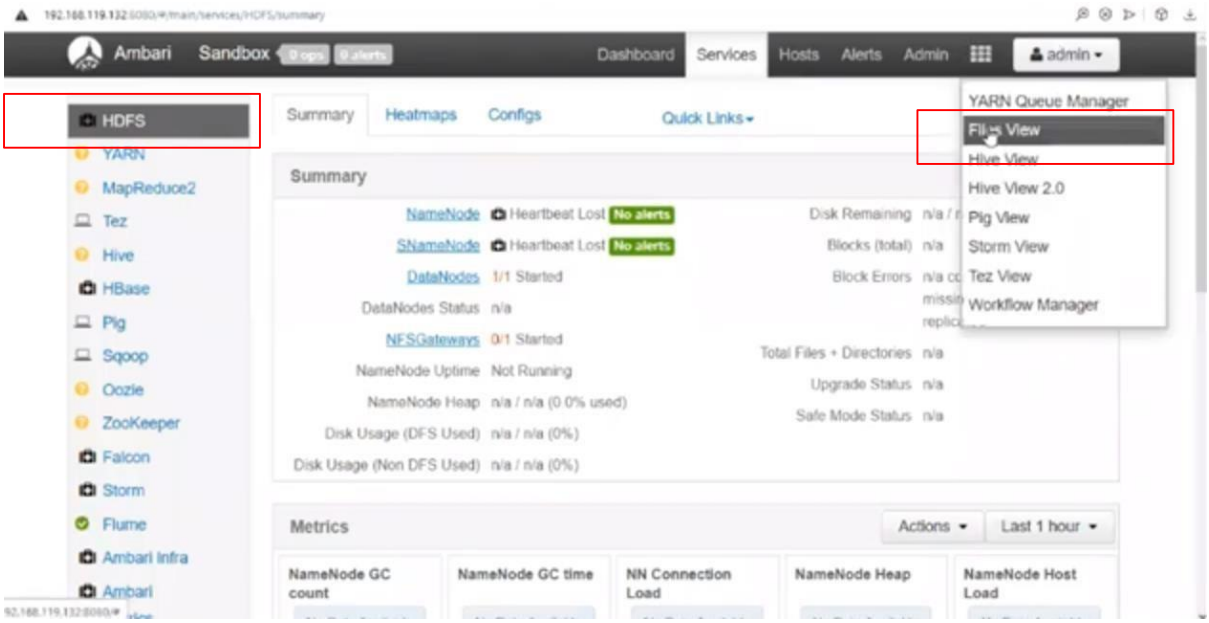
Enter the username and password for admin login.



Below is the Hadoop server.



To view file in HDFS click on HDFS and click on File view.

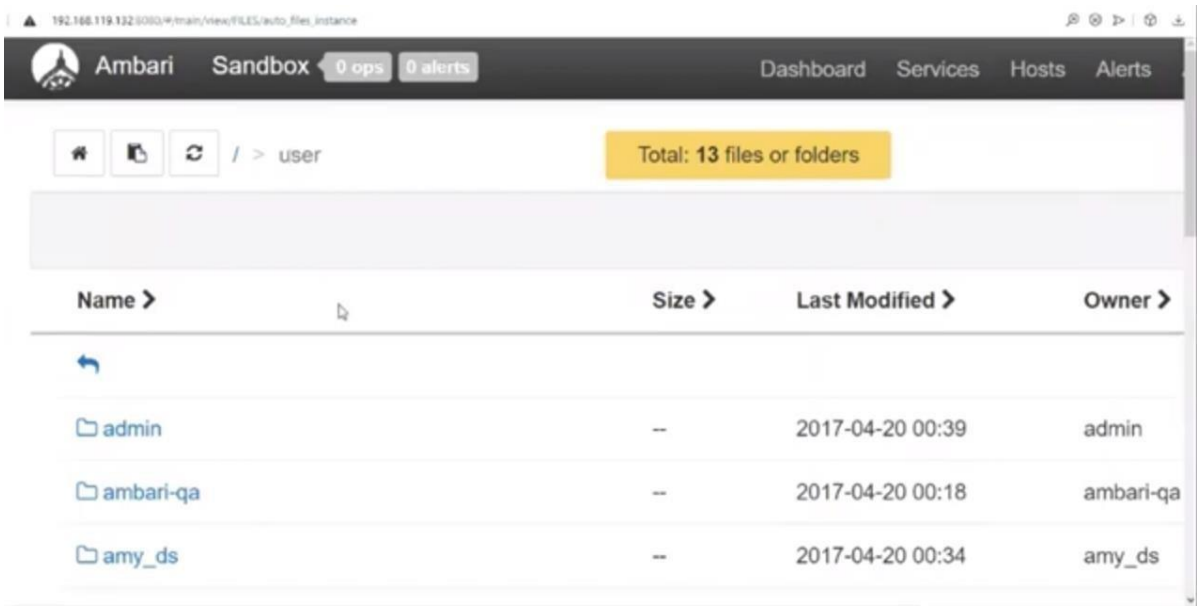


Commands:

1) To view root folder file from terminal use command hdfs dfs -ls /user and press enter. It will display all the files in the root user that we see in UI(Screenshot 2).

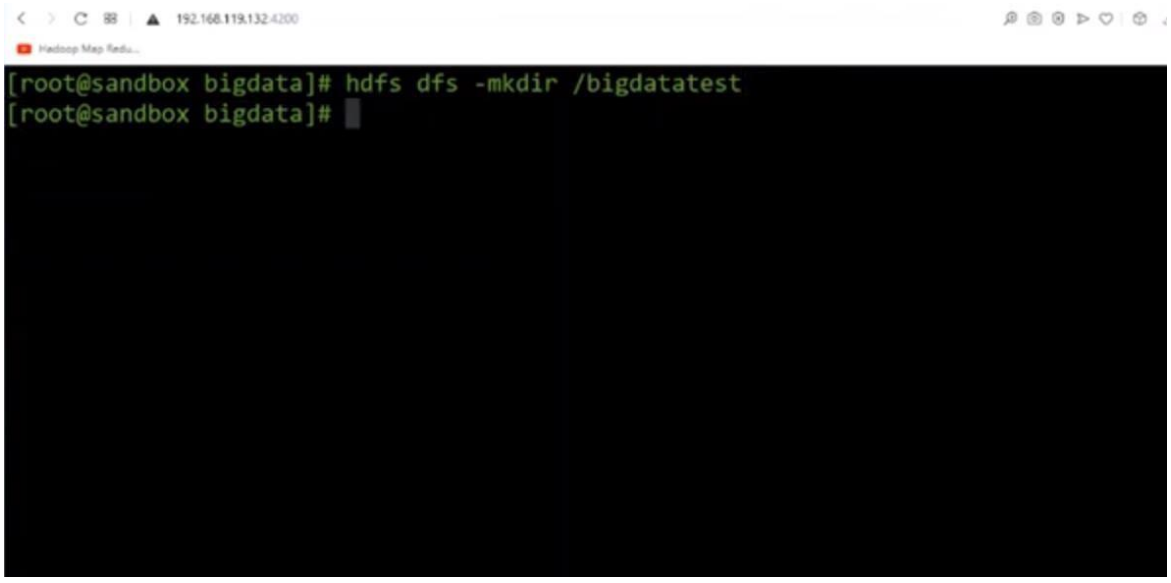
   ls: This command is used to list all the files

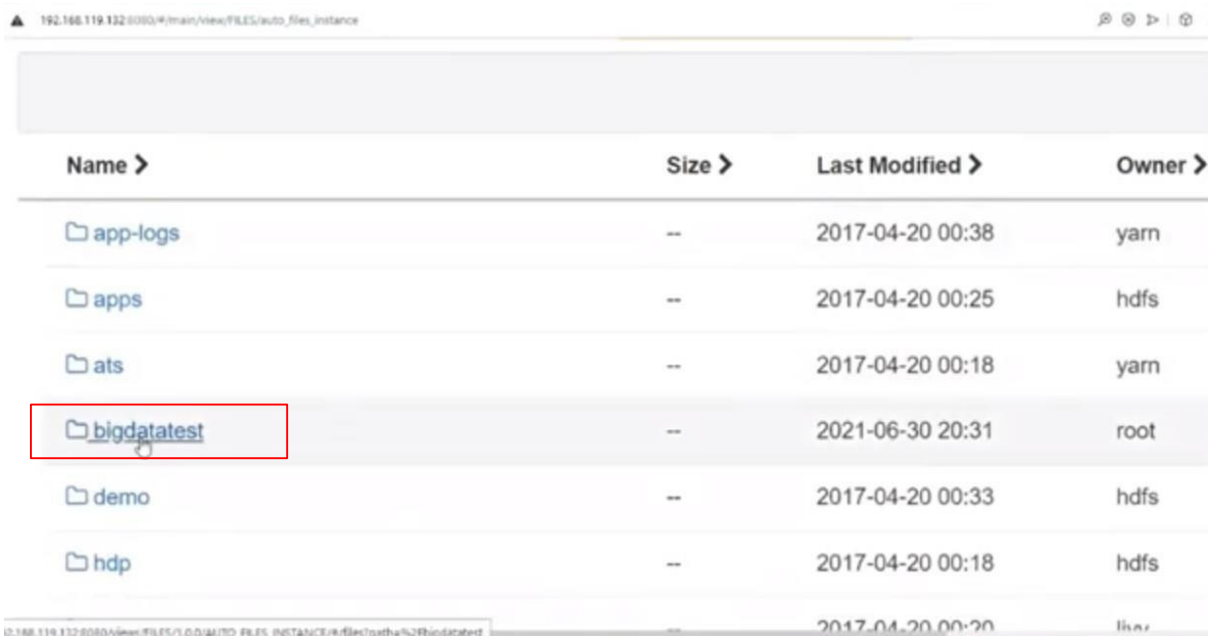
```
DB configs consistency check: no errors and warnings were found.
[root@sandbox ~]# hdfs dfs -ls /user
Found 13 items
drwxr-xr-x   - admin      hdfs          0 2017-04-19 19:09 /user/admin
drwxrwx---   - ambari-qa  hdfs          0 2017-04-19 18:48 /user/ambari-qa
drwxr-xr-x   - amy_ds     hdfs          0 2017-04-19 19:04 /user/amy_ds
drwxr-xr-x   - hbase      hdfs          0 2017-04-19 18:48 /user/hbase
drwxr-xr-x   - hcat       hdfs          0 2017-04-19 18:51 /user/hcat
drwxr-xr-x   - hive       hdfs          0 2017-04-19 19:08 /user/hive
drwxr-xr-x   - holger_gov hdfs          0 2017-04-19 19:05 /user/holger_gov
drwxrwxr-x   - livy       hdfs          0 2017-04-19 18:49 /user/livy
drwxr-xr-x   - maria_dev  hdfs          0 2017-04-19 18:58 /user/maria_dev
drwxrwxr-x   - oozie      hdfs          0 2017-04-19 18:52 /user/oozie
drwxr-xr-x   - raj_ops    hdfs          0 2017-04-19 19:06 /user/raj_ops
drwxrwxr-x   - spark      hdfs          0 2017-04-19 18:49 /user/spark
drwxr-xr-x   - zeppelin   hdfs          0 2017-04-19 18:49 /user/zeppelin
[root@sandbox ~]#
```



2) **mkdir**: To create a directory.

Create a folder in Hadoop directory. Type command hdfs dfs -mkdir /bigdatatest and enter. After it execute the command, we will see whether it is created folder in UI.



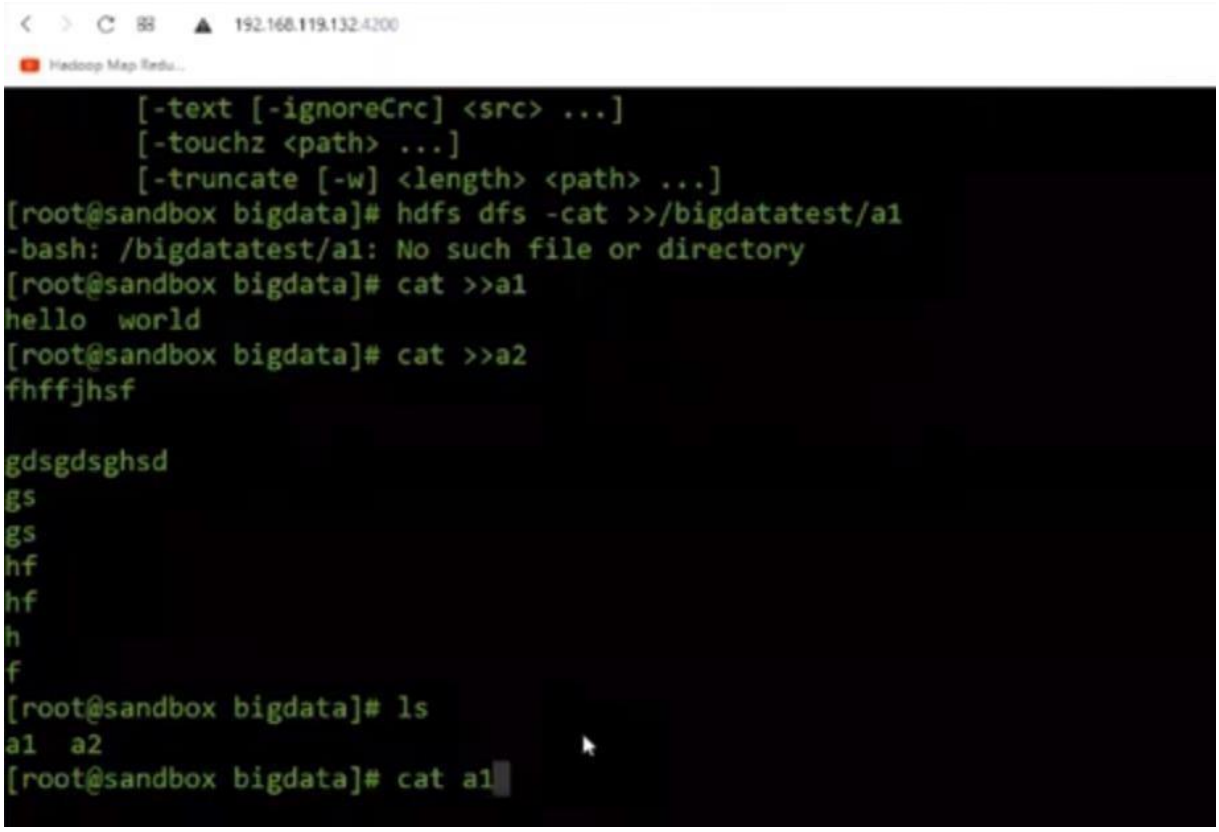


3) Create a file in local directory

Cat: Create a file.

Cat>>

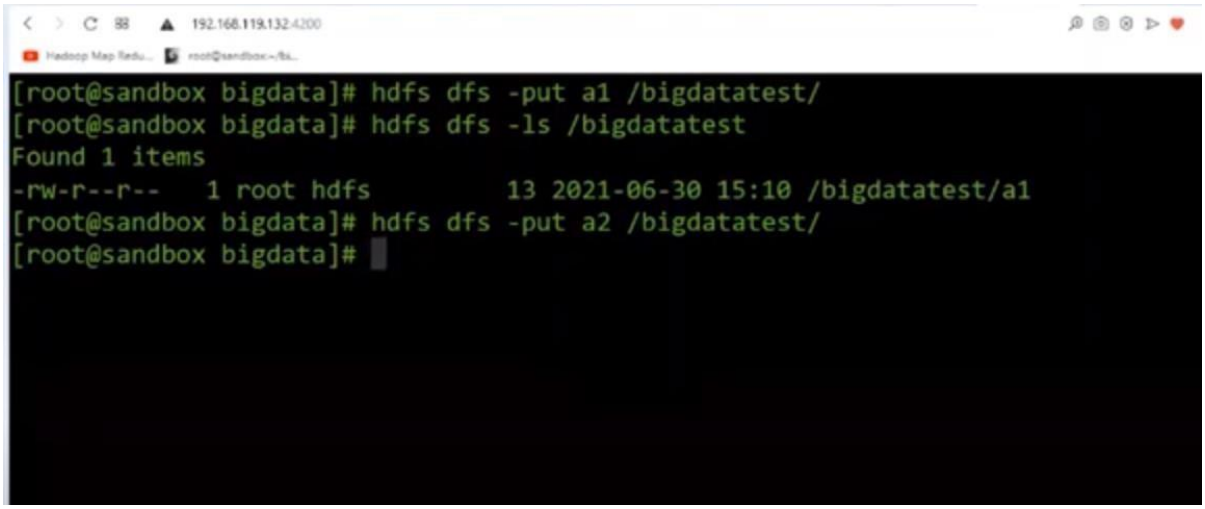
To terminate press ctrl+d



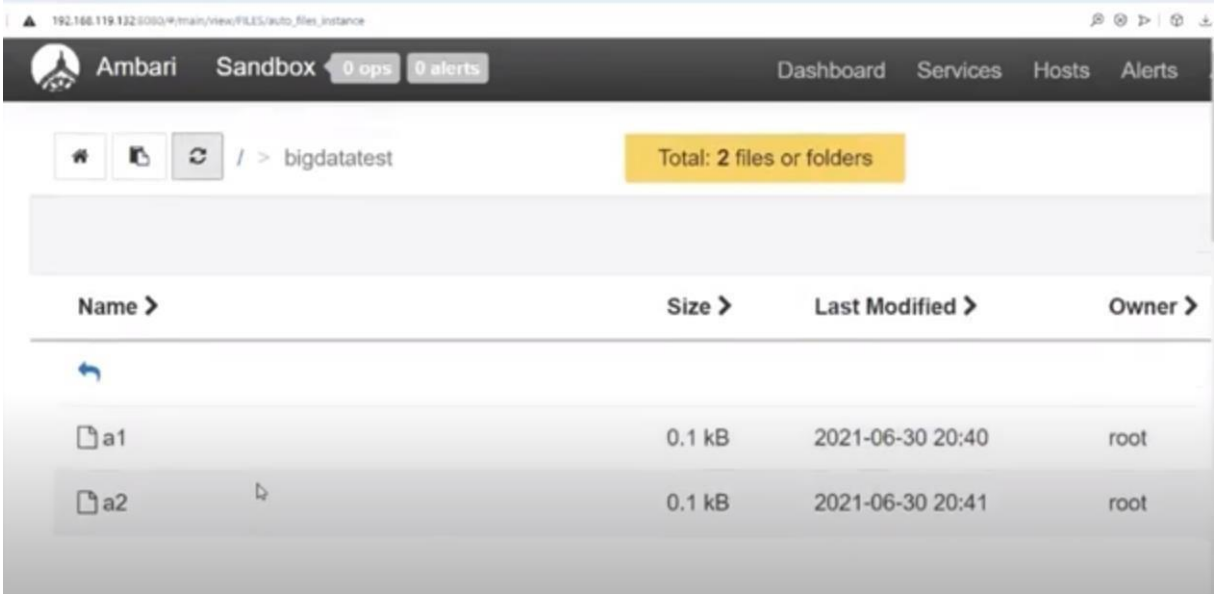
4) To upload files/directory to from local to HDFS

Put: to move a local file or directories into the distributed file system

Command: hdfs dfs -put a1 /bigdatatest/ and hdfs dfs -put a2 /bigdatatest/ will upload both the files.



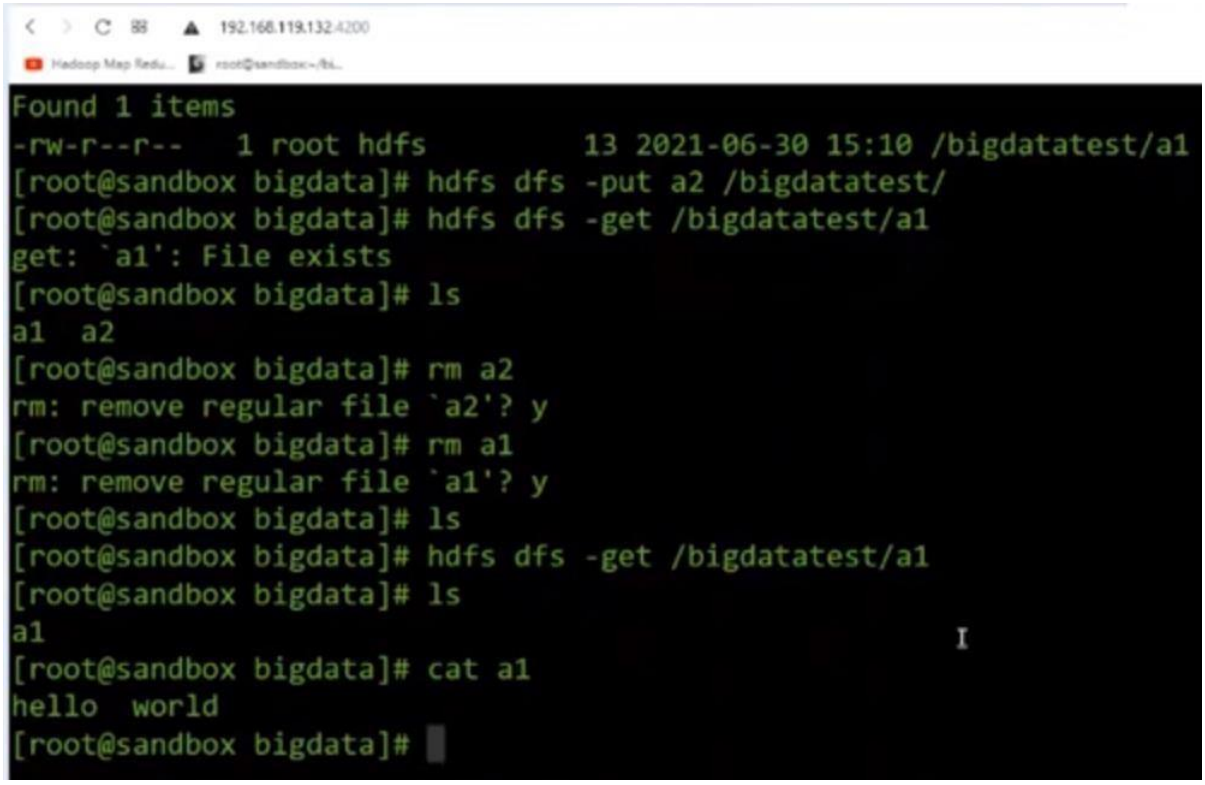
Refresh the user interface we can see both the files.



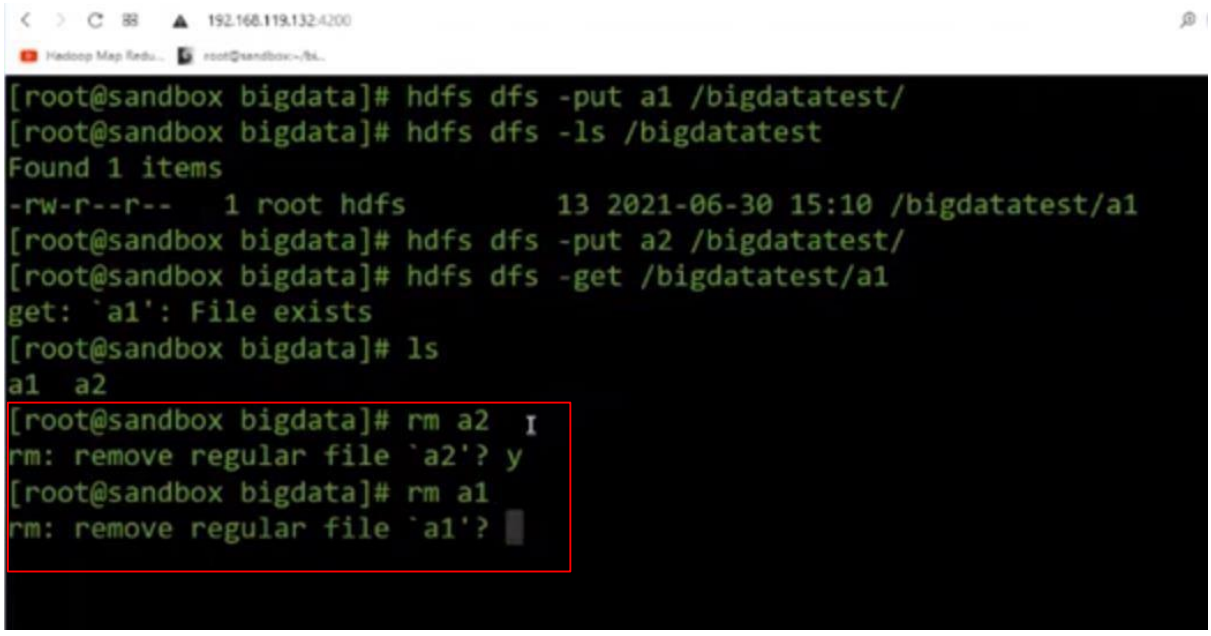
5) To download files/directories to from hdfs to local

Get: To copy files/folders from hdfs store to local file system.

Command: hdfs dfs -get /bigdatatest/a1 and hdfs dfs -get /bigdatatest/ a2 will upload both the files.

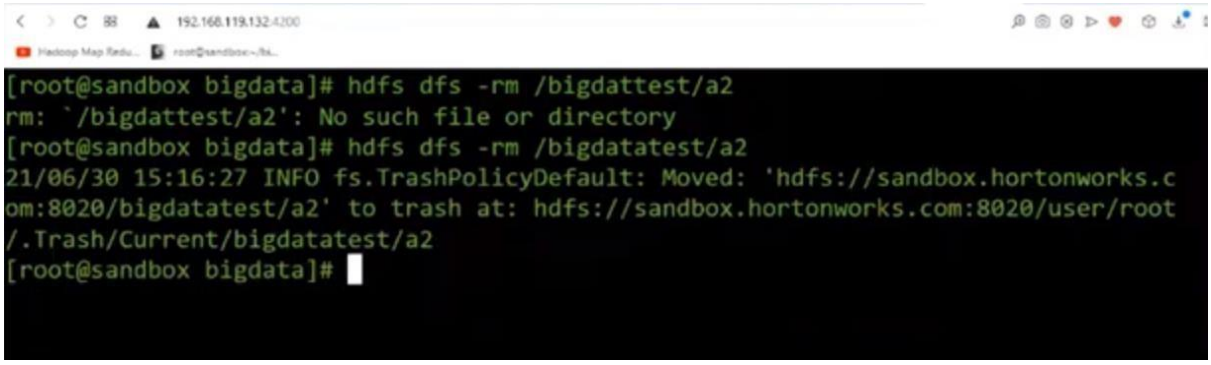


6) To remove file from local use rm command

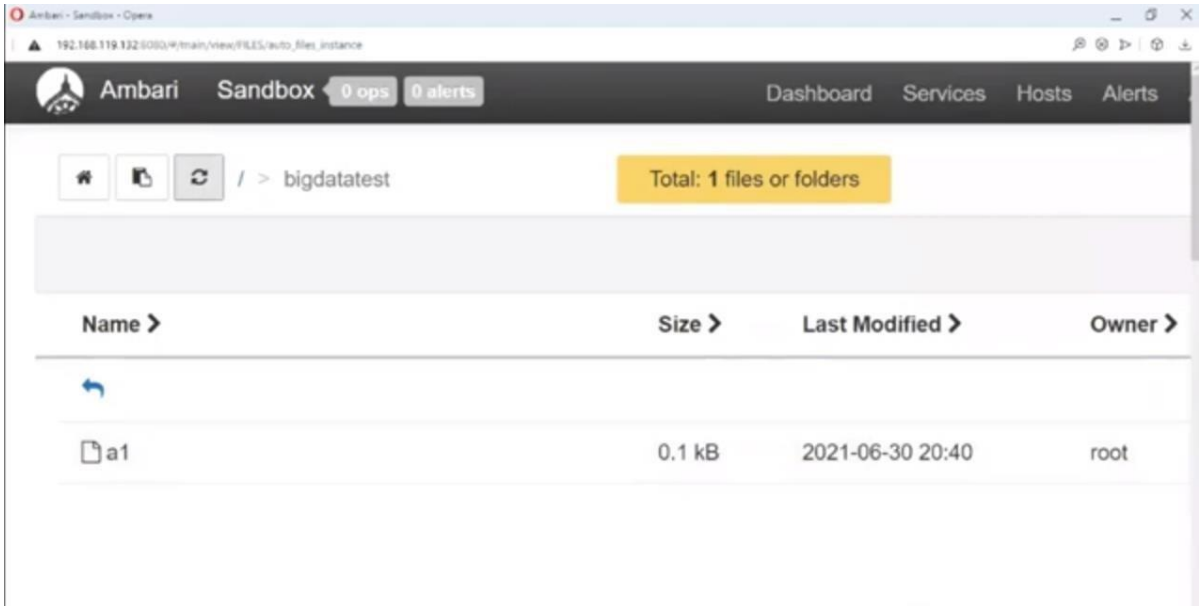


7) To remove file from Hadoop directory

Command: hdfs dfs -rm a2 /gibdatatest/a2

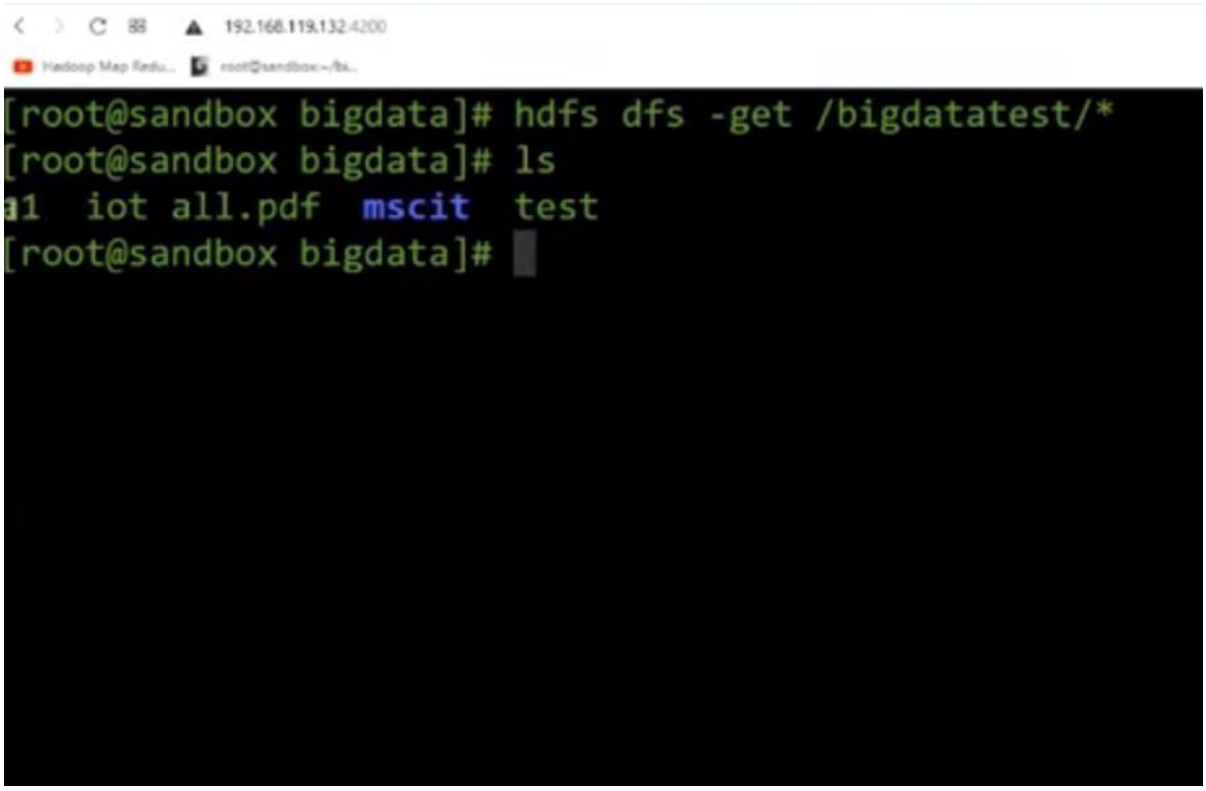


Now refresh the UI and the file will be deleted.



8) To download all the files from hdfs to local

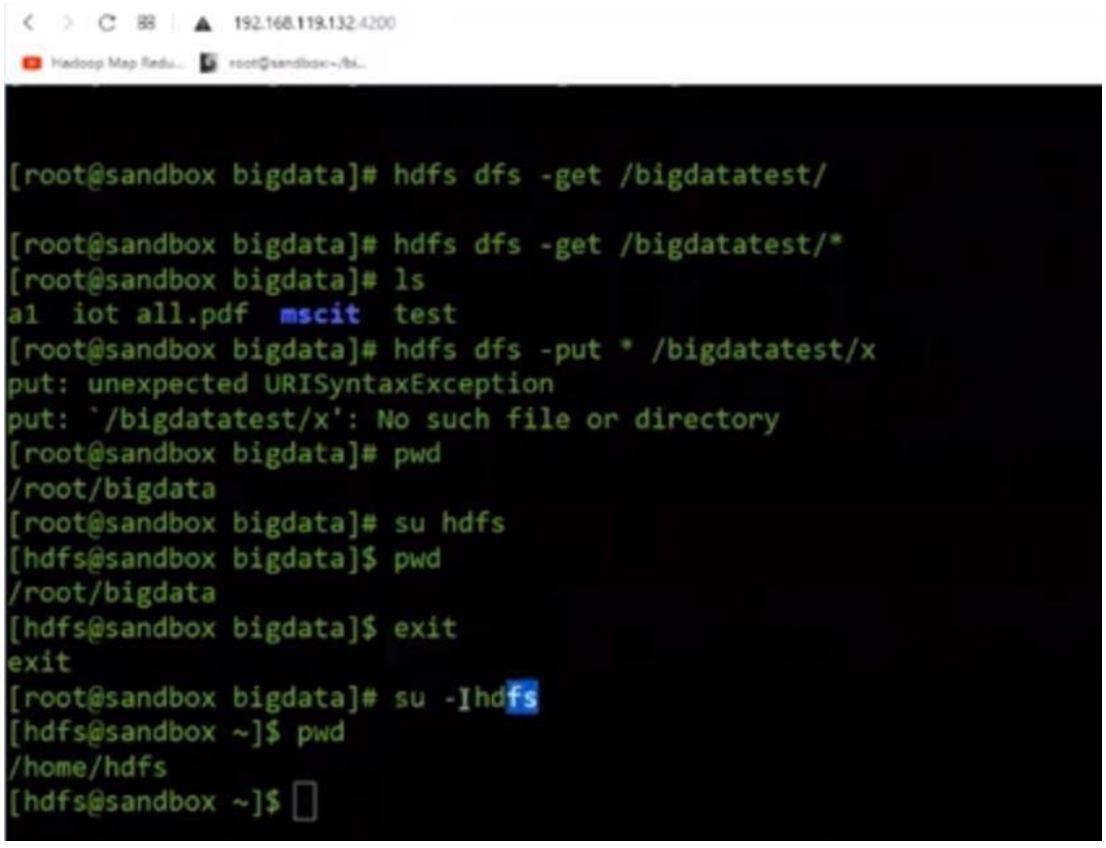
Command: hdfs dfs -get /bigdatatest/*



9) Change user and directory and change user only

Command: su – hdfs (Change user and directory)

Su hdfs (change user only)



Practical 2

Implement word count / frequency programs using MapReduce

Map Reduce as two component Map and Reduce.

Java program:

```
write program save as WordCount.java

//////////////////////
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()) {//"This is the output is the"
 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
 {//is,3
 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");
```

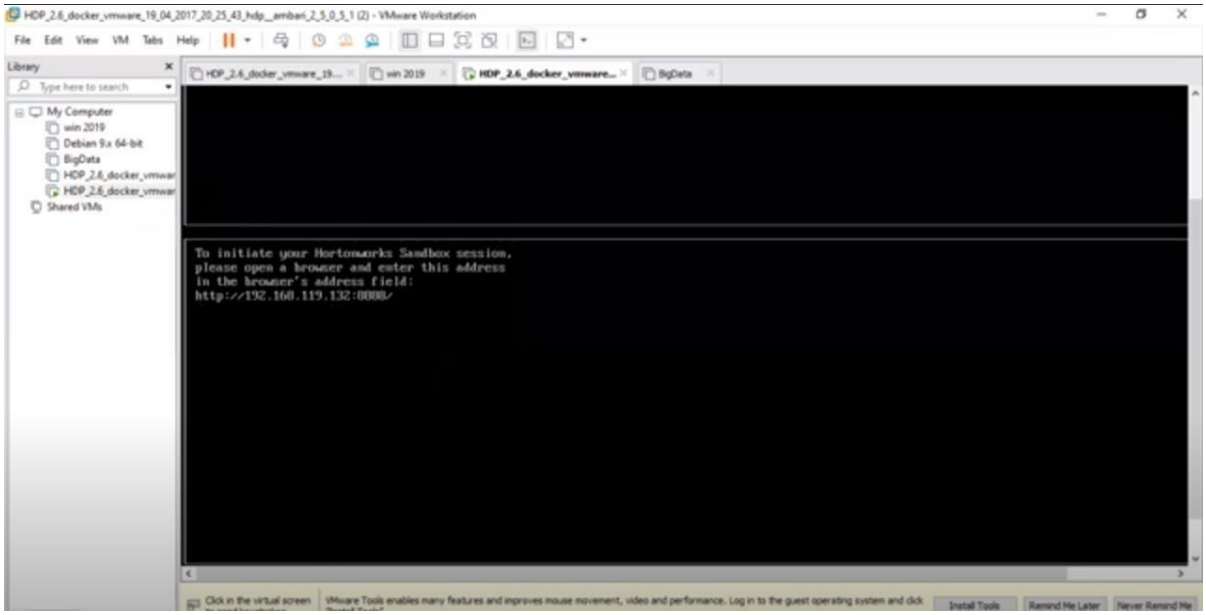
```
 job.setJarByClass(WordCount.class);
 job.setMapperClass(TokenizerMapper.class);
 job.setCombinerClass(IntSumReducer.class);
 job.setReducerClass(IntSumReducer.class);
 job.setOutputKeyClass(Text.class);
 job.setOutputValueClass(IntWritable.class);
 FileInputFormat.addInputPath(job, new Path(args[0]));
 FileOutputFormat.setOutputPath(job, new Path(args[1]));
 System.exit(job.waitForCompletion(true)?0:1);
 }
}
//////////////////////
```

Text File:

Hello World

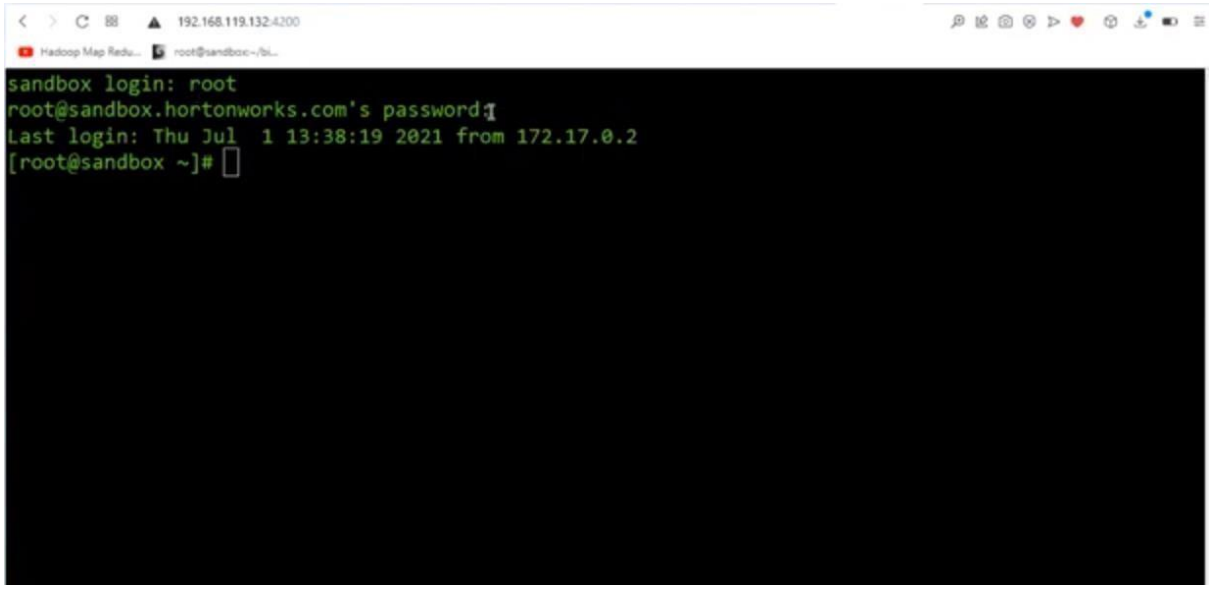
This is the output is the

Start the server (Horton Sandbox)



Open the terminal with 192.168.119.132/4200

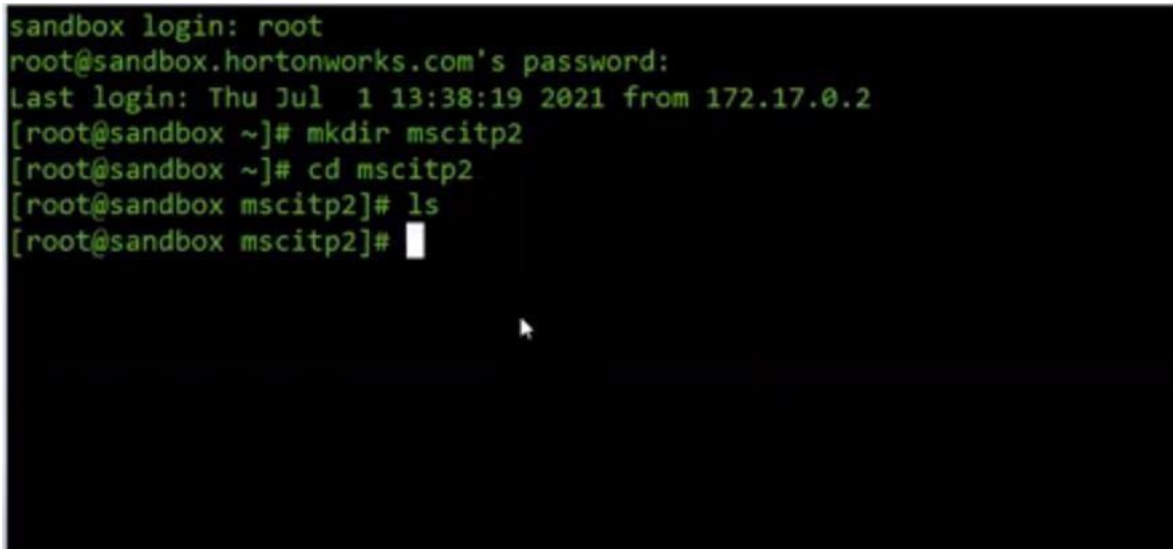
Enter the login: root and the password and enter



Create a folder in local directory.

Command: mkdir mscitp2

Change the directory cd mscitp2



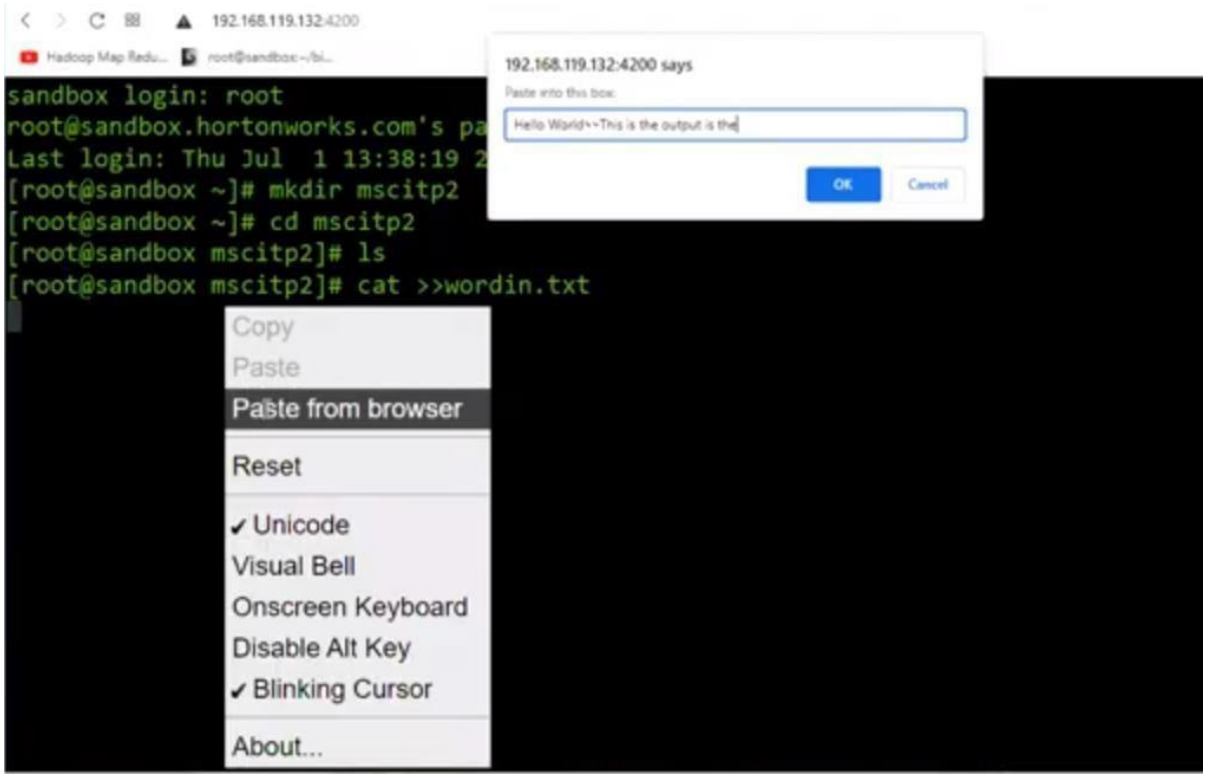
Now create input file

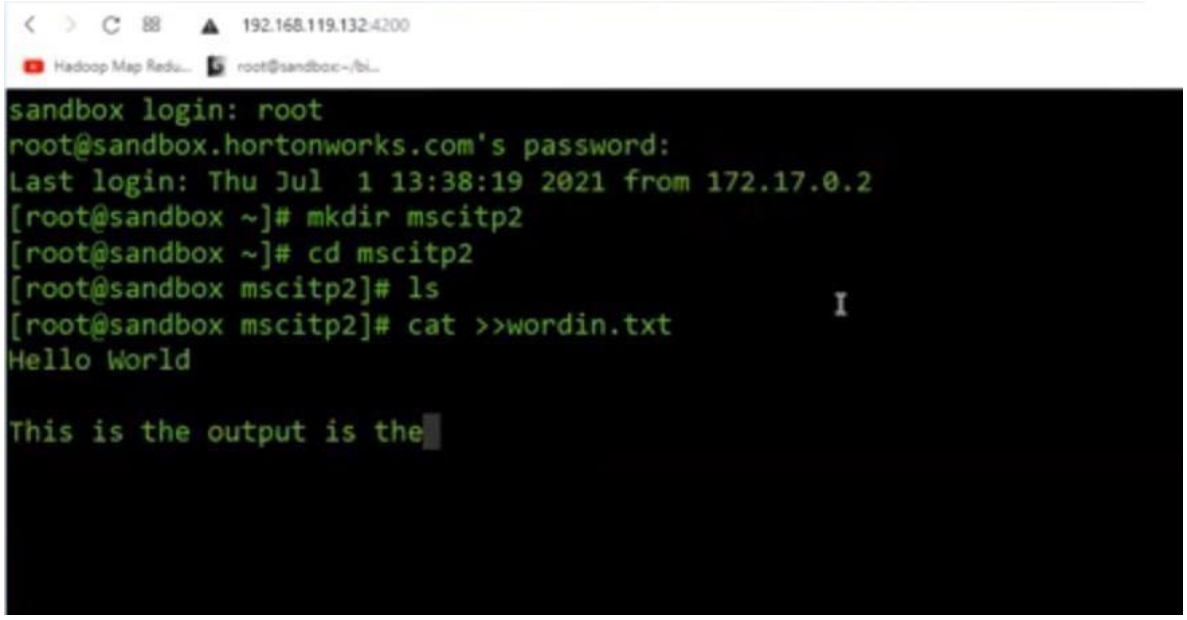
Command: cat >> wordin.txt

Paste the text by right clicking on terminal

Hello World

This is the output is the



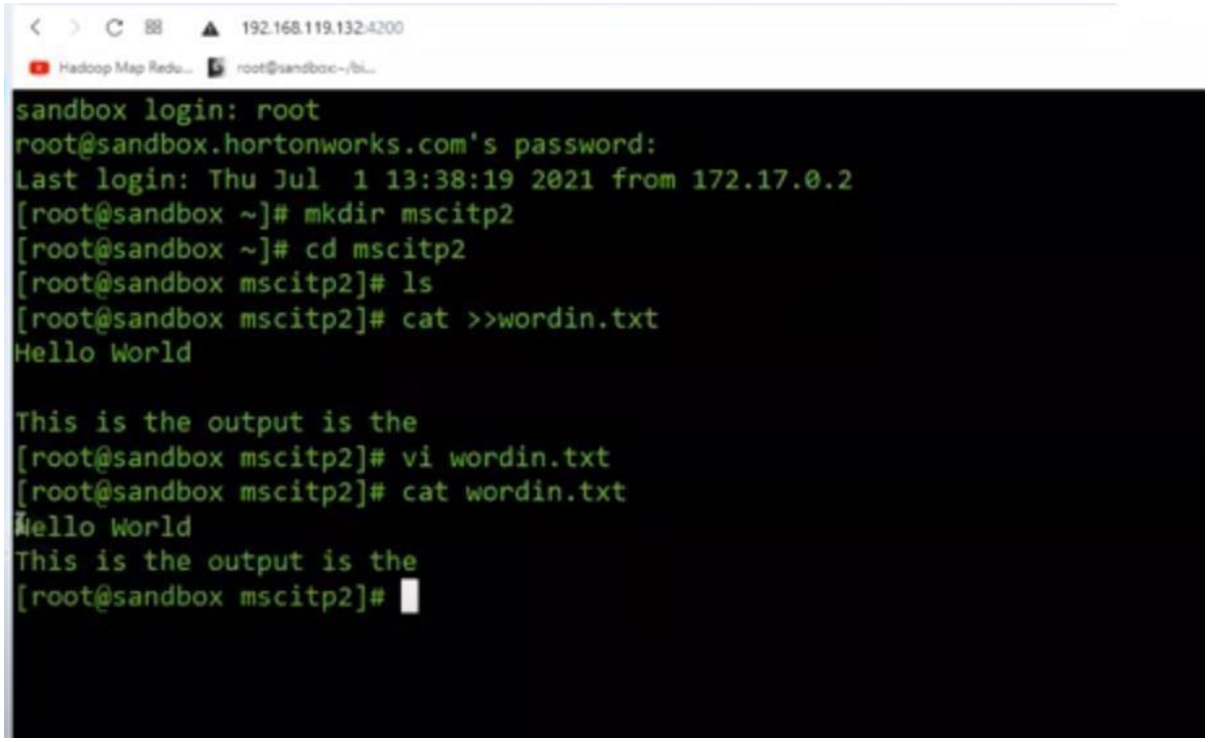


To remove the extra space type command

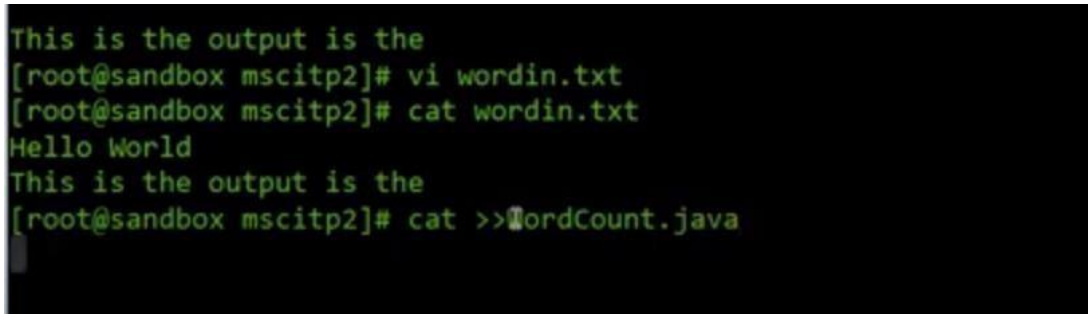
vi wordin.txt

After removing the extra space check the content of the file

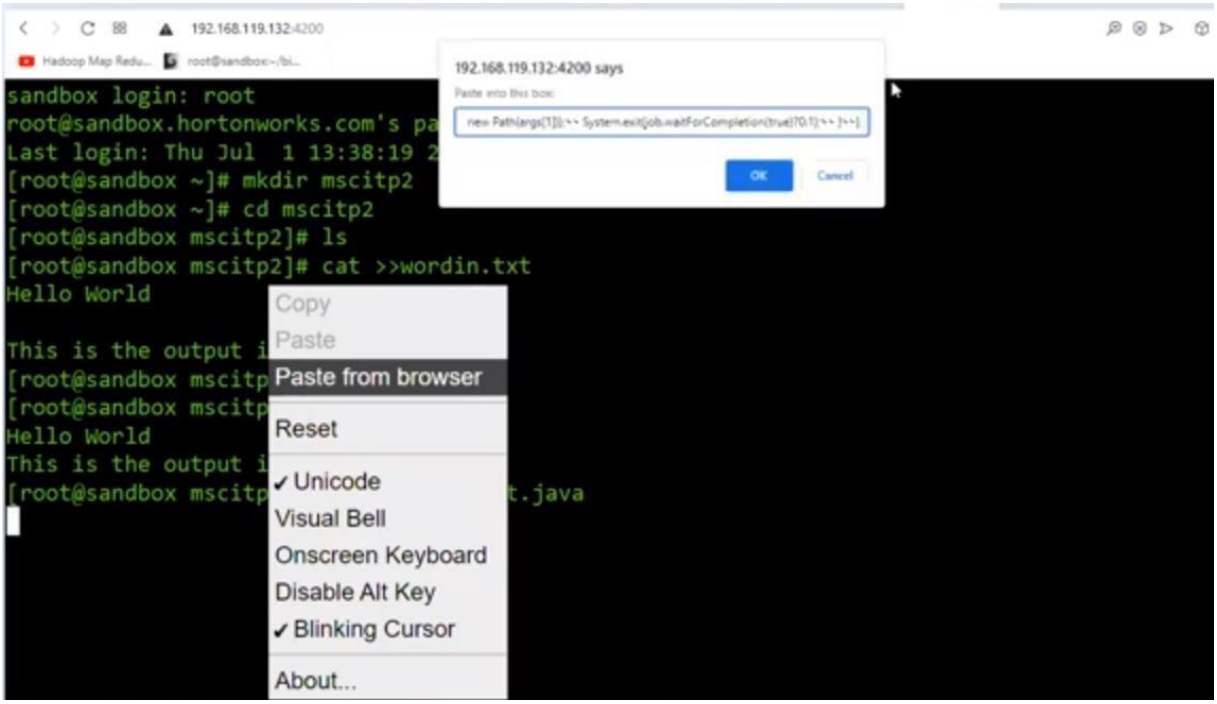
cat wordin.txt



Create another file wordcount.java

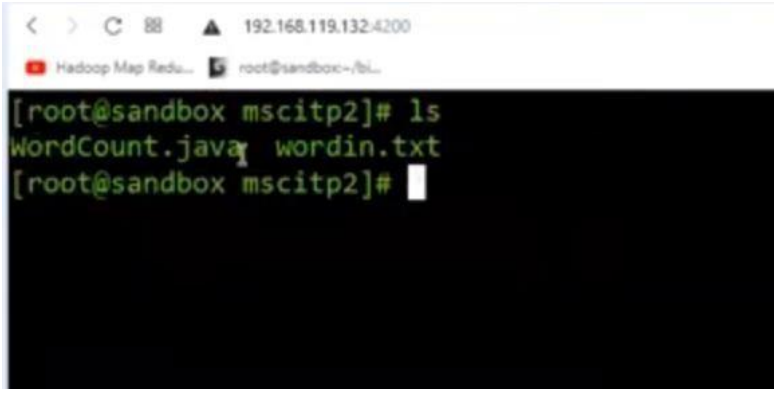


Paste the java code.



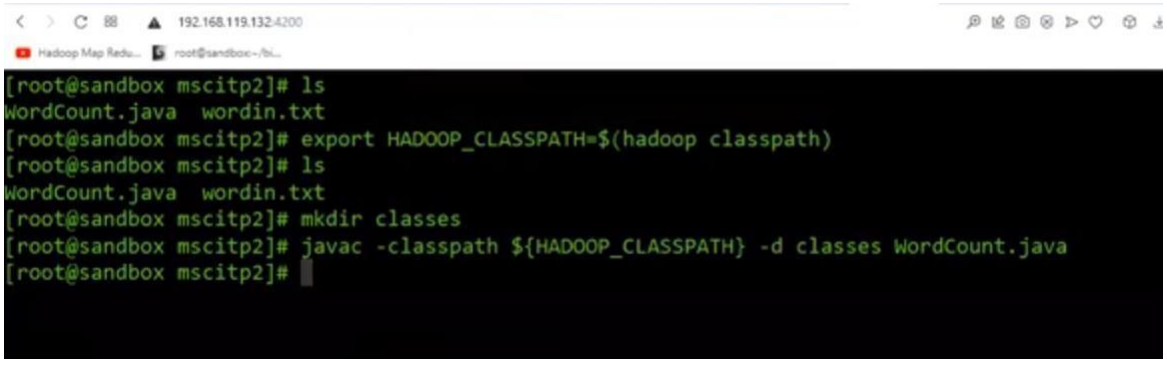
Press control d to save the file

Check both the files create with command ls



Now, to compile the java file

```
export HADOOP_CLASSPATH=$(hadoop classpath)
mkdir classes (To keep the compile files)
javac -classpath ${HADOOP_CLASSPATH} -d classes WordCount.java
```



Check class files are created with command ls classes

```
WordCount.java  wordin.txt
[root@sandbox mscitp2]# mkdir classes
[root@sandbox mscitp2]# javac -classpath ${HADOOP_CLASSPATH} -d classes WordCount.java
[root@sandbox mscitp2]# ls classes
WordCount.class  WordCount$IntSumReducer.class  WordCount$TokenizerMapper.class
[root@sandbox mscitp2]#
```

Now we have to bind all the class into single jar file with below command

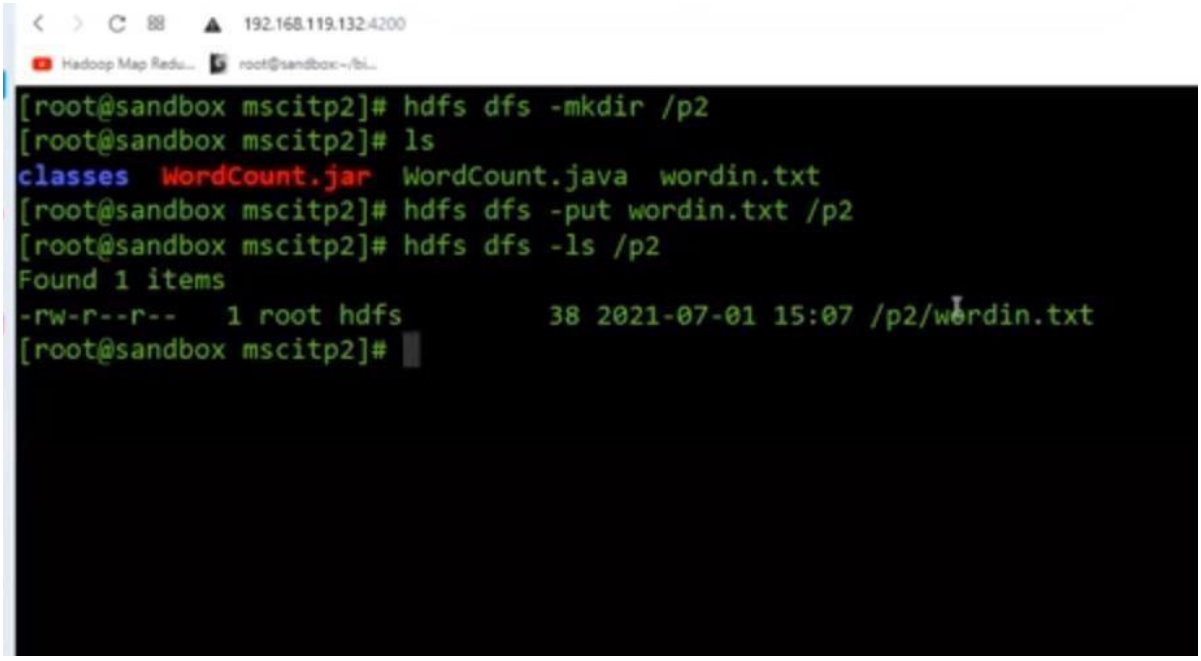
```
jar -cvf WordCount.jar -C classes/ .
```

```
[root@sandbox mscitp2]# ls
WordCount.java  wordin.txt
[root@sandbox mscitp2]# export HADOOP_CLASSPATH=$(hadoop classpath)
[root@sandbox mscitp2]# ls
WordCount.java  wordin.txt
[root@sandbox mscitp2]# mkdir classes
[root@sandbox mscitp2]# javac -classpath ${HADOOP_CLASSPATH} -d classes WordCount.java
[root@sandbox mscitp2]# ls classes
WordCount.class  WordCount$IntSumReducer.class  WordCount$TokenizerMapper.class
[root@sandbox mscitp2]# ls
classes  WordCount.java  wordin.txt
[root@sandbox mscitp2]# jar -cvf WordCount.jar -C classes/ .
added manifest
adding: WordCount$IntSumReducer.class(in = 1739) (out= 742)(deflated 57%)
adding: WordCount$TokenizerMapper.class(in = 1736) (out= 756)(deflated 56%)
adding: WordCount.class(in = 1491) (out= 813)(deflated 45%)
[root@sandbox mscitp2]#
```

Run ls command we can see jar file is created.

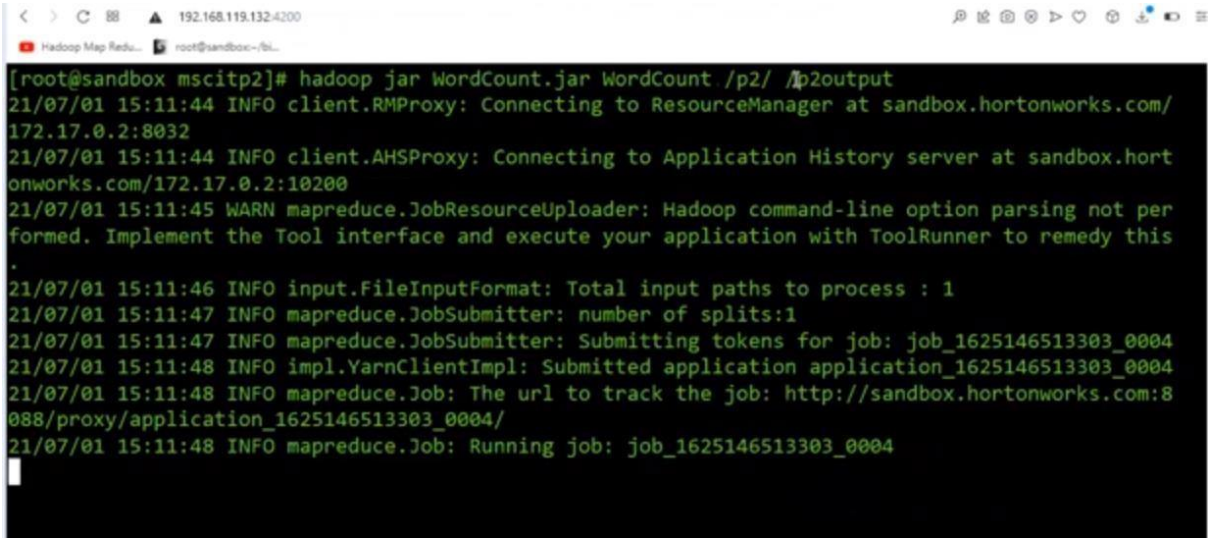
```
[root@sandbox mscitp2]# ls
classes  WordCount.jar  WordCount.java  wordin.txt
[root@sandbox mscitp2]#
```

wordin.txt should be present in word directory of hdfs. So we need to upload wordin.txt file.



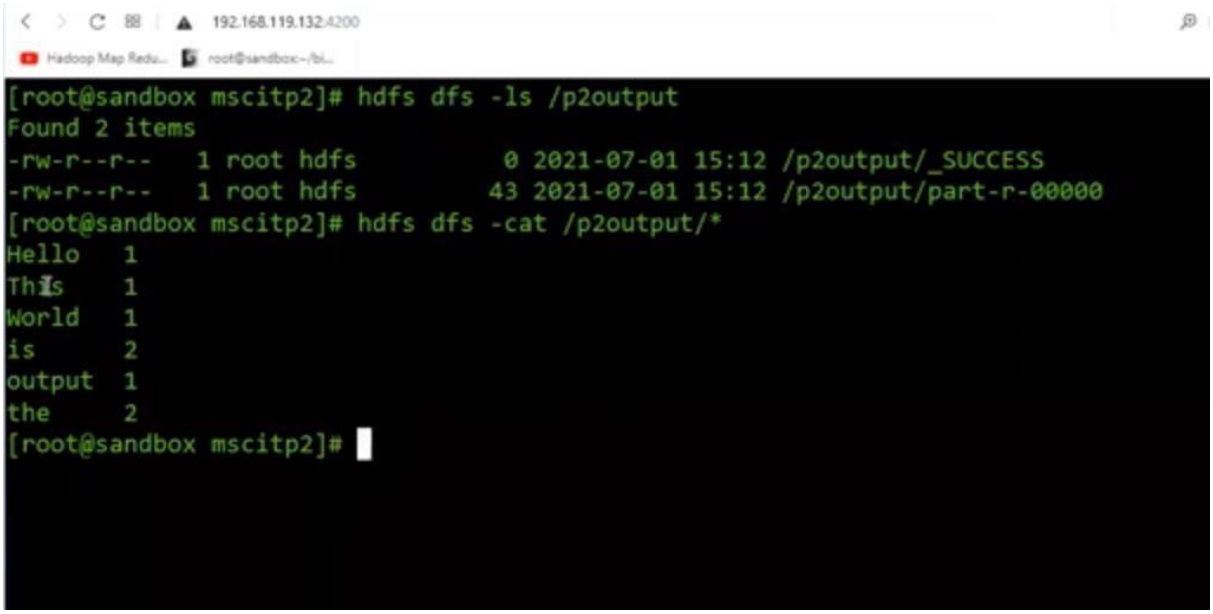
We need to put the final output p2output.

hadoop jar WordCount.jar WordCount /p2/ /p2output



Print the content of the output file

Command: hdfs dfs -cat /p2ouput/*



Ctrl + l to clear the screen.

vi filename.txt= this command will create/ open filename.txt

two modes of vi editor

1) Insert mode – i (press i key)
2) Command mode – esc key

:wq is to save and exit

Practical 3

Implement an MapReduce program that processes a weather dataset.

Java program:

MyMaxMin.java

```
//////////////////////////
// importing Libraries
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.Job;
import  org.apache.hadoop.mapreduce.Mapper;
import  org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.conf.Configuration;

public class MyMaxMin {


        // Mapper


        /*MaxTemperatureMapper class is static
        * and extends Mapper abstract class
        * having four Hadoop generics type
        * LongWritable, Text, Text, Text.
        */
```

```
public static class MaxTemperatureMapper extends
                Mapper<LongWritable, Text, Text, Text> {

        /**
        * @method map
        * This method takes the input as a text data type.
        * Now leaving the first five tokens, it takes
        * 6th token is taken as temp_max and
        * 7th token is taken as temp_min. Now
        * temp_max > 30 and temp_min < 15 are
        * passed to the reducer.
        */

// the data in our data set with
// this value is inconsistent data
public static final int MISSING = 9999;

@Override
        public void map(LongWritable arg0, Text Value, Context context)
                        throws IOException, InterruptedException {

        // Convert the single row(Record) to
        // String and store it in String
        // variable name line

        String line = Value.toString();

                // Check for the empty line
                if (!(line.length() == 0)) {
```

```
                        // from character 6 to 14 we have
                        // the date in our dataset
                        String date = line.substring(6, 14);

                        // similarly we have taken the maximum
                        // temperature from 39 to 45 characters
                        float temp_Max = Float.parseFloat(line.substring(39, 45).trim());

                        // similarly we have taken the minimum
                        // temperature from 47 to 53 characters

                        float temp_Min = Float.parseFloat(line.substring(47, 53).trim());

                        // if maximum temperature is
                        // greater than 30, it is a hot day
                        if (temp_Max > 30.0) {

                                // Hot day
                                context.write(new Text("The Day is Hot Day :" + date),
                                                                        new
Text(String.valueOf(temp_Max)));
                        }

                        // if the minimum temperature is
                        // less than 15, it is a cold day
                        if (temp_Min < 15) {

                                // Cold day
                                context.write(new Text("The Day is Cold Day :" + date),
                                                new Text(String.valueOf(temp_Min)));
                        }
```

```
                }
            }

        }

// Reducer

        /*MaxTemperatureReducer class is static
        and extends Reducer abstract class
        having four Hadoop generics type
        Text, Text, Text, Text.
        */
        //The Day is Cold Day :20150101 ,-21.8

        public static class MaxTemperatureReducer extends
                Reducer<Text, Text, Text, Text> {

            /**
            * @method reduce
            * This method takes the input as key and
            * list of values pair from the mapper,
            * it does aggregation based on keys and
            * produces the final context.
            */

            public void reduce(Text Key, Iterator<Text> Values, Context context)
                    throws IOException, InterruptedException {
                // putting all the values in
                // temperature variable of type String
                String temperature = Values.next().toString();
                context.write(Key, new Text(temperature));
```

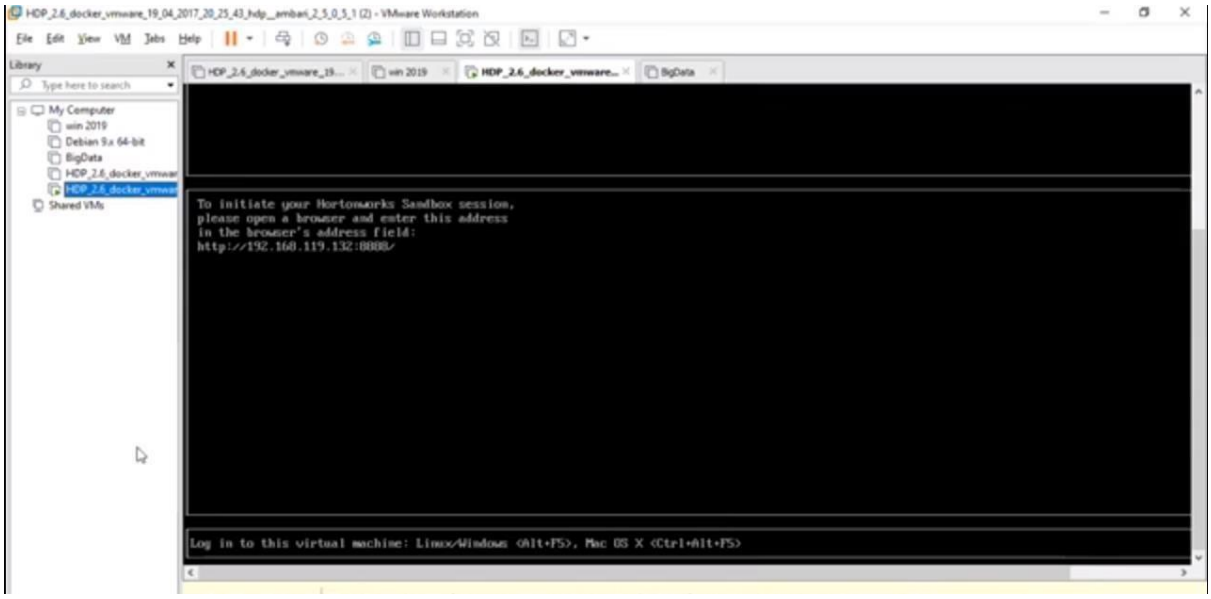
```
        }

}



/**
* @method main
* This method is used for setting
* all the configuration properties.
* It acts as a driver for map-reduce
* code.
*/

public static void main(String[] args) throws Exception {

        // reads the default configuration of the
        // cluster from the configuration XML files
        Configuration conf = new Configuration();

        // Initializing the job with the
        // default configuration of the cluster
        Job job = new Job(conf, "weather example");

        // Assigning the driver class name
        job.setJarByClass(MyMaxMin.class);

        // Key type coming out of mapper
        job.setMapOutputKeyClass(Text.class);

        // value type coming out of mapper
```

```
job.setMapOutputValueClass(Text.class);

// Defining the mapper class name
job.setMapperClass(MaxTemperatureMapper.class);

// Defining the reducer class name
job.setReducerClass(MaxTemperatureReducer.class);

// Defining input Format class which is
// responsible to parse the dataset
// into a key value pair
job.setInputFormatClass(TextInputFormat.class);

// Defining output Format class which is
// responsible to parse the dataset
// into a key value pair
job.setOutputFormatClass(TextOutputFormat.class);

// setting the second argument
// as a path in a path variable
Path OutputPath = new Path(args[1]);

// Configuring the input path
// from the filesystem into the job
FileInputFormat.addInputPath(job, new Path(args[0]));

// Configuring the output path from
// the filesystem into the job
FileOutputFormat.setOutputPath(job, new Path(args[1]));

// deleting the context path automatically
```

```
            // from hdfs so that we don't have
            // to delete it explicitly
            OutputPath.getFileSystem(conf).delete(OutputPath);

            // exiting the job only if the
            // flag value becomes false
            System.exit(job.waitForCompletion(true) ? 0 : 1);

        }
}
```

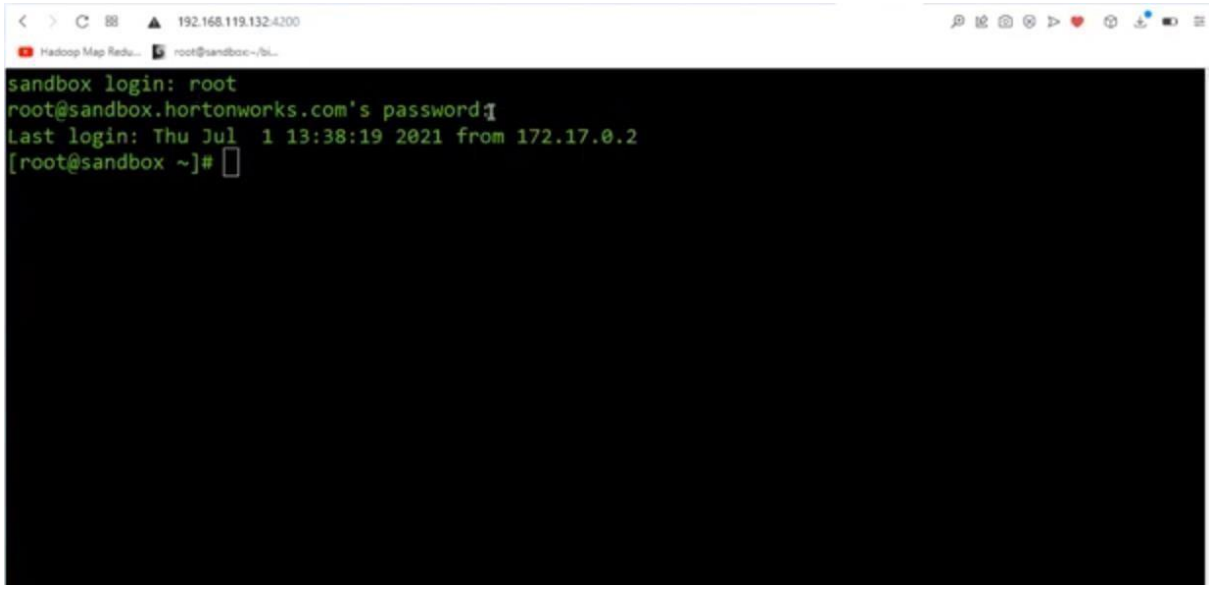
//////////////////////

Start the server



Open the terminal with 192.168.119.132/4200

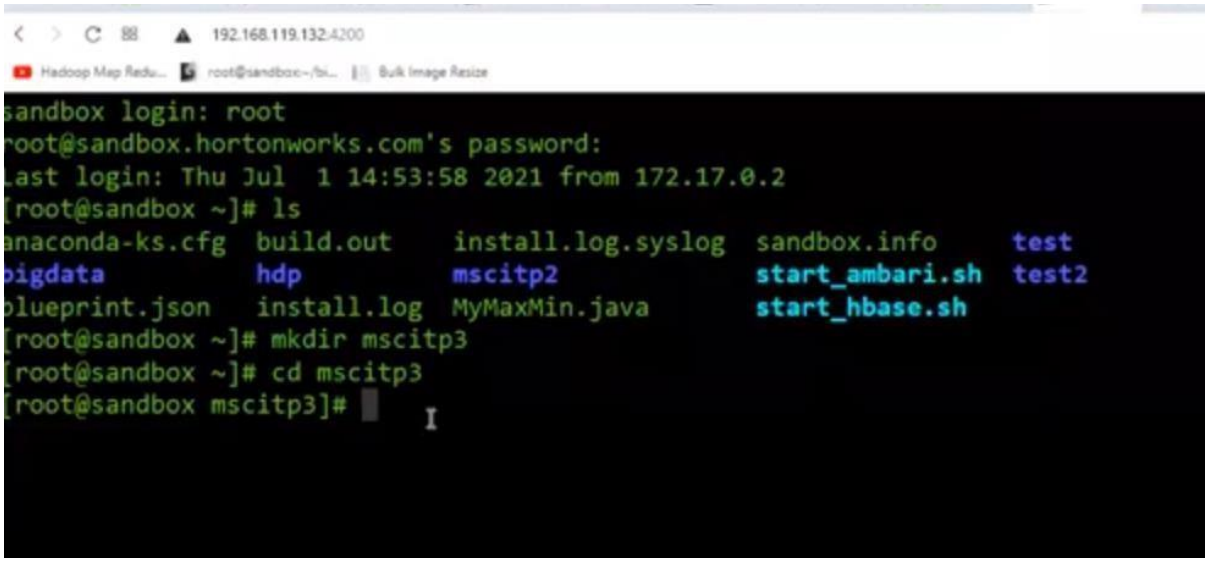
Enter the login: root and the password and enter



Create a folder in local directory.

Command: mkdir mscitp3

Change the directory cd mscitp3



Now create input file

Command: cat >> weatherin2.txt

Paste the weather dataset by right clicking on terminal

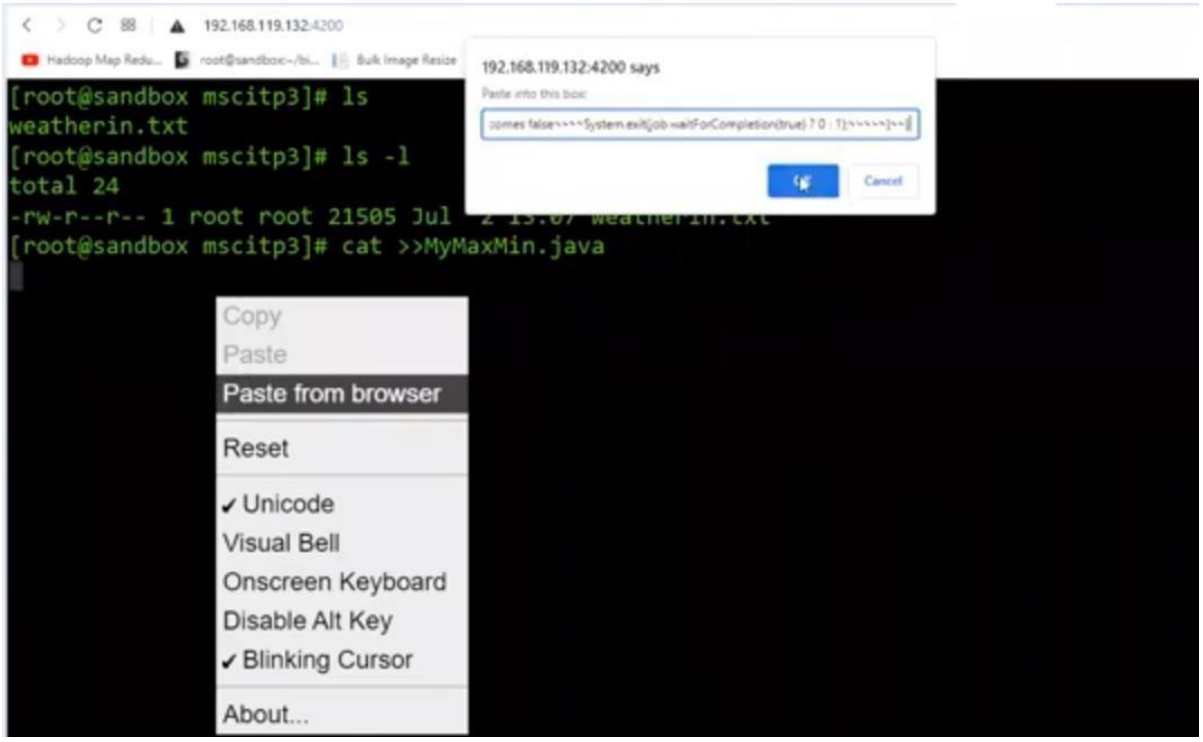
Ctrl d will save the file

Run command ls to see the file.

Create java file

Command: cat >>MyMaxMin.java

Paste the java code and ctrl d to save the file



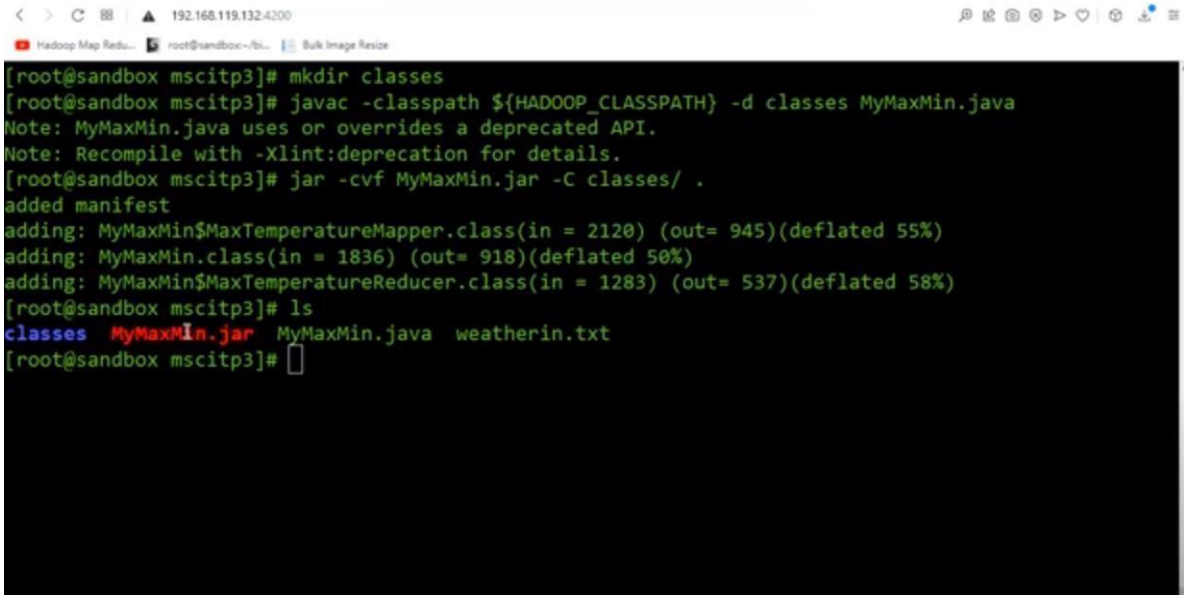
export HADOOP_CLASSPATH=$(hadoop classpath) ////compile and to create jar file

mkdir classes

javac -classpath ${HADOOP_CLASSPATH} -d classes MyMaxMin.java

After compile need to create a jar file

jar -cvf MyMaxMin.jar -C classes/ .



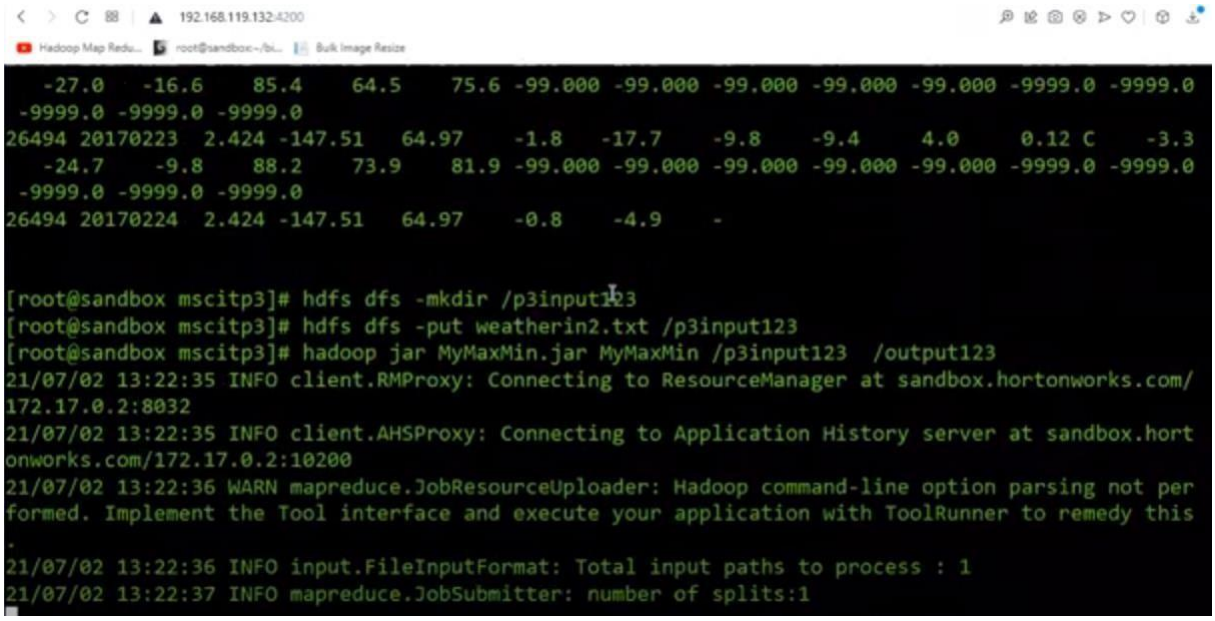
Now, put weatherin.txt in hdfs

Before that create a folder

Command: hdfs dfs -mkdir /p3input123

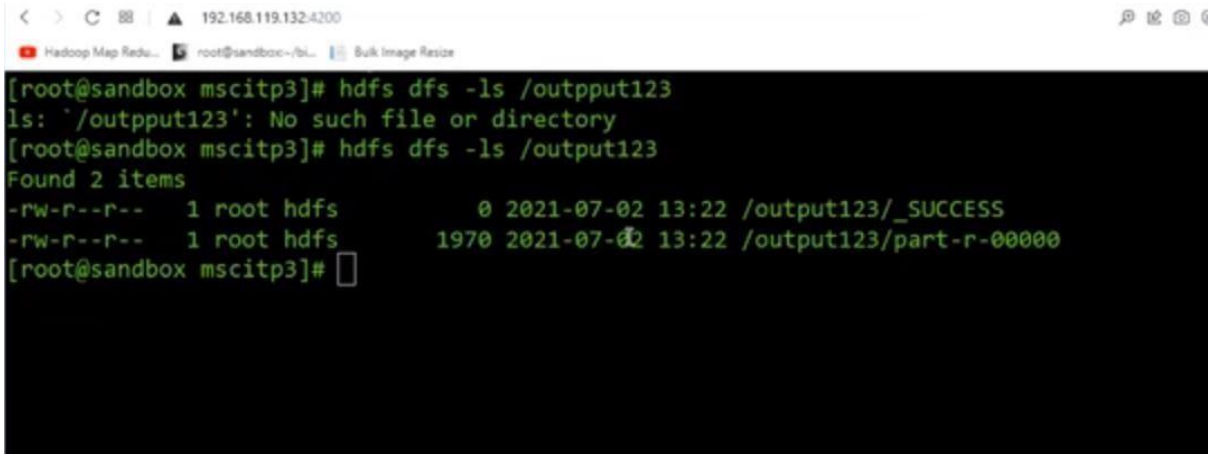
Then run command: hdfs dfs -put weatherin2.txt /p3input123

hadoop jar MyMaxMin.jar MyMaxMin /p3inputw /output123

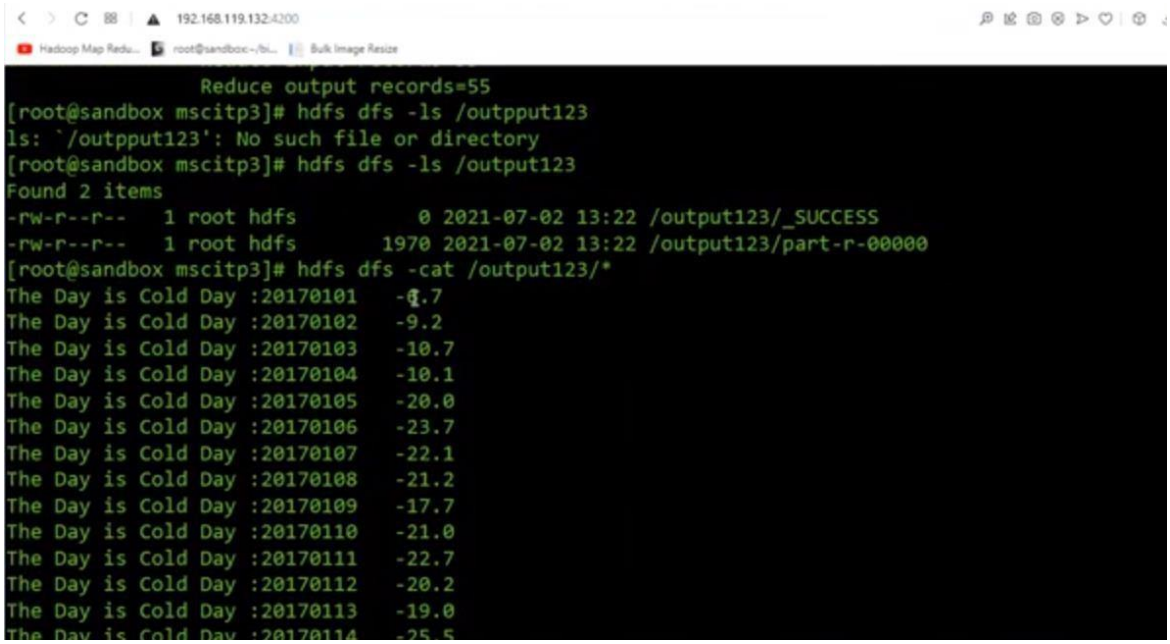


Check outfile is created

Command: hdfs dfs -ls /output123



hdfs dfs -cat /output123/*


192.168.119.132:4200
Hadoop Map Redu...
root@sandbox:~/bi...
Bulk Image Resize
Reduce output records=55
[root@sandbox mscitp3]# hdfs dfs -ls /outpput123
ls: `/outpput123': No such file or directory
[root@sandbox mscitp3]# hdfs dfs -ls /output123
Found 2 items
-rw-r--r--   1 root hdfs   0 2021-07-02 13:22 /output123/_SUCCESS
-rw-r--r--   1 root hdfs   1970 2021-07-02 13:22 /output123/part-r-00000
[root@sandbox mscitp3]# hdfs dfs -cat /output123/*
The Day is Cold Day :20170101   -6.7
The Day is Cold Day :20170102   -9.2
The Day is Cold Day :20170103   -10.7
The Day is Cold Day :20170104   -10.1
The Day is Cold Day :20170105   -20.0
The Day is Cold Day :20170106   -23.7
The Day is Cold Day :20170107   -22.1
The Day is Cold Day :20170108   -21.2
The Day is Cold Day :20170109   -17.7
The Day is Cold Day :20170110   -21.0
The Day is Cold Day :20170111   -22.7
The Day is Cold Day :20170112   -20.2
The Day is Cold Day :20170113   -19.0

Practical 4

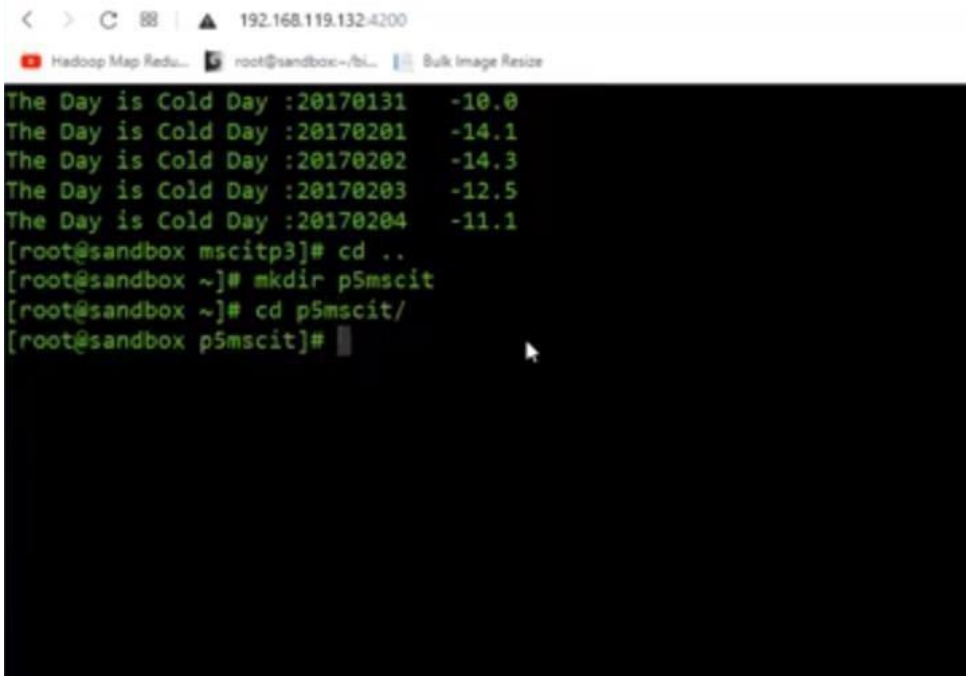
Implement the program using Pig.

Dataset:

```
001,Rajiv,Reddy,21,9848022337,Hyderabad
002,siddarth,Battacharya,22,9848022338,Kolkata
003,Rajesh,Khanna,22,9848022339,Delhi
004,Preethi,Agarwal,21,9848022330,Pune
005,Trupthi,Mohanthy,23,9848022336,Bhuwaneshwar
006,Archana,Mishra,23,9848022335,Chennai
007,Komal,Nayak,24,9848022334,trivendram
008,Bharathi,Nambiayar,24,9848022333,Chennai
#student.txt
```

create a directory and get into that directory

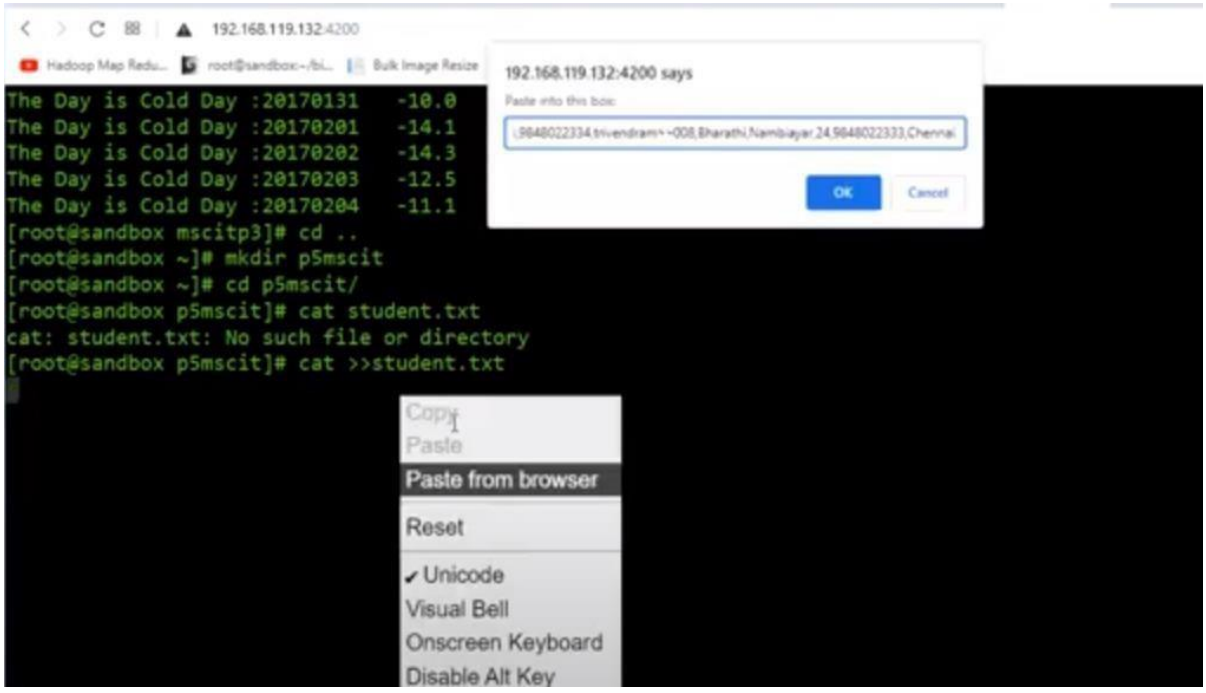
Command: mkdir p5mscit

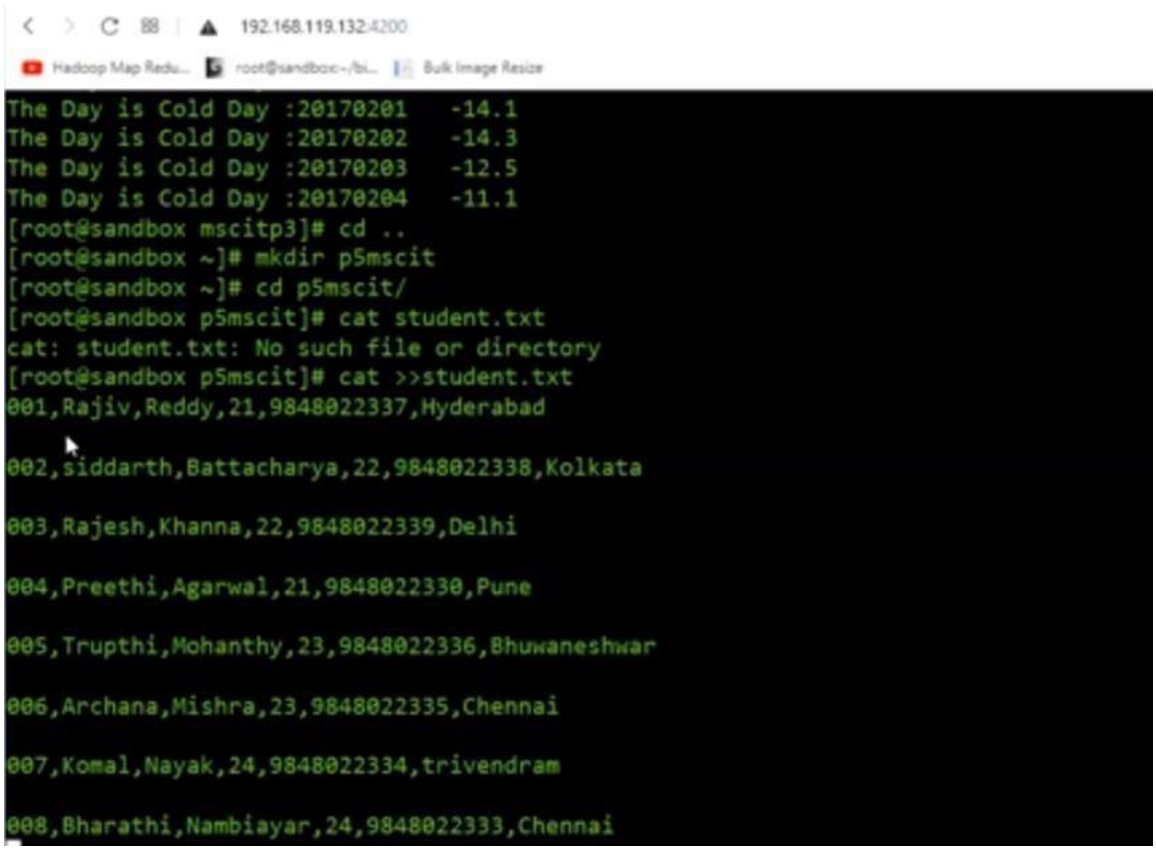


Create a file

Command: cat >>student.txt

Right click and paste the text



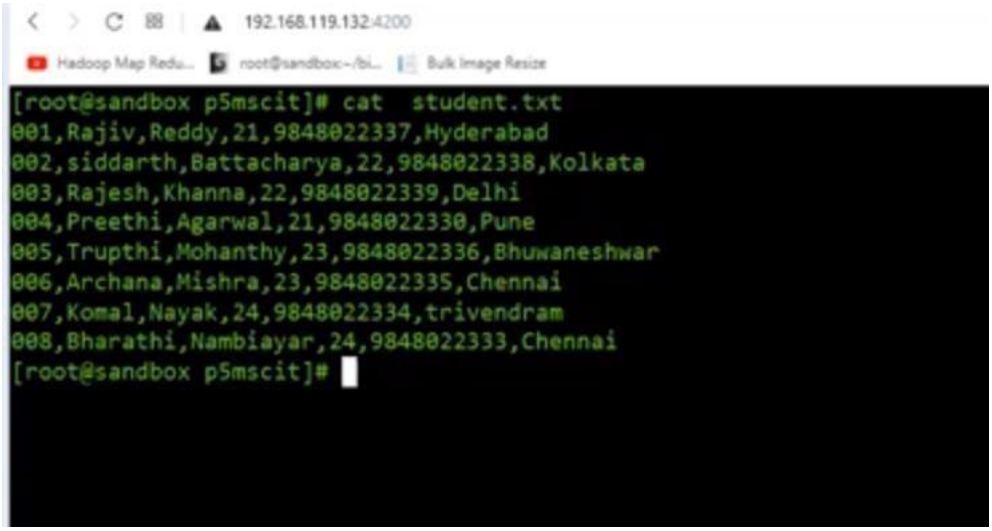


Remove the space with vi editor

Command: vi student.txt and press i for insert mode

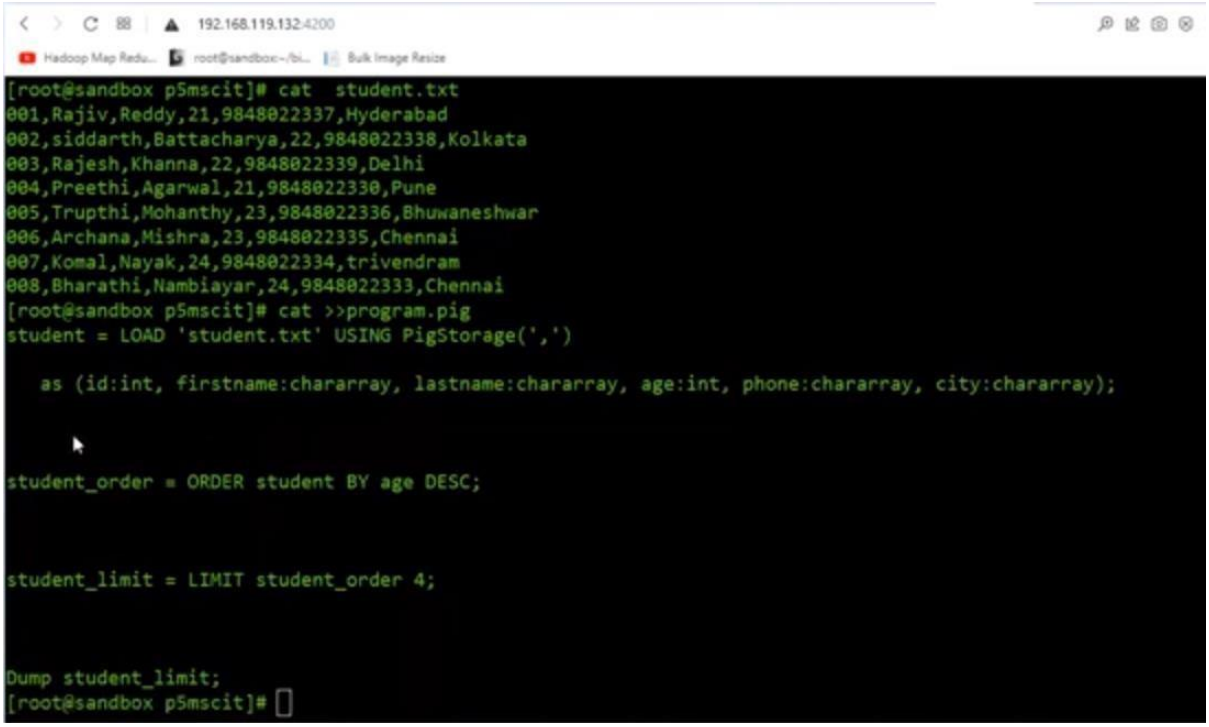
After editing: wq and enter

Print the content and see the text



Create a program file

```
//////////////////////////script start
student = LOAD 'student.txt' USING PigStorage(',')
   as (id:int, firstname:chararray, lastname:chararray, age:int,
phone:chararray, city:chararray);

student_order = ORDER student BY age DESC;

student_limit = LIMIT student_order 4;

Dump student_limit;
////////script end
```



Upload student on hdfs

Command: hdfs dfs -put student.txt /user/root/

Run the pig program

```
[root@sandbox p5mscit]# hdfs dfs -put student.txt /user/root/
[root@sandbox p5mscit]# pig program.pig
```

Output:

```
dent.txt"

Output(s):
Successfully stored 4 records (211 bytes) in: "hdfs://sandbox.hortonworks.com:8020/tmp/temp81689
1614/tmp500376273"

2021-07-02 13:51:34,079 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Tot
al input paths to process : 1
2021-07-02 13:51:34,079 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUt
il - Total input paths to process : 1
(7,Komal,Nayak,24,9848022334,trivendram)
(8,Bharathi,Nambiayar,24,9848022333,Chennai)
(5,Trupthi,Mohanthy,23,9848022336,Bhuwaneshwar)
(6,Archana,Mishra,23,9848022335,Chennai)
2021-07-02 13:51:34,563 [main] INFO  org.apache.pig.Main - Pig script completed in 39 seconds an
d 726 milliseconds (39726 ms)
2021-07-02 13:51:34,575 [main] INFO  org.apache.pig.backend.hadoop.executionengine.tez.TezLaunch
er - Shutting down thread pool
2021-07-02 13:51:34,608 [pool-1-thread-1] INFO  org.apache.pig.backend.hadoop.executionengine.te
z.TezSessionManager - Shutting down Tez session org.apache.tez.client.TezClient@132aba44
2021-07-02 13:51:34 Shutting down Tez session , sessionName=PigLatin:program.pig, applicationId=
application_1625229346715_0004
```

Practical 5

Implement the application in Hive.

Dataset:

```
001,Rajiv,Reddy,21,9848022337,Hyderabad
002,siddarth,Battacharya,22,9848022338,Kolkata
003,Rajesh,Khanna,22,9848022339,Delhi
004,Preethi,Agarwal,21,9848022330,Pune
005,Trupthi,Mohanthy,23,9848022336,Bhuwaneshwar
006,Archana,Mishra,23,9848022335,Chennai
007,Komal,Nayak,24,9848022334,trivendram
008,Bharathi,Nambiayar,24,9848022333,Chennai
#student.txt
```

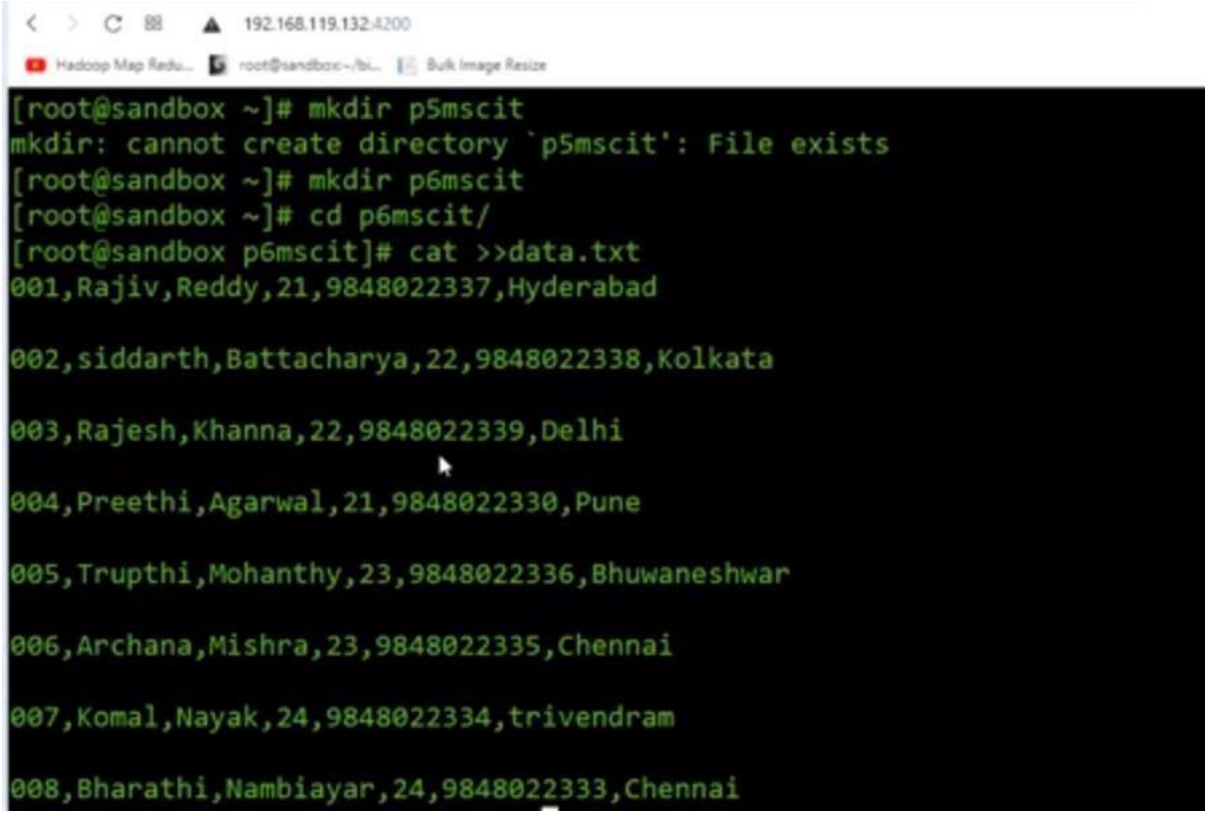
create a directory and get into that directory

Command: mkdir p6mscit

Create a file

Command: cat >>data.txt

Right click and paste the text



Remove the space with vi editor

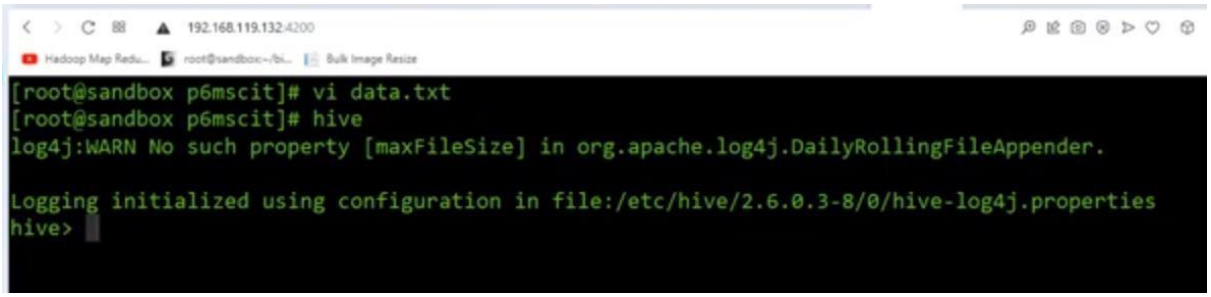
Command: vi student.txt and press i for insert mode

After editing: wq and enter

Print the content and see the text

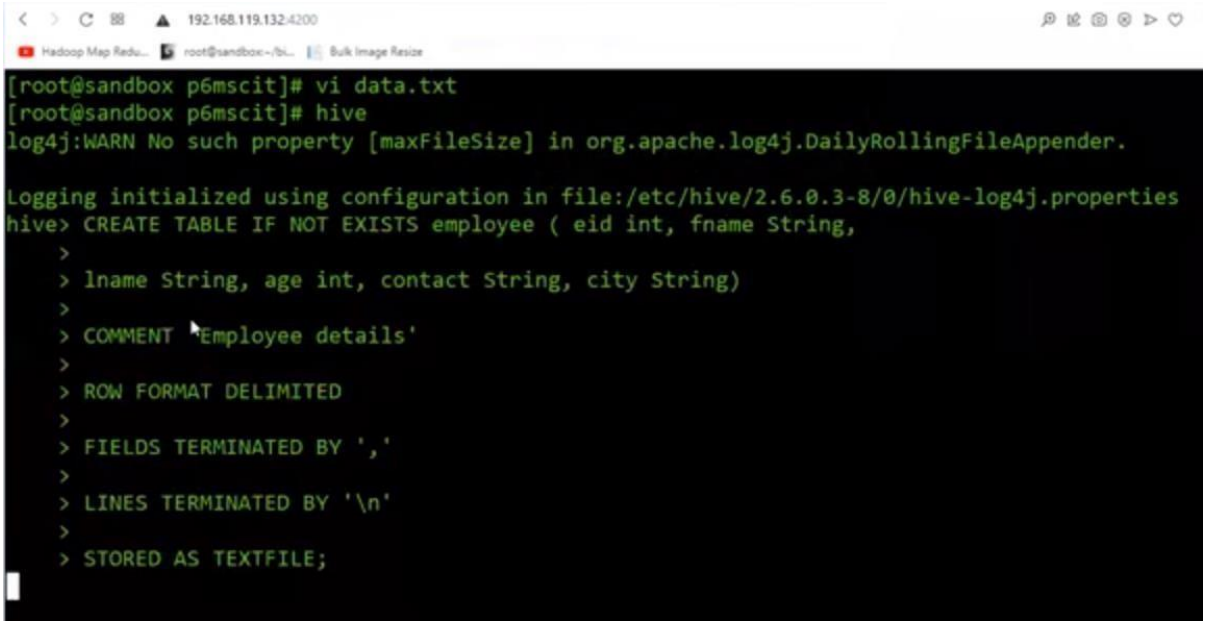
Now start the hive terminal

Command: hive



Copy paste below command on hive and enter

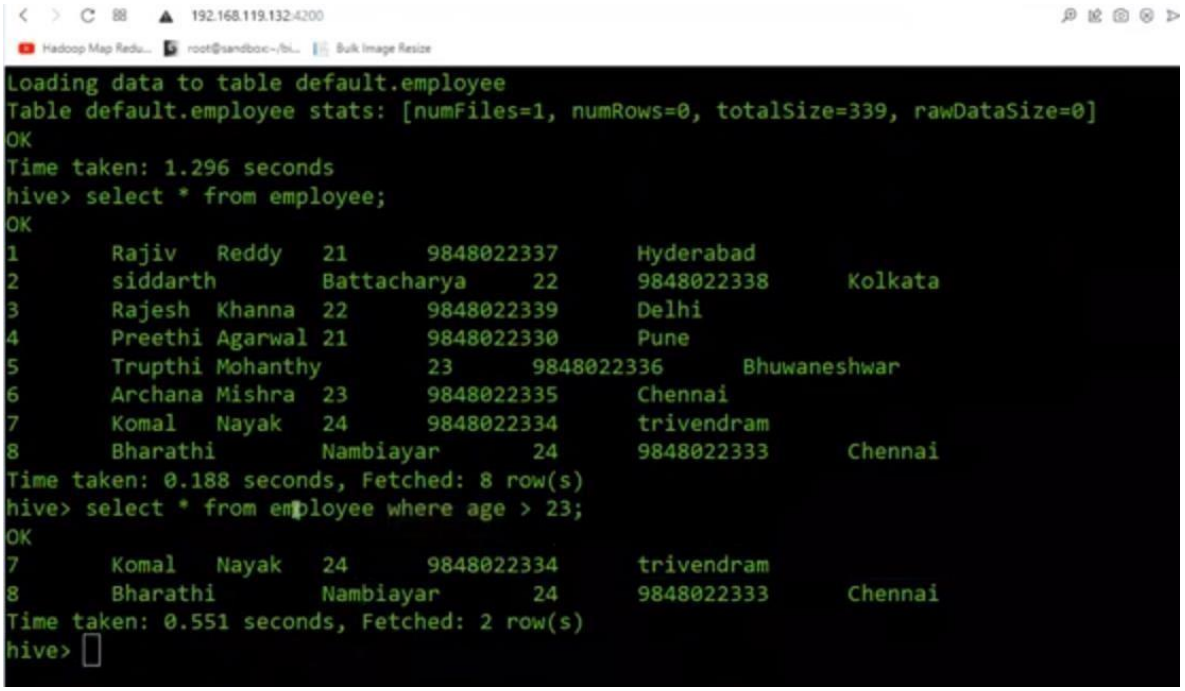
```
create table
CREATE TABLE IF NOT EXISTS employee ( eid int, fname String,
lname String, age int, contact String, city String)
COMMENT 'Employee details'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED AS TEXTFILE;
```




```
Run command: LOAD DATA LOCAL INPATH 'data.txt' OVERWRITE INTO TABLE
employee;
```

```
hive> LOAD DATA LOCAL INPATH 'data.txt' OVERWRITE INTO TABLE employee;
Loading data to table default.employee
Table default.employee stats: [numFiles=1, numRows=0, totalSize=339, rawDataSize=0]
OK
Time taken: 1.296 seconds
hive> ;
```

Run the command like select * from employee;



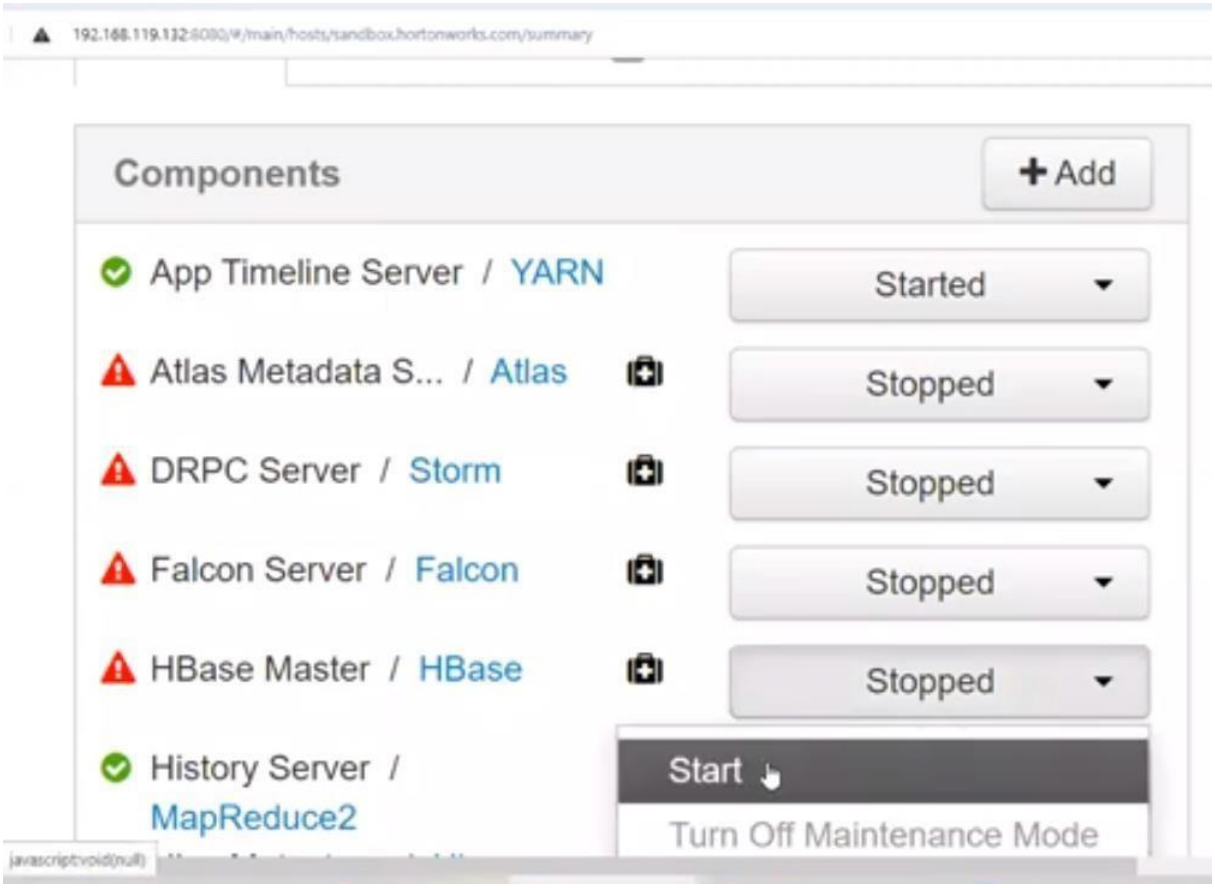
Practical 6

Implement an application that stores big data in Hbase/ Python

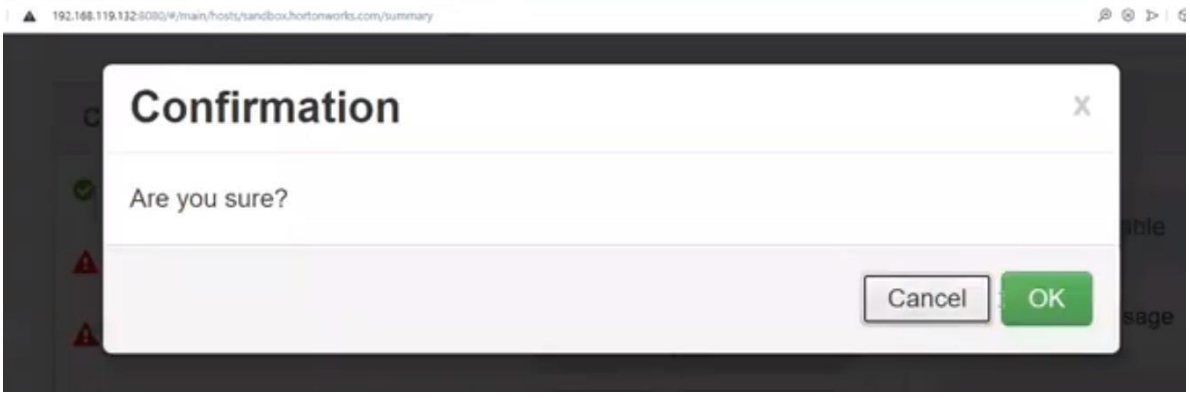
What is HBase?

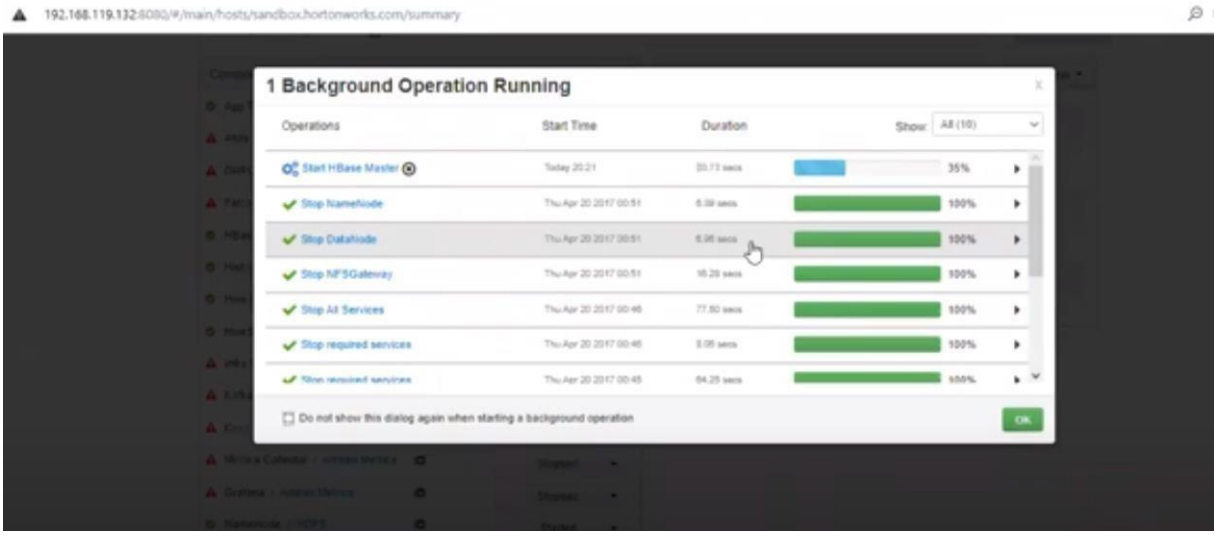
HBase is a distributed column-oriented database built on top of the Hadoop file system. It is an open-source project and is horizontally scalable. It is a part of the Hadoop ecosystem that provides random real-time read/write access to data in the Hadoop File System.

Go to GUI page and start the hbase service.

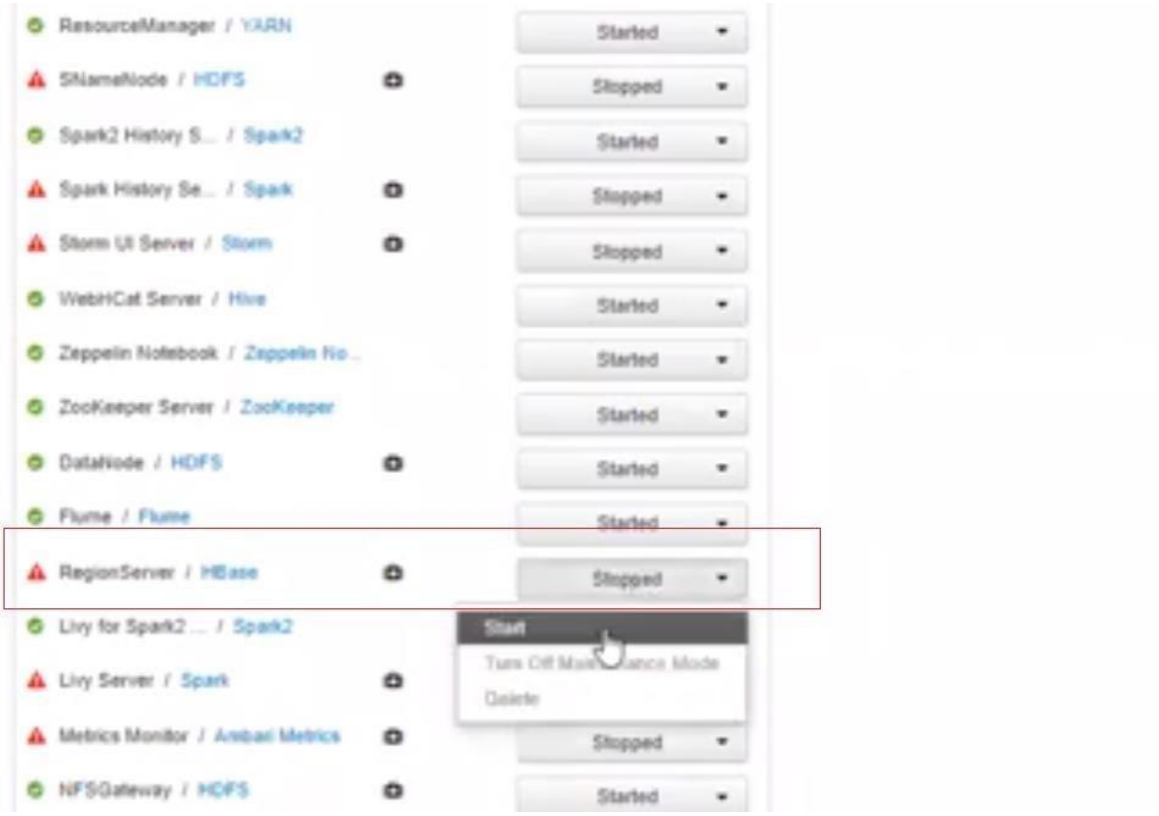


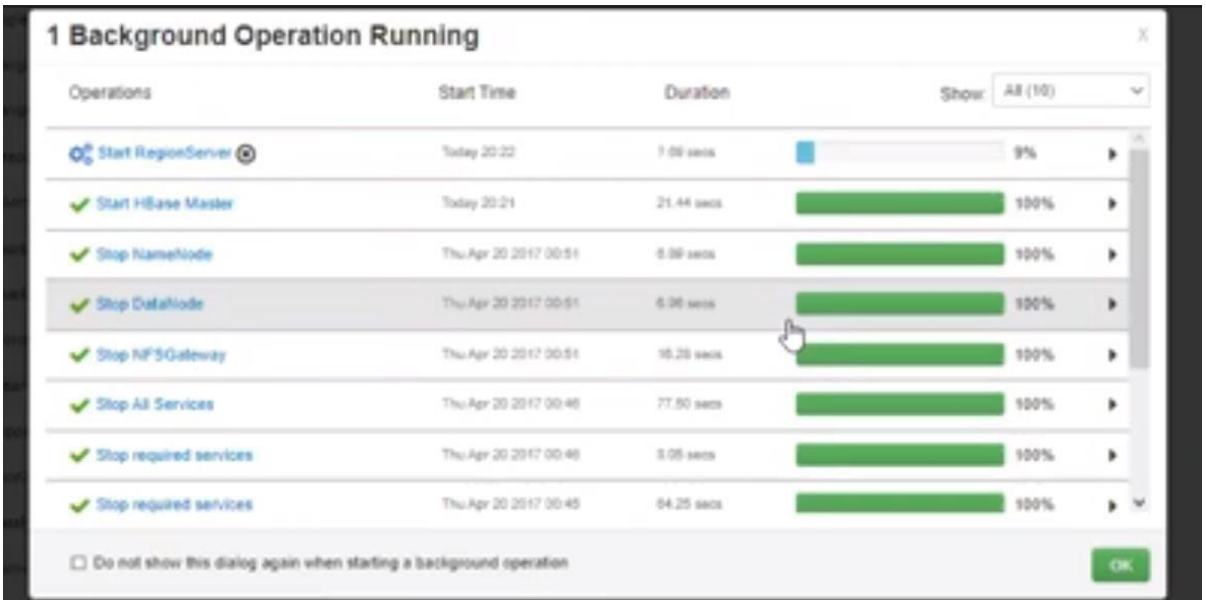
Click on OK to start the service.



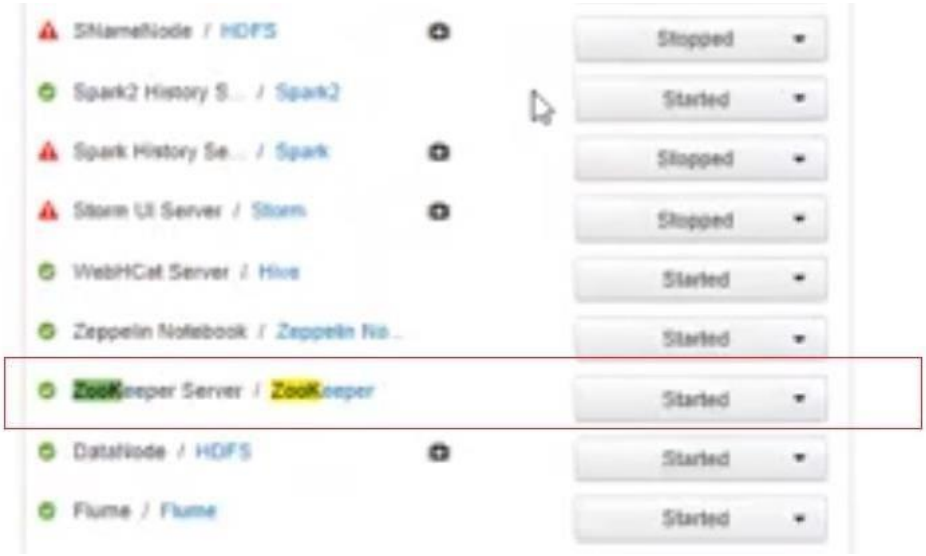


Now we must start region server.

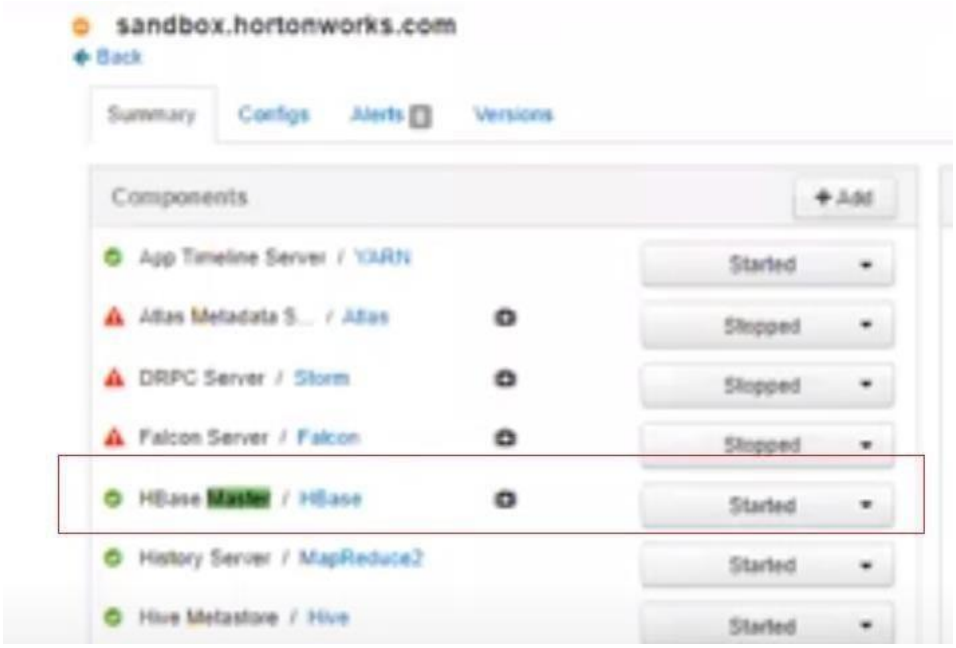




Check zooperkeeper server is started.



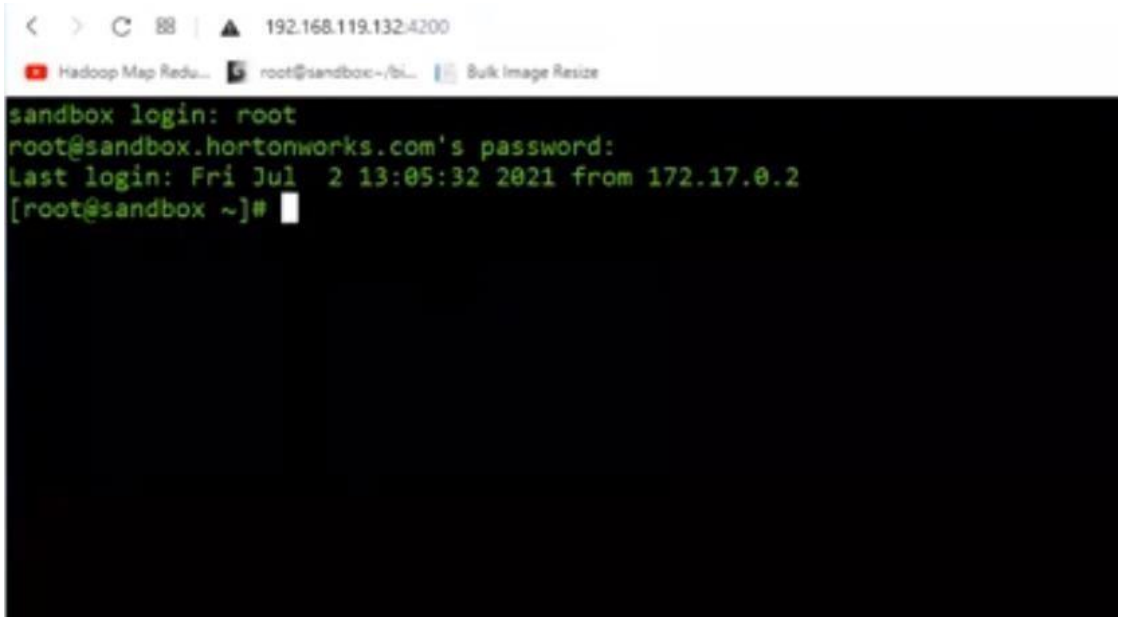
Check hbase and region server are started.



Command: **which application-name** gives directory in which application-name is installed.

Open the shell

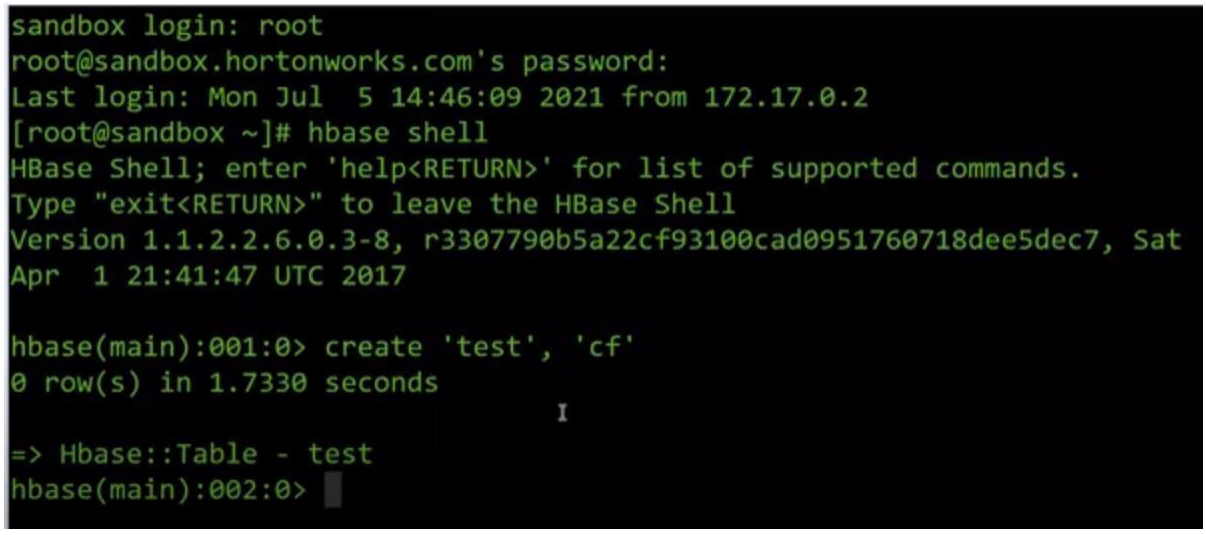
192.168.119.132:4200



Command: hbase shell

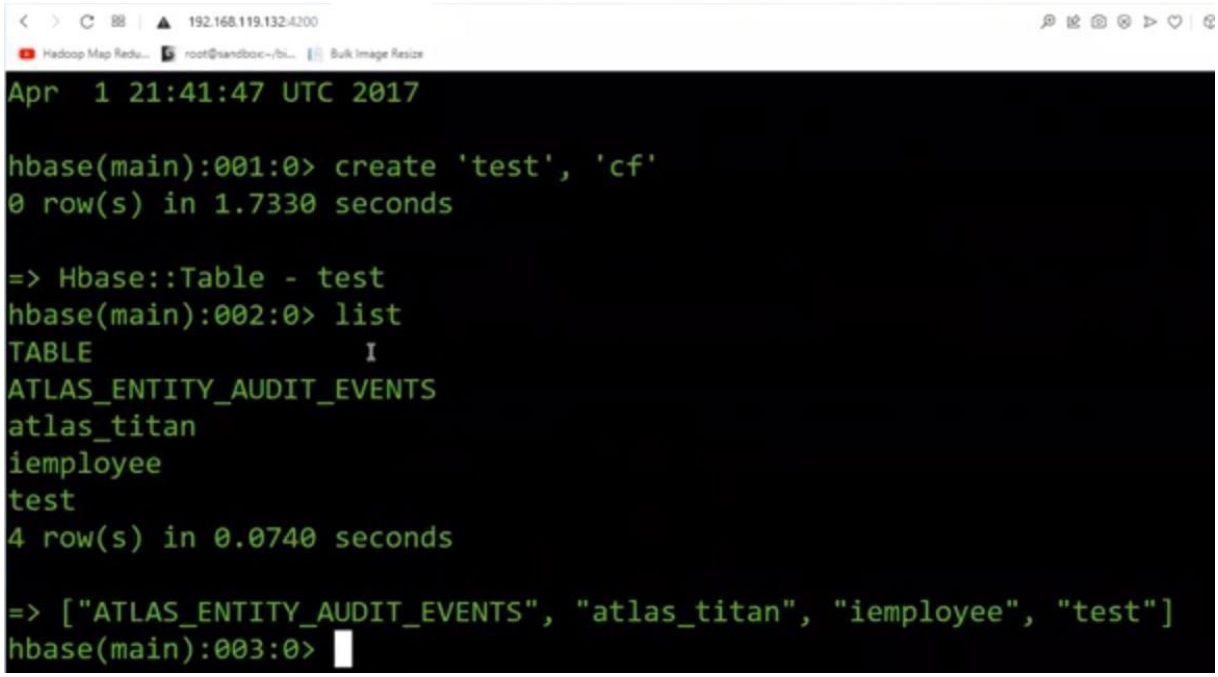
It will start the server

Enter the command create ‘test’, ‘cf’ and it will create the table



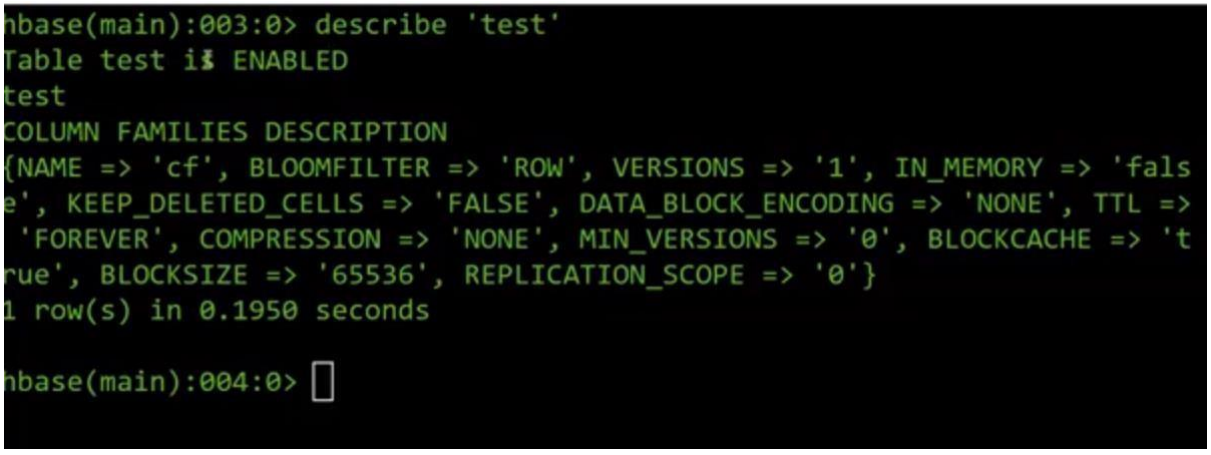
Check the table is created with command

List- It will list all the tables created.



If we want to see column description of a table.

Command- describe tablename



Now, we have to put the values in table

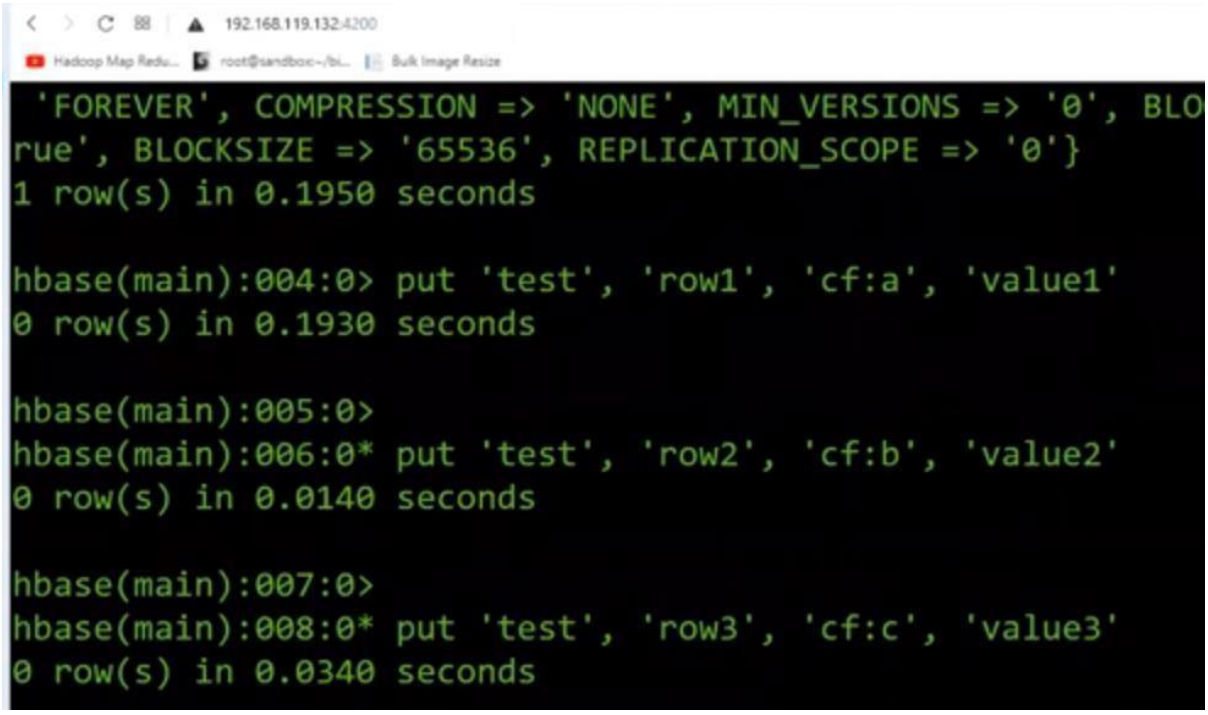
Values:

put ‘test’, ‘row1’, ‘cf:a’, ‘value1’

put ‘test’, ‘row2’, ‘cf:b’, ‘value2’

put ‘test’, ‘row3’, ‘cf:c’, ‘value3’

copy paste the data in shell.



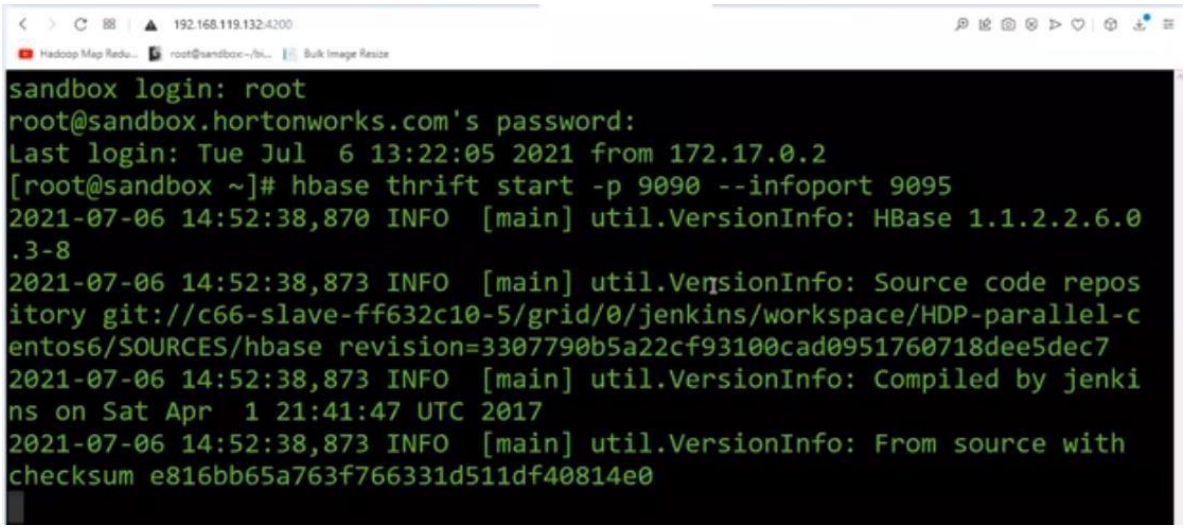
We to display the records of table

Command: scan 'test'

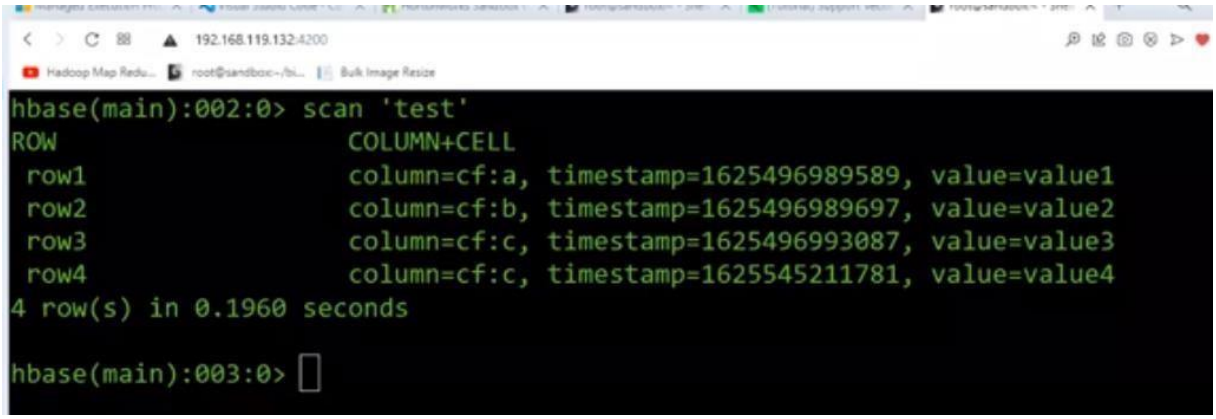
Python: storage/retrieval

Start the service with command

Hbase thrift start -p 9090 –inforport 9095



Create the table the way we did it in hbase and see the records using scan command



Create a program file

Import happybase as hb

conn=hb.connection('192.168.119.132', 9090)

print(conn.table('test').row('row1')

print(conn.table('test').row('row2')

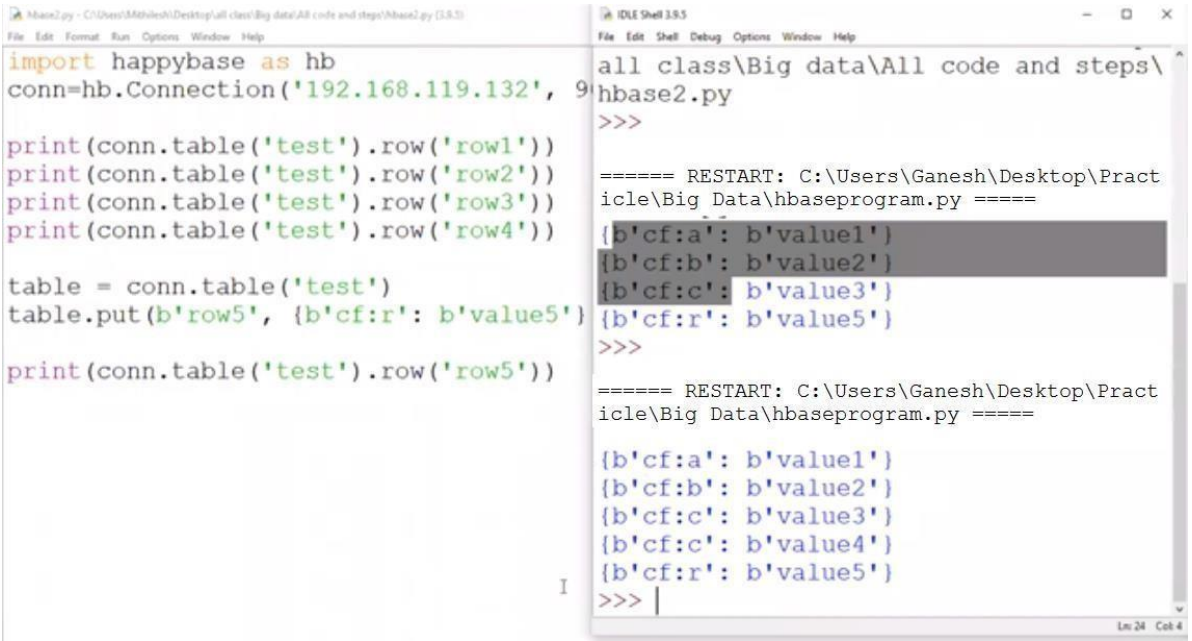
print(conn.table('test').row('row3')

print(conn.table('test').row('row4')

table = conn.table('test')

table.put(b'row5', {b'cf:r': b'value5'})

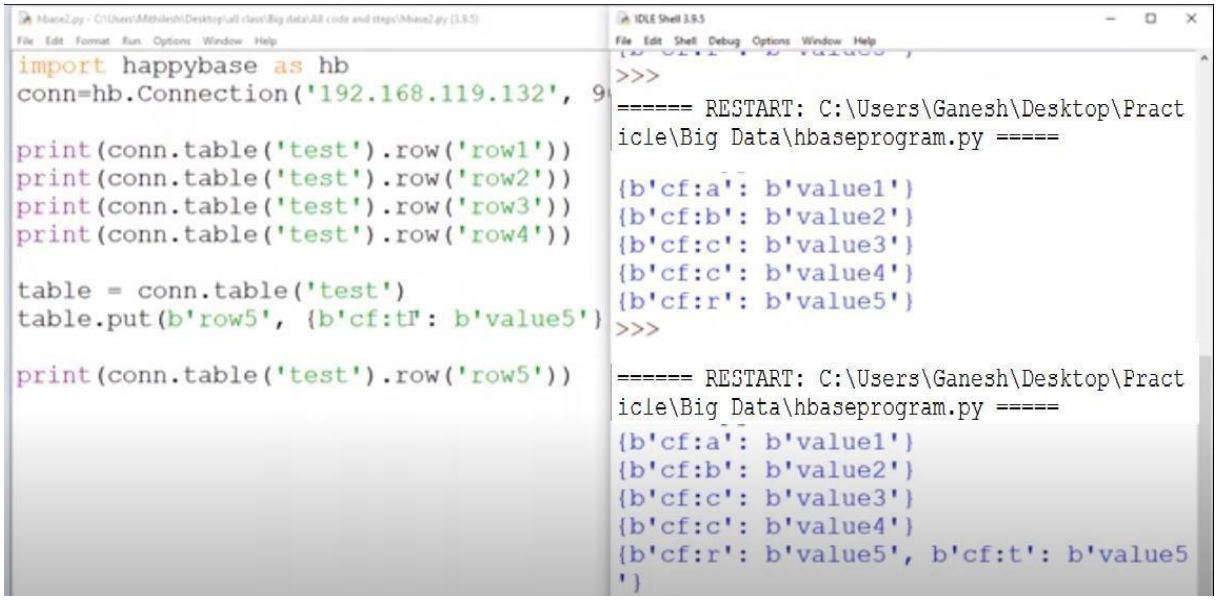
print(conn.table('test').row('row5')



Run a scan command on shell to display the values

```
hbase(main):004:0> scan 'test'
ROW                   COLUMN+CELL
 row1                 column=cf:a, timestamp=1625496989589, value=value1
 row2                 column=cf:b, timestamp=1625496989697, value=value2
 row3                 column=cf:c, timestamp=1625496993087, value=value3
 row4                 column=cf:c, timestamp=1625545211781, value=value4
 row5                 column=cf:r, timestamp=1625583481042, value=value5
5 row(s) in 0.0320 seconds
```

Now, try with duplicate value at row 5 say value t



Run a scan command on shell to display the values

When there is unique value, it will create a record. If duplicate value it will not create a record

```
hbase(main):005:0> scan 'test'
ROW                   COLUMN+CELL
 row1                 column=cf:a, timestamp=1625496989589, value=value1
 row2                 column=cf:b, timestamp=1625496989697, value=value2
 row3                 column=cf:c, timestamp=1625496993087, value=value3
 row4                 column=cf:c, timestamp=1625545211781, value=value4
 row5                 column=cf:r, timestamp=1625583481042, value=value5
 row5                 column=cf:t, timestamp=1625583505297, value=value5
5 row(s) in 0.1320 seconds
```

Practical 7

Implement Decision tree classification techniques

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation.

Using the Iris dataset, we can construct a tree as follows:

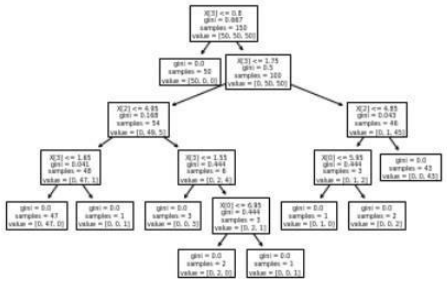
+ Code + Text

```
from sklearn.datasets import load_iris
from sklearn import tree
iris = load_iris()
X, y = iris.data, iris.target
clf = tree.DecisionTreeClassifier()
clf = clf.fit(X, y)
```

Once trained, we can plot the tree with the `plot_tree` function:

```
[4]  tree.plot_tree(clf)
```

```
[Text(167.4, 199.32, 'X[3] <= 0.8\ngini = 0.667\nsamples = 150\nvalue = [50, 50, 50]'),
 Text(141.64615384615385, 163.07999999999998, 'gini = 0.0\nsamples = 50\nvalue = [50, 0, 0]'),
 Text(193.15384615384616, 163.07999999999998, 'X[3] <= 1.75\ngini = 0.5\nsamples = 100\nvalue = [0, 50, 50]'),
 Text(103.01538461538462, 126.83999999999999, 'X[2] <= 4.95\ngini = 0.168\nsamples = 54\nvalue = [0, 49, 5]'),
 Text(51.50769230769231, 90.6, 'X[3] <= 1.65\ngini = 0.041\nsamples = 48\nvalue = [0, 47, 1]'),
 Text(25.753846153846155, 54.359999999999985, 'gini = 0.0\nsamples = 47\nvalue = [0, 47, 0]'),
 Text(77.26153846153846, 54.359999999999985, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 1]'),
 Text(154.52307692307693, 90.6, 'X[3] <= 1.55\ngini = 0.444\nsamples = 6\nvalue = [0, 2, 4]'),
 Text(128.76923076923077, 54.359999999999985, 'gini = 0.0\nsamples = 3\nvalue = [0, 0, 3]'),
 Text(180.27692307692308, 54.359999999999985, 'X[0] <= 6.95\ngini = 0.444\nsamples = 3\nvalue = [0, 2, 1]'),
 Text(154.52307692307693, 18.119999999999976, 'gini = 0.0\nsamples = 2\nvalue = [0, 2, 0]'),
 Text(206.03076923076924, 18.119999999999976, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 1]'),
 Text(283.2923076923077, 126.83999999999999, 'X[2] <= 4.85\ngini = 0.043\nsamples = 46\nvalue = [0, 1, 45]'),
 Text(257.53846153846155, 90.6, 'X[0] <= 5.95\ngini = 0.444\nsamples = 3\nvalue = [0, 1, 2]'),
 Text(231.7846153846154, 54.359999999999985, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0]'),
 Text(283.2923076923077, 54.359999999999985, 'gini = 0.0\nsamples = 2\nvalue = [0, 0, 2]'),
 Text(309.04615384615386, 90.6, 'gini = 0.0\nsamples = 43\nvalue = [0, 0, 43]')]
```



Practical 8

Implement SVM classification techniques

Support Vector Machines

Generally, Support Vector Machines is considered to be a classification approach, it but can be employed in both types of classification and regression problems. It can easily handle multiple continuous and categorical variables. SVM constructs a hyperplane in multidimensional space to separate different classes. SVM generates optimal hyperplane in an iterative manner, which is used to minimize an error. The core idea of SVM is to find a maximum marginal hyperplane (MMH) that best divides the dataset into classes.

Loading data:

```
[1]  #Import scikit-learn dataset library
     from sklearn import datasets

     #Load dataset
     cancer = datasets.load_breast_cancer()
```

Exploring data:

```
# print the names of the 13 features
print("Features: ", cancer.feature_names)

# print the label type of cancer('malignant' 'benign')
print("Labels: ", cancer.target_names)
```

```
Features:  ['mean radius' 'mean texture' 'mean perimeter' 'mean area'
 'mean smoothness' 'mean compactness' 'mean concavity'
 'mean concave points' 'mean symmetry' 'mean fractal dimension'
 'radius error' 'texture error' 'perimeter error' 'area error'
 'smoothness error' 'compactness error' 'concavity error'
 'concave points error' 'symmetry error' 'fractal dimension error'
 'worst radius' 'worst texture' 'worst perimeter' 'worst area'
 'worst smoothness' 'worst compactness' 'worst concavity'
 'worst concave points' 'worst symmetry' 'worst fractal dimension']
Labels:  ['malignant' 'benign']
```

Check the shape of the dataset using shape.

```
# print data(feature)shape
cancer.data.shape
```

```
(569, 30)
```

Check top 5 records of the feature set.

```
[4]  # print the cancer data features (top 5 records)
     print(cancer.data[0:5])
```

```
[[1.799e+01 1.038e+01 1.228e+02 1.001e+03 1.184e-01 2.776e-01 3.001e-01
  1.471e-01 2.419e-01 7.871e-02 1.095e+00 9.053e-01 8.589e+00 1.534e+02
  6.399e-03 4.904e-02 5.373e-02 1.587e-02 3.003e-02 6.193e-03 2.538e+01
  1.733e+01 1.846e+02 2.019e+03 1.622e-01 6.656e-01 7.119e-01 2.654e-01
  4.601e-01 1.189e-01]
 [2.057e+01 1.777e+01 1.329e+02 1.326e+03 8.474e-02 7.864e-02 8.690e-02
  7.017e-02 1.812e-01 5.667e-02 5.435e-01 7.339e-01 3.398e+00 7.408e+01
  5.225e-03 1.308e-02 1.860e-02 1.340e-02 1.389e-02 3.532e-03 2.499e+01
  2.341e+01 1.588e+02 1.956e+03 1.238e-01 1.866e-01 2.416e-01 1.860e-01
  2.750e-01 8.902e-02]
 [1.969e+01 2.125e+01 1.300e+02 1.203e+03 1.096e-01 1.599e-01 1.974e-01
  1.279e-01 2.069e-01 5.999e-02 7.456e-01 7.869e-01 4.585e+00 9.403e+01
  6.150e-03 4.006e-02 3.832e-02 2.058e-02 2.250e-02 4.571e-03 2.357e+01
  2.553e+01 1.525e+02 1.709e+03 1.444e-01 4.245e-01 4.504e-01 2.430e-01
  3.613e-01 8.758e-02]
 [1.142e+01 2.038e+01 7.758e+01 3.861e+02 1.425e-01 2.839e-01 2.414e-01
  1.052e-01 2.597e-01 9.744e-02 4.956e-01 1.156e+00 3.445e+00 2.723e+01
  9.110e-03 7.458e-02 5.661e-02 1.867e-02 5.963e-02 9.208e-03 1.491e+01
  2.650e+01 9.887e+01 5.677e+02 2.098e-01 8.663e-01 6.869e-01 2.575e-01
  6.638e-01 1.730e-01]
 [2.029e+01 1.434e+01 1.351e+02 1.297e+03 1.003e-01 1.328e-01 1.980e-01
  1.043e-01 1.809e-01 5.883e-02 7.572e-01 7.813e-01 5.438e+00 9.444e+01
  1.149e-02 2.461e-02 5.688e-02 1.885e-02 1.756e-02 5.115e-03 2.254e+01
  1.667e+01 1.522e+02 1.575e+03 1.374e-01 2.050e-01 4.000e-01 1.625e-01
  2.364e-01 7.678e-02]]
```

Target set:

```
[5]  # print the cancer labels (0:malignant, 1:benign)
     print(cancer.target)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 1 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 1 1 1 0 1 0 0 1 1 1 1 0 1 0 0
 1 0 1 0 0 1 1 1 0 0 1 0 0 0 1 1 1 0 1 1 0 0 1 1 1 0 0 1 1 1 1 0 1 1 0 1 1
 1 1 1 1 1 1 0 0 0 1 0 0 1 1 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 1 1 1 1 0 1
 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 1 0 1 1 0 0 1 1 0 0 1 1 1 1 0 1 1 0 0 0 1 0
 1 0 1 1 1 0 1 1 0 0 1 0 0 0 0 1 0 0 0 1 0 1 0 1 1 0 1 0 0 0 0 1 1 0 0 1 1
 1 0 1 1 1 1 1 0 0 1 1 0 1 1 0 0 1 0 1 1 1 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 1 1 1 1 1 1 0 1 0 1 1 0 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1
 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 0 1 1 1 1 0 0 0 1 1
 1 1 0 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 1 0 0
 0 1 0 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 0 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1
 1 0 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 0 1 1 1 1 1 0 1 1
 0 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1
 1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1 1 1 0 0 1 0 1 0 1 1 1 1 1 0 1 1 0 1 0 1 0 0
 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 0 0 0 0 0 0 1]
```

Splitting Data:

To understand model performance, dividing the dataset into a training set and a test set is a good strategy.

Split the dataset by using the function train_test_split(). you need to pass 3 parameters features, target, and test_set size. Additionally, you can use random_state to select records randomly.

```
[7]  # Import train_test_split function
     from sklearn.model_selection import train_test_split

     # Split dataset into training set and test set
     X_train, X_test, y_train, y_test = train_test_split(cancer.data, cancer.target, test_size=0.3,random_state=109) # 70% training and 30% test
```

Generate Model:

Let's build support vector machine model. First, import the SVM module and create support vector classifier object by passing argument kernel as the linear kernel in SVC() function.

Then, fit your model on train set using fit() and perform prediction on the test set using predict().

```
[8]  #Import svm model
     from sklearn import svm

     #Create a svm Classifier
     clf = svm.SVC(kernel='linear') # Linear Kernel

     #Train the model using the training sets
     clf.fit(X_train, y_train)

     #Predict the response for test dataset
     y_pred = clf.predict(X_test)
```

Evaluating the Model:

Let's estimate how accurately the classifier or model can predict the breast cancer of patients. Accuracy can be computed by comparing actual test set values and predicted values.

```
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics

# Model Accuracy: how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.9649122807017544
```

Smt. Durgadevi Sharma Charitable Trust's

Chandrabhan Sharma College

of Arts, Commerce & Science

(Hindi Linguistic Minority Institution)

(Affiliated to the University of Mumbai)

NAAC Re-Accredited 'A' Grade (CGPA 3.10)

MASTER OF SCIENCE (INFORMATION TECHNOLOGY)

Subject Name : Modern Networking

Name : Yash Chandrakant Saundalkar

Seat No : 1312229

Teaching Faculty : Mr. Sandeep Vishwakarma

DEPARTMENT OF INFORMATION TECHNOLOGY

CHANDRABHAN SHARMA COLLEGE OF ARTS,

COMMERCE & SCIENCE

NAAC RE-ACCREDITED 'A' Grade (CGPA 3.10)

(Affiliated to the University of Mumbai)

(Autonomous)

MUMBAI, 400076

MAHARASHTRA

2024-2025

Smt. Durgadevi Sharma Charitable Trust's

Chandrabhan Sharma College

of Arts, Commerce & Science

(Hindi Linguistic Minority Institution)

(Affiliated to the University of Mumbai)

NAAC Re-Accredited 'A' Grade (CGPA 3.10)

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that Mr./~~Miss~~ **Yash Chandrakant Saundalkar** having Exam Seat No. **1312229** of M.Sc.IT **(Semester II)** has completed the Practical work in the subject of "**Modern Networking**" during the **Academic year 2024-25** under the guidance of **Mr. Sandeep Vishwakarma** being the Partial requirement for The fulfilment of the curriculum of Degree of Master of Science in Information Technology, University of Mumbai.

Internal Examiner **Co-ordinator**

Date: **College Seal**

External Examiner

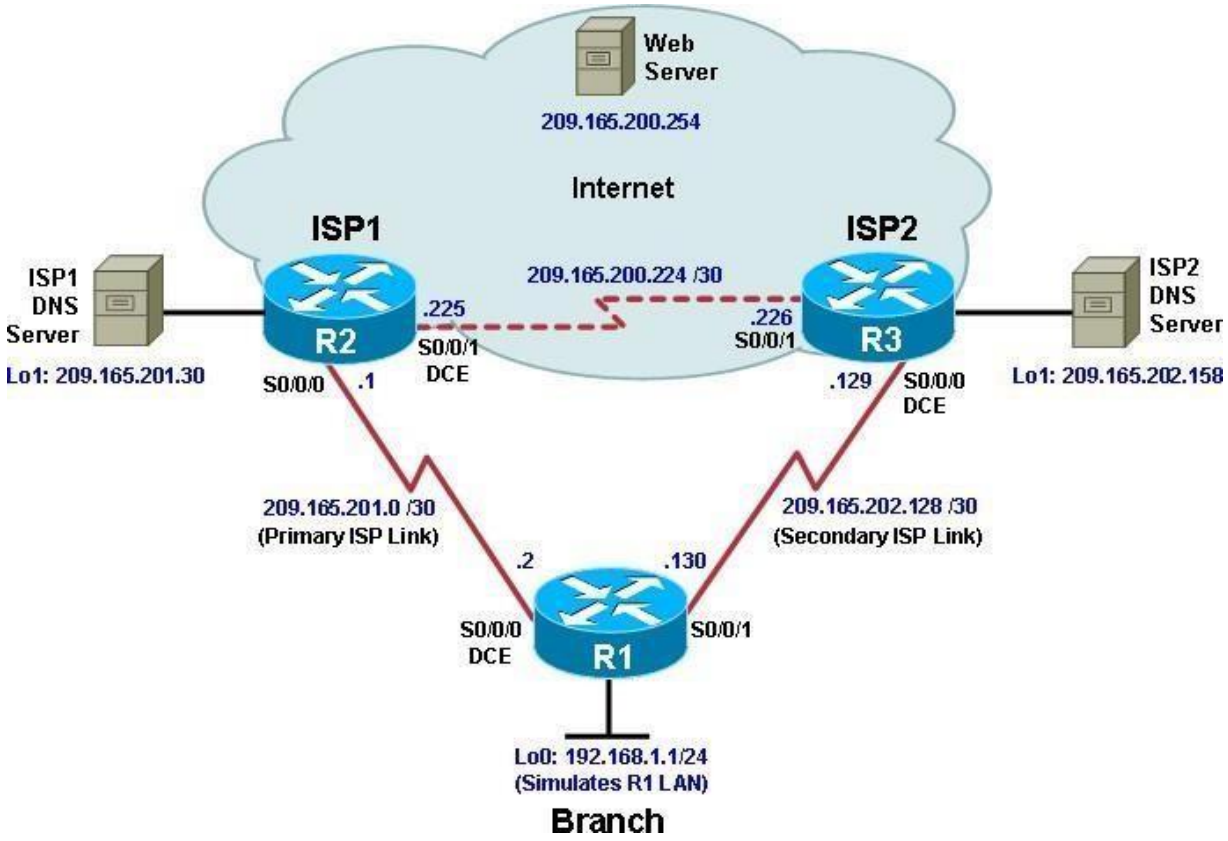
INDEX

Sr. No	Practical	Date	Sign
1	Configure IP SLA Tracking and Path Control		
2	Using the AS_PATH Attribute		
3	Configuring IBGP and EBGP Sessions		
4	Secure the Management Plane		
5	Configure and Verify Path Control Using PB		
6	IP Service Level Agreements and Remote SPAN in a Campus Environment		
7	Inter-VLAN Routing		
8	Simulating MPLS environment		
9	Simulating VRF		
10	Simulating SDN with OpenDaylight SDN Controller with the Mininet Network Emulator		
11	Simulating OpenFlow Using MININET		

Practical 1

Configure IP SLA Tracking and Path Control

Topology



Objectives

- Configure and verify the IP SLA feature.
- Test the IP SLA tracking feature.
- Verify the configuration and operation using **show** and **debug** commands.

Background

You want to experiment with the Cisco IP Service Level Agreement (SLA) feature to study how it could be of value to your organization.

At times, a link to an ISP could be operational, yet users cannot connect to any other outside Internet resources. The problem might be with the ISP or downstream from them. Although policy-based routing (PBR) can be implemented to alter path control, you will implement the Cisco IOS SLA feature to monitor this behavior and intervene by injecting another default route to a backup ISP.

To test this, you have set up a three-router topology in a lab environment. Router R1 represents a branch office connected to two different ISPs. ISP1 is the preferred connection to the Internet, while ISP2 provides a backup link. ISP1 and ISP2 can also interconnect, and both can reach the web server. To monitor ISP1 for failure, you will configure IP SLA probes to track the reachability to the ISP1 DNS server. If connectivity to the ISP1 server fails, the SLA probes detect the failure and alter the default static route to point to the ISP2 server.

Note: This lab uses Cisco 1941 routers with Cisco IOS Release 15.2 with IP Base. Depending on the router or switch model and Cisco IOS Software version, the commands available and output produced might vary from what is shown in this lab.

Required Resources

- 3 routers (Cisco IOS Release 15.2 or comparable)
- Serial and Ethernet cables

Step 1: Configure loopbacks and assign addresses.

a. Cable the network as shown in the topology diagram. Erase the startup configuration and reload each router to clear the previous configurations. Using the addressing scheme in the diagram, create the loopback interfaces and apply IP addresses to them as well as the serial interfaces on R1, ISP1, and ISP2.

You can copy and paste the following configurations into your routers to begin.

Note: Depending on the router model, interfaces might be numbered differently than those listed. You might need to alter them accordingly.

Router R1

```
hostname R1

interface Loopback 0
 description R1 LAN
 ip address 192.168.1.1 255.255.255.0

interface Serial0/0/0
 description R1 --> ISP1
 ip address 209.165.201.2 255.255.255.252
 clock rate 128000
 bandwidth 128
 no shutdown

interface Serial0/0/1
 description R1 --> ISP2
 ip address 209.165.202.130 255.255.255.252
 bandwidth 128
 no shutdown
```

Router ISP1 (R2)

```
hostname ISP1

interface Loopback0
 description Simulated Internet Web Server
 ip address 209.165.200.254 255.255.255.255

interface Loopback1
 description ISP1 DNS Server
 ip address 209.165.201.30 255.255.255.255
```

```
interface Serial0/0/0
 description ISP1 --> R1
 ip address 209.165.201.1 255.255.255.252
 bandwidth 128
 no shutdown

interface Serial0/0/1
 description ISP1 --> ISP2
 ip address 209.165.200.225 255.255.255.252
 clock rate 128000
 bandwidth 128
 no shutdown
```

Router ISP2 (R3)

```
hostname ISP2

interface Loopback0
 description Simulated Internet Web Server
 ip address 209.165.200.254 255.255.255.255

interface Loopback1
 description ISP2 DNS Server
 ip address 209.165.202.158 255.255.255.255

interface Serial0/0/0
 description ISP2 --> R1
 ip address 209.165.202.129 255.255.255.252
 clock rate 128000
 bandwidth 128
 no shutdown

interface Serial0/0/1
 description ISP2 --> ISP1
 ip address 209.165.200.226 255.255.255.252
 bandwidth 128
 no shutdown
```

b. Verify the configuration by using the **show interfaces description** command. The output from router R1 is shown here as an example.

```
R1# show interfaces description | include up
Se0/0/0                        up             up       R1 --> ISP1
Se0/0/1                        up             up       R1 --> ISP2
Lo0                            up             up       R1 LAN
R1#
```

All three interfaces should be active. Troubleshoot if necessary.

Step 2: Configure static routing.

The current routing policy in the topology is as follows:

- Router R1 establishes connectivity to the Internet through ISP1 using a default static route.
- ISP1 and ISP2 have dynamic routing enabled between them, advertising their respective public address pools.
- ISP1 and ISP2 both have static routes back to the ISP LAN.

Note: For the purpose of this lab, the ISPs have a static route to an RFC 1918 private network address on the branch router R1. In an actual branch implementation, Network Address Translation (NAT) would

be configured for all traffic exiting the branch LAN. Therefore, the static routes on the ISP routers would be pointing to the provided public pool of the branch office.

a. Implement the routing policies on the respective routers. You can copy and paste the following configurations.

Router R1

```
R1(config)# ip route 0.0.0.0 0.0.0.0 209.165.201.1
R1(config)#
```

Router ISP1 (R2)

```
ISP1(config)# router eigrp 1
ISP1(config-router)# network 209.165.200.224 0.0.0.3
ISP1(config-router)# network 209.165.201.0 0.0.0.31
ISP1(config-router)# no auto-summary
ISP1(config-router)# exit
ISP1(config)#
ISP1(config-router)# ip route 192.168.1.0 255.255.255.0 209.165.201.2
ISP1(config)#
```

Router ISP2 (R3)

```
ISP2(config)# router eigrp 1
ISP2(config-router)# network 209.165.200.224 0.0.0.3
ISP2(config-router)# network 209.165.202.128 0.0.0.31
ISP2(config-router)# no auto-summary
ISP2(config-router)# exit
ISP2(config)#
ISP2(config)# ip route 192.168.1.0 255.255.255.0 209.165.202.130
ISP2(config)#
```

EIGRP neighbor relationship messages on ISP1 and ISP2 should be generated. Troubleshoot if necessary.

b. The Cisco IOS IP SLA feature enables an administrator to monitor network performance between Cisco devices (switches or routers) or from a Cisco device to a remote IP device. IP SLA probes continuously check the reachability of a specific destination, such as a provider edge router interface, the DNS server of the ISP, or any other specific destination, and can conditionally announce a default route only if the connectivity is verified.

Before implementing the Cisco IOS SLA feature, you must verify reachability to the Internet servers. From router R1, ping the web server, ISP1 DNS server, and ISP2 DNS server to verify connectivity. You can copy the following Tcl script and paste it into R1.

```
foreach address {
209.165.200.254
209.165.201.30
209.165.202.158
} {
ping $address source 192.168.1.1
}
```

All pings should be successful. Troubleshoot if necessary.

c. Trace the path taken to the web server, ISP1 DNS server, and ISP2 DNS server. You can copy the following Tcl script and paste it into R1.

```
foreach address {
209.165.200.254
209.165.201.30
```

```
209.165.202.158
} {
trace $address source 192.168.1.1
}
```

Through which ISP is traffic flowing?

Step 3: Configure IP SLA probes.

When the reachability tests are successful, you can configure the Cisco IOS IP SLAs probes. Different types of probes can be created, including FTP, HTTP, and jitter probes.

In this scenario, you will configure ICMP echo probes.

a. Create an ICMP echo probe on R1 to the primary DNS server on ISP1 using the **ip sla** command.

```
R1(config)# ip sla 11
R1(config-ip-sla)# icmp-echo 209.165.201.30
R1(config-ip-sla-echo)# frequency 10
R1(config-ip-sla-echo)# exit
R1(config)#
R1(config)# ip sla schedule 11 life forever start-time now
R1(config)#
```

The operation number of 11 is only locally significant to the router. The **frequency 10** command schedules the connectivity test to repeat every 10 seconds. The probe is scheduled to start now and to run forever.

b. Verify the IP SLAs configuration of operation 11 using the **show ip sla configuration 11** command.

```
R1# show ip sla configuration 11
IP SLAs Infrastructure Engine-III
Entry number: 11
Owner:
Tag:
Operation timeout (milliseconds): 5000
Type of operation to perform: icmp-echo
Target address/Source address: 209.165.201.30/0.0.0.0
Type Of Service parameter: 0x0
Request size (ARR data portion): 28
Verify data: No
Vrf Name:
Schedule:
   Operation frequency (seconds): 10  (not considered if randomly scheduled)
   Next Scheduled Start Time: Start Time already passed
   Group Scheduled : FALSE
   Randomly Scheduled : FALSE
   Life (seconds): Forever
   Entry Ageout (seconds): never
   Recurring (Starting Everyday): FALSE
   Status of entry (SNMP RowStatus): Active
Threshold (milliseconds): 5000
Distribution Statistics:
   Number of statistic hours kept: 2
   Number of statistic distribution buckets kept: 1
   Statistic distribution interval (milliseconds): 20
Enhanced History:
History Statistics:
```

```
    Number of history Lives kept: 0
    Number of history Buckets kept: 15
    History Filter Type: None


R1#
```

The output lists the details of the configuration of operation 11. The operation is an ICMP echo to 209.165.201.30, with a frequency of 10 seconds, and it has already started (the start time has already passed).

c. Issue the **show ip sla statistics** command to display the number of successes, failures, and results of the latest operations.

```
R1# show ip sla statistics
IPSLAs Latest Operation Statistics

IPSLA operation id: 11
        Latest RTT: 8 milliseconds
Latest operation start time: 10:33:18 UTC Sat Jan 10 2015
Latest operation return code: OK
Number of successes: 51
Number of failures: 0
Operation time to live: Forever

R1#
```

You can see that operation 11 has already succeeded five times, has had no failures, and the last operation returned an OK result.

d. Although not actually required because IP SLA session 11 alone could provide the desired fault tolerance, create a second probe, 22, to test connectivity to the second DNS server located on router ISP2.

```
R1(config)# ip sla 22
R1(config-ip-sla)# icmp-echo 209.165.202.158
R1(config-ip-sla-echo)# frequency 10
R1(config-ip-sla-echo)# exit
R1(config)#
R1(config)# ip sla schedule 22 life forever start-time now
R1(config)# end
R1#
```

e. Verify the new probe using the **show ip sla configuration** and **show ip sla statistics** commands.

```
R1# show ip sla configuration 22
IP SLAs Infrastructure Engine-III
Entry number: 22
Owner:
Tag:
Operation timeout (milliseconds): 5000
Type of operation to perform: icmp-echo
Target address/Source address: 209.165.202.158/0.0.0.0
Type Of Service parameter: 0x0
Request size (ARR data portion): 28
Verify data: No
Vrf Name:
Schedule:
    Operation frequency (seconds): 10  (not considered if randomly scheduled)
    Next Scheduled Start Time: Start Time already passed
    Group Scheduled : FALSE
    Randomly Scheduled : FALSE
```

```
   Life (seconds): Forever
   Entry Ageout (seconds): never
   Recurring (Starting Everyday): FALSE
   Status of entry (SNMP RowStatus): Active
Threshold (milliseconds): 5000
Distribution Statistics:
   Number of statistic hours kept: 2
   Number of statistic distribution buckets kept: 1
   Statistic distribution interval (milliseconds): 20
Enhanced History:
History Statistics:
   Number of history Lives kept: 0
   Number of history Buckets kept: 15
   History Filter Type: None


R1#


R1# show ip sla configuration 22
IP SLAs, Infrastructure Engine-II.
Entry number: 22
Owner:
Tag:
Type of operation to perform: icmp-echo
Target address/Source address: 209.165.201.158/0.0.0.0
Type Of Service parameter: 0x0
Request size (ARR data portion): 28
Operation timeout (milliseconds): 5000
Verify data: No
Vrf Name:
Schedule:
   Operation frequency (seconds): 10  (not considered if randomly scheduled)
   Next Scheduled Start Time: Start Time already passed
   Group Scheduled : FALSE
   Randomly Scheduled : FALSE
   Life (seconds): Forever
   Entry Ageout (seconds): never
   Recurring (Starting Everyday): FALSE
   Status of entry (SNMP RowStatus): Active
Threshold (milliseconds): 5000 (not considered if react RTT is configured)
Distribution Statistics:
   Number of statistic hours kept: 2
   Number of statistic distribution buckets kept: 1
   Statistic distribution interval (milliseconds): 20
History Statistics:
   Number of history Lives kept: 0
   Number of history Buckets kept: 15
   History Filter Type: None
Enhanced History:

R1#
R1# show ip sla statistics 22
IPSLAs Latest Operation Statistics

IPSLA operation id: 22
      Latest RTT: 16 milliseconds
Latest operation start time: 10:38:29 UTC Sat Jan 10 2015
```

```
Latest operation return code: OK
Number of successes: 82
Number of failures: 0
Operation time to live: Forever


R1#
```

The output lists the details of the configuration of operation 22. The operation is an ICMP echo to 209.165.202.158, with a frequency of 10 seconds, and it has already started (the start time has already passed). The statistics also prove that operation 22 is active.

Step 4: Configure tracking options.

Although PBR could be used, you will configure a floating static route that appears or disappears depending on the success or failure of the IP SLA.

a. On R1, remove the current default route and replace it with a floating static route having an administrative distance of 5.

```
R1(config)# no ip route 0.0.0.0 0.0.0.0 209.165.201.1
R1(config)# ip route 0.0.0.0 0.0.0.0 209.165.201.1 5
R1(config)# exit
```

b. Verify the routing table.

```
R1# show ip route | begin Gateway
Gateway of last resort is 209.165.201.1 to network 0.0.0.0

S*    0.0.0.0/0 [5/0] via 209.165.201.1
      192.168.1.0/24 is variably subnetted, 2 subnets, 2 masks
C       192.168.1.0/24 is directly connected, Loopback0
L       192.168.1.1/32 is directly connected, Loopback0
      209.165.201.0/24 is variably subnetted, 2 subnets, 2 masks
C       209.165.201.0/30 is directly connected, Serial0/0/0
L       209.165.201.2/32 is directly connected, Serial0/0/0
      209.165.202.0/24 is variably subnetted, 2 subnets, 2 masks
C         209.165.202.128/30 is directly connected, Serial0/0/1
L       209.165.202.130/32 is directly connected, Serial0/0/1
R1#
```

Notice that the default static route is now using the route with the administrative distance of 5. The first tracking object is tied to IP SLA object 11.

c. From global configuration mode on R1, use the **track 1 ip sla 11 reachability** command to enter the config-track subconfiguration mode.

```
R1(config)# track 1 ip sla 11 reachability
R1(config-track)#
```

d. Specify the level of sensitivity to changes of tracked objects to 10 seconds of down delay and 1 second of up delay using the **delay down 10 up 1** command. The delay helps to alleviate the effect of flapping objects–objects that are going down and up rapidly. In this situation, if the DNS server fails momentarily and comes back up within 10 seconds, there is no impact.

```
R1(config-track)# delay down 10 up 1
R1(config-track)# exit
R1(config)#
```

e. To view routing table changes as they happen, first enable the **debug ip routing** command.

```
R1# debug ip routing
```

```
IP routing debugging is on
R1#
```

f. Configure the floating static route that will be implemented when tracking object 1 is active. Use the **ip route 0.0.0.0 0.0.0.0 209.165.201.1 2 track 1** command to create a floating static default route via 209.165.201.1 (ISP1). Notice that this command references the tracking object number 1, which in turn references IP SLA operation number 11.

```
R1(config)# ip route 0.0.0.0 0.0.0.0 209.165.201.1 2 track 1
R1(config)#
Jan 10 10:45:39.119: RT: updating static 0.0.0.0/0 (0x0)  :
    via 209.165.201.1   0 1048578

Jan 10 10:45:39.119: RT: closer admin distance for 0.0.0.0, flushing 1 routes
Jan 10 10:45:39.119: RT: add 0.0.0.0/0 via 209.165.201.1, static metric [2/0]
Jan 10 10:45:39.119: RT: updating static 0.0.0.0/0 (0x0)  :
    via 209.165.201.1   0 1048578

Jan 10 10:45:39.119: RT: rib update return code: 17
Jan 10 10:45:39.119: RT: updating static 0.0.0.0/0 (0x0)  :
    via 209.165.201.1   0 1048578

Jan 10 10:45:39.119: RT: rib update return code: 17
R1(config)#
```

Notice that the default route with an administrative distance of 5 has been immediately flushed because of a route with a better admin distance. It then adds the new default route with the admin distance of 2.

g. Repeat the steps for operation 22, track number 2, and assign the static route an admin distance higher than track 1 and lower than 5. On R1, copy the following configuration, which sets an admin distance of 3.

```
R1(config)# track 2 ip sla 22 reachability
R1(config-track)# delay down 10 up 1
R1(config-track)# exit
R1(config)#
R1(config)# ip route 0.0.0.0 0.0.0.0 209.165.202.129 3 track 2
R1(config)#
```

h. Verify the routing table again.

```
R1#show ip route | begin Gateway
Gateway of last resort is 209.165.201.1 to network 0.0.0.0

S*    0.0.0.0/0 [2/0] via 209.165.201.1
      192.168.1.0/24 is variably subnetted, 2 subnets, 2 masks
C       192.168.1.0/24 is directly connected, Loopback0
L       192.168.1.1/32   is   directly   connected,   Loopback0
      209.165.201.0/24 is variably subnetted, 2 subnets, 2 masks
C       209.165.201.0/30 is directly connected, Serial0/0/0
L      209.165.201.2/32 is directly connected, Serial0/0/0
      209.165.202.0/24 is variably subnetted, 2 subnets, 2 masks
C        209.165.202.128/30 is directly connected, Serial0/0/1
L       209.165.202.130/32 is directly connected, Serial0/0/1
R1#
```

Although a new default route was entered, its administrative distance is not better than 2. Therefore, it does not replace the previously entered default route.

Step 5: Verify IP SLA operation.

In this step you observe and verify the dynamic operations and routing changes when tracked objects fail. The following summarizes the process:

- Disable the DNS loopback interface on ISP1 (R2).
- Observe the output of the **debug** command on R1.
- Verify the static route entries in the routing table and the IP SLA statistics of R1.
- Re-enable the loopback interface on ISP1 (R2) and again observe the operation of the IP SLA tracking feature.

a. On ISP1, disable the loopback interface 1.

```
ISP1(config-if)# int lo1
ISP1(config-if)# shutdown
ISP1(config-if)#
Jan 10 10:53:25.091: %LINK-5-CHANGED: Interface Loopback1, changed state to
administratively down
Jan 10 10:53:26.091: %LINEPROTO-5-UPDOWN: Line protocol on Interface Loopback1,
changed state to down
ISP1(config-if)#
```

b. On R1, observe the **debug** output being generated. Recall that R1 will wait up to 10 seconds before initiating action therefore several seconds will elapse before the output is generated.

```
R1#
Jan 10 10:53:59.551: %TRACK-6-STATE: 1 ip sla 11 reachability Up -> Down
Jan 10 10:53:59.551: RT: del 0.0.0.0 via 209.165.201.1, static metric [2/0]
Jan 10 10:53:59.551: RT: delete network route to 0.0.0.0/0
Jan 10 10:53:59.551: RT: default path has been cleared
Jan 10 10:53:59.551: RT: updating static 0.0.0.0/0 (0x0)  :
    via 209.165.202.129  0 1048578

Jan 10 10:53:59.551: RT: add 0.0.0.0/0 via 209.165.202.129, static metric [3/0]
Jan 10 10:53:59.551: RT: default path is now 0.0.0.0 via 209.165.202.129
Jan 10 10:53:59.551: RT: updating static 0.0.0.0/0 (0x0)  :
    via 209.165.201.1  0 1048578

Jan 10 10:53:59.551: RT: rib update return code: 17
Jan 10 10:53:59.551: RT: updating static 0.0.0.0/0 (0x0)  :
    via 209.165.202.129  0 1048578

Jan 10 10:53:59.551: RT: updating static 0.0.0.0/0 (0x0)  :
    via 209.165.201.1  0 1048578

Jan 10 10:53:59.551: RT: rib update return code: 17
R1#
```

The tracking state of track 1 changes from up to down. This is the object that tracked reachability for IP SLA object 11, with an ICMP echo to the ISP1 DNS server at 209.165.201.30.

R1 then proceeds to delete the default route with the administrative distance of 2 and installs the next highest default route to ISP2 with the administrative distance of 3.

c. On R1, verify the routing table.

```
R1# show ip route | begin Gateway
Gateway of last resort is 209.165.202.129 to network 0.0.0.0
```

```
S*    0.0.0.0/0 [3/0] via 209.165.202.129
      192.168.1.0/24 is variably subnetted, 2 subnets, 2 masks
C       192.168.1.0/24 is directly connected, Loopback0
L       192.168.1.1/32   is   directly   connected,   Loopback0
      209.165.201.0/24 is variably subnetted, 2 subnets, 2 masks
C       209.165.201.0/30 is directly connected, Serial0/0/0
L       209.165.201.2/32 is directly connected, Serial0/0/0
      209.165.202.0/24 is variably subnetted, 2 subnets, 2 masks
C         209.165.202.128/30 is directly connected, Serial0/0/1
L       209.165.202.130/32 is directly connected, Serial0/0/1
R1#
```

The new static route has an administrative distance of 3 and is being forwarded to ISP2 as it should.

d. Verify the IP SLA statistics.

```
R1# show ip sla statistics
IPSLAs Latest Operation Statistics

IPSLA operation id: 11
      Latest RTT: NoConnection/Busy/Timeout
Latest operation start time: 11:01:08 UTC Sat Jan 10 2015
Latest operation return code: Timeout
Number of successes: 173
Number of failures: 45
Operation time to live: Forever



IPSLA operation id: 22
      Latest RTT: 8 milliseconds
Latest operation start time: 11:01:09 UTC Sat Jan 10 2015
Latest operation return code: OK
Number of successes: 218
Number of failures: 0
Operation time to live: Forever


R1#
```

Notice that the latest return code is **Timeout** and there have been 45 failures on IP SLA object 11.

e. On R1, initiate a trace to the web server from the internal LAN IP address.

```
R1# trace 209.165.200.254 source 192.168.1.1
Type escape sequence to abort.
Tracing the route to 209.165.200.254
VRF info: (vrf in name/id, vrf out name/id)
  1 209.165.202.129 4 msec *  *
R1#
```

This confirms that traffic is leaving router R1 and being forwarded to the ISP2 router.

f. On ISP1, re-enable the DNS address by issuing the **no shutdown** command on the loopback 1 interface to examine the routing behavior when connectivity to the ISP1 DNS is restored.

```
ISP1(config-if)# no shutdown
Jan 10 11:05:45.847: %LINK-3-UPDOWN: Interface Loopback1, changed state to up
Jan 10 11:05:46.847: %LINEPROTO-5-UPDOWN: Line protocol on Interface Loopback1,
changed state to up
```

```
ISP1(config-if)#
```

Notice the output of the **debug ip routing** command on R1.

```
R1#
Jan 10 11:06:20.551: %TRACK-6-STATE: 1 ip sla 11 reachability Down -> Up
Jan 10 11:06:20.551: RT: updating static 0.0.0.0/0 (0x0)  :
    via 209.165.201.1   0 1048578

Jan 10 11:06:20.551: RT: closer admin distance for 0.0.0.0, flushing 1 routes
Jan 10 11:06:20.551: RT: add 0.0.0.0/0 via 209.165.201.1, static metric [2/0]
Jan 10 11:06:20.551: RT: updating static 0.0.0.0/0 (0x0)  :
    via 209.165.202.129   0 1048578

Jan 10 11:06:20.551: RT: rib update return code: 17
Jan 10 11:06:20.551: RT: u
R1#pdating static 0.0.0.0/0 (0x0)  :
    via 209.165.202.129   0 1048578

Jan 10 11:06:20.551: RT: rib update return code: 17
Jan 10 11:06:20.551: RT: updating static 0.0.0.0/0 (0x0)  :
    via 209.165.201.1   0 1048578

Jan 10 11:06:20.551: RT: rib update return code: 17
R1#
```

Now the IP SLA 11 operation transitions back to an up state and reestablishes the default static route to ISP1 with an administrative distance of 2.

g. Again examine the IP SLA statistics.

```
R1# show ip sla statistics
IPSLAs Latest Operation Statistics

IPSLA operation id: 11
      Latest RTT: 8 milliseconds
Latest operation start time: 11:07:38 UTC Sat Jan 10 2015
Latest operation return code: OK
Number of successes: 182
Number of failures: 75
Operation time to live: Forever



IPSLA operation id: 22
      Latest RTT: 16 milliseconds
Latest operation start time: 11:07:39 UTC Sat Jan 10 2015
Latest operation return code: OK
Number of successes: 257
Number of failures: 0
Operation time to live: Forever


R1#
```

The IP SLA 11 operation is active again, as indicated by the OK return code, and the number of successes is incrementing.

h. Verify the routing table.

```
R1# show ip route | begin Gateway
Gateway of last resort is 209.165.201.1 to network 0.0.0.0

S*    0.0.0.0/0 [2/0] via 209.165.201.1
      192.168.1.0/24 is variably subnetted, 2 subnets, 2 masks
C       192.168.1.0/24 is directly connected, Loopback0
L       192.168.1.1/32   is   directly   connected,   Loopback0
      209.165.201.0/24 is variably subnetted, 2 subnets, 2 masks
C       209.165.201.0/30 is directly connected, Serial0/0/0
L      209.165.201.2/32 is directly connected, Serial0/0/0
      209.165.202.0/24 is variably subnetted, 2 subnets, 2 masks
C        209.165.202.128/30 is directly connected, Serial0/0/1
L       209.165.202.130/32 is directly connected, Serial0/0/1
R1#
```

The default static through ISP1 with an administrative distance of 2 is reestablished.

There are many possibilities available with object tracking and Cisco IOS IP SLAs. As shown in this lab, a probe can be based on reachability, changing routing operations, and path control based on the ability to reach an object. However, Cisco IOS IP SLAs also allow paths to be changed based on network conditions such as delay, load, and other factors.

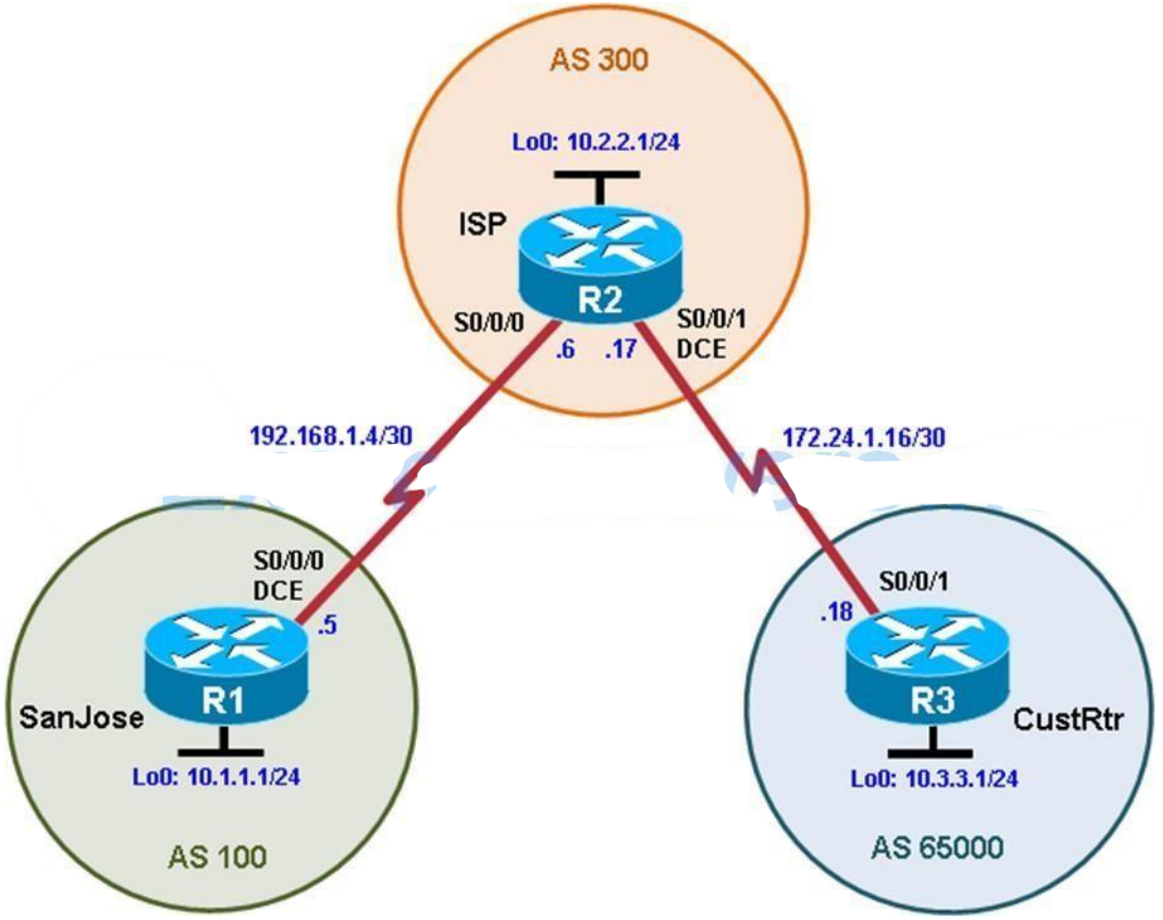
Before deploying a Cisco IOS IP SLA solution, the impact of the additional probe traffic being generated should be considered, including how that traffic affects bandwidth utilization, and congestion levels. Tuning the configuration (for example, with the **delay** and **frequency** commands) is critical to mitigate possible issues related to excessive transitions and route changes in the presence of flapping tracked objects.

The benefits of running IP SLAs should be carefully evaluated. The IP SLA is an additional task that must be performed by the router's CPU. A large number of intensive SLAs could be a significant burden on the CPU, possibly interfering with other router functions and having detrimental impact on the overall router performance. The CPU load should be monitored after the SLAs are deployed to verify that they do not cause excessive utilization of the router CPU.

Practical 2

Using the AS_PATH Attribute

Topology



Objectives

- Use BGP commands to prevent private AS numbers from being advertised to the outside world.
- Use the AS_PATH attribute to filter BGP routes based on their source AS numbers.

Background

The International Travel Agency's ISP has been assigned an AS number of 300. This provider uses BGP to exchange routing information with several customer networks. Each customer network is assigned an AS number from the private range, such as AS 65000. Configure the ISP router to remove the

private AS numbers from the AS Path information of CustRtr. In addition, the ISP would like to prevent its customer networks from receiving route information from International Travel Agency's AS 100. Use the AS_PATH attribute to implement this policy.

Note: This lab uses Cisco 1941 routers with Cisco IOS Release 15.4 with IP Base. The switches are Cisco WS-C2960-24TT-L with Fast Ethernet interfaces, therefore the router will use routing metrics associated with a 100 Mb/s interface. Depending on the router or switch model and Cisco IOS Software version, the commands available and output produced might vary from what is shown in this lab.

Required Resources

- 3 routers (Cisco IOS Release 15.2 or comparable)
- Serial and Ethernet cables

Step 0: Suggested starting configurations.

a. Apply the following configuration to each router along with the appropriate **hostname.** The **exec-timeout 0 0** command should only be used in a lab environment.

```
Router(config)# no ip domain-lookup
Router(config)# line con 0
Router(config-line)# logging synchronous
Router(config-line)# exec-timeout 0 0
```

Step 1: Configure interface addresses.

b. Using the addressing scheme in the diagram, create the loopback interfaces and apply IPv4 addresses to these and the serial interfaces on SanJose (R1), ISP (R2), and CustRtr (R3). The ISP loopbacks simulate real networks. Set a clock rate on the DCE serial interfaces.

```
SanJose(config)# interface Loopback0
SanJose(config-if)# ip address 10.1.1.1 255.255.255.0
SanJose(config-if)# exit
SanJose(config)# interface Serial0/0/0
SanJose(config-if)# ip address 192.168.1.5 255.255.255.252
SanJose(config-if)# clock rate 128000
SanJose(config-if)# no shutdown
SanJose(config-if)# end
SanJose#

ISP(config)# interface Loopback0
ISP(config-if)# ip address 10.2.2.1 255.255.255.0
ISP(config-if)# interface Serial0/0/0
ISP(config-if)# ip address 192.168.1.6 255.255.255.252
ISP(config-if)# no shutdown
ISP(config-if)# exit
ISP(config)# interface Serial0/0/1
ISP(config-if)# ip address 172.24.1.17 255.255.255.252
ISP(config-if)# clock rate 128000
ISP(config-if)# no shutdown
ISP(config-if)# end
ISP#

CustRtr(config)# interface Loopback0
CustRtr(config-if)# ip address 10.3.3.1 255.255.255.0
CustRtr(config-if)# exit
CustRtr(config)# interface Serial0/0/1
CustRtr(config-if)# ip address 172.24.1.18 255.255.255.252
CustRtr(config-if)# no shutdown
CustRtr(config-if)# end
CustRtr#
```

c. Use **ping** to test the connectivity between the directly connected routers. **Note:** SanJose will not be able to reach either ISP's loopback (10.2.2.1) or CustRtr's loopback (10.3.3.1), nor will it be able to reach either end of the link joining ISP to CustRtr (172.24.1.17 and 172.24.1.18).

Step 2: Configure BGP.

a. Configure BGP for normal operation. Enter the appropriate BGP commands on each router so that they identify their BGP neighbors and advertise their loopback networks.

```
SanJose(config)# router bgp 100
SanJose(config-router)# neighbor 192.168.1.6 remote-as 300
SanJose(config-router)# network 10.1.1.0 mask 255.255.255.0

ISP(config)# router bgp 300
ISP(config-router)# neighbor 192.168.1.5 remote-as 100
ISP(config-router)# neighbor 172.24.1.18 remote-as 65000
ISP(config-router)# network 10.2.2.0 mask 255.255.255.0

CustRtr(config)# router bgp 65000
CustRtr(config-router)# neighbor 172.24.1.17 remote-as 300
CustRtr(config-router)# network 10.3.3.0 mask 255.255.255.0
```

b. Verify that these routers have established the appropriate neighbor relationships by issuing the **show ip bgp neighbors** command on each router.

```
ISP# show ip bgp neighbors
BGP neighbor is 172.24.1.18, remote AS 65000, external link
  BGP version 4, remote router ID 10.3.3.1
  BGP state = Established, up for 00:00:28
  Last read 00:00:28, last write 00:00:28, hold time is 180, keepalive interv
<output omitted>
```

```
BGP neighbor is 192.168.1.5, remote AS 100, external link
  BGP version 4, remote router ID 10.1.1.1
  BGP state = Established, up for 00:01:34
  Last read 00:00:33, last write 00:00:06, hold time is 180, keepalive interv
<output omitted>
```

Step 3: Remove the private AS.

a. Display the SanJose routing table using the **show ip route** command. SanJose should have a route to both 10.2.2.0 and 10.3.3.0. Troubleshoot if necessary.

```
SanJose#show ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static rout
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override

Gateway of last resort is not set

      10.0.0.0/8 is variably subnetted, 4 subnets, 2 masks
C        10.1.1.0/24 is directly connected, Loopback0
L        10.1.1.1/32 is directly connected, Loopback0
B        10.2.2.0/24 [20/0] via 192.168.1.6, 00:04:22
B        10.3.3.0/24 [20/0] via 192.168.1.6, 00:03:14
      192.168.1.0/24 is variably subnetted, 2 subnets, 2 masks
C        192.168.1.4/30 is directly connected, Serial0/0/0
L        192.168.1.5/32 is directly connected, Serial0/0/0
SanJose#
```

b. Ping the 10.3.3.1 address from SanJose.

Why does this fail?

_____

c. Ping again, this time as an extended ping, sourcing from the Loopback0 interface address.

```
SanJose# ping
Protocol [ip]:
Target IP address: 10.3.3.1
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface: 10.1.1.1
Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.3.3.1, timeout is 2 seconds:
Packet sent with a source address of 10.1.1.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/28/28 ms
SanJose#
```

Note: You can bypass extended ping mode and specify a source address using one of these commands:

```
SanJose# ping 10.3.3.1 source 10.1.1.1
```

or

```
SanJose# ping 10.3.3.1 source Lo0
```

d. Check the BGP table from SanJose by using the **show ip bgp** command. Note the AS path for the 10.3.3.0 network. The AS 65000 should be listed in

the path to 10.3.3.0.

```
SanJose# show ip bgp
BGP table version is 5, local router ID is 10.1.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - interna
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter
              x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
 *>  10.1.1.0/24      0.0.0.0                  0         32768 i
 *>  10.2.2.0/24      192.168.1.6              0             0 300 i
 *>  10.3.3.0/24      192.168.1.6                            0 300 65000 i
SanJose#
```

Why is this a problem?

e. Configure ISP to strip the private AS numbers from BGP routes exchanged with SanJose using the following commands.

```
ISP(config)# router bgp 300
ISP(config-router)# neighbor 192.168.1.5 remove-private-as
```

f. After issuing these commands, use the **clear ip bgp *** command on ISP to reestablish the BGP relationship between the three routers. Wait several seconds and then return to SanJose to check its routing table.
Note: The **clear ip bgp * soft** command can also be used to force each router to resend its BGP table.

```
ISP# clear ip bgp *
ISP#
*Sep  8 18:40:03.551: %BGP-5-ADJCHANGE: neighbor 172.24.1.18 Down User reset
*Sep  8 18:40:03.551: %BGP_SESSION-5-ADJCHANGE: neighbor 172.24.1.18 IPv4 Uni
*Sep  8 18:40:03.551: %BGP-5-ADJCHANGE: neighbor 192.168.1.5 Down User reset
*Sep  8 18:40:03.551: %BGP_SESSION-5-ADJCHANGE: neighbor 192.168.1.5 IPv4 Uni
*Sep  8 18:40:04.515: %BGP-5-ADJCHANGE: neighbor 172.24.1.18 Up
*Sep  8 18:40:04.519: %BGP-
ISP#5-ADJCHANGE: neighbor 192.168.1.5 Up
ISP#

SanJose# show ip route


      10.0.0.0/8 is variably subnetted, 4 subnets, 2 masks
C        10.1.1.0/24 is directly connected, Loopback0
L        10.1.1.1/32 is directly connected, Loopback0
B        10.2.2.0/24 [20/0] via 192.168.1.6, 00:00:20
B        10.3.3.0/24 [20/0] via 192.168.1.6, 00:01:02
      192.168.1.0/24 is variably subnetted, 2 subnets, 2 masks
C        192.168.1.4/30 is directly connected, Serial0/0/0
L        192.168.1.5/32 is directly connected, Serial0/0/0
SanJose#
```

Does SanJose still have a route to 10.3.3.0?

SanJose should be able to ping 10.3.3.1 using its loopback 0 interface as the source of the ping.

```
SanJose# ping 10.3.3.1 source lo0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.3.3.1, timeout is 2 seconds:
Packet sent with a source address of 10.1.1.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/28/32 ms
```

g. Now check the BGP table on SanJose. The AS_ PATH to the 10.3.3.0 network should be AS 300. It no longer has the private AS in the path.

```
SanJose# show ip bgp
BGP table version is 9, local router ID is 10.1.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - interna
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter
              x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
 *>  10.1.1.0/24      0.0.0.0                  0         32768 i
 *>  10.2.2.0/24      192.168.1.6              0             0 300 i
 *>  10.3.3.0/24      192.168.1.6                            0 300 i
SanJose#
```

Step 4: Use the AS_PATH attribute to filter routes.

As a final configuration, use the AS_PATH attribute to filter routes based on their origin. In a complex environment, you can use this attribute to enforce routing policy. In this case, the provider router, ISP, must be configured so that it does not propagate routes that originate from AS 100 to the customer router CustRtr.

AS-path access lists are read like regular access lists. The statements are read sequentially, and there is an implicit deny at the end. Rather than matching an address in each statement like a conventional access list, AS path access lists match on something called a regular expression. Regular expressions are a way of matching text patterns and have many uses. In this case, you will be using them in the AS path access list to match text patterns in AS paths.

```
ISP(config)# ip as-path access-list 1 deny ^100$
ISP(config)# ip as-path access-list 1 permit .*
```

The first command uses the ^ character to indicate that the AS path must

begin with the given number 100. The $ character indicates that the AS_PATH attribute must also end with 100. Essentially, this statement matches only paths that are sourced from AS 100. Other paths, which might include AS 100 along the way, will not match this list.

In the second statement, the . (period) is a wildcard, and the * (asterisk) stands for a repetition of the wildcard. Together, .* matches any value of the AS_PATH attribute, which in effect permits any update that has not been denied by the previous access-list statement.

For more details on configuring regular expressions on Cisco routers, see:

http://www.cisco.com/c/en/us/td/docs/ios/12_2/termserv/configuration/guide/ftersv_c/tcfaapre.html
http://www.cisco.com/c/en/us/support/docs/ip/border-gateway-protocol-bgp/13754-26.html

b. Apply the configured access list using the neighbor command with the **filter-list** option.

```
ISP(config)# router bgp 300
ISP(config-router)# neighbor 172.24.1.18 filter-list 1 out
```

The **out** keyword specifies that the list is applied to routing information sent to this neighbor.

c. Use the **clear ip bgp *** command to reset the routing information. Wait several seconds and then check the routing table for ISP. The route to 10.1.1.0 should be in the routing table.

Note: To force the local router to resend its BGP table, a less disruptive option is to use the **clear ip bgp * out** or **clear ip bgp * soft** command (the second command performs both outgoing and incoming route resync).

```
ISP# clear ip bgp *
ISP#
*Sep  8 18:48:04.915: %BGP-5-ADJCHANGE: neighbor 172.24.1.18 Down User reset
*Sep  8 18:48:04.915: %BGP_SESSION-5-ADJCHANGE: neighbor 172.24.1.18 IPv4 Uni
*Sep  8 18:48:04.915: %BGP-5-ADJCHANGE: neighbor 192.168.1.5 Down User reset
*Sep  8 18:48:04.915: %BGP_SESSION-5-ADJCHANGE: neighbor 192.168.1.5 IPv4 Uni
*Sep  8 18:48:04.951: %BGP-5-ADJCHANGE: neighbor 172.24.1.18 Up
*Sep  8 18:48:04.955: %BGP-
ISP#5-ADJCHANGE: neighbor 192.168.1.5 Up
ISP#

ISP# show ip route


      10.0.0.0/8 is variably subnetted, 4 subnets, 2 masks
B        10.1.1.0/24 [20/0] via 192.168.1.5, 00:00:29
C        10.2.2.0/24 is directly connected, Loopback0
L        10.2.2.1/32 is directly connected, Loopback0
B        10.3.3.0/24 [20/0] via 172.24.1.18, 00:00:29
      172.24.0.0/16 is variably subnetted, 2 subnets, 2 masks
C        172.24.1.16/30 is directly connected, Serial0/0/1
L        172.24.1.17/32 is directly connected, Serial0/0/1
      192.168.1.0/24 is variably subnetted, 2 subnets, 2 masks
C        192.168.1.4/30 is directly connected, Serial0/0/0
L        192.168.1.6/32 is directly connected, Serial0/0/0
ISP#
```

d. Check the routing table for CustRtr. It should not have a route to 10.1.1.0 in its routing table.

```
CustRtr# show ip route


      10.0.0.0/8 is variably subnetted, 3 subnets, 2 masks
B        10.2.2.0/24 [20/0] via 172.24.1.17, 00:00:32
C        10.3.3.0/24 is directly connected, Loopback0
L        10.3.3.1/32 is directly connected, Loopback0
      172.24.0.0/16 is variably subnetted, 2 subnets, 2 masks
C        172.24.1.16/30 is directly connected, Serial0/0/1
L        172.24.1.18/32 is directly connected, Serial0/0/1
CustRtr#
```

e. Return to ISP and verify that the filter is working as intended. Issue the **show ip bgp regexp ^100$** command.

```
ISP# show ip bgp regexp ^100$
BGP table version is 4, local router ID is 10.2.2.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - interna
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter
              x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
 *>  10.1.1.0/24     192.168.1.5              0             0 100 i
ISP#
```

The output of this command shows all matches for the regular expressions that were used in the access list. The path to 10.1.1.0 matches the access list and is filtered from updates to CustRtr.

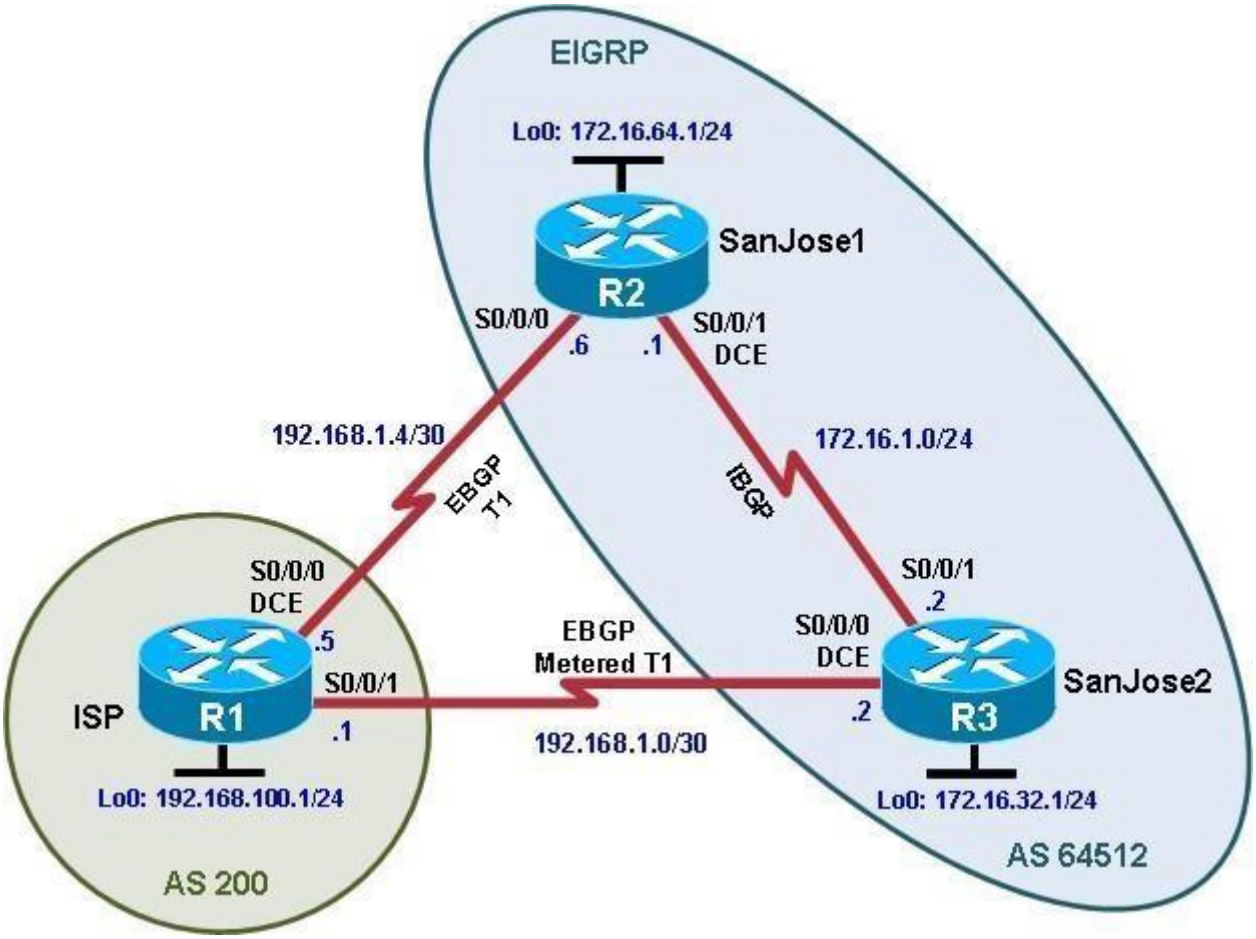
f. Run the following Tcl script on all routers to verify whether there is connectivity. All pings from ISP should be successful. SanJose should not be able to ping the CustRtr loopback 10.3.3.1 or the WAN link 172.24.1.16/30. CustRtr should not be able to ping the SanJose loopback 10.1.1.1 or the WAN link 192.168.1.4/30.

```
ISP# tclsh

foreach address {
10.1.1.1
10.2.2.1
10.3.3.1
192.168.1.5
192.168.1.6
172.24.1.17
172.24.1.18
} {
ping $address }
```

Practical 3

Configuring IBGP and EBGP Sessions, Local Preference, andMED

Topology



Objectives

- For IBGP peers to correctly exchange routing information, use the **next-hop-self** command with the **Local-Preference** and **MED** attributes.
- Ensure that the flat-rate, unlimited-use T1 link is used for sending and receiving data to and from the AS 200 on ISP and that the metered T1 only be used in the event that the primary T1 link has failed.

Background

The International Travel Agency runs BGP on its SanJose1 and SanJose2 routers externally with the ISP router in AS 200. IBGP is run internally between SanJose1 and SanJose2. Your job is to configure both EBGP and IBGP for this

internetwork to allow for redundancy. The metered T1 should only be used in the event that the primary T1 link has failed. Traffic sent across the metered T1 link offers the same bandwidth of the primary link but at a huge expense. Ensure that this link is not used unnecessarily.

Note: This lab uses Cisco 1941 routers with Cisco IOS Release 15.4 with IP Base. The switches are Cisco WS-C2960-24TT-L with Fast Ethernet interfaces, therefore the router will use routing metrics associated with a 100 Mb/s interface. Depending on the router or switch model and Cisco IOS Software version, the commands available and output produced might vary from what is shown in this lab.

Required Resources

- 3 routers (Cisco IOS Release 15.2 or comparable)
- Serial and Ethernet cables

Step 0: Suggested starting configurations.

a. Apply the following configuration to each router along with the appropriate **hostname**. The **exec-timeout 0 0** command should only be used in a lab environment.

```
Router(config)# no ip domain-lookup
Router(config)# line con 0
Router(config-line)# logging synchronous
Router(config-line)# exec-timeout 0 0
```

Step 1: Configure interface addresses.

a. Using the addressing scheme in the diagram, create the loopback interfaces and apply IPv4 addresses to these and the serial interfaces on ISP (R1), SanJose1 (R2), and SanJose2 (R3).

Router R1 (hostname ISP)

```
ISP(config)# interface Loopback0
ISP(config-if)# ip address 192.168.100.1 255.255.255.0
ISP(config-if)# exit
ISP(config)# interface Serial0/0/0
ISP(config-if)# ip address 192.168.1.5 255.255.255.252
ISP(config-if)# clock rate 128000
ISP(config-if)# no shutdown
ISP(config-if)# exit
ISP(config)# interface Serial0/0/1
ISP(config-if)# ip address 192.168.1.1 255.255.255.252
ISP(config-if)# no shutdown
ISP(config-if)# end
ISP#
```

Router R2 (hostname SanJose1)

```
SanJose1(config)# interface Loopback0
SanJose1(config-if)# ip address 172.16.64.1 255.255.255.0
SanJose1(config-if)# exit
SanJose1(config)# interface Serial0/0/0
SanJose1(config-if)# ip address 192.168.1.6 255.255.255.252
SanJose1(config-if)# no shutdown
SanJose1(config-if)# exit
SanJose1(config)# interface Serial0/0/1
SanJose1(config-if)# ip address 172.16.1.1 255.255.255.0
SanJose1(config-if)# clock rate 128000
SanJose1(config-if)# no shutdown
SanJose1(config-if)# end
SanJose1#
```

Router R3 (hostname SanJose2)

```
SanJose2(config)# interface Loopback0
SanJose2(config-if)# ip address 172.16.32.1 255.255.255.0
SanJose2(config-if)# exit
SanJose2(config)# interface Serial0/0/0
SanJose2(config-if)# ip address 192.168.1.2 255.255.255.252
SanJose2(config-if)# clock rate 128000
SanJose2(config-if)# no shutdown
SanJose2(config-if)# exit
SanJose2(config)# interface Serial0/0/1
SanJose2(config-if)# ip address 172.16.1.2 255.255.255.0
SanJose2(config-if)# no shutdown
SanJose2(config-if)# end
SanJose2#
```

b. Use **ping** to test the connectivity between the directly connected routers. Both SanJose routers should be able to ping each other and their local ISP serial link IP address. The ISP router cannot reach the segment between SanJose1 and SanJose2.

Step 2: Configure EIGRP.

Configure EIGRP between the SanJose1 and SanJose2 routers. (Note: If using an IOS prior to 15.0, use the no auto-summary router configuration command to disable automatic summarization. This command is the default beginning with IOS 15.)

```
SanJose1(config)# router eigrp 1
SanJose1(config-router)# network 172.16.0.0

SanJose2(config)# router eigrp 1
SanJose2(config-router)# network 172.16.0.0
```

Step 3: Configure IBGP and verify BGP neighbors.

a. Configure IBGP between the SanJose1 and SanJose2 routers. On the SanJose1 router, enter the following configuration.

```
SanJose1(config)# router bgp 64512
SanJose1(config-router)# neighbor 172.16.32.1 remote-as 64512
SanJose1(config-router)# neighbor 172.16.32.1 update-source lo0
```

If multiple pathways to the BGP neighbor exist, the router can use multiple IP interfaces to communicate with the neighbor. The source IP address therefore depends on the outgoing interface. The **update-source lo0** command instructs the router to use the IP address of the interface Loopback0 as the source IP address for all BGP messages sent to that neighbor.

b. Complete the IBGP configuration on SanJose2 using the following commands.

```
SanJose2(config)# router bgp 64512
SanJose2(config-router)# neighbor 172.16.64.1 remote-as 64512
SanJose2(config-router)# neighbor 172.16.64.1 update-source lo0
```

c. Verify that SanJose1 and SanJose2 become BGP neighbors by issuing the **show ip bgp neighbors** command on SanJose1. View the following partial output. If the BGP state is not established, troubleshoot the connection.

```
SanJose2# show ip bgp neighbors
BGP neighbor is 172.16.64.1,  remote AS 64512, internal link
  BGP version 4, remote router ID 172.16.64.1
  BGP state = Established, up for 00:00:22
  Last read 00:00:22, last write 00:00:22, hold time is 180, keepalive interval is 60
seconds
<output omitted>
```

The link between SanJose1 and SanJose2 should be identified as an internal link indicating an IBGP peering relationship, as shown in the output.

Step 4: Configure EBGP and verify BGP neighbors.

a. Configure ISP to run EBGP with SanJose1 and SanJose2. Enter the following commands on ISP.

```
ISP(config)# router bgp 200
ISP(config-router)# neighbor 192.168.1.6 remote-as 64512
ISP(config-router)# neighbor 192.168.1.2 remote-as 64512
ISP(config-router)# network 192.168.100.0
```

Because EBGP sessions are almost always established over point-to-point links, there is no reason to use the **update-source** keyword in this configuration. Only one path exists between the peers. If this path goes down, alternative paths are not available.

b. Configure a discard static route for the 172.16.0.0/16 network. Any packets that do not have a more specific match (longer match) for a 172.16.0.0 subnet will be dropped instead of sent to the ISP. Later in this lab we will configure a default route to the ISP.

```
SanJose1(config)# ip route 172.16.0.0 255.255.0.0 null0
```

c. Configure SanJose1 as an EBGP peer to ISP.

```
SanJose1(config)# router bgp 64512
SanJose1(config-router)# neighbor 192.168.1.5 remote-as 200
SanJose1(config-router)# network 172.16.0.0
```

d. Use the **show ip bgp neighbors** command to verify that SanJose1 and ISP have reached the established state. Troubleshoot if necessary.

```
SanJose1# show ip bgp neighbors
BGP neighbor is 172.16.32.1,  remote AS 64512, internal link
  BGP version 4, remote router ID 172.16.32.1
  BGP state = Established, up for 00:12:43
<output omitted>

BGP neighbor is 192.168.1.5,  remote AS 200, external link
  BGP version 4, remote router ID 192.168.100.1
  BGP state = Established, up for 00:06:49
  Last read 00:00:42, last write 00:00:45, hold time is 180, keepalive interval is 60
seconds
<output omitted>
```

Notice that the “external link” indicates that an EBGP peering session has been established. You should also see an informational message indicating the establishment of the BGP neighbor relationship.

```
*Sep 8 21:09:59.699: %BGP-5-ADJCHANGE: neighbor 192.168.1.5 Up
```

e. Configure a discard static route for 172.16.0.0/16 on SanJose2 and as an EBGP peer to ISP.

```
SanJose2(config)# ip route 172.16.0.0 255.255.0.0 null0
SanJose2(config)# router bgp 64512
SanJose2(config-router)# neighbor 192.168.1.1 remote-as 200
SanJose2(config-router)# network 172.16.0.0
```

Step 5: View BGP summary output.

In Step 4, the **show ip bgp neighbors** command was used to verify that SanJose1 and ISP had reached the established state. A useful alternative command is **show ip bgp summary**. The output should be similar to the following.

```
SanJose2# show ip bgp summary
BGP router identifier 172.16.32.1, local AS number 64512
```

```
BGP table version is 6, main routing table version 6
2 network entries using 288 bytes of memory
4 path entries using 320 bytes of memory
4/2 BGP path/bestpath attribute entries using 640 bytes of memory
1 BGP AS-PATH entries using 24 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 1272 total bytes of memory
BGP activity 2/0 prefixes, 4/0 paths, scan interval 60 secs

Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down
State/PfxRcd
172.16.64.1     4        64512      27      26        6    0    0 00:18:15        2
192.168.1.1     4          200      10       7        6    0    0 00:01:42        1
SanJose2#
```

Step 6: Verify which path the traffic takes.

f. Clear the IP BGP conversation with the **clear ip bgp *** command on ISP. Wait for the conversations to reestablish with each SanJose router.

```
ISP# clear ip bgp *
ISP#
*Nov 9 22:05:32.427: %BGP-5-ADJCHANGE: neighbor 192.168.1.2 Down User reset
*Nov  9 22:05:32.427: %BGP_SESSION-5-ADJCHANGE: neighbor 192.168.1.2 IPv4 Unicast
   topology base removed from session  User reset
*Nov 9 22:05:32.427: %BGP-5-ADJCHANGE: neighbor 192.168.1.6 Down User reset
*Nov  9 22:05:32.427: %BGP_SESSION-5-ADJCHANGE: neighbor 192.168.1.6 IPv4 Unicast
   topology base removed from session  User reset
*Nov 9 22:05:32.851: %BGP-5-ADJCHANGE: neighbor 192.168.1.2 Up
*Nov  9 22:05:32.851: %BGP-
ISP#5-ADJCHANGE: neighbor 192.168.1.6 Up
ISP#
```

g. Test whether ISP can ping the loopback 0 address of 172.16.64.1 on SanJose1 and the serial link between SanJose1 and SanJose2, 172.16.1.1.

```
ISP# ping 172.16.64.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.64.1, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
ISP#
ISP# ping 172.16.1.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.1.1, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
ISP#
```

h. Now ping from ISP to the loopback 0 address of 172.16.32.1 on SanJose2 and the serial link between SanJose1 and SanJose2, 172.16.1.2.

```
ISP# ping 172.16.32.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.32.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 12/14/16 ms
ISP# ping 172.16.1.2
Type escape sequence to abort.
```

```
Sending 5, 100-byte ICMP Echos to 172.16.1.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 12/13/16 ms
ISP#
```

You should see successful pings to each IP address on SanJose2 router. Ping attempts to 172.16.64.1 and 172.16.1.1 should fail. Why does this happen?

i. Issue the **show ip bgp** command on ISP to verify BGP routes and metrics.

```
ISP# show ip bgp
BGP table version is 3, local router ID is 192.168.100.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
 *   172.16.0.0       192.168.1.6              0             0 64512 i
 *>                   192.168.1.2              0             0 64512 i
 *>  192.168.100.0    0.0.0.0                  0         32768 i
ISP#
ISP# show ip bgp
```

Notice that ISP has two valid routes to the 172.16.0.0 network, as indicated by the **.** However, the link to SanJose2 has been selected as the best path, indicated by the inclusion of the ">". Why did the ISP prefer the link to SanJose2 over SanJose1?

Would changing the bandwidth metric on each link help to correct this issue? Explain.

BGP operates differently than all other protocols. Unlike other routing protocols that use complex algorithms involving factors such as bandwidth, delay, reliability, and load to formulate a metric, BGP is policy-based. BGP determines the best path based on variables, such as AS path, weight, local preference, MED, and so on. If all things are equal, BGP prefers the route leading to the BGP speaker with the lowest BGP router ID. The SanJose2 router with BGP router ID 172.16.32.1 was preferred to the higher BGP router ID of the SanJose1 router (172.16.64.1).

j. At this point, the ISP router should be able to get to each network connected to SanJose1 and SanJose2 from the loopback address 192.168.100.1. Use the extended **ping** command and specify the source address of ISP Lo0 to test.

```
ISP# ping 172.16.1.1 source 192.168.100.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.1.1, timeout is 2 seconds:
Packet sent with a source address of 192.168.100.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 20/21/24 ms
```

```
ISP# ping 172.16.32.1 source 192.168.100.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.32.1, timeout is 2 seconds:
Packet sent with a source address of 192.168.100.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 12/15/16 ms

ISP# ping 172.16.1.2 source 192.168.100.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.1.2, timeout is 2 seconds:
Packet sent with a source address of 192.168.100.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 12/15/16 ms
ISP#

ISP# ping 172.16.64.1 source 192.168.100.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.64.1, timeout is 2 seconds:
Packet sent with a source address of 192.168.100.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 20/21/24 ms
```

You can also use the extended ping dialogue to specify the source address, as shown in this example.

```
ISP# ping
Protocol [ip]:
Target IP address: 172.16.64.1
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface: 192.168.100.1
Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.64.1, timeout is 2 seconds:
Packet sent with a source address of 192.168.100.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 20/20/24 ms
ISP#
```

Complete reachability has been demonstrated between the ISP router and both SanJose1 and SanJose2.

Step 7: Configure the BGP next-hop-self feature.

SanJose1 is unaware of the link between ISP and SanJose2, and SanJose2 is unaware of the link between ISP and SanJose1. Before ISP can successfully ping all the internal serial interfaces of AS 64512, these serial links should be advertised via BGP on the ISP router. This can also be resolved via EIGRP on each SanJose router. One method is for ISP to advertise these links.

a. Issue the following commands on the ISP router.

```
ISP(config)# router bgp 200
```

```
ISP(config-router)# network 192.168.1.0 mask 255.255.255.252
ISP(config-router)# network 192.168.1.4 mask 255.255.255.252
```

b. Issue the **show ip bgp** command to verify that the ISP is correctly injecting its own WAN links into BGP.

```
ISP# show ip bgp
BGP table version is 5, local router ID is 192.168.100.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
 *   172.16.0.0       192.168.1.6              0             0 64512 i
 *>                   192.168.1.2              0             0 64512 i
 *>  192.168.1.0/30   0.0.0.0                  0         32768 i
 *>  192.168.1.4/30   0.0.0.0                  0         32768 i
 *>  192.168.100.0    0.0.0.0                  0         32768 i
ISP#
```

c. Verify on SanJose1 and SanJose2 that the opposite WAN link is included in the routing table. The output from SanJose2 is as follows.

```
SanJose2# show ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override

Gateway of last resort is not set

      172.16.0.0/16 is variably subnetted, 6 subnets, 3 masks
S        172.16.0.0/16 is directly connected, Null0
C        172.16.1.0/24 is directly connected, Serial0/0/1
L        172.16.1.2/32 is directly connected, Serial0/0/1
C        172.16.32.0/24 is directly connected, Loopback0
L        172.16.32.1/32 is directly connected, Loopback0
D        172.16.64.0/24 [90/2297856] via 172.16.1.1, 00:52:03, Serial0/0/1
     192.168.1.0/24 is variably subnetted, 3 subnets, 2 masks
C        192.168.1.0/30 is directly connected, Serial0/0/0
L        192.168.1.2/32 is directly connected, Serial0/0/0
B        192.168.1.4/30 [20/0] via 192.168.1.1, 00:01:03
B     192.168.100.0/24 [20/0] via 192.168.1.1, 00:25:20
SanJose2#
```

The next issue to consider is BGP policy routing between autonomous systems. The next-hop attribute of a route in a different AS is set to the IP address of the border router in the next AS toward the destination, and this attribute is not modified by default when advertising this route through IBGP. Therefore, for all IBGP peers, it is either necessary to know the route to that border router (in a different neighboring AS), or our own border router needs to advertise the foreign routes using the next-hop-self feature, overriding the next-hop address with its own IP address. The SanJose2 router is passing a policy to SanJose1 and vice versa. The policy for routing from AS 64512 to AS 200 is to forward packets to the 192.168.1.1 interface. SanJose1 has a similar yet opposite policy: it forwards requests to the 192.168.1.5 interface. If either WAN link fails, it is critical that the opposite router become a valid gateway. This is achieved if the **next-hop-self** command is configured on SanJose1 and SanJose2.

d. To better understand the **next-hop-self** command we will remove ISP advertising its two WAN links and shutdown the WAN link between ISP and SanJose2. The only possible path from SanJose2 to ISP's 192.168.100.0/24 is through SanJose1.

```
ISP(config)# router bgp 200
ISP(config-router)# no network 192.168.1.0 mask 255.255.255.252
ISP(config-router)# no network 192.168.1.4 mask 255.255.255.252
ISP(config-router)# exit
ISP(config)# interface serial 0/0/1
ISP(config-if)# shutdown
ISP(config-if)#
```

e. Display SanJose2's BGP table using the **show ip bgp** command and the IPv4 routing table with **show ip route**.

```
SanJose2# show ip bgp
BGP table version is 1, local router ID is 172.16.32.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
 * i 172.16.0.0       172.16.64.1              0    100      0 i
 * i 192.168.100.0    192.168.1.5              0    100      0 200 i
SanJose2#
```

```
SanJose2# show ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override

Gateway of last resort is not set

      172.16.0.0/16 is variably subnetted, 6 subnets, 3 masks
S       172.16.0.0/16 is directly connected, Null0
C        172.16.1.0/24 is directly connected, Serial0/0/1
L       172.16.1.2/32 is directly connected, Serial0/0/1
C      172.16.32.0/24 is directly connected, Loopback0 L
172.16.32.1/32 is directly connected, Loopback0
D        172.16.64.0/24 [90/2297856] via 172.16.1.1, 02:41:46, Serial0/0/1
SanJose2#
```

Notice that SanJose2 has 192.168.100.0 in it's BGP table but not in its routing table. The BGP table shows the next hop to 192.168.100.0 as 192.168.1.5. Because SanJose2 does not have a route to this next hop address of 192.168.1.5 in its routing table, it will not install the 192.168.100.0 network into the routing table. It won't install a route if it doesn't know how to get to the next hop.

EBGP next hop addresses are carried into IBGP unchanged. As we saw previously, we could advertise the WAN link using BGP, but this is not always desirable. It means advertising additional routes when we are usually trying to minimize the size of the routing table. Another option is to have the routers within the IGP domain advertise themselves as the next hop router using the **next-hop-self** command.

f. Issue the **next-hop-self** command on SanJose1 and SanJose2 to advertise themselves as the next hop to their IBGP peer.

```
SanJose1(config)# router bgp 64512
SanJose1(config-router)# neighbor 172.16.32.1 next-hop-self

SanJose2(config)# router bgp 64512
SanJose2(config-router)# neighbor 172.16.64.1 next-hop-self
```

g. Reset BGP operation on either router with the **clear ip bgp *** command.

```
SanJose1# clear ip bgp *
SanJose1#

SanJose2# clear ip bgp *
SanJose2#
```

h. After the routers have returned to established BGP speakers, issue the **show ip bgp** command on SanJose2 and notice that the next hop is now SanJose1 instead of ISP.

```
SanJose2# show ip bgp
BGP table version is 5, local router ID is 172.16.32.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
 *>  172.16.0.0       0.0.0.0                  0         32768 i
 * i                  172.16.64.1              0    100      0 i
 *>i 192.168.100.0    172.16.64.1              0    100      0 200 i
SanJose2#
```

i. The **show ip route** command on SanJose2 now displays the 192.168.100.0/24 network because SanJose1 is the next hop, 172.16.64.1, which is reachable from SanJose2.

```
SanJose2# show ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override

Gateway of last resort is not set

      172.16.0.0/16 is variably subnetted, 6 subnets, 3 masks
S        172.16.0.0/16 is directly connected, Null0
C       172.16.1.0/24 is directly connected, Serial0/0/1
L       172.16.1.2/32 is directly connected, Serial0/0/1
C      172.16.32.0/24 is directly connected, Loopback0 L
172.16.32.1/32 is directly connected, Loopback0
D       172.16.64.0/24 [90/2297856] via 172.16.1.1, 04:27:19, Serial0/0/1
B     192.168.100.0/24 [200/0] via 172.16.64.1, 00:00:46
SanJose2#
```

j. Before configuring the next BGP attribute, restore the WAN link between ISP and SanJose3. This will change the BGP table and routing table on both routers. For example, SanJose2's routing table shows 192.168.100.0/24 will now have a better path through ISP.

```
ISP(config)# interface serial 0/0/1
ISP(config-if)# no shutdown
ISP(config-if)#

SanJose2# show ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override

Gateway of last resort is not set

      172.16.0.0/16 is variably subnetted, 6 subnets, 3 masks
S       172.16.0.0/16 is directly connected, Null0
C        172.16.1.0/24 is directly connected, Serial0/0/1
L       172.16.1.2/32 is directly connected, Serial0/0/1
C      172.16.32.0/24 is directly connected, Loopback0 L
172.16.32.1/32 is directly connected, Loopback0
D        172.16.64.0/24 [90/2297856] via 172.16.1.1, 04:37:34, Serial0/0/1
      192.168.1.0/24 is variably subnetted, 2 subnets, 2 masks
C         192.168.1.0/30 is directly connected, Serial0/0/0
L        192.168.1.2/32 is directly connected, Serial0/0/0
B     192.168.100.0/24 [20/0] via 192.168.1.1, 00:01:35
SanJose2#
```

Step 8: Set BGP local preference.

At this point, everything looks good, with the exception of default routes, the outbound flow of data, and inbound packet flow.

a. Because the local preference value is shared between IBGP neighbors, configure a simple route map that references the local preference value on SanJose1 and SanJose2. This policy adjusts outbound traffic to prefer the link off the SanJose1 router instead of the metered T1 off SanJose2.

```
SanJose1(config)# route-map PRIMARY_T1_IN permit 10
SanJose1(config-route-map)# set local-preference 150
SanJose1(config-route-map)# exit
SanJose1(config)# router bgp 64512
SanJose1(config-router)# neighbor 192.168.1.5 route-map PRIMARY_T1_IN in

SanJose2(config)# route-map SECONDARY_T1_IN permit 10
SanJose2(config-route-map)# set local-preference 125
SanJose1(config-route-map)# exit
SanJose2(config)# router bgp 64512
SanJose2(config-router)# neighbor 192.168.1.1 route-map SECONDARY_T1_IN in
```

b. Use the **clear ip bgp * soft** command after configuring this new policy. When the conversations have been reestablished, issue the **show ip bgp** command on SanJose1 and SanJose2.

```
SanJose1# clear ip bgp * soft
```

```
SanJose2# clear ip bgp * soft

SanJose1# show ip bgp
BGP table version is 3, local router ID is 172.16.64.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
 * i 172.16.0.0       172.16.32.1              0    100      0 i
 *>                   0.0.0.0                  0         32768 i
 *>  192.168.100.0    192.168.1.5              0    150      0 200 i
SanJose1#

SanJose2# show ip bgp
BGP table version is 7, local router ID is 172.16.32.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
 * i 172.16.0.0       172.16.64.1              0    100      0 i
 *>                   0.0.0.0                  0         32768 i
 *>i 192.168.100.0    172.16.64.1              0    150      0 200 i
 *                    192.168.1.1              0    125      0 200 i
SanJose2#
```

This now indicates that routing to the loopback segment for ISP 192.168.100.0 /24 can be reached only through the link common to SanJose1 and ISP. SanJose2's next hop to 192.168.100.0/24 is SanJose1 because both routers have been configured using the **next-hop-self** command.

Step 9: Set BGP MED.

a. In the previous step we saw that SanJose1 and SanJose2 will route traffic for 192.168.100.0/24 using the link between SanJose1 and ISP. Examine what the return path ISP takes to reach AS 64512. Notice that the return path is different from the original path. This is known as asymmetric routing and is not necessarily an unwanted trait.

```
ISP# show ip bgp
BGP table version is 22, local router ID is 192.168.100.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
 *   172.16.0.0       192.168.1.6              0             0 64512 i
 *>                   192.168.1.2              0             0 64512 i
 *>  192.168.100.0    0.0.0.0                  0         32768 i
ISP# show ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
```

```
        E1 - OSPF external type 1, E2 - OSPF external type 2
        i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
        ia - IS-IS inter area, * - candidate default, U - per-user static route
        o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
        a - application route
        + - replicated route, % - next hop override

Gateway of last resort is not set

B    172.16.0.0/16 [20/0] via 192.168.1.2, 00:12:45
     192.168.1.0/24 is variably subnetted, 4 subnets, 2 masks
C       192.168.1.0/30 is directly connected, Serial0/0/1
L       192.168.1.1/32 is directly connected, Serial0/0/1
C       192.168.1.4/30 is directly connected, Serial0/0/0
L       192.168.1.5/32 is directly connected, Serial0/0/0
     192.168.100.0/24 is variably subnetted, 2 subnets, 2 masks
C  192.168.100.0/24 is directly connected, Loopback0
L   192.168.100.1/32 is directly connected, Loopback0ISP#
```

How will traffic from network 192.168.100.0 /24 on ISP return to SanJose1 or SanJose2? Will it be routed through SanJose1 or SanJose2?

To verify this, the simplest solution is to issue the **show ip bgp** command on the ISP router as was done above. What if access was not given to the ISP router? Traffic returning from the Internet should not be passed across the metered T1. Is there a simple way to verify before receiving the monthly bill? How can it be checked instantly?

a. Use an extended **ping** command to verify this situation. Specify the **record** option and compare your output to the following. Notice the return path using the exit interface 192.168.1.1 to SanJose2.

```
SanJose2# ping
Protocol [ip]:
Target IP address: 192.168.100.1
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface: 172.16.32.1
Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]: record
Number of hops [ 9 ]:
Loose, Strict, Record, Timestamp, Verbose[RV]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.100.1, timeout is 2 seconds:
Packet sent with a source address of 172.16.32.1
Packet has IP options:  Total option bytes= 39, padded length=40
```

```
 Record route: <*>
    (0.0.0.0)
    (0.0.0.0)
    (0.0.0.0)
    (0.0.0.0)
    (0.0.0.0)
    (0.0.0.0)
    (0.0.0.0)
    (0.0.0.0)
    (0.0.0.0)

Reply to request 0 (20 ms).  Received packet has options
 Total option bytes= 40, padded length=40
 Record route:
    (172.16.1.2)
    (192.168.1.6)
    (192.168.100.1)
    (192.168.1.1)
    (172.16.32.1) <*>
    (0.0.0.0)
    (0.0.0.0)
    (0.0.0.0)
    (0.0.0.0)
 End of list

Reply to request 1 (20 ms).  Received packet has options
 Total option bytes= 40, padded length=40
 Record route:
    (172.16.1.2)
    (192.168.1.6)
    (192.168.100.1)
    (192.168.1.1)
    (172.16.32.1) <*>
    (0.0.0.0)
    (0.0.0.0)
    (0.0.0.0)
    (0.0.0.0)
 End of list

Reply to request 2 (20 ms).  Received packet has options
 Total option bytes= 40, padded length=40
 Record route:
    (172.16.1.2)
    (192.168.1.6)
    (192.168.100.1)
    (192.168.1.1)
    (172.16.32.1) <*>
    (0.0.0.0)
    (0.0.0.0)
    (0.0.0.0)
    (0.0.0.0)
 End of list

Reply to request 3 (24 ms).  Received packet has options
 Total option bytes= 40, padded length=40
 Record route:
    (172.16.1.2)
    (192.168.1.6)
```

```
   (192.168.100.1)
   (192.168.1.1)
   (172.16.32.1) <*>
   (0.0.0.0)
   (0.0.0.0)
   (0.0.0.0)
   (0.0.0.0)
 End of list

Reply to request 4 (20 ms).  Received packet has options
 Total option bytes= 40, padded length=40
 Record route:
   (172.16.1.2)
   (192.168.1.6)
   (192.168.100.1)
   (192.168.1.1)
   (172.16.32.1) <*>
   (0.0.0.0)
   (0.0.0.0)
   (0.0.0.0)
   (0.0.0.0)
 End of list

Success rate is 100 percent (5/5), round-trip min/avg/max = 20/20/24 ms
SanJose2#
```

If you are unfamiliar with the **record** option, the important thing to note is that each IP address in brackets is an outgoing interface. The output can be interpreted as follows:

1. A ping that is sourced from 172.16.32.1 exits SanJose2 through s0/0/1, 172.16.1.2. It then arrives at the s0/0/1 interface for SanJose1.
2. SanJose1 S0/0/0, 192.168.1.6, routes the packet out to arrive at the S0/0/0 interface of ISP.
3. The target of 192.168.100.1 is reached: 192.168.100.1.
4. The packet is next forwarded out the S0/0/1, 192.168.1.1 interface for ISP and arrives at the S0/0/0 interface for SanJose2.
5. SanJose2 then forwards the packet out the last interface, loopback 0, 172.16.32.1.

Although the unlimited use of the T1 from SanJose1 is preferred here, ISP currently takes the link from SanJose2 for all return traffic.

b. Create a new policy to force the ISP router to return all traffic via SanJose1. Create a second route map utilizing the MED (metric) that is shared between EBGP neighbors.

```
SanJose1(config)#route-map PRIMARY_T1_MED_OUT permit 10
SanJose1(config-route-map)#set Metric 50
SanJose1(config-route-map)#exit
SanJose1(config)#router bgp 64512
SanJose1(config-router)#neighbor 192.168.1.5 route-map PRIMARY_T1_MED_OUT out

SanJose2(config)#route-map SECONDARY_T1_MED_OUT permit 10
SanJose2(config-route-map)#set Metric 75
SanJose2(config-route-map)#exit
SanJose2(config)#router bgp 64512
SanJose2(config-router)#neighbor 192.168.1.1 route-map SECONDARY_T1_MED_OUT out
```

c. Use the **clear ip bgp * soft** command after issuing this new policy. Issuing the **show ip bgp** command as follows on SanJose1 or SanJose2 does not indicate anything about this newly defined policy.

```
SanJose1# clear ip bgp * soft
SanJose2# clear ip bgp * soft

SanJose1# show ip bgp
BGP table version is 4, local router ID is 172.16.64.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
 * i 172.16.0.0       172.16.32.1              0    100      0 i
 *>                   0.0.0.0                  0         32768 i
 *>  192.168.100.0    192.168.1.5              0    150      0 200 i
SanJose1#

SanJose2# show ip bgp
BGP table version is 8, local router ID is 172.16.32.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
 * i 172.16.0.0       172.16.64.1              0    100      0 i
 *>                   0.0.0.0                  0         32768 i
 *>i 192.168.100.0    172.16.64.1              0    150      0 200 i
 *                    192.168.1.1              0    125      0 200 i
SanJose2#
```

d. Reissue an extended **ping** command with the **record** command. Notice the change in return path using the exit interface 192.168.1.5 to SanJose1.

```
SanJose2# ping
Protocol [ip]:
Target IP address: 192.168.100.1
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface: 172.16.32.1
Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]: record
Number of hops [ 9 ]:
Loose, Strict, Record, Timestamp, Verbose[RV]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.100.1, timeout is 2 seconds:
Packet sent with a source address of 172.16.32.1
Packet has IP options:  Total option bytes= 39, padded length=40
 Record route: <*>
   (0.0.0.0)
   (0.0.0.0)
```

```
    (0.0.0.0)
    (0.0.0.0)
    (0.0.0.0)
    (0.0.0.0)
    (0.0.0.0)
    (0.0.0.0)
    (0.0.0.0)

Reply to request 0 (28 ms).  Received packet has options
 Total option bytes= 40, padded length=40
 Record route:
    (172.16.1.2)
    (192.168.1.6)
    (192.168.100.1)
    (192.168.1.5)
    (172.16.1.1)
    (172.16.32.1) <*>
    (0.0.0.0)
    (0.0.0.0)
    (0.0.0.0)
 End of list

Reply to request 1 (28 ms).  Received packet has options
 Total option bytes= 40, padded length=40
 Record route:
    (172.16.1.2)
    (192.168.1.6)
    (192.168.100.1)
    (192.168.1.5)
    (172.16.1.1)
    (172.16.32.1) <*>
    (0.0.0.0)
    (0.0.0.0)
    (0.0.0.0)
 End of list

Reply to request 2 (28 ms).  Received packet has options
 Total option bytes= 40, padded length=40
 Record route:
    (172.16.1.2)
    (192.168.1.6)
    (192.168.100.1)
    (192.168.1.5)
    (172.16.1.1)
    (172.16.32.1) <*>
    (0.0.0.0)
    (0.0.0.0)
    (0.0.0.0)
 End of list

Reply to request 3 (28 ms).  Received packet has options
 Total option bytes= 40, padded length=40
 Record route:
    (172.16.1.2)
    (192.168.1.6)
    (192.168.100.1)
    (192.168.1.5)
    (172.16.1.1)
```

```
   (172.16.32.1) <*>
   (0.0.0.0)
   (0.0.0.0)
   (0.0.0.0)
 End of list

Reply to request 4 (28 ms).  Received packet has options
 Total option bytes= 40, padded length=40
 Record route:
   (172.16.1.2)
   (192.168.1.6)
   (192.168.100.1)
   (192.168.1.5)
   (172.16.1.1)
   (172.16.32.1) <*>
   (0.0.0.0)
   (0.0.0.0)
   (0.0.0.0)
 End of list

Success rate is 100 percent (5/5), round-trip min/avg/max = 28/28/28 ms
SanJose2#
```

Does the output look correct? Does the 192.168.1.5 above mean that the ISP now prefers SanJose1 for return traffic?

The newly configured policy MED shows that the lower MED value is considered best. The ISP now prefers the route with the lower MED value of 50 to AS 64512. This is just opposite from the **local-preference** command configured earlier.

```
ISP# show ip bgp
BGP table version is 24, local router ID is 192.168.100.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
 *>  172.16.0.0       192.168.1.6              50             0 64512 i
 *                    192.168.1.2              75             0 64512 i
 *>  192.168.100.0    0.0.0.0                   0         32768 i
ISP#
```

Step 10: Establish a default route.

The final step is to establish a default route that uses a policy statement that adjusts to changes in the network.

a. Configure ISP to inject a default route to both SanJose1 and SanJose2 using BGP using the **default-originate** command. This command does not require the presence of 0.0.0.0 in the ISP router. Configure the 10.0.0.0/8 network which will not be advertised using BGP. This network will be used to test the default route on SanJose1 and SanJose2.

```
ISP(config)# router bgp 200
ISP(config-router)# neighbor 192.168.1.6 default-originate
ISP(config-router)# neighbor 192.168.1.2 default-originate
```

```
ISP(config-router)# exit
ISP(config)# interface loopback 10
ISP(config-if)# ip address 10.0.0.1 255.255.255.0
ISP(config-if)#
```

b. Verify that both routers have received the default route by examining the routing tables on SanJose1 and SanJose2. Notice that both routers prefer the route between SanJose1 and ISP.

```
SanJose1# show ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override

Gateway of last resort is 192.168.1.5 to network 0.0.0.0

B*    0.0.0.0/0 [20/0] via 192.168.1.5, 00:00:36
      172.16.0.0/16 is variably subnetted, 6 subnets, 3 masks
S        172.16.0.0/16 is directly connected, Null0
C        172.16.1.0/24 is directly connected, Serial0/0/1
L        172.16.1.1/32 is directly connected, Serial0/0/1
D        172.16.32.0/24 [90/2297856] via 172.16.1.2, 05:47:24, Serial0/0/1
C        172.16.64.0/24 is directly connected, Loopback0
L        172.16.64.1/32 is directly connected, Loopback0
      192.168.1.0/24 is variably subnetted, 2 subnets, 2 masks
C        192.168.1.4/30 is directly connected, Serial0/0/0
L        192.168.1.6/32 is directly connected, Serial0/0/0
SanJose1#

SanJose2# show ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override

Gateway of last resort is 172.16.64.1 to network 0.0.0.0

B*    0.0.0.0/0 [200/0] via 172.16.64.1, 00:00:45
      172.16.0.0/16 is variably subnetted, 6 subnets, 3 masks
S       172.16.0.0/16 is directly connected, Null0
C        172.16.1.0/24 is directly connected, Serial0/0/1
L       172.16.1.2/32 is directly connected, Serial0/0/1
C      172.16.32.0/24 is directly connected, Loopback0 L
172.16.32.1/32 is directly connected, Loopback0
D        172.16.64.0/24 [90/2297856] via 172.16.1.1, 05:47:33, Serial0/0/1
      192.168.1.0/24 is variably subnetted, 2 subnets, 2 masks
C       192.168.1.0/30 is directly connected, Serial0/0/0
L     192.168.1.2/32 is directly connected, Serial0/0/0
SanJose2#
```

c. The preferred default route is by way of SanJose1 because of the higher local preference attribute configured on SanJose1 earlier.

```
SanJose2# show ip bgp
BGP table version is 38, local router ID is 172.16.32.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
 *>i 0.0.0.0          172.16.64.1              0    150      0 200 i
 *                    192.168.1.1                   125      0 200 i
 * i 172.16.0.0       172.16.64.1              0    100      0 i
 *>                   0.0.0.0                  0         32768 i
 *>i 192.168.100.0    172.16.64.1              0    150      0 200 i
 *                    192.168.1.1              0    125      0 200 i
SanJose2#
```

d. Using the traceroute command verify that packets to 10.0.0.1 is using the default route through SanJose1.

```
SanJose2# traceroute 10.0.0.1
Type escape sequence to abort.
Tracing the route to 10.0.0.1
VRF info: (vrf in name/id, vrf out name/id)
  1 172.16.1.1 8 msec 4 msec 8 msec
  2 192.168.1.5 [AS 200] 12 msec *  12 msec
SanJose2#
```

e. Next, test how BGP adapts to using a different default route when the path between SanJose1 and ISP goes down.

```
ISP(config)# interface serial 0/0/0
ISP(config-if)# shutdown
ISP(config-if)#
```

f. Verify that both routers are modified their routing tables with the default route using the path between SanJose2 and ISP.

```
SanJose1# show ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override

Gateway of last resort is 172.16.32.1 to network 0.0.0.0

B*    0.0.0.0/0 [200/0] via 172.16.32.1, 00:00:06
      172.16.0.0/16 is variably subnetted, 6 subnets, 3 masks
S        172.16.0.0/16 is directly connected, Null0
C        172.16.1.0/24 is directly connected, Serial0/0/1
L        172.16.1.1/32 is directly connected, Serial0/0/1
```

```
D       172.16.32.0/24 [90/2297856] via 172.16.1.2, 05:49:25, Serial0/0/1
C       172.16.64.0/24 is directly connected, Loopback0
L       172.16.64.1/32 is directly connected, Loopback0
B  192.168.100.0/24 [200/0] via 172.16.32.1, 00:00:06
SanJose1#

SanJose2# show ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override

Gateway of last resort is 192.168.1.1 to network 0.0.0.0

B*    0.0.0.0/0 [20/0] via 192.168.1.1, 00:00:30
      172.16.0.0/16 is variably subnetted, 6 subnets, 3 masks
S       172.16.0.0/16 is directly connected, Null0
C         172.16.1.0/24 is directly connected, Serial0/0/1
L       172.16.1.2/32 is directly connected, Serial0/0/1
C      172.16.32.0/24 is directly connected, Loopback0 L
172.16.32.1/32 is directly connected, Loopback0
D         172.16.64.0/24 [90/2297856] via 172.16.1.1, 05:49:49, Serial0/0/1
      192.168.1.0/24 is variably subnetted, 2 subnets, 2 masks
C       192.168.1.0/30 is directly connected, Serial0/0/0
L         192.168.1.2/32 is directly connected, Serial0/0/0
B     192.168.100.0/24 [20/0] via 192.168.1.1, 00:00:30
SanJose2#
```

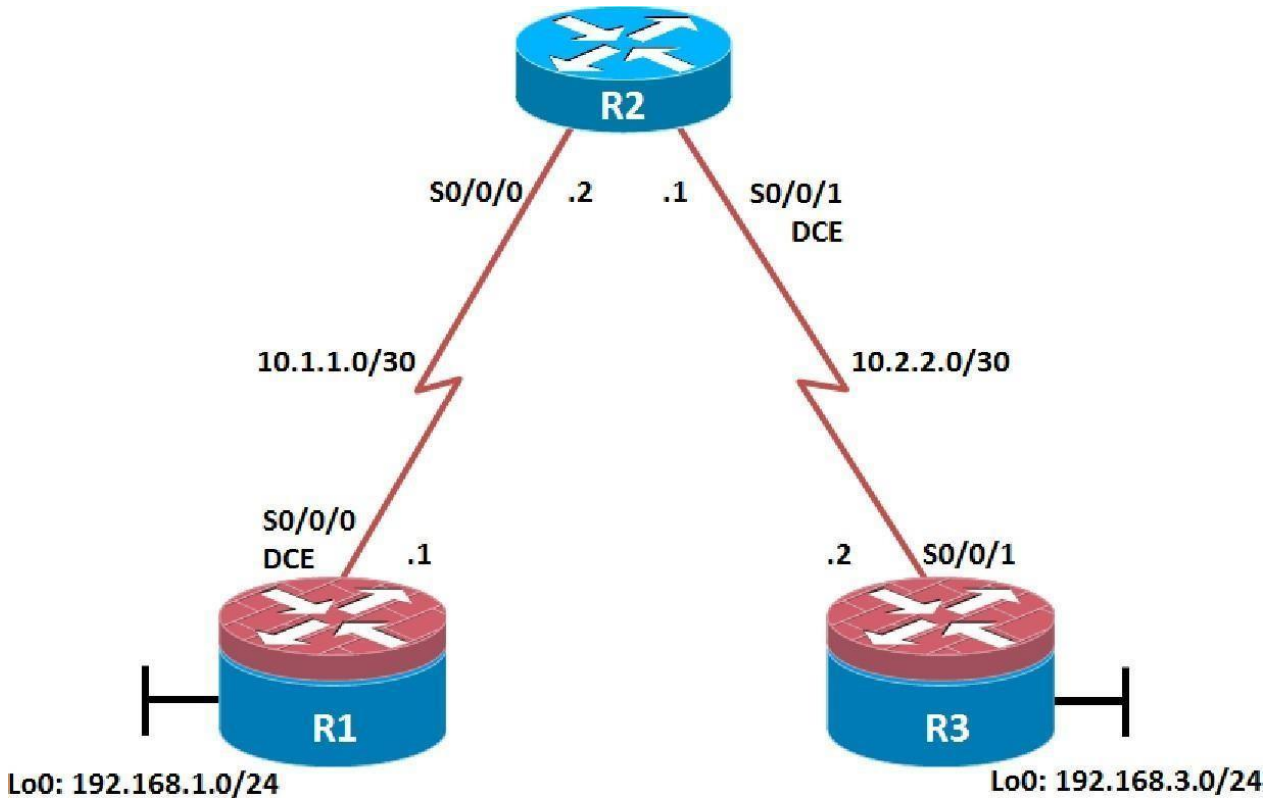
g. Verify the new path using the traceroute command to 10.0.0.1 from SanJose1. Notice the default route is now through SanJose2.

```
SanJose1# trace 10.0.0.1
Type escape sequence to abort.
Tracing the route to 10.0.0.1
VRF info: (vrf in name/id, vrf out name/id)
  1 172.16.1.2 8 msec 8 msec 8 msec
  2 192.168.1.1 [AS 200] 12 msec *  12 msec
SanJose1#
```

Practical 4

Secure the Management Plane

Topology



Objectives

- Secure management access.
- Configure enhanced username password security.
- Enable AAA RADIUS authentication.
- Enable secure remote management.

Background

The management plane of any infrastructure device should be protected as much as possible. Controlling access to routers and enabling reporting on routers are critical to network security and should be part of a comprehensive security policy.

In this lab, you build a multi-router network and secure the management plane of routers R1 and R3.

Note: This lab uses Cisco 1941 routers with Cisco IOS Release 15.2 with IP Base. Depending on the router or switch model and Cisco IOS Software version, the commands available and output produced might vary from what is shown in this lab.

Required Resources

- 3 routers (Cisco IOS Release 15.2 or comparable)
- Serial and Ethernet cables

Step 1: Configure loopbacks and assign addresses.

Cable the network as shown in the topology diagram. Erase the startup configuration and reload each router to clear previous configurations. Using the addressing scheme in the diagram, apply the IP addresses to the interfaces on the R1, R2, and R3 routers.

You can copy and paste the following configurations into your routers to begin.

Note: Depending on the router model, interfaces might be numbered differently than those listed. You might need to alter the designations accordingly.

R1

```
hostname R1

interface Loopback 0
 description R1 LAN
 ip address 192.168.1.1 255.255.255.0
exit
!
interface Serial0/0/0
 description R1 --> R2
 ip address 10.1.1.1 255.255.255.252
 clock rate 128000
 no shutdown
exit
!
end
```

R2

```
hostname R2
!
interface Serial0/0/0
 description R2 --> R1
 ip address 10.1.1.2 255.255.255.252
 no shutdown
exit
```

```
interface Serial0/0/1
 description R2 --> R3
 ip address 10.2.2.1 255.255.255.252
 clock rate 128000
 no shutdown
exit
!
end
```

R3

```
hostname R3
!
interface Loopback0
 description R3 LAN
 ip address 192.168.3.1 255.255.255.0
exit


interface Serial0/0/1
 description R3 --> R2
 ip address 10.2.2.2 255.255.255.252
 no shutdown
exit
!
end
```

Step 2: Configure static routes.

a. On R1, configure a default static route to ISP.

```
R1(config)# ip route 0.0.0.0 0.0.0.0 10.1.1.2
```

b. On R3, configure a default static route to ISP.

```
R3(config)# ip route 0.0.0.0 0.0.0.0 10.2.2.1
```

c. On R2, configure two static routes.

```
R2(config)# ip route 192.168.1.0 255.255.255.0 10.1.1.1
R2(config)# ip route 192.168.3.0 255.255.255.0 10.2.2.2
```

d. From the R1 router, run the following Tcl script to verify connectivity.

```
foreach address {
192.168.1.1
10.1.1.1
10.1.1.2
10.2.2.1
10.2.2.2
192.168.3.1
} { ping $address }

R1# tclsh

R1(tcl)#foreach address {
+>(tcl)#192.168.1.1
```

```
+>(tcl)#10.1.1.1
+>(tcl)#10.1.1.2
+>(tcl)#10.2.2.1
+>(tcl)#10.2.2.2
+>(tcl)#192.168.3.1
+>(tcl)#} { ping $address }
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.1.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.2.2.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.2.2.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 12/14/16 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.3.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 12/15/16 ms
R1(tcl)#
```

Are the pings now successful?

Yes. If not, troubleshoot.

Step 3: Secure management access.

a. On R1, use the **security passwords** command to set a minimum password length of 10 characters.

```
R1(config)# security passwords min-length 10
```

b. Configure the enable secret encrypted password on both routers.

```
R1(config)# enable secret class12345
```

How does configuring an enable secret password help protect a router from being compromised by an attack?

_____

_____

_____

The goal is to always prevent unauthorized users from accessing a device using Telnet, SSH, or via the console. If attackers are able to penetrate this first layer of defense, using an enable secret password prevents them from being able to alter the configuration of the device. Unless the enable secret password is known, a user

cannot go into privileged EXEC mode where they can display the running config and enter various configuration commands to make changes to the router. This provides an additional layer of security.

Note: Passwords in this task are set to a minimum of 10 characters but are relatively simple for the benefit of performing the lab. More complex passwords are recommended in a production network.

c. Configure a console password and enable login for routers. For additional security, the **exec-timeout** command causes the line to log out after 5 minutes of inactivity. The **logging synchronous** command prevents console messages from interrupting command entry.

Note: To avoid repetitive logins during this lab, the **exec-timeout** command can be set to 0 0, which prevents it from expiring. However, this is not considered a good security practice.

```
R1(config)# line console 0
R1(config-line)# password ciscoconpass
R1(config-line)# exec-timeout 5 0
R1(config-line)# login
R1(config-line)# logging synchronous
R1(config-line)# exit
R1(config)#
```

d. Configure the password on the vty lines for router R1.

```
R1(config)# line vty 0 4
R1(config-line)# password ciscovtypass
R1(config-line)# exec-timeout 5 0
R1(config-line)# login
R1(config-line)# exit
R1(config)#
```

e. The aux port is a legacy port used to manage a router remotely using a modem and is hardly ever used. Therefore, disable the aux port.

```
R1(config)# line aux 0
R1(config-line)# no exec
R1(config-line)# end
R1#
```

f. Enter privileged EXEC mode and issue the **show run** command. Can you read the enable secret password? Why or why not?

_____

_____

No. The enable secret password is encrypted automatically using the MD5 or SHA hash algorithm. . IOS 15.0(1)S and later default to SHA256 hashing algorithm. SHA256 which is considered to be a very strong hashing algorithm and is extremely difficult to reverse. Earlier IOS versions use the weaker MD5 hashing algorithm.

Note: If the **enable secret** password command is lost or forgotten, it must be replaced using the Cisco router password recovery procedure. Refer to cisco.com for more information.

Can you read the console, aux, and vty passwords? Why or why not?

Yes. They are all in clear text.

g. Use the **service password-encryption** command to encrypt the line console and vty passwords.

```
R1(config)# service password-encryption
R1(config)#
```

Note: Password encryption is applied to all the passwords, including the **username** passwords, the authentication key passwords, the privileged command password, the console and the virtual terminal line access passwords, and the BGP neighbor passwords.

h. Issue the **show run** command. Can you read the console, aux, and vty passwords? Why or why not?

No. The passwords are now encrypted.

Note: Type 7 passwords are encrypted using a Vigenère cipher which can be easily reversed. Therefore this command primarily protects from shoulder surfing attacks.

i. Configure a warning to unauthorized users with a message-of-the-day (MOTD) banner using the **banner motd** command. When a user connects to one of the routers, the MOTD banner appears before the login prompt. In this example, the dollar sign ($) is used to start and end the message.

```
R1(config)# banner motd $Unauthorized access strictly prohibited!$
R1(config)# exit
```

j. Issue the **show run** command. What does the $ convert to in the output?

The $ is converted to ^C when the running-config is displayed.

k. Exit privileged EXEC mode using the **disable** or **exit** command and press **Enter** to get started. Does the MOTD banner look like what you created with the **banner motd** command? If the MOTD banner is not as you wanted it, recreate it using the **banner motd** command.

l. Repeat the configuration portion of steps 3a through 3k on router R3.

Step 4: Configure enhanced username password security.

To increase the encryption level of console and VTY lines, it is recommended to enable authentication using the local database. The local database consists of usernames and password combinations that are created locally on each device. The local and VTY lines are configured to refer to the local database when authenticating a user.

a. To create local database entry encrypted to level 4 (SHA256), use the **username** *name* **secret** *password* global configuration command. In global configuration mode, enter the following command:

```
R1(config)# username JR-ADMIN secret class12345
R1(config)# username ADMIN secret class54321
```

Note: An older method for creating local database entries is to use the **username** *name* **password** *password* command.

b. Set the console line to use the locally defined login accounts.

```
R1(config)# line console 0
R1(config-line)# login local
R1(config-line)# exit
R1(config)#
```

c. Set the vty lines to use the locally defined login accounts.

```
R1(config)# line vty 0 4
R1(config-line)# login local
R1(config-line)# end
R1(config)#
```

d. Repeat the steps 4a to 4c on R3.

e. To verify the configuration, telnet to R3 from R1 and login using the ADMIN local database account.

```
R1# telnet 10.2.2.2
Trying 10.2.2.2 ... Open
Unauthorized access strictly prohibited!
User Access Verification

Username: ADMIN
Password:
R3>
```

Step 5: Enabling AAA RADIUS Authentication with Local User for Backup.

Authentication, authorization, and accounting (AAA) is a standards-based framework that can be implemented to control who is permitted to access a network (authenticate), what they can do on that network (authorize), and audit what they did while accessing the network (accounting).

Users must authenticate against an authentication database which can be stored:

- **Locally**: Users are authenticated against the local device database which is created using the username secret command. Sometimes referred to self-contained AAA.

- **Centrally**: A client-server model where users are authenticated against AAA servers. This provides improved scalability, manageability and control. Communication between the device and AAA servers is secured using either the RADIUS or TACACS+ protocols.

In this step, we will configure AAA authentication to use a RADIUS server and the local database as a backup. Specifically, the authentication will be validated against one of two RADIUS servers. If the servers are not available, then authentication will be validated against the local database.

a. Always have local database accounts created before enabling AAA. Since we created two local database accounts in the previous step, then we can proceed and enable AAA on R1.

```
R1(config)# aaa new-model
```

Note: Although the following configuration refers to two RADIUS servers, the actual RADIUS server implementation is beyond the scope. Therefore, the goal of this step is to provide an example of how to configure a router to access the servers.

b. Configure the specifics for the first RADIUS server located at 192.168.1.101. Use **RADIUS-1-pa55w0rd** as the server password.

```
R1(config)# radius server RADIUS-1
R1(config-radius-server)# address ipv4 192.168.1.101
R1(config-radius-server)# key RADIUS-1-pa55w0rd
R1(config-radius-server)# exit
R1(config)#
```

c. Configure the specifics for the second RADIUS server located at 192.168.1.102. Use **RADIUS-2-pa55w0rd** as the server password.

```
R1(config)# radius server RADIUS-2
R1(config-radius-server)# address ipv4 192.168.1.102
R1(config-radius-server)# key RADIUS-2-pa55w0rd
R1(config-radius-server)# exit
R1(config)#
```

d. Assign both RADIUS servers to a server group.

```
R1(config)# aaa group server radius RADIUS-GROUP
R1(config-sg-radius)# server name RADIUS-1
R1(config-sg-radius)# server name RADIUS-2
R1(config-sg-radius)# exit
R1(config)#
```

e. Enable the default AAA authentication login to attempt to validate against the server group. If they are not available, then authentication should be validated against the local database..

```
R1(config)# aaa authentication login default group RADIUS-GROUP local
R1(config)#
```

Note: Once this command is configured, all line access methods default to the default authentication method. The **local** option enables AAA to refer to the local database. Only the password is case sensitive.

f. Enable the default AAA authentication Telnet login to attempt to validate against the server group. If they are not available, then authentication should be validated against a case sensitive local database.

```
R1(config)# aaa authentication login TELNET-LOGIN group RADIUS-GROUP local-case
R1(config)#
```

Note: Unlike the **local** option that makes the password is case sensitive, local-case makes the username and password case sensitive.

g. Alter the VTY lines to use the TELNET-LOGIN AAA authentiaito0n method.

```
R1(config)# line vty 0 4
R1(config-line)# login authentication TELNET-LOGIN
R1(config-line)# exit
R1(config)#
```

h. Repeat the steps 5a to 5g on R3.

i. To verify the configuration, telnet to R3 from R1 and login using the ADMIN local database account.

```
R1# telnet 10.2.2.2
Trying 10.2.2.2 ... Open
Unauthorized access strictly prohibited!

User Access Verification

Username: admin
Password:

% Authentication failed

Username: ADMIN
Password:

R3>
```

Note: The first login attempt did not use the correct username (i.e., ADMIN) which is why it failed.

Note: The actual login time is longer since the RADIUS servers are not available.

Step 6: Enabling secure remote management using SSH.

Traditionally, remote access on routers was configured using Telnet on TCP port 23. However, Telnet was developed in the days when security was not an issue; therefore, all Telnet traffic is forwarded in plaintext.

Secure Shell (SSH) is a network protocol that establishes a secure terminal emulation connection to a router or other networking device. SSH encrypts all information that passes over the network link and provides authentication of the remote computer. SSH is rapidly replacing Telnet as the remote login tool of choice for network professionals.

Note: For a router to support SSH, it must be configured with local authentication, (AAA services, or username) or password authentication. In this task, you configure an SSH username and local authentication.

In this step, you will enable R1 and R3 to support SSH instead of Telnet.

a. SSH requires that a device name and a domain name be configured. Since the router already has a name assigned, configure the domain name.

```
R1(config)# ip domain-name ccnasecurity.com
```

b. The router uses the RSA key pair for authentication and encryption of transmitted SSH data. Although optional it may be wise to erase any existing key pairs on the router.

```
R1(config)# crypto key zeroize rsa
```

Note: If no keys exist, you might receive this message: `% No Signature RSA Keys found in configuration.`

c. Generate the RSA encryption key pair for the router. Configure the RSA keys with **1024** for the number of modulus bits. The default is 512, and the range is from 360 to 2048.

```
R1(config)# crypto key generate rsa general-keys modulus 1024
The name for the keys will be: R1.ccnasecurity.com

% The key modulus size is 1024 bits
% Generating 1024 bit RSA keys, keys will be non-exportable...[OK]

R1(config)#
Jan 10 13:44:44.711: %SSH-5-ENABLED: SSH 1.99 has been enabled
R1(config)#
```

d. Cisco routers support two versions of SSH:

- **SSH version 1 (SSHv1)**: Original version but has known vulnerabilities.
- **SSH version 2 (SSHv2)**: Provides better security using the Diffie-Hellman key exchange and the strong integrity-checking message authentication code (MAC).

The default setting for SSH is SSH version 1.99. This is also known as compatibility mode and is merely an indication that the server supports both SSH version 2 and SSH version 1. However, best practices are to enable version 2 only.

Configure SSH version 2 on R1.

```
R1(config)# ip ssh version 2
```

```
R1(config)#
```

e. Configure the vty lines to use only SSH connections.

```
R1(config)# line vty 0 4
R1(config-line)# transport input ssh
R1(config-line)# end
```

Note: SSH requires that the **login local** command be configured. However, in the previous step we enabled AAA authentication using the TELNET-LOGIN authentication method, therefore **login local** is not necessary.

Note: If you add the keyword **telnet** to the **transport input** command, users can log in using Telnet as well as SSH. However, the router will be less secure. If only SSH is specified, the connecting host must have an SSH client installed.

f. Verify the SSH configuration using the **show ip ssh** command.

```
R1# show ip ssh
SSH Enabled - version 2.0
Authentication timeout: 120 secs; Authentication retries: 3
Minimum expected Diffie Hellman key size : 1024 bits
IOS Keys in SECSH format(ssh-rsa, base64 encoded):
ssh-rsa  AAAAB3NzaC1yc2EAAAADAQABAAAAgQC3Lehh7ReYlgyDzls6wq+mFzxqzoaZFr9XGx+Q/yio
dFYw00hQo80tZy1W1Ff3Pz6q7Qi0y00urwddHZ0kBZceZK9EzJ6wZ+9a87KKDETCWrGSLi6c8lE/y4K+
Z/oVrMMZk7bpTM1MFdP41YgkTf35utYv+TcqbsYo++KJiYk+xw==
R1#
```

g. Repeat the steps 6a to 6f on R3.

h. Although a user can SSH from a host using the SSH option of TeraTerm of PuTTY, a router can also SSH to another SSH enabled device. SSH to R3 from R1.

```
R1# ssh -l ADMIN 10.2.2.2
Password:
Unauthorized access strictly prohibited!
R3>
R3> en
Password:
R3#
```

Device Configurations

Router R1

```
service password-encryption
!
hostname R1
!
security passwords min-length 10
enable secret 5 $1$t6eK$FZ.JdmMLj8QSgNkpChyZz.
!
aaa new-model
!
!
aaa group server radius RADIUS-GROUP
 server name RADIUS-1
 server name RADIUS-2
!
aaa authentication login default group RADIUS-GROUP local
aaa authentication login TELNET-LOGIN group RADIUS-GROUP local-case
!
ip domain name ccnasecurity.com
!
username JR-ADMIN secret 5 $1$0u0q$lwimCZIAuQtV4C1ezXL1S0
username ADMIN secret 5 $1$NSVD$/YjzB7Auyes1sAt4qMfpd.
!
ip ssh version 2
!
interface Loopback0
 description R1 LAN
 ip address 192.168.1.1 255.255.255.0
!
interface Serial0/0/0
 description R1 --> R2
 ip address 10.1.1.1 255.255.255.252
 no fair-queue
!
ip route 0.0.0.0 0.0.0.0 10.1.1.2
!
radius server RADIUS-1
 address ipv4 192.168.1.101 auth-port 1645 acct-port 1646
 key 7 107C283D2C2221465D493A2A717D24653017
!
radius server RADIUS-2
 address ipv4 192.168.1.102 auth-port 1645 acct-port 1646
 key 7 03367A2F2F3A12011C44090442471C5C162E
!
banner motd ^CUnauthorized access strictly prohibited!^C
!
line con 0
 exec-timeout 5 0
 password 7 070C285F4D061A0A19020A1F17
 logging synchronous
!
line aux 0
```

```
 no exec
!
password 7 060506324F411F0D1C0713181F
 login authentication TELNET-LOGIN
 transport input ssh
!
end
```

Router R2

```
hostname R2
!
enable secret 5 $1$DJS7$xvJDW87zLs8pSJDFUlCPB1
!
interface Serial0/0/0
 description R2 --> R1
 ip address 10.1.1.2 255.255.255.252
 no fair-queue
 clock rate 2000000
!
interface Serial0/0/1
 description R2 --> R3
 ip address 10.2.2.1 255.255.255.252
 clock rate 128000
!
ip route 192.168.1.0 255.255.255.0 10.1.1.1
ip route 192.168.3.0 255.255.255.0 10.2.2.2
!
line con 0
 exec-timeout 0 0
 logging synchronous
!
line vty 0 4
 password cisco
 login
!
end
```

Router R3

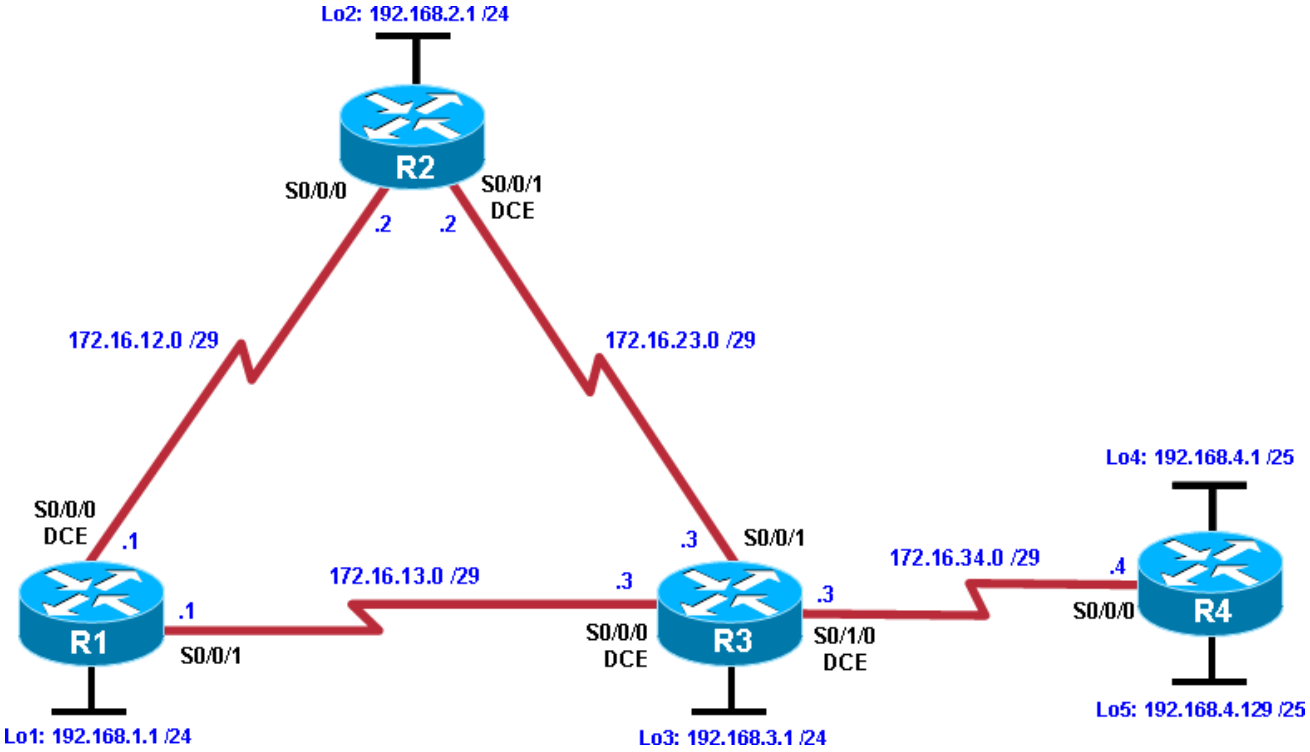
```
service password-encryption
!
hostname R3
!
security passwords min-length 10
enable secret 5 $1$5OY4$4J6VFlvGNKjwQ8XtajgUk1
!
aaa new-model
!
!
aaa group server radius RADIUS-GROUP
 server name RADIUS-1
 server name RADIUS-2
!
aaa authentication login default group RADIUS-GROUP local
aaa authentication login TELNET-LOGIN group RADIUS-GROUP local-case
```

```
!
ip domain name ccnasecurity.com
!
username JR-ADMIN secret 5 $1$b4m1$RVmjL9S3gxKh1xr8qzNqr/
username ADMIN secret 5 $1$zGV7$pVgSEbinvXQ7f7uyxeKBj0
!
ip ssh version 2
!
interface Loopback0
 description R3 LAN
 ip address 192.168.3.1 255.255.255.0
!
interface Serial0/0/1
 description R3 --> R2
 ip address 10.2.2.2 255.255.255.252
!
ip route 0.0.0.0 0.0.0.0 10.2.2.1
!
radius server RADIUS-1
 address ipv4 192.168.1.101 auth-port 1645 acct-port 1646
 key 7 01212720723E354270015E084C5000421908
!
radius server RADIUS-2
 address ipv4 192.168.1.102 auth-port 1645 acct-port 1646
 key 7 003632222D6E384B5D6C5C4F5C4C1247000F
!
banner motd ^CUnauthorized access strictly prohibited!^C
!
line con 0
 exec-timeout 5 0
 password 7 104D000A0618110402142B3837
 logging synchronous
!
line aux 0
 no exec
!
line vty 0 4
 exec-timeout 5 0
 password 7 070C285F4D060F110E020A1F17
 login authentication TELNET-LOGIN
 transport input ssh
!
end
```

Practical 5

Configure and Verify Path Control using PBR.

Topology



Objectives

- Configure and verify policy-based routing.
- Select the required tools and commands to configure policy-based routing operations.
- Verify the configuration and operation by using the proper **show** and **debug** commands.

Background

You want to experiment with policy-based routing (PBR) to see how it is implemented and to study how it could be of value to your organization. To this end, you have interconnected and configured a test network with four routers. All routers are exchanging routing information using EIGRP.

Note: This lab uses Cisco 1841 routers with Cisco IOS Release 12.4(24)T1, and the Advanced IP Services image c1841-advipservicesk9-mz.124-24.T1.bin. You can use other routers (such as 2801 or 2811) and Cisco IOS Software versions if they have comparable capabilities and features. Depending on the router and software version, the commands available and output produced might vary from what is shown in this lab.

Required Resources

- 4 routers (Cisco 1841 with Cisco IOS Release 12.4(24)T1 Advanced IP Services or comparable)
- Serial and console cables

Step 1: Prepare the routers for the lab.

Cable the network as shown in the topology diagram. Erase the startup configuration, and reload each router to clear previous configurations.

Step 2: Configure router hostname and interface addresses.

a. Using the addressing scheme in the diagram, create the loopback interfaces and apply IP addresses to these and the serial interfaces on R1, R2, R3, and R4. On the serial interfaces connecting R1 to R3 and R3 to R4, specify the bandwidth as 64 Kb/s and set a clock rate on the DCE using the **clock rate 64000** command. On the serial interfaces connecting R1 to R2 and R2 to R3, specify the bandwidth as 128 Kb/s and set a clock rate on the DCE using the **clock rate 128000** command.

You can copy and paste the following configurations into your routers to begin.

Note: Depending on the router model, interfaces might be numbered differently than those listed. You might need to alter them accordingly.

Router R1

```
hostname R1
!
interface Lo1
 description R1 LAN
 ip address 192.168.1.1 255.255.255.0
!
interface Serial0/0/0
 description R1 --> R2
 ip address 172.16.12.1 255.255.255.248
 clock rate 128000
 bandwidth 128
 no shutdown
!
interface Serial0/0/1
 description R1 --> R3
 ip address 172.16.13.1 255.255.255.248
 bandwidth 64
 no shutdown
!
end
```

Router R2

```
hostname R2
!
interface Lo2
 description R2 LAN
 ip address 192.168.2.1 255.255.255.0
!
interface Serial0/0/0
 description R2 --> R1
 ip address 172.16.12.2 255.255.255.248
 bandwidth 128
 no shutdown

interface Serial0/0/1
 description R2 --> R3
 ip address 172.16.23.2 255.255.255.248
 clock rate 128000
```

```
 bandwidth 128
 no shutdown
!
end
```

Router R3

```
hostname R3
!
interface Lo3
 description R3 LAN
 ip address 192.168.3.1 255.255.255.0
!
interface Serial0/0/0
 description R3 --> R1
 ip address 172.16.13.3 255.255.255.248
 clock rate 64000
 bandwidth 64
 no shutdown
!
interface Serial0/0/1
 description R3 --> R2
 ip address 172.16.23.3 255.255.255.248
 bandwidth 128
 no shutdown
!
interface Serial0/1/0
 description R3 --> R4
 ip address 172.16.34.3 255.255.255.248
 clock rate 64000
 bandwidth 64
 no shutdown
!
end
```

Router R4

```
hostname R4
!
interface Lo4
 description R4 LAN A
 ip address 192.168.4.1 255.255.255.128
!
interface Lo5
 description R4 LAN B
 ip address 192.168.4.129 255.255.255.128
!
interface Serial0/0/0
 description R4 --> R3
 ip address 172.16.34.4 255.255.255.248
 bandwidth 64
 no shutdown
!
end
```

b. Verify the configuration with the **show ip interface brief**, **show protocols,** and **show interfaces description** commands. The output from router R3 is shown here as an example.

```
R3# show ip interface brief
Interface              IP-Address      OK? Method Status                Protocol
FastEthernet0/0        unassigned      YES manual administratively down down

FastEthernet0/1   unassigned   YES   unset   administratively   down   down

Serial0/0/0       172.16.13.3  YES manual up                    up

Serial0/0/1         172.16.23.3    YES manual up                    up

Serial0/1/0       172.16.34.3   YES manual up                    up

Serial0/1/1         unassigned      YES unset  administratively down  down

Loopback3           192.168.3.1     YES manual up                    up

R3# show protocols
Global values:
  Internet Protocol routing is enabled
FastEthernet0/0 is administratively down, line protocol is down
FastEthernet0/1 is administratively down, line protocol is down
Serial0/0/0 is up, line protocol is up
  Internet address is 172.16.13.3/29
Serial0/0/1 is up, line protocol is up
  Internet address is 172.16.23.3/29
Serial0/1/0 is up, line protocol is up
  Internet address is 172.16.34.3/29
Serial0/1/1 is administratively down, line protocol is down
Loopback3 is up, line protocol is up
  Internet address is 192.168.3.1/24

R3# show interfaces description
Interface                      Status         Protocol Description
Fa0/0                          admin down     down
Fa0/1                          admin down     down
Se0/0/0                        up             up       R3 --> R1
Se0/0/1                        up             up       R3 --> R2
Se0/1/0                        up             up       R3 --> R4
Se0/1/1                        admin down     down
Lo3                            up             up       R3 LAN
```

Step 3: Configure basic EIGRP.

a. Implement EIGRP AS 1 over the serial and loopback interfaces as you have configured it for the other EIGRP labs.

b. Advertise networks 172.16.12.0/29, 172.16.13.0/29, 172.16.23.0/29, 172.16.34.0/29, 192.168.1.0/24, 192.168.2.0/24, 192.168.3.0/24, and 192.168.4.0/24 from their respective routers.

You can copy and paste the following configurations into your routers.

Router R1

```
router eigrp 1
 network 192.168.1.0
 network 172.16.12.0 0.0.0.7
 network 172.16.13.0 0.0.0.7
 no auto-summary
```

Router R2

```
router eigrp 1
 network 192.168.2.0
 network 172.16.12.0 0.0.0.7
 network 172.16.23.0 0.0.0.7
 no auto-summary
```

Router R3

```
router eigrp 1
 network 192.168.3.0
 network 172.16.13.0 0.0.0.7
 network 172.16.23.0 0.0.0.7
 network 172.16.34.0 0.0.0.7
 no auto-summary
```

Router R4

```
router eigrp 1
 network 192.168.4.0
 network 172.16.34.0 0.0.0.7
 no auto-summary
```

You should see EIGRP neighbor relationship messages being generated.

Step 4: Verify EIGRP connectivity.

a. Verify the configuration by using the **show ip eigrp neighbors** command to check which routers have EIGRP adjacencies.

```
R1# show ip eigrp neighbors
IP-EIGRP neighbors for process 1
H   Address                 Interface       Hold Uptime    SRTT   RTO  Q  Seq
                                            (sec)          (ms)       Cnt Num
1   172.16.13.3             Se0/0/1          12 00:00:58   127  2280  0  16
0   172.16.12.2             Se0/0/0          13 00:01:20     8  1140  0  17

R2# show ip eigrp neighbors
IP-EIGRP neighbors for process 1
H   Address                 Interface       Hold Uptime    SRTT   RTO  Q  Seq
                                            (sec)          (ms)       Cnt Num
1   172.16.23.3             Se0/0/1          10 00:01:30   15  1140  0  17
0   172.16.12.1             Se0/0/0          11 00:01:43   14  1140  0  180

R3# show ip eigrp neighbors
IP-EIGRP neighbors for process 1
H   Address                 Interface       Hold Uptime    SRTT   RTO  Q  Seq
                                            (sec)          (ms)       Cnt Num
2   172.16.34.4             Se0/1/0          10 00:02:51   27  2280  0  3
0   172.16.13.1             Se0/0/0          12 00:03:08   45  2280  0  19
1   172.16.23.2             Se0/0/1          12 00:03:13   12  1140  0  16

R4# show ip eigrp neighbors
IP-EIGRP neighbors for process 1
H   Address                 Interface       Hold Uptime    SRTT   RTO  Q  Seq
                                            (sec)          (ms)       Cnt Num
0   172.16.34.3             Se0/0/0          13 00:03:33   40  2280  0  15
```

Did you receive the output you expected?

b. Run the following Tcl script on all routers to verify full connectivity.

```
R1# tclsh

foreach address {
172.16.12.1
172.16.12.2
172.16.13.1
172.16.13.3
172.16.23.2
172.16.23.3
172.16.34.3
172.16.34.4
192.168.1.1
192.168.2.1
192.168.3.1
192.168.4.1
192.168.4.129
} { ping $address }
```

You should get ICMP echo replies for every address pinged. Make sure to run the Tcl script on each router.

Step 5: Verify the current path.

Before you configure PBR, verify the routing table on R1.

a. On R1, use the **show ip route** command. Notice the next-hop IP address for all networks discovered by EIGRP.

```
R1# show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static
route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     172.16.0.0/29 is subnetted, 4 subnets
D       172.16.34.0 [90/41024000] via 172.16.13.3, 00:05:18, Serial0/0/1
D       172.16.23.0 [90/21024000] via 172.16.12.2, 00:05:18, Serial0/0/0
C       172.16.12.0 is directly connected, Serial0/0/0
C       172.16.13.0 is directly connected, Serial0/0/1
     192.168.4.0/25 is subnetted, 2 subnets
D       192.168.4.0 [90/41152000] via 172.16.13.3, 00:05:06, Serial0/0/1
D       192.168.4.128 [90/41152000] via 172.16.13.3, 00:05:06, Serial0/0/1
C    192.168.1.0/24 is directly connected, Loopback1
D    192.168.2.0/24 [90/20640000] via 172.16.12.2, 00:05:18, Serial0/0/0
D    192.168.3.0/24 [90/21152000] via 172.16.12.2, 00:05:18, Serial0/0/0
```

b. On R4, use the **traceroute** command to the R1 LAN address and source the ICMP packet from R4 LAN A and LAN B.

Note: You can specify the source as the interface address (for example 192.168.4.1) or the interface designator (for example, Fa0/0).

```
R4# traceroute 192.168.1.1 source 192.168.4.1

Type escape sequence to abort.
Tracing the route to 192.168.1.1

  1 172.16.34.3 12 msec 12 msec 16 msec
  2 172.16.23.2 20 msec 20 msec 20 msec
  3 172.16.12.1 28 msec 24 msec *

R4# traceroute 192.168.1.1 source 192.168.4.129

Type escape sequence to abort.
Tracing the route to 192.168.1.1

  1 172.16.34.3 12 msec 12 msec 16 msec
  2 172.16.23.2 20 msec 20 msec 20 msec
  3 172.16.12.1 28 msec 24 msec *
```

Notice that the path taken for the packets sourced from the R4 LANs are going through R3 --> R2 --> R1.

Why are the R4 interfaces not using the R3 --> R1 path?

c. On R3, use the **show ip route** command and note that the preferred route from R3 to R1 LAN 192.168.1.0/24 is via R2 using the R3 exit interface S0/0/1.

```
R3# show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static
route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     172.16.0.0/29 is subnetted, 4 subnets
C       172.16.34.0 is directly connected, Serial0/1/0
C       172.16.23.0 is directly connected, Serial0/0/1
D       172.16.12.0 [90/21024000] via 172.16.23.2, 00:15:07, Serial0/0/1
C       172.16.13.0 is directly connected, Serial0/0/0
     192.168.4.0/25 is subnetted, 2 subnets
D       192.168.4.0 [90/40640000] via 172.16.34.4, 00:14:55, Serial0/1/0
D       192.168.4.128 [90/40640000] via 172.16.34.4, 00:14:55, Serial0/1/0
D    192.168.1.0/24 [90/21152000] via 172.16.23.2, 00:15:07, Serial0/0/1
D    192.168.2.0/24 [90/20640000] via 172.16.23.2, 00:15:07, Serial0/0/1
C    192.168.3.0/24 is directly connected, Loopback3
```

d. On R3, use the **show interfaces serial 0/0/0** and **show interfaces s0/0/1** commands.

```
R3# show interfaces s0/0/0
Serial0/0/0 is up, line protocol is up
  Hardware is GT96K Serial
  Description: R3 --> R1
  Internet address is 172.16.13.3/29
  MTU 1500 bytes, BW 64 Kbit/sec, DLY 20000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation HDLC, loopback not set
  Keepalive set (10 sec)
  CRC checking enabled
  Last input 00:00:00, output 00:00:00, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: weighted fair
  Output queue: 0/1000/64/0 (size/max total/threshold/drops)
     Conversations 0/1/256 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 48 kilobits/sec
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     771 packets input, 53728 bytes, 0 no buffer
     Received 489 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     768 packets output, 54404 bytes, 0 underruns
     0 output errors, 0 collisions, 6 interface resets
     0 unknown protocol drops
     0 output buffer failures, 0 output buffers swapped out
     1 carrier transitions
     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up

R3# show interfaces s0/0/1
Serial0/0/1 is up, line protocol is up
  Hardware is GT96K Serial
  Description: R3 --> R2
  Internet address is 172.16.23.3/29
  MTU 1500 bytes, BW 128 Kbit/sec, DLY 20000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation HDLC, loopback not set
  Keepalive set (10 sec)
  CRC checking enabled
  Last input 00:00:00, output 00:00:01, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: weighted fair
  Output queue: 0/1000/64/0 (size/max total/threshold/drops)
     Conversations 0/1/256 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 1158 kilobits/sec
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     894 packets input, 65653 bytes, 0 no buffer
     Received 488 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     895 packets output, 66785 bytes, 0 underruns
     0 output errors, 0 collisions, 6 interface resets
     0 unknown protocol drops
```

```
     0 output buffer failures, 0 output buffers swapped out
     1 carrier transitions
     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
```

Notice that the bandwidth of the serial link between R3 and R1 (S0/0/0) is set to 64 Kb/s, while the bandwidth of the serial link between R3 and R2 (S0/0/1) is set to 128 Kb/s.

e. Confirm that R3 has a valid route to reach R1 from its serial 0/0/0 interface using the **show ip eigrp topology 192.168.1.0** command.

```
R3# show ip eigrp topology 192.168.1.0
IP-EIGRP (AS 1): Topology entry for 192.168.1.0/24
  State is Passive, Query origin flag is 1, 1 Successor(s), FD is 21152000
  Routing Descriptor Blocks:
  172.16.23.2 (Serial0/0/1), from 172.16.23.2, Send flag is 0x0
      Composite metric is (21152000/20640000), Route is Internal
      Vector metric:
        Minimum bandwidth is 128 Kbit
        Total delay is 45000 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 2
  172.16.13.1 (Serial0/0/0), from 172.16.13.1, Send flag is 0x0
      Composite metric is (40640000/128256), Route is Internal
      Vector metric:
        Minimum bandwidth is 64 Kbit
        Total delay is 25000 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 1
```

As indicated, R4 has two routes to reach 192.168.1.0. However, the metric for the route to R1 (172.16.13.1) is much higher (40640000) than the metric of the route to R2 (21152000), making the route through R2 the successor route.

Step 6: Configure PBR to provide path control.

Now you will deploy source-based IP routing by using PBR. You will change a default IP routing decision based on the EIGRP-acquired routing information for selected IP source-to-destination flows and apply a different next-hop router.

Recall that routers normally forward packets to destination addresses based on information in their routing table. By using PBR, you can implement policies that selectively cause packets to take different paths based on source address, protocol type, or application type. Therefore, PBR overrides the router's normal routing behavior.

Configuring PBR involves configuring a route map with **match** and **set** commands and then applying the route map to the interface.

The steps required to implement path control include the following:

- Choose the path control tool to use. Path control tools manipulate or bypass the IP routing table. For PBR, **route-map** commands are used.
- Implement the traffic-matching configuration, specifying which traffic will be manipulated. The **match** commands are used within route maps.
- Define the action for the matched traffic using **set** commands within route maps.

- Apply the route map to incoming traffic.

As a test, you will configure the following policy on router R3:

- All traffic sourced from R4 LAN A must take the R3 --> R2 --> R1 path.
- All traffic sourced from R4 LAN B must take the R3 --> R1 path.

a. On router R3, create a standard access list called **PBR-ACL** to identify the R4 LAN B network.

```
R3(config)# ip access-list standard PBR-ACL
R3(config-std-nacl)# remark ACL matches R4 LAN B traffic
R3(config-std-nacl)# permit 192.168.4.128 0.0.0.127
R3(config-std-nacl)# exit
```

b. Create a route map called **R3-to-R1** that matches PBR-ACL and sets the next-hop interface to the R1 serial 0/0/1 interface.

```
R3(config)# route-map R3-to-R1 permit
R3(config-route-map)# match ip address PBR-ACL
R3(config-route-map)# set ip next-hop 172.16.13.1
R3(config-route-map)# exit
```

c. Apply the R3-to-R1 route map to the serial interface on R3 that receives the traffic from R4. Use the **ip policy route-map** command on interface S0/1/0.

```
R3(config)# interface s0/1/0
R3(config-if)# ip policy route-map R3-to-R1
R3(config-if)# end
```

d. On R3, display the policy and matches using the **show route-map** command.

```
R3# show route-map
route-map R3-to-R1, permit, sequence 10
  Match clauses:
    ip address (access-lists): PBR-ACL
  Set clauses:
    ip next-hop 172.16.13.1
  Policy routing matches: 0 packets, 0 bytes
```

Note: There are currently no matches because no packets matching the ACL have passed through R3 S0/1/0.

Step 7: Test the policy.

Now you are ready to test the policy configured on R3. Enable the **debug ip policy** command on R3 so that you can observe the policy decision-making in action. To help filter the traffic, first create a standard ACL that identifies all traffic from the R4 LANs.

a. On R3, create a standard ACL which identifies all of the R4 LANs.

```
R3# conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)# access-list 1 permit 192.168.4.0 0.0.0.255
R3(config)# exit
```

b. Enable PBR debugging only for traffic that matches the R4 LANs.

```
R3# debug ip policy ?
  <1-199> Access list
  dynamic  dynamic PBR
  <cr>
```

```
R3# debug ip policy 1
Policy routing debugging is on for access list 1
```

c. Test the policy from R4 with the **traceroute** command, using R4 LAN A as the source network.

```
R4# traceroute 192.168.1.1 source 192.168.4.1

Type escape sequence to abort.
Tracing the route to 192.168.1.1

  1 172.16.34.3 0 msec 0 msec 4 msec
  2 172.16.23.2 0 msec 0 msec 4 msec
  3 172.16.12.1 4 msec 0 msec *
```

Notice the path taken for the packet sourced from R4 LAN A is still going through R3 --> R2 --> R1.

As the traceroute was being executed, router R3 should be generating the following debug output.

```
R3#
*Feb 23 06:59:20.931: IP: s=192.168.4.1 (Serial0/1/0), d=192.168.1.1, len
28, policy rejected -- normal forwarding
*Feb 23 06:59:29.935: IP: s=192.168.4.1 (Serial0/1/0), d=192.168.1.1, len
28, policy rejected -- normal forwarding
*Feb 23 06:59:29.939: IP: s=192.168.4.1 (Serial0/1/0), d=192.168.1.1, len
28, policy rejected -- normal forwarding
*Feb 23 06:59:29.939: IP: s=192.168.4.1 (Serial0/1/0), d=192.168.1.1, len
28, FIB policy rejected(no match) - normal forwarding
*Feb 23 06:59:38.943: IP: s=192.168.4.1 (Serial0/1/0), d=192.168.1.1, len
28, FIB policy rejected(no match) - normal forwarding
*Feb 23 06:59:38.947: IP: s=192.168.4.1 (Serial0/1/0), d=192.168.1.1, len
28, FIB policy rejected(no match) - normal forwarding
*Feb 23 06:59:38.947: IP: s=192.168.4.1 (Serial0/1/0), d=192.168.1.1, len
28, FIB policy rejected(no match) - normal forwarding
*Feb 23 06:59:47.951: IP: s=192.168.4.1 (Serial0/1/0), d=192.168.1.1, len
28, FIB policy rejected(no match) - normal forwarding
*Feb 23 06:59:47.955: IP: s=192.168.4.1 (Serial0/1/0), d=192.168.1.1, len
28, FIB policy rejected(no match) - normal forwarding
```

Why is the traceroute traffic not using the R3 --> R1 path as specified in the R3-to-R1 policy?

d. Test the policy from R4 with the **traceroute** command, using R4 LAN B as the source network.

```
R4# traceroute 192.168.1.1 source 192.168.4.129

Type escape sequence to abort.
Tracing the route to 192.168.1.1

  1 172.16.34.3 12 msec 12 msec 16 msec
  2 172.16.13.1 28 msec 28 msec *
```

Now the path taken for the packet sourced from R4 LAN B is R3 --> R1, as expected.

The debug output on R3 also confirms that the traffic meets the criteria of the R3-to-R1 policy.

```
R3#
*Feb 23 07:07:46.467: IP: s=192.168.4.129 (Serial0/1/0), d=192.168.1.1, le
n 28, policy match
*Feb 23 07:07:46.467: IP: route map R3-to-R1, item 10, permit
```

```
*Feb 23 07:07:46.467: IP: s=192.168.4.129 (Serial0/1/0), d=192.168.1.1 (Se
rial0/0/0), len 28, policy routed
*Feb 23 07:07:46.467: IP: Serial0/1/0 to Serial0/0/0 172.16.13.1
*Feb 23 07:07:55.471: IP: s=192.168.4.129 (Serial0/1/0), d=192.168.1.1, le
n 28, policy match
*Feb 23 07:07:55.471: IP: route map R3-to-R1, item 10, permit
*Feb 23 07:07:55.471: IP: s=192.168.4.129 (Serial0/1/0), d=192.168.1.1 (Se
rial0/0/0), len 28, policy routed
*Feb 23 07:07:55.471: IP: Serial0/1/0 to Serial0/0/0 172.16.13.1
*Feb 23 07:07:55.471: IP: s=192.168.4.129 (Serial0/1/0), d=192.168.1.1, le
n 28, policy match
*Feb 23 07:07:55.471: IP: route map R3-to-R1, item 10, permit
*Feb 23 07:07:55.475: IP: s=192.168.4.129 (Serial0/1/0), d=192.168.1.1 (Se
rial0/0/0), len 28, policy routed
*Feb 23 07:07:55.475: IP: Serial0/1/0 to Serial0/0/0 172.16.13.1
*Feb 23 07:07:55.475: IP: s=192.168.4.129 (Serial0/1/0), d=192.168.1.1, le
n 28, FIB policy match
*Feb 23 07:07:55.475: IP: s=192.168.4.129 (Serial0/1/0), d=192.168.1.1, g=
172.16.13.1, len 28, FIB policy routed
*Feb 23 07:08:04.483: IP: s=192.168.4.129 (Serial0/1/0), d=192.168.1.1, le
n 28, FIB policy match
*Feb 23 07:08:04.483: IP: s=192.168.4.129 (Serial0/1/0), d=192.168.1.1, g=
172.16.13.1, len 28, FIB policy routed
*Feb 23 07:08:04.491: IP: s=192.168.4.129 (Serial0/1/0), d=192.168.1.1, le
n 28, FIB policy match
*Feb 23 07:08:04.491: IP: s=192.168.4.129 (Serial0/1/0), d=192.168.1.1, g=
172.16.13.1, len 28, FIB policy routed
```

e. On R3, display the policy and matches using the **show route-map** command.

```
R3# show route-map
route-map R3-to-R1, permit, sequence 10
  Match clauses:
    ip address (access-lists): PBR-ACL
  Set clauses:
    ip next-hop 172.16.13.1
  Policy routing matches: 12 packets, 384 bytes
```

Note: There are now matches to the policy because packets matching the ACL have passed through R3 S0/1/0.

Router Interface Summary Table

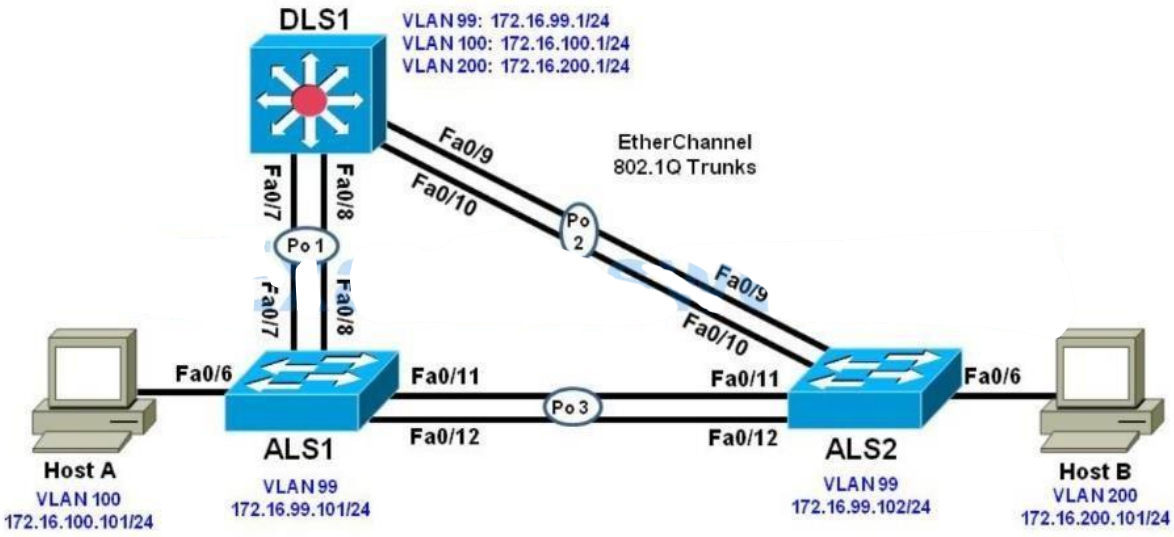
Router Interface Summary				
Router Model	Ethernet Interface #1	Ethernet Interface #2	Serial Interface #1	Serial Interface #2
1700	Fast Ethernet 0 (FA0)	Fast Ethernet 1 (FA1)	Serial 0 (S0)	Serial 1 (S1)
1800	Fast Ethernet 0/0 (FA0/0)	Fast Ethernet 0/1 (FA0/1)	Serial 0/0/0 (S0/0/0)	Serial 0/0/1 (S0/0/1)
2600	Fast Ethernet 0/0 (FA0/0)	Fast Ethernet 0/1 (FA0/1)	Serial 0/0 (S0/0)	Serial 0/1 (S0/1)
2800	Fast Ethernet 0/0 (FA0/0)	Fast Ethernet 0/1 (FA0/1)	Serial 0/0/0 (S0/0/0)	Serial 0/0/1 (S0/0/1)

Note: To find out how the router is configured, look at the interfaces to identify the type of router and how many interfaces the router has. Rather than list all combinations of configurations for each router class, this table includes identifiers for the possible combinations of Ethernet and serial interfaces in the device. The table does not include any other type of interface, even though a specific router might contain one. For example, for an ISDN BRI interface, the string in parenthesis is the legal abbreviation that can be used in Cisco IOS commands to represent the interface.

Practical 6

IP Service LEVEL Agrements and Remote SPAN in a Campus Environment

Topology



Objectives

1. Configure trunking, VTP, and SVIs.
2. Implement IP SLAs to monitor various network performance characteristics.
3. Implement Remote SPAN

Background

Cisco IOS IP service level agreements (SLAs) allow users to monitor network performance between Cisco devices (switches or routers) or from a Cisco device to a remote IP device. Cisco IOS IP SLAs can be applied to VoIP and video applications as well as monitoring end-to-end IP network performance.

Part 1: Prepare for the Lab

Step 1: Prepare the switches for the lab

Use the **reset.tcl** script you created in Lab 1 "Preparing the Switch" to set your switches up for this lab. Then load the file BASE.CFG into the running-config with the command **copy flash:BASE.CFG running-config**. An example from DLS1:

```
DLS1# tclsh reset.tcl
Erasing the nvram filesystem will remove all configuration files! Continue? [c
[OK]
Erase of nvram: complete
Reloading the switch in 1 minute, type reload cancel to halt

Proceed with reload? [confirm]

*Mar 7 18:41:40.403: %SYS-7-NV_BLOCK_INIT: Initialized the geometry of nvram
*Mar 7 18:41:41.141: %SYS-5-RELOAD: Reload requested by console. Reload Reason
Reload command.
<switch reloads - output omitted>

Would you like to enter the initial configuration dialog? [yes/no]: n
Switch> en
*Mar 1 00:01:30.915: %LINK-5-CHANGED: Interface Vlan1, changed state to
administratively down
Switch# copy BASE.CFG running-config
Destination filename [running-config]?
184 bytes copied in 0.310 secs (594 bytes/sec)
DLS1#
```

Step 2: Configure basic switch parameters.

Configure an IP address on the management VLAN according to the diagram. VLAN 1 is the default management VLAN, but following best practice, we will use a different VLAN. In this case, VLAN 99.

Enter basic configuration commands on each switch according to the diagram.DLS1 example:

```
DLS1# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
DLS1(config)# interface vlan 99
DLS1(config-if)# ip address 172.16.99.1 255.255.255.0
DLS1(config-if)# no shutdown
```

The interface VLAN 99 will not come up immediately, because the broadcast domain it is associated with (VLAN 99) doesn't exist on the switch. We will fix that in a few moments.

(Optional) On each switch, create an enable secret password and configure the VTY lines to allow remote access from other network devices.

DLS1 example:

```
DLS1(config)# enable secret class
DLS1(config)# line vty 0 15
DLS1(config-line)# password cisco
DLS1(config-line)# login
```

Note: The passwords configured here are required for NETLAB compatibility only and are NOT recommended for use in a live environment.

Note(2): For purely lab environment purposes, it is possible to configure the VTY lines so that they accept any Telnet connection immediately, without asking for a password, and place the user into the privileged EXEC mode directly. The configuration would be similar to the following example for DLS1:

```
DLS1(config)# enable secret class
DLS1(config)# line vty 0 15
DLS1(config-line)# no login
DLS1(config-line)# privilege level 15
```

Note: The %PKI-6-AUTOSAVE message tells you that your BASE.CFG has been saved as the startup-config, so a simple reload will revert the switch back to BASE configuration

a. Configure default gateways on ALS1 and ALS2. These are access layer switches operating as Layer 2 devices and need a default gateway to send traffic from their management interface to other networks. Configure both ALS1 and ALS2. An example from ALS1 is shown:

```
ALS1(config)# ip default-gateway 172.16.99.1
```

Step 3: Configure host PCs.

Configure PCs Host A and Host B with the IP address and subnet mask shown in the topology. Host A is in VLAN 100 with a default gateway of 172.16.100.1. Host B is in VLAN 200 with a default gateway of 172.16.200.1.

Step 4: Configure trunks and EtherChannels between switches.

Configure trunking according to the diagram. LACP is used for EtherChannel negotiation for these trunks. Examples from DLS1 and ALS1 are shown. Configure all the switches with the channel groups shown in the topology:

Configure the trunks and EtherChannel from DLS1 to ALS1 and ALS2.

```
DLS1(config)# vlan 666
DLS1(config-vlan)# name NATIVE_DO_NOT_USE
DLS1(config-vlan)# exit
DLS1(config)# int ran f0/7-10
DLS1(config-if-range)# switchport trunk encapsulation dot1q
DLS1(config-if-range)# switchport trunk native vlan 666
DLS1(config-if-range)# switchport nonegotiate
DLS1(config-if-range)# switchport mode trunk
DLS1(config-if-range)# exit
DLS1(config)# int ran f0/7-8
DLS1(config-if-range)# channel-group 1 mode active
DLS1(config-if-range)# description EtherChannel to ALS1
DLS1(config-if-range)# no shut
DLS1(config-if-range)# exit
DLS1(config)# int ran f0/9-10
DLS1(config-if-range)# channel-group 2 mode active
DLS1(config-if-range)# description EtherChannel to ALS2
DLS1(config-if-range)# no shut
DLS1(config-if-range)# exit
```

Configure the trunks and EtherChannel between ALS1 and ALS2.

```
ALS1(config)# interface range fastEthernet 0/11 - 12
ALS1(config-if-range)# switchport mode trunk
ALS1(config-if-range)# channel-group 3 mode active
ALS1(config-if-range)# no shut
```

Step 5: Configure VTP on ALS1 and ALS2.

Change the VTP mode of ALS1 and ALS2 to client.

```
ALS1(config)# vtp mode client
Setting device to VTP CLIENT mode.

ALS2(config)# vtp mode client
Setting device to VTP CLIENT mode.
```

Step 6: Configure VTP on DLS1.

Create the VTP domain on DLS1, and create VLANs 100 and 200 for the domain.

```
DLS1(config)# vtp domain SWPOD
DLS1(config)# vtp version 2

DLS1(config)# vlan 99
DLS1(config-vlan)# name Management
DLS1(config-vlan)# vlan 100
DLS1(config-vlan)# name Finance
DLS1(config-vlan)# vlan 200
DLS1(config-vlan)# name Engineering
DLS1(config-vlan)# exit
DLS1(config)#
```

Step 7: Configure access ports.

Configure the host ports for the appropriate VLANs according to the diagram.

```
ALS1(config)# interface fastEthernet 0/6
ALS1(config-if)# switchport mode access
ALS1(config-if)# switchport access vlan 100
ALS1(config-if)# no shut

ALS2(config)# interface fastEthernet 0/6
ALS2(config-if)# switchport mode access
ALS2(config-if)# switchport access vlan 200
ALS1(config-if)# no shut
```

Step 8: Configure VLAN interfaces and enable routing.

On DLS1, create the SVIs for VLANs 100 and 200. Note that the corresponding Layer 2 VLANs must be configured for the Layer 3 SVIs to activate. This was done in Step 6.

```
DLS1(config)# interface vlan 100
DLS1(config-if)# ip address 172.16.100.1 255.255.255.0
DLS1(config-if)# interface vlan 200
DLS1(config-if)# ip address 172.16.200.1 255.255.255.0
```

The **ip routing** command is also needed to allow the DLS1 switch to act as a Layer 3 device to route between these VLANs. Because the VLANs are all considered directly connected, a routing protocol is not needed at this time. The default configuration on 3560 switches is **no ip routing**.

```
DLS1(config)# ip routing
```

Verify the configuration using the **show ip route** command on DLS1.

```
DLS1# show ip route | begin Gateway
Gateway of last resort is not set

      172.16.0.0/16 is variably subnetted, 6 subnets, 2 masks
C        172.16.99.0/24 is directly connected, Vlan99
L        172.16.99.1/32 is directly connected, Vlan99
C        172.16.100.0/24 is directly connected, Vlan100
L        172.16.100.1/32 is directly connected, Vlan100
C        172.16.200.0/24 is directly connected, Vlan200
L        172.16.200.1/32 is directly connected, Vlan200
DLS1#
```

Run the following Tcl script on DLS1 to verify full connectivity. If these pings are not successful, troubleshoot.

```
DLS1# tclsh
foreach address {
172.16.99.1
172.16.99.101
172.16.99.102
172.16.100.1
172.16.200.1
172.16.100.101
172.16.200.101
} {
ping $address }
```

Part 2: Configure Cisco IOS IP SLA

Step 1: Configure Cisco IOS IP SLA responders.

IP SLA responders are Cisco IOS devices that support the IP SLA control protocol. An IP SLA responder uses the Cisco IOS IP SLA Control Protocol for notification configuration and on which port to listen and respond. Some operations, such as ICMP echo, do not require a dedicated IP SLA responder.

Use the **ip sla responder** command on ALS1 and ALS2 to enable sending and receiving IP SLAs control packets.

Note: This command replaces the ip sla monitor responder command. All commands that used to begin with “ip sla monitor” now begin with “ip sla” (without “monitor”). Configure this on both ALS1 and ALS2. An example from ALS1:

```
ALS1(config)# ip sla responder
```

Configure ALS1 and ALS2 as IP SLA responders for UDP jitter using the **ip sla responder udp-echo ipaddress** command. Specify the IP address of DLS1 VLAN 1 to act as the destination IP address for the reflected UDP traffic on both ALS1 and ALS2. Configure this on both ALS1 and ALS2. An example from ALS1:

```
ALS1(config)# ip sla responder udp-echo ipaddress 172.16.99.1 port 5000
```

Step 2: Configure the Cisco IOS IP SLA source to measure network performance.

IP SLA uses generated traffic to measure network performance between two networking devices.

On DLS1, create an IP SLA operation and enter IP SLA configuration mode with the **ip sla operation-number** command.

```
DLS1(config)# ip sla 1
DLS1(config-ip-sla)#
```

Configure an IP SLA ICMP echo operation using the icmp-echo command in IP SLA configuration mode. The IP SLA ICMP echo operation does not require a dedicated Cisco IOS IP SLA responder (the destination device can be a non-Cisco device, such as a PC). By default, the ICMP operation repeats every 60 seconds. On DLS1, for ICMP echo operation 1, specify the IP address of Host A as the target. For ICMP echo operation 2, specify the IP address of Host B as the target.

```
DLS1(config-ip-sla)# icmp-echo 172.16.100.101
DLS1(config-ip-sla-echo)# exit

DLS1(config)# ip sla 2
DLS1(config-ip-sla)# icmp-echo 172.16.200.101
DLS1(config-ip-sla-echo)# exit
```

Jitter means inter-packet delay variance. UDP-based voice traffic associated with IP phone and PC softphone applications at the access layer require strict adherence to delay and jitter thresholds. To configure an IP SLA UDP jitter operation, use the udp-jitter command in IP SLA configuration mode. By default, the UDP jitter operation repeats every 60 seconds. For UDP jitter operation 3, specify the destination IP address of the ALS1 VLAN 99 interface as the target. For operation 4, specify the destination IP address of the ALS2 VLAN 99 interface as the target. The IP SLA communication port is 5000 for both operations.

```
DLS1(config)# ip sla 3
DLS1(config-ip-sla)# udp-jitter 172.16.99.101 5000
DLS1(config-ip-sla-jitter)# exit
DLS1(config)# ip sla 4
DLS1(config-ip-sla)# udp-jitter 172.16.99.102 5000
DLS1(config-ip-sla-jitter)# exit
```

Schedule the IP SLAs operations to run indefinitely beginning immediately using the ip sla schedule global configuration mode command.

```
DLS1(config)# ip sla schedule 1 life forever start-time now
DLS1(config)# ip sla schedule 2 life forever start-time now
DLS1(config)# ip sla schedule 3 life forever start-time now
DLS1(config)# ip sla schedule 4 life forever start-time now
```

Step 3: Monitor IP SLAs operations.

View the IP SLA configuration for IP SLA 1 on DLS1. The output for IP SLA 2 is similar.

```
DLS1# show ip sla configuration 1
IP SLAs Infrastructure Engine-III
Entry number: 1
Owner:
Tag:
Operation timeout (milliseconds): 5000
Type of operation to perform: icmp-echo
Target address/Source address: 172.16.100.101/0.0.0.0
Type Of Service parameter: 0x0
Request size (ARR data portion): 28
Verify data: No
Vrf Name:
Schedule:
   Operation frequency (seconds): 60 (not considered if randomly scheduled)
   Next Scheduled Start Time: Start Time already passed
   Group Scheduled : FALSE
   Randomly Scheduled : FALSE
   Life (seconds): Forever
   Entry Ageout (seconds): never
   Recurring (Starting Everyday): FALSE
   Status of entry (SNMP RowStatus): Active
Threshold (milliseconds): 5000
Distribution Statistics:
   Number of statistic hours kept: 2
   Number of statistic distribution buckets kept: 1
   Statistic distribution interval (milliseconds): 20
Enhanced History:
History Statistics:
   Number of history Lives kept: 0
   Number of history Buckets kept: 15
   History Filter Type: None
```

What type of operation is being performed with IP SLA 1?

View the IP SLA configuration for IP SLA 3 on DLS1. The output for IP SLA 4 is similar.

```
DLS1# show ip sla configuration 3
IP SLAs Infrastructure Engine-III
Entry number: 2
Owner:
Tag:
Operation timeout (milliseconds): 5000
Type of operation to perform: icmp-echo
Target address/Source address: 172.16.200.101/0.0.0.0
Type Of Service parameter: 0x0
Request size (ARR data portion): 28
Verify data: No
Vrf Name:
Schedule:
   Operation frequency (seconds): 60 (not considered if randomly scheduled)
   Next Scheduled Start Time: Start Time already passed
   Group Scheduled : FALSE
   Randomly Scheduled : FALSE
   Life (seconds): Forever
   Entry Ageout (seconds): never
   Recurring (Starting Everyday): FALSE
   Status of entry (SNMP RowStatus): Active
Threshold (milliseconds): 5000
Distribution Statistics:
   Number of statistic hours kept: 2
   Number of statistic distribution buckets kept: 1
   Statistic distribution interval (milliseconds): 20
Enhanced History:
History Statistics:
   Number of history Lives kept: 0
   Number of history Buckets kept: 15
   History Filter Type: None
```

What type of operation is being performed with IP SLA 3?

Display global information about Cisco IOS IP SLAs on DLS1.

```
DLS1# show ip sla application

IP Service Level Agreements
Version: Round Trip Time MIB 2.2.0, Infrastructure Engine-III

Supported Operation Types:
        icmpEcho, path-echo, path-jitter, udpEcho, tcpConnect, http
        dns, udpJitter, dhcp, ftp, video, udpApp, wspApp

Supported Features:
        IPSLAs Event Publisher

IP SLAs low memory water mark: 9359471
Estimated system max number of entries: 6855

Estimated number of configurable operations: 6817
Number of Entries configured  : 4
Number of active Entries      : 4
Number of pending Entries     : 0
Number of inactive Entries    : 0
Time of last change in whole IP SLAs: 13:54:00.025 CDT Fri Jul 31 2015
```

Display information about Cisco IOS IP SLA responders on ALS1. The ALS2 output is similar.

```
ALS1# show ip sla responder
General IP SLA Responder on Control port 1967
General IP SLA Responder is: Enabled
Number of control message received: 26 Number of errors: 0
Recent sources:
        172.16.99.1 [14:17:28.775 CDT Fri Jul 31 2015]
        172.16.99.1 [14:16:28.780 CDT Fri Jul 31 2015]
        172.16.99.1 [14:15:28.776 CDT Fri Jul 31 2015]
        172.16.99.1 [14:14:28.781 CDT Fri Jul 31 2015]
        172.16.99.1 [14:13:28.777 CDT Fri Jul 31 2015]
Recent error sources:

       Permanent Port IP SLA Responder
Permanent Port IP SLA Responder is: Enabled

udpEcho Responder:
  IP Address Port
  172.16.99.1 5000
```

Display IP SLA statistics on DLS1 for IP SLA 1. The IP SLA 2 output is similar.

```
DLS1# show ip sla statistics 1

IPSLAs Latest Operation Statistics

IPSLA operation id: 1
        Latest RTT: 1 milliseconds
Latest operation start time: 14:17:00 CDT Fri Jul 31 2015
Latest operation return code: OK
Number of successes: 26
Number of failures: 0
Operation time to live: Forever
```

From this output, you can see that the latest round-trip time (RTT) for SLA operation Index 1 (icmp-echo) is 1 millisecond (ms). The number of packets sent successfully from DLS1 to PC Host A was 26, and there were no failures.

Display IP SLA statistics on DLS1 for IP SLA 3. The IP SLA 4 output is similar.

```
DLS1# show ip sla statistics 3

IPSLAs Latest Operation Statistics

IPSLA operation id: 3
Type of operation: udp-jitter
        Latest RTT: 3 milliseconds
Latest operation start time: 14:18:01 CDT Fri Jul 31 2015
Latest operation return code: OK
RTT Values:
        Number Of RTT: 10                   RTT Min/Avg/Max: 3/3/5 millisecond
Latency one-way time:
        Number of Latency one-way Samples: 0
        Source to Destination Latency one way Min/Avg/Max: 0/0/0 milliseconds
        Destination to Source Latency one way Min/Avg/Max: 0/0/0 milliseconds
Jitter Time:
        Number of SD Jitter Samples: 9
        Number of DS Jitter Samples: 9
        Source to Destination Jitter Min/Avg/Max: 0/1/1 milliseconds
        Destination to Source Jitter Min/Avg/Max: 0/1/1 milliseconds
Packet Loss Values:
        Loss Source to Destination: 0
        Source to Destination Loss Periods Number: 0
        Source to Destination Loss Period Length Min/Max: 0/0
        Source to Destination Inter Loss Period Length Min/Max: 0/0
        Loss Destination to Source: 0
        Destination to Source Loss Periods Number: 0
        Destination to Source Loss Period Length Min/Max: 0/0
        Destination to Source Inter Loss Period Length Min/Max: 0/0
        Out Of Sequence: 0 Tail Drop: 0
        Packet Late Arrival: 0 Packet Skipped: 0
Voice Score Values:
        Calculated Planning Impairment Factor (ICPIF): 0
        Mean Opinion Score (MOS): 0
Number of successes: 27
Number of failures: 0
Operation time to live: Forever
```

From this output, you can see that the latest RTT for SLA operation Index 3 (udp-jitter) is 3 ms. Jitter time from source to destination and from destination to source is averaging 1 ms, which is acceptable for voice applications. The number of packets sent successfully from DLS1 to ALS1 was 27, and there were no failures.

Disable interface VLAN 99 on ALS1 using the **shutdown** command.

```
ALS1(config)# interface vlan 99
ALS1(config-if)# shutdown
```

Allow a few minutes to pass and then issue the **show ip sla statistics 3** command on DLS1. The output should look similar to the following.

```
DLS1# show ip sla statistics 3

IPSLAs Latest Operation Statistics

IPSLA operation id: 3
Type of operation: udp-jitter
        Latest RTT: NoConnection/Busy/Timeout
Latest operation start time: 14:22:01 CDT Fri Jul 31 2015
Latest operation return code: No connection
RTT Values:
        Number Of RTT: 0                    RTT Min/Avg/Max: 0/0/0 milliseco
Latency one-way time:
        Number of Latency one-way Samples: 0
        Source to Destination Latency one way Min/Avg/Max: 0/0/0 milliseconds
        Destination to Source Latency one way Min/Avg/Max: 0/0/0 milliseconds
Jitter Time:
        Number of SD Jitter Samples: 0
        Number of DS Jitter Samples: 0
        Source to Destination Jitter Min/Avg/Max: 0/0/0 milliseconds
        Destination to Source Jitter Min/Avg/Max: 0/0/0 milliseconds
Packet Loss Values:
        Loss Source to Destination: 0
        Source to Destination Loss Periods Number: 0
        Source to Destination Loss Period Length Min/Max: 0/0
        Source to Destination Inter Loss Period Length Min/Max: 0/0
        Loss Destination to Source: 0
        Destination to Source Loss Periods Number: 0
        Destination to Source Loss Period Length Min/Max: 0/0
        Destination to Source Inter Loss Period Length Min/Max: 0/0
        Out Of Sequence: 0 Tail Drop: 0
        Packet Late Arrival: 0 Packet Skipped: 0
Voice Score Values:
        Calculated Planning Impairment Factor (ICPIF): 0
        Mean Opinion Score (MOS): 0
Number of successes: 29
Number of failures: 2
Operation time to live: Forever
```

If there is a connectivity problem between IP SLA source DLS1 and responder ALS1 or ALS2, the communication to the responder will be lost and statistics will cease to be collected, except for the number of failed tests.

Note: The IP SLA itself is an additional task that must be performed by the switch CPU. A large number of intensive SLAs could create a significant burden on the CPU, possibly interfering with other switch functions and having detrimental impact on the overall device performance. Therefore, you should carefully evaluate the benefits of running IP SLAs. The CPU load should be monitored after the SLAs are deployed to verify that they do not stress the device's CPU above safe limits.

Re-enable ALS1's interface vlan 99 before continuing.

Part 3: Switch Port Analyzer (SPAN) Feature

SPAN is tool that allows for monitoring and troubleshooting a network. There are different variations of the SPAN tool. There is local SPAN, Remote Span, and VLAN span. Local Span allows an administrator to monitor traffic from a source and have it sent to a destination port on the same switch running a protocol analyzer on the same switch. The source and destination port used to create the monitor session must be on the same switch. Remote SPAN allows the source and destination ports to be on different switches. In order for this to work, it uses a VLAN configured only for remote span functionality. The source port then places the transmitted or received data onto the remote span VLAN. The remote span VLAN is carried across trunks. The receiving switch takes the data sourced from the remote VLAN and sends it to the destination port running the protocol analyzer.

In this lab, we will demonstrate the use of remote SPAN (RSPAN). VLAN 300 will be created and used as the remote span VLAN. We will set up a monitoring session for the host connected to port fa0/6 on switch ALS1. Ultimately, the destination port will be the host connected to fa0/6 of ALS2. The ALS2 host is collect the transmit and receive data using Wireshark.

Step 1: Configure Remote SPAN (RSPAN).

Create the RSPAN VLAN on DLS1 using the VLAN 300 command from global configuration mode.

```
DLS1(config)# vlan 300
DLS1(config-vlan)# name REMOTE_SPAN
DLS1(config-vlan)# remote-span
```

Use the show vlan remote-span command to verify the vlan 300 is configured correctly and is designated as the remote-span vlan. Ensure that the VLAN propagates across the VTP Domain

with show vlan brief command. Use the show interface trunk command to ensure the RSPAN VLAN
is allowed on the trunks. The RSPAN VLAN should not be a DATA VLAN. Its purpose is strictly for carrying the monitored traffic across trunk links from one switch to another.

Verify the output on DLS1

```
DLS1# show vlan brief | include active
1    default                          active    Fa0/1, Fa0/2, Fa0/3, Fa0/4
99   Management                       active
100  Finance                          active
200  Engineering                      active
300  REMOTE_SPAN                      active
666  NATIVE_DO_NOT_USE                active
DLS1#
```

Verify the output on ALS1.

```
ALS1# show vlan brief | include active
1    default                          active    Fa0/1, Fa0/2, Fa0/3, Fa0/4
99   Management                       active
100  Finance                          active    Fa0/6
200  Engineering                      active
300  REMOTE_SPAN                      active
666  NATIVE_DO_NOT_USE                active
ALS1#
```

Now configure the monitor session on ALS1 with a source interface of fa0/6 and a destination of remote vlan 300. Because the captured traffic must traverse the local switch to a remote switch, we must use the remote VLAN as the destination.

```
ALS1(config)# monitor session 1 source interface Fa0/6
ALS1(config)# monitor session 1 destination remote vlan 300
```

Verify the configuration using the show monitor command

```
ALS1# show monitor
Session 1
---------
Type                   : Remote Source Session
Source Ports           :
    Both               : Fa0/6
Dest RSPAN VLAN        : 300
```

Move to the ALS2 switch and configure it to collect the desired traffic. The source port on ALS2 will be the remote span vlan 300 and the destination port will be the Engineering client connected to port fa0/6.

It is important to note that the PC-B host should be running a protocol analyzer to view the contents of the captured traffic and perform traffic analysis. Both transmit and receive traffic of the source port will be captured. The configuration can be modified to only capture transmit or receive traffic if necessary.

Configure ALS2 for the remote span session.

```
ALS2(config)# monitor session 10 source remote vlan 300
ALS2(config)# monitor session 10 destination interface Fa0/6
```

Our configuration shows the use of a different session number than the one used on ALS1. The session numbers do not have to match from device to device.

Verify the configuration using the show monitor command. The source port should show VLAN 300 and the destination port should be interface fa0/6.

```
ALS2# show monitor
Session 10
----------
Type                        : Remote Destination Session
Source RSPAN VLAN           : 300
Destination Ports           : Fa0/6
Encapsulation               : Native
Ingress                     : Disabled
```

Use the **show interfaces fa0/6** to command to view the status of the interface. Notice from the output the line protocol is down. When a port is used as a destination in monitoring session, it cannot be used to transmit and receive regular network traffic.

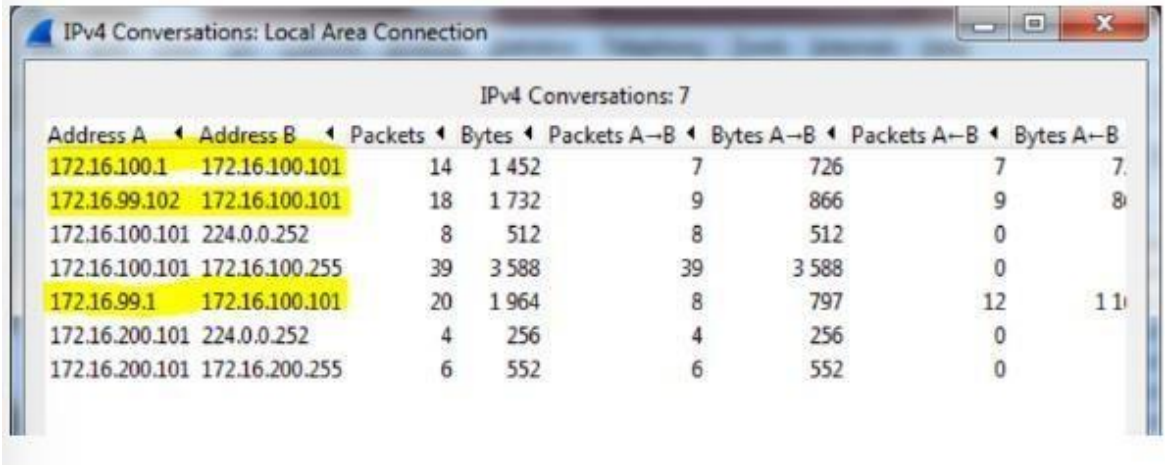
```
ALS2# show interface f0/6
FastEthernet0/6 is up, line protocol is down (monitoring)
   Hardware is Fast Ethernet, address is 5017.ff84.0a86 (bia 5017.ff84.0a86)
<output omitted>
```

Step 2: Test RSPAN operation

On PC-B, turn on Wireshark and capture all interface traffic.In order to test the RSPAN configuration implemented on ALS1 and ALS2, we need to generate traffic from the source host, PC-A.

o Initiate a ping from PC-A to the 172.16.99.102 address
o Open a web browser. Browse to the following url: http://172.16.99.1
o From ALS2, initiate a ping to PC-A, 172.16.100.101.
o From DLS1, initiate a ping to PC-A, 172.16.100.101.

In the Wireshark application that is running on PC-B, select the STOP button then use the Statistics > Converstion List > IPv4 menu to view the IPv4 conversations contained in the capture. You will see that 172.16.200.101 (the address of PC-B) is not involved in any conversations except for traffic to 224.0.0.252 and 172.16.200.255.


IPv4 Conversations: Local Area Connection
IPv4 Conversations: 7
Address A
Address B
Packets
Bytes
Packets A→B
Bytes A→B
Packets A←B
Bytes A←B
172.16.100.1 172.16.100.101 14 1 452 7 726 7 7
172.16.99.102 172.16.100.101 18 1 732 9 866 9 8
172.16.100.101 224.0.0.252 8 512 8 512 0
172.16.100.101 172.16.100.255 39 3 588 39 3 588 0
172.16.99.1 172.16.100.101 20 1 964 8 797 12 1 1
172.16.200.101 224.0.0.252 4 256 4 256 0
172.16.200.101 172.16.200.255 6 552 6 552 0

Device Configurations:

Below are the final configurations for each switch.

DLS1:

```
DLS1# show run brief | exclude !
Building configuration...

Current configuration : 3174 bytes
version 15.0
no service pad
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
hostname DLS1
boot-start-marker
boot-end-marker
enable secret 5 $1$394h$4u5EACpRNVCKAFz1Ntit6/
no aaa new-model
clock timezone CST -6 0
clock summer-time CDT recurring
system mtu routing 1500
ip routing
no ip domain-lookup
ip domain-name CCNP.NET
spanning-tree mode pvst
spanning-tree extend system-id
vlan internal allocation policy ascending
interface Port-channel1
switchport trunk encapsulation dot1q
switchport trunk native vlan 666
switchport mode trunk
interface Port-channel2
switchport trunk encapsulation dot1q
switchport trunk native vlan 666
switchport mode trunk
interface FastEthernet0/1
shutdown
interface FastEthernet0/2
shutdown
interface FastEthernet0/3
shutdown
interface FastEthernet0/4
shutdown
```

```
interface FastEthernet0/5
shutdown
interface FastEthernet0/6
shutdown
interface FastEthernet0/7
description EtherChannel to ALS1
switchport trunk encapsulation dot1q
switchport trunk native vlan 666
switchport mode trunk
channel-group 1 mode active
interface FastEthernet0/8
description EtherChannel to ALS1
switchport trunk encapsulation dot1q
switchport trunk native vlan 666
switchport mode trunk
channel-group 1 mode active
interface FastEthernet0/9
description EtherChannel to ALS2
switchport trunk encapsulation dot1q
switchport trunk native vlan 666
switchport mode trunk
channel-group 2 mode active
interface FastEthernet0/10
description EtherChannel to ALS2
switchport trunk encapsulation dot1q
switchport trunk native vlan 666
switchport mode trunk
channel-group 2 mode active
interface FastEthernet0/11
shutdown
interface FastEthernet0/12
shutdown
interface FastEthernet0/13
shutdown
interface FastEthernet0/14
shutdown
interface FastEthernet0/15
shutdown
interface FastEthernet0/16
shutdown
interface FastEthernet0/17
shutdown
interface FastEthernet0/18
shutdown
interface FastEthernet0/19
shutdown
interface FastEthernet0/20
shutdown
```

```
interface FastEthernet0/21
shutdown
interface FastEthernet0/22
shutdown
interface FastEthernet0/23
shutdown
interface FastEthernet0/24
shutdown
interface GigabitEthernet0/1
shutdown
interface GigabitEthernet0/2
shutdown
interface Vlan1
no ip address
interface Vlan99
ip address 172.16.99.1 255.255.255.0
interface Vlan100
ip address 172.16.100.1 255.255.255.0
interface Vlan200
ip address 172.16.200.1 255.255.255.0
ip http server
ip http secure-server
ip sla 1
icmp-echo 172.16.100.101
ip sla schedule 1 life forever start-time now
ip sla 2
icmp-echo 172.16.200.101
ip sla schedule 2 life forever start-time now
ip sla 3
udp-jitter 172.16.99.101 5000
ip sla schedule 3 life forever start-time now
ip sla 4
udp-jitter 172.16.99.102 5000
ip sla schedule 4 life forever start-time now
line con 0
exec-timeout 0 0
logging synchronous
line vty 0 4
password cisco
login
line vty 5 15
password cisco
login
end
DLS1#
```

ALS1:

```
ALS1# show run brief | exclude !
Building configuration...

Current configuration : 2692 bytes
version 15.0
no service pad
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
hostname ALS1
boot-start-marker
boot-end-marker
enable secret 5 $1$eOw5$elPah.syL1h.EPTDwlpnG0
no aaa new-model
clock timezone CST -6 0
clock summer-time CDT recurring
system mtu routing 1500
no ip domain-lookup
ip domain-name CCNP.NET
spanning-tree mode pvst
spanning-tree extend system-id
vlan internal allocation policy ascending
interface Port-channel1
switchport trunk native vlan 666
switchport mode trunk
interface Port-channel3
switchport trunk native vlan 666
switchport mode trunk
interface FastEthernet0/1
shutdown
interface FastEthernet0/2
shutdown
interface FastEthernet0/3
shutdown
interface FastEthernet0/4
shutdown
interface FastEthernet0/5
shutdown
interface FastEthernet0/6
switchport access vlan 100
switchport mode access
interface FastEthernet0/7
description EtherChannel to DLS1
switchport trunk native vlan 666
switchport mode trunk
channel-group 1 mode active
interface FastEthernet0/8
description EtherChannel to DLS1
```

```
switchport trunk native vlan 666
switchport mode trunk
channel-group 1 mode active
interface FastEthernet0/9
shutdown
interface FastEthernet0/10
shutdown
interface FastEthernet0/11
description EtherChannel to ALS2
switchport trunk native vlan 666
switchport mode trunk
channel-group 3 mode active
interface FastEthernet0/12
description EtherChannel to ALS2
switchport trunk native vlan 666
switchport mode trunk
channel-group 3 mode active
interface FastEthernet0/13
shutdown
interface FastEthernet0/14
shutdown
interface FastEthernet0/15
shutdown
interface FastEthernet0/16
shutdown
interface FastEthernet0/17
shutdown
interface FastEthernet0/18
shutdown
interface FastEthernet0/19
shutdown
interface FastEthernet0/20
shutdown
interface FastEthernet0/21
shutdown
interface FastEthernet0/22
shutdown
interface FastEthernet0/23
shutdown
interface FastEthernet0/24
shutdown
interface GigabitEthernet0/1
shutdown
```

```
interface GigabitEthernet0/2
shutdown
interface Vlan1
no ip address
interface Vlan99
ip address 172.16.99.101 255.255.255.0
ip default-gateway 172.16.99.1
ip http server
ip http secure-server
ip sla responder
ip sla responder udp-echo ipaddress 172.16.99.1 port 5000
line con 0
exec-timeout 0 0
logging synchronous
line vty 0 4
password cisco
login
line vty 5 15
password cisco
login
monitor session 1 source interface Fa0/6
monitor session 1 destination remote vlan 300
end
ALS1#
```

ALS2:

```
ALS2# show run brief | exclude !
Building configuration...

Current configuration : 2694 bytes
version 15.0
no service pad
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
hostname ALS2
boot-start-marker
boot-end-marker
enable secret 5 $1$QTDK$sDfAMuL5FIHWCO5U4q9F50
no aaa new-model
clock timezone CST -6 0
clock summer-time CDT recurring
system mtu routing 1500
no ip domain-lookup
ip domain-name CCNP.NET
spanning-tree mode pvst
spanning-tree extend system-id
vlan internal allocation policy ascending
interface Port-channel2
switchport trunk native vlan 666
switchport mode trunk
interface Port-channel3
switchport trunk native vlan 666
switchport mode trunk
interface FastEthernet0/1
shutdown
interface FastEthernet0/2
shutdown
interface FastEthernet0/3
shutdown
interface FastEthernet0/4
shutdown
interface FastEthernet0/5
shutdown
interface FastEthernet0/6
switchport access vlan 200
switchport mode access
```

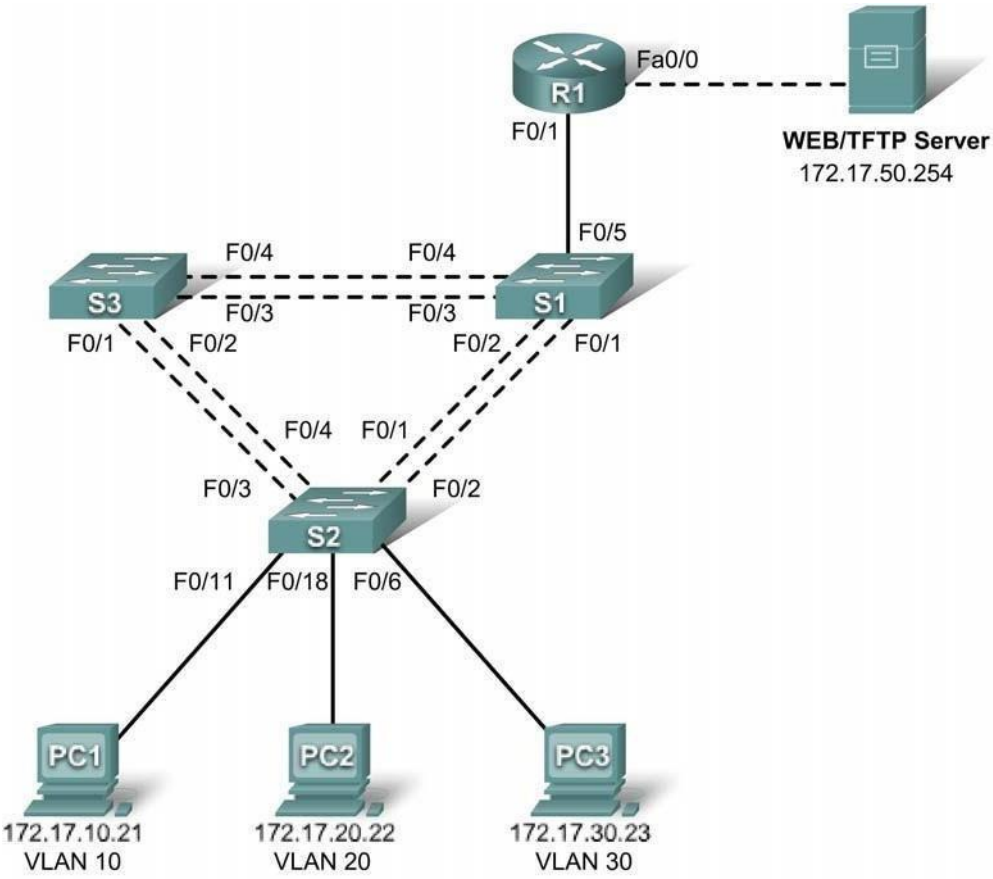
```
interface FastEthernet0/7
shutdown
interface FastEthernet0/8
shutdown
interface FastEthernet0/9
description EtherChannel to DLS1
switchport trunk native vlan 666
switchport mode trunk
channel-group 2 mode active
interface FastEthernet0/10
description EtherChannel to DLS1
switchport trunk native vlan 666
switchport mode trunk
channel-group 2 mode active
interface FastEthernet0/11
description EtherChannel to ALS1
switchport trunk native vlan 666
switchport mode trunk
channel-group 3 mode active
interface FastEthernet0/12
description EtherChannel to ALS1
switchport trunk native vlan 666
switchport mode trunk
channel-group 3 mode active
interface FastEthernet0/13
shutdown
interface FastEthernet0/14
shutdown
interface FastEthernet0/15
shutdown
interface FastEthernet0/16
shutdown
interface FastEthernet0/17
shutdown
interface FastEthernet0/18
shutdown
interface FastEthernet0/19
shutdown
interface FastEthernet0/20
shutdown
```

```
interface FastEthernet0/21
shutdown
interface FastEthernet0/22
shutdown
interface FastEthernet0/23
shutdown
interface FastEthernet0/24
shutdown
interface GigabitEthernet0/1
shutdown
interface GigabitEthernet0/2
shutdown
interface Vlan1
no ip address
interface Vlan99
ip address 172.16.99.102 255.255.255.0
ip default-gateway 172.16.99.1
ip http server
ip http secure-server
ip sla responder
ip sla responder udp-echo ipaddress 172.16.99.1 port 5000
line con 0
exec-timeout 0 0
logging synchronous
line vty 0 4
password cisco
login
line vty 5 15
password cisco
login
monitor session 10 destination interface Fa0/6
monitor session 10 source remote vlan 300
end
ALS2#
```

Practical 7

Inter-VLAN Routing

Topology Diagram



Addressing Table

Device	Interface	IP Address	Subnet Mask	Default Gateway
S1	**VLAN 99**	172.17.99.11	255.255.255.0	172.17.99.1
S2	**VLAN 99**	172.17.99.12	255.255.255.0	172.17.99.1
S3	**VLAN 99**	172.17.99.13	255.255.255.0	172.17.99.1
R1	**Fa0/0**	See Interface Configuration Table		N/A
	Fa0/1	172.17.50.1	255.255.255.0	N/A
PC1	**NIC**	172.17.10.21	255.255.255.0	172.17.10.1
PC2	**NIC**	172.17.20.22	255.255.255.0	172.17.20.1
PC3	**NIC**	172.17.30.23	255.255.255.0	172.17.30.1
Server	**NIC**	172.17.50.254	255.255.255.0	172.17.50.1

Port Assignments – S2

Ports	Assignment	Network
Fa0/1 - 0/5	802.1q Trunks (Native VLAN 99)	172.17.99.0 /24
Fa0/6 - 0/10	VLAN 30 - Guests(Default)	172.17.30.0 /24
Fa0/11 - 0/17	VLAN 10 - Faculty/Staff	172.17.10.0 /24
Fa0/18 - 0/24	VLAN 20 - Students	172.17.20.0 /24

Subinterface Configuration Table – R1

Interface	Assignment	IP Address
Fa0/0.1	VLAN 1	172.17.1.1 /24
Fa0/0.10	VLAN 10	172.17.10.1 /24
Fa0/0.20	VLAN 20	172.17.20.1 /24
Fa0/0.30	VLAN 30	172.17.30.1 /24
Fa0/0.99	VLAN 99	172.17.99.1 /24

Learning Objectives

- Perform basic switch configurations
- Configure the Ethernet interfaces on the host PCs
- Configure VTP on the switches
- Configure the router and the remote server LAN

Introduction

In this activity, you will perform basic switch configurations, configure addressing on PCs, configure VTP and inter-VLAN routing.

Task 1: Perform Basic Switch Configurations

Configure the S1, S2, and S3 switches according to the addressing table and the following guidelines:

- Configure the switch hostname.
- Disable DNS lookup.
- Configure the default gateway.
- Configure an EXEC mode password of **class**.
- Configure a password of **cisco** for console connections.
- Configure a password of **cisco** for vty connections.
- Configure the default gateway on each switch.

```
Switch>enable
Switch#config term
Enter configuration commands, one per line.  End with CNTL/Z.
Switch(config)#hostname S1
S1(config)#enable secret class
S1(config)#no ip domain-lookup
```

```
S1(config)#ip default-gateway 172.17.99.1
S1(config)#line console 0
S1(config-line)#password cisco
S1(config-line)#login
S1(config-line)#line vty 0 15
S1(config-line)#password cisco
S1(config-line)#login
S1(config-line)#end
%SYS-5-CONFIG_I: Configured from console by console
S1#copy running-config startup-config
Destination filename [startup-config]? [enter]
Building configuration...
```

Task 2: Configure the Ethernet Interfaces on the Host PCs

Configure the Ethernet interfaces of PC1, PC2 and PC3 with the IP addresses from the addressing table.

Task 3: Configure VTP on the Switches

Step 1. Enable the user ports on S2 in access mode.

```
S2(config)#interface fa0/6
S2(config-if)#switchport mode access
S2(config-if)#no shutdown
S2(config-if)#interface fa0/11
S2(config-if)#switchport mode access
S2(config-if)#no shutdown
S2(config-if)#interface fa0/18
S2(config-if)#switchport mode access
S2(config-if)#no shutdown
```

Step 2. Configure VTP.

Configure VTP on the three switches using the following table. Remember that VTP domain names and passwords are case-sensitive.

Switch Name	VTP Operating Mode	VTP Domain	VTP Password
S1	Server	Lab5	cisco
S2	Client	Lab5	cisco
S3	Client	Lab5	cisco

```
S1(config)#vtp mode server
Device mode already VTP SERVER.
S1(config)#vtp domain Lab6
Changing VTP domain name from NULL to Lab6
S1(config)#vtp password cisco
Setting device VLAN database password to cisco
S1(config)#end
```

```
S2(config)#vtp mode client
Setting device to VTP CLIENT mode
S2(config)#vtp domain Lab6
Changing VTP domain name from NULL to Lab6
```

```
S2(config)#vtp password cisco
Setting device VLAN database password to cisco
S2(config)#end

S3(config)#vtp mode client
Setting device to VTP CLIENT mode
S3(config)#vtp domain Lab6
Changing VTP domain name from NULL to Lab6
S3(config)#vtp password cisco
Setting device VLAN database password to cisco
S3(config)#end
```

Step 3. Configure trunking ports and designate the native VLAN for the trunks.

Configure Fa0/1 through Fa0/5 as trunking ports, and designate VLAN 99 as the native VLAN for these trunks. When this activity was started, these ports were disabled and must be re-enabled now using the **no shutdown** command.

Only the commands for the FastEthernet0/1 interface on each switch are shown, but the commands should be applied up to the FastEthernet0/5 interface.

```
S1(config)#interface fa0/1
S1(config-if)#switchport mode trunk
S1(config-if)#switchport trunk native  vlan 99
S1(config-if)#no shutdown
S1(config)#end

S2(config)#interface fa0/1
S2(config-if)#switchport mode trunk
S2(config-if)#switchport trunk native  vlan 99
S2(config-if)#no shutdown
S2(config-if)#end

S3(config)#interface fa0/1
S3(config-if#switchport mode trunk
S3(config-if)#switchport trunk native  vlan 99
S3(config-if)#no shutdown
S3(config-if-#end
```

Step 4. Configure the VTP server with VLANs.

Configure the following VLANS on the VTP server:

VLAN	VLAN Name
VLAN 99	management
VLAN 10	faculty-staff
VLAN 20	students
VLAN 30	guest

```
S1(config)#vlan 99
S1(config-vlan)#name management
S1(config)#vlan 10
S1(config-vlan)#name faculty-staff
S1(config)#vlan 20
S1(config-vlan)#name students
S1(config)#vlan 30
```

```
S1(config-vlan)#name guest
S1(config-vlan)#end
```

Verify that the VLANs have been created on S1 with the show vlan brief command.

Step 5. Verify that the VLANs created on S1 have been distributed to S2 and S3.

Use the **show vlan brief** command on S2 and S3 to verify that all four VLANs have been distributed to the client switches.

```
S2#show vlan brief

VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
1    default                          active    Fa0/1, Fa0/2, Fa0/4, Fa0/5
                                                Fa0/6, Fa0/7, Fa0/8, Fa0/9
                                                Fa0/10, Fa0/11, Fa0/12,Fa0/13
                                                Fa0/14, Fa0/15, Fa0/16,Fa0/17
                                                Fa0/18, Fa0/19, Fa0/20,Fa0/21
                                                Fa0/22, Fa0/23, Fa0/24, Gi0/1
                                                Gi0/2
10   faculty/staff                    active
20   students                         active
30   guest                            active
99   management                       active

S3#show vlan brief

VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
1    default                          active    Fa0/5, Fa0/6, Fa0/7, Fa0/8
                                                Fa0/9, Fa0/10, Fa0/11, Fa0/12
                                                Fa0/13, Fa0/14, Fa0/15, Fa0/16
                                                Fa0/17, Fa0/18, Fa0/19, Fa0/20
                                                Fa0/21, Fa0/22, Fa0/23, Fa0/24
                                                Gig1/1, Gig1/2
10   faculty-staff                    active
20   students                         active
30   guest                            active
99   management                       active
1002 fddi-default                     active
1003 token-ring-default               active
1004 fddinet-default                  active
1005 trnet-default                    active
```

Step 6. Configure the management interface address on all three switches.

```
S1(config)#interface vlan99
S1(config-if)#ip address 172.17.99.11  255.255.255.0

S2(config)#interface vlan99
S2(config-if)#ip address 172.17.99.12  255.255.255.0

S3(config)#interface vlan99
S3(config-if)#ip address 172.17.99.13  255.255.255.0
```

Verify that the switches are correctly configured by pinging between them. From S1, ping the management interface on S2 and S3. From S2, ping the management interface on S3.

Were the pings successful? _____

If not, troubleshoot the switch configurations and try again.

Step 7. Assign switch ports to VLANs on S2.

Port assignments are listed in the table at the beginning of the activity. However, since Packet Tracer 4.11 does not support the **interface range** command, only assign the first port from each range.

```
S2(config)#interface fa0/6
S2(config-if)#switchport access vlan 30
S2(config-if)#interface fa0/11
S2(config-if)#switchport access vlan 10
S2(config-if)#interface fa0/18
S2(config-if)#switchport access vlan 20
S2(config-if)#end
S2#copy running-config startup-config
Destination filename [startup-config]? [enter]
Building configuration...
[OK]
S2#
```

Step 8. Check connectivity between VLANs.

Open the Command Prompt on the three PCs.

- Ping from PC1 to PC2 (172.17.20.22)
- Ping from PC2 to PC3 (172.17.30.23)
- Ping from PC3 to PC1 (172.17.30.21)

Are the pings successful? _____

If not, why do these pings fail?

Task 4: Configure the Router and the Remote Server LAN

Step 1. Create a basic configuration on the router.

- Configure the router with hostname R1.
- Disable DNS lookup.
- Configure an EXEC mode password of **class**.
- Configure a password of **cisco** for console connections.
- Configure a password of **cisco** for vty connections.

Step 2. Configure the trunking interface on R1.

You have demonstrated that connectivity between VLANs requires routing at the network layer, exactly like connectivity between any two remote networks. There are a couple of options for configuring routing between VLANs.

The first is something of a brute force approach. An L3 device, either a router or a Layer 3 capable switch, is connected to a LAN switch with multiple connections--a separate connection for each VLAN that requires inter-VLAN connectivity. Each of the switch ports used by the L3 device are configured in a different VLAN on the switch. After IP addresses are assigned to the interfaces on the L3 device, the routing table has directly connected routes for all VLANs, and inter-VLAN routing is enabled. The limitations to this approach are the lack of sufficient Fast Ethernet ports on routers, under-utilization of ports on L3 switches and routers, and excessive wiring and manual configuration. The topology used in this lab does not use this approach.

An alternative approach is to create one or more Fast Ethernet connections between the L3 device (the router) and the distribution layer switch, and to configure these connections as **dot1q** trunks. This allows all inter-VLAN traffic to be carried to and from the routing device on a single trunk. However, it requires that the L3 interface be configured with multiple IP addresses. This can be done by creating virtual interfaces, called subinterfaces, on one of the router Fast Ethernet ports and configuring them to be **dot1q** aware.

Using the subinterface configuration approach requires these steps:

- Enter subinterface configuration mode
- Establish trunking encapsulation
- Associate a VLAN with the subinterface
- Assign an IP address from the VLAN to the subinterface

The commands are as follows:

```
R1(config)#interface fastethernet 0/0
R1(config-if)#no shutdown
R1(config-if)#interface fastethernet 0/0.1
R1(config-subif)#encapsulation dot1q 1
R1(config-subif)#ip address 172.17.1.1 255.255.255.0
R1(config-subif)#interface fastethernet 0/0.10
R1(config-subif)#encapsulation dot1q 10
R1(config-subif)#ip address 172.17.10.1 255.255.255.0
R1(config-subif)#interface fastethernet 0/0.20
R1(config-subif)#encapsulation dot1q 20
R1(config-subif)#ip address 172.17.20.1 255.255.255.0
R1(config-subif)#interface fastethernet 0/0.30
R1(config-subif)#encapsulation dot1q 30
R1(config-subif)#ip address 172.17.30.1 255.255.255.0
R1(config-subif)#interface fastethernet 0/0.99
R1(config-subif)#encapsulation dot1q 99 native
R1(config-subif)#ip address 172.17.99.1 255.255.255.0
```

Note the following points in this configuration:

- The physical interface is enabled using the **no shutdown** command, because router interfaces are down by default. The subinterface will then be up by default.
- The subinterface can use any number that can be described with 32 bits, but it is good practice to assign the number of the VLAN as the interface number, as has been done here.
- The native VLAN is specified on the L3 device so that it is consistent with the switches. Otherwise, VLAN 1 is native by default, and there is no communication between the router and the management VLAN on the switches.

Step 3. Configure the server LAN interface on R1.

```
R1(config)#interface FastEthernet0/1
R1(config-if)#ip address 172.17.50.1 255.255.255.0
R1(config-if)#description server interface
R1(config-if)#no shutdown
R1(config-if)#end
```

There are now six networks configured. Verify that you can route packets to all six by checking the routing table on R1.

```
R1#show ip route
<output omitted>
```

```
Gateway of last resort is not set

     172.17.0.0/24 is subnetted, 6 subnets
C       172.17.1.0 is directly connected, FastEthernet0/0.1
C       172.17.10.0 is directly connected, FastEthernet0/0.10
C       172.17.20.0 is directly connected, FastEthernet0/0.20
C       172.17.30.0 is directly connected, FastEthernet0/0.30
C       172.17.50.0 is directly connected, FastEthernet0/1
C       172.17.99.0 is directly connected, FastEthernet0/0.99
```

If your routing table does not show all six networks, troubleshoot your configuration and resolve the problem before proceeding.

Step 4. Verify Inter-VLAN routing.

From PC1, verify that you can ping the remote server (172.17.50.254) and the other two hosts (172.17.20.22 and 172.17.30.23). It may take a couple of pings before the end-to-end path is established.

These pings should be successful. If not, troubleshoot your configuration. Check to make sure that the default gateways have been set on all PCs and all switches.

Task 5: Reflection

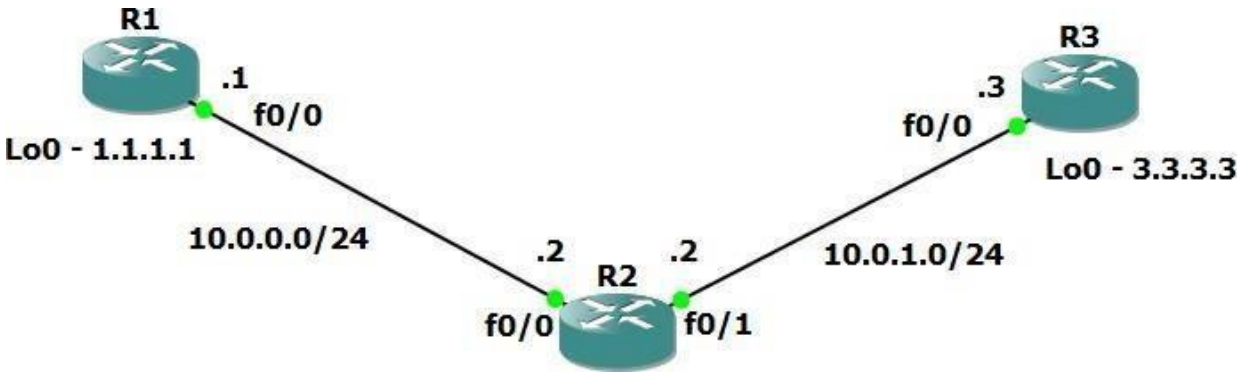
In Task 4, you configured VLAN 99 as the native VLAN in the router Fa0/0.99 interface configuration. Why would packets from the router or hosts fail when trying to reach the switch management interfaces if the native VLAN were left in default?

Practical 8

Simulating MPLS Environment

Step 1 – IP addressing of MPLS Core and OSPF

First bring 3 routers into your topology R1, R2, R3 position them as below. We are going to address the routers and configure ospf to ensure loopbackto loopback connectivity between R1 and R3




```
R1

hostname R1

int lo0

ip add 1.1.1.1 255.255.255.255

ip ospf 1 area 0


int f0/0

ip add 10.0.0.1 255.255.255.0

no shut

ip ospf 1 area 0


R2

hostname R2

int lo0
```

```
ip add 2.2.2.2 255.255.255.255
ip ospf 1 are 0


int f0/0
ip add 10.0.0.2 255.255.255.0
no shut
ip ospf 1 area 0


int f0/1
ip add 10.0.1.2 255.255.255.0
no shut
ip ospf 1 area 0



R3
hostname R3
int lo0
ip add 3.3.3.3 255.255.255.255
ip ospf 1 are 0


int f0/0
ip add 10.0.1.3 255.255.255.0
no shut
ip ospf 1 area 0
```

You should now have full ip connectivity between R1, R2, R3 to verify thiswe need to see if we can ping between the loopbacks of R1 and R3

```
R1#ping 3.3.3.3 source lo0



Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 3.3.3.3, timeout is 2
seconds:

Packet sent with a source address of 1.1.1.1

!!!!!

Success rate is 100 percent (5/5), round-trip
min/avg/max = 40/52/64 ms

R1#
```

You could show the routing table here, but the fact that you can ping between the loopbacks is verification enough and it is safe to move on.

Step 2 – Configure LDP on all the interfaces in the MPLS Core

In order to run MPLS you need to enable it, there are two ways to do this.

- At each interface enter the **mpls ip** command
- Under the ospf process use the **mpls ldp autoconfig** command

For this tutorial we will be using the second option, so go int the ospf process and enter mpls ldp autoconfig – this will enable mpls label distribution protocol on every interface running ospf under that specificprocess.

```
R1

router ospf 1

mpls ldp autoconfig



R2

router ospf 1
```

```
mpls ldp autoconfig



R3

router ospf 1

mpls ldp autoconfig
```

You should see log messages coming up showing the LDP neighbors are up.

```
R2#

*Mar 1 00:31:53.643: %SYS-5-CONFIG_I: Configured from
console

*Mar  1 00:31:54.423: %LDP-5-NBRCHG: LDP Neighbor
1.1.1.1:0 (1) is UP

R2#

*Mar  1 00:36:09.951: %LDP-5-NBRCHG: LDP Neighbor
3.3.3.3:0 (2) is UP
```

To verify the mpls interfaces the command is very simple – **sh mpls interface**
This is done on R2 and you can see that both interfaces are running mplsand using LDP

```
R2#sh mpls interface

Interface              IP            Tunnel
Operational

FastEthernet0/0        Yes (ldp)     No       Yes

FastEthernet0/1        Yes (ldp)     No       Yes
```

You can also verify the LDP neighbors with the **sh mpls ldp neighbors** command.

```
R2#sh mpls ldp neigh

    Peer LDP Ident: 1.1.1.1:0; Local LDP Ident 2.2.2.2:0

        TCP connection: 1.1.1.1.646 - 2.2.2.2.37909
```

One more verification to confirm LDP is running ok is to do a trace between R1 and R3 and verify if you get MPLS Labels show up in the trace.

```
R1#trace 3.3.3.3


Type escape sequence to abort.

Tracing the route to 3.3.3.3


  1 10.0.0.2 [MPLS: Label 17 Exp 0] 84 msec 72 msec 44
msec

  2 10.0.1.3 68 msec 60 msec *
```

As you can see the trace to R2 used an MPLS Label in the path, as this is a very small MPLS core only one label was used as R3 was the final hop.

So to review we have now configured IP addresses on the MPLS core,enabled OSPF and full IP connectivity between all routers and finally enabled mpls on all the interfaces in the core and have established ldpneighbors between all routers.

The next step is to configure MP-BGP between R1 and R3

This is when you start to see the layer 3 vpn configuration come to life

Step 3 – MPLS BGP Configuration between R1 and R3

We need to establish a Multi Protocol BGP session between R1 and R3 thisis done by configuring the vpnv4 address family as below

```
R1#
router bgp 1
 neighbor 3.3.3.3 remote-as 1
 neighbor 3.3.3.3 update-source Loopback0
 no auto-summary
 !
 address-family vpnv4
  neighbor 3.3.3.3 activate


R3#
router bgp 1
 neighbor 1.1.1.1 remote-as 1
 neighbor 1.1.1.1 update-source Loopback0
 no auto-summary
 !
 address-family vpnv4
  neighbor 1.1.1.1 activate
```

```
*Mar  1 00:45:01.047: %BGP-5-ADJCHANGE: neighbor 1.1.1.1
Up
```

You should see log messages showing the BGP sessions coming up.

To verify the BGP session between R1 and R3 issue the command **sh bgpvpnv4 unicast all summary**

You can see here that we do have a bgp vpnv4 peering to R3 – looking at the PfxRcd you can see it says 0 this is because we have not got any routesin BGP. We are now going to add two more routers to the topology. These will be the customer sites connected to R1 and R3. We will then create a VRF on each router and put the interfaces connected to each site router intothat VRF.

Step 4 – Add two more routers, create VRFs

We will add two more routers into the topology so it now looks like the finaltopology

Router 4 will peer OSPF using process number 2 to a VRF configured onR1. It will use the local site addressing of 192.168.1.0/24.

```
R4

int lo0

ip add 4.4.4.4 255.255.255.255

ip ospf 2 area 2

int f0/0
```

```
ip add 192.168.1.4 255.255.255.0
ip ospf 2 area 2
no shut


R1
int f0/1
no shut
ip add 192.168.1.1 255.255.255.0
```

Now at this point we have R4 peering to R1 but in the global routing table of R1 which is not what we want.

We are now going to start using VRF's

What is a VRF in networking?

Virtual routing and forwarding (**VRF**) is a technology included in IP (Internet Protocol) that allows multiple instances of a routing table to co-exist in a router and work together but not interfere with each other.. This increases functionality by allowing network paths to be segmented without using multiple devices.

As an example if R1 was a PE Provider Edge router of an ISP and it had two customers that were both addressed locally with the 192.168.1.0/24 addressspace it could accommodate both their routing tables in different VRFs – it distinguishes between the two of them using a Route Distinguisher

So back to the topology – we now need to create a VRF on R1For

this mpls tutorial I will be using VRF RED

```
R1
ip vrf RED
rd 4:4
route-target both 4:4
```

The RD and route-target do not need to be the same – and for a full explanation please read this post on Route Distinguishers Route Distinguisher vs Route Target before proceeding.

So now we have configured the VRF on R1 we need to move the interfaceF0/1 into that VRF

```
R1

int f0/1

ip vrf forwarding RED
```

Now notice what happens when you do that – the IP address is removed

```
R1(config-if)#ip vrf fo

R1(config-if)#ip vrf forwarding RED

% Interface FastEthernet0/1 IP address 192.168.1.1
removed due to enabling VRF RED
```

You just need to re-apply it

```
R1

int f0/1

ip address 192.168.1.1 255.255.255.0
```

Now if we view the config on R1 int f0/1 you can see the VRF configured.

```
R1




R1#sh run int f0/1

Building configuration...


Current configuration : 119 bytes

!
```

```
interface FastEthernet0/1

 ip vrf forwarding RED

 ip address 192.168.1.1 255.255.255.0

 duplex auto

 speed auto

end


R1#
```

Now we can start to look int VRF's and how they operate – you need to understand now that there are 2 routing tables within R1

- The Global Routing Table
- The Routing Table for VRF RED

If you issue the command **sh ip route** this shows the routes in the globaltable and you will notice that you do not see 192.168.1.0/24

```
R1#sh ip route

Codes: C - connected, S - static, R - RIP, M - mobile, B
- BGP

 D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF
inter area

 N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external
type 2

 E1 - OSPF external type 1, E2 - OSPF external type 2

 i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 -
IS-IS level-2

 ia - IS-IS inter area, * - candidate default, U - per-
user static route

 o - ODR, P - periodic downloaded static route


Gateway of last resort is not set
```

```
1.0.0.0/32 is subnetted, 1 subnets

C 1.1.1.1 is directly connected, Loopback0

 2.0.0.0/32 is subnetted, 1 subnets

O 2.2.2.2 [110/11] via 10.0.0.2, 01:03:48,
FastEthernet0/0

 3.0.0.0/32 is subnetted, 1 subnets

O 3.3.3.3 [110/21] via 10.0.0.2, 01:02:29,
FastEthernet0/0

 10.0.0.0/24 is subnetted, 2 subnets

C 10.0.0.0 is directly connected, FastEthernet0/0

O 10.0.1.0 [110/20] via 10.0.0.2, 01:02:39,
FastEthernet0/0

R1#
```

If you now issue the command sh ip route vrf red – this will show the routes in the routing table for VRF RED

```
R1#sh ip route vrf red

% IP routing table red does not exist
```

NOTE: The VRF name is case sensitive!

```
R1#sh ip route vrf RED



Routing Table: RED

Codes: C - connected, S - static, R - RIP, M - mobile, B
- BGP

 D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF
inter area

 N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external
type 2
```

```
 E1 - OSPF external type 1, E2 - OSPF external type 2

 i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 -
IS-IS level-2

 ia - IS-IS inter area, * - candidate default, U - per-
user static route

 o - ODR, P - periodic downloaded static route



Gateway of last resort is not set



C 192.168.1.0/24 is directly connected, FastEthernet0/1

R1#
```

We just need to enable OSPF on this interface and get the loopback address for R4 in the VRF RED routing table before proceeding.

```
R1

 int f0/1

 ip ospf 2 area 2
```

You should see a log message showing the OSPF neighbor come up

```
R1(config-if)#

  *Mar 1 01:12:54.323: %OSPF-5-ADJCHG: Process 2, Nbr
4.4.4.4

 on FastEthernet0/1 from LOADING to FULL, Loading Done
```

If we now check the routes in the VRF RED routing table you should see 4.4.4.4 in there as well.

```
R1#sh ip route vrf RED



Routing Table: RED

 Codes: C - connected, S - static, R - RIP, M - mobile,
B - BGP
```

```
 D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF
inter area

 N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external
type 2

 E1 - OSPF external type 1, E2 - OSPF external type 2

 i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 -
IS-IS level-2

 ia - IS-IS inter area, * - candidate default, U - per-
user static route

 o - ODR, P - periodic downloaded static route



Gateway of last resort is not set



4.0.0.0/32 is subnetted, 1 subnets

 O 4.4.4.4 [110/11] via 192.168.1.4, 00:00:22,
FastEthernet0/1

 C 192.168.1.0/24 is directly connected, FastEthernet0/1

 R1#
```

We now need to repeat this process for R3 & R6

Router 6 will peer OSPF using process number 2 to a VRF configured onR3. It will use the local site addressing of 192.168.2.0/24.

```
R6

int lo0

ip add 6.6.6.6 255.255.255.255

ip ospf 2 area 2

int f0/0

ip add 192.168.2.6 255.255.255.0

ip ospf 2 area 2
```

```
no shut


R3

int f0/1

no shut

ip add 192.168.2.3 255.255.255.0
```

We also need to configure a VRF onto R3 as well.

```
R3

ip vrf RED

rd 4:4

route-target both 4:4
```

So now we have configured the VRF on R3 we need to move the interface F0/1 into that VRF

```
R3

int f0/1

ip vrf forwarding RED
```

Now notice what happens when you do that – the IP address is removed

```
R3(config-if)#ip vrf forwarding RED

% Interface FastEthernet0/1 IP address 192.168.2.1
removed due to enabling VRF RED
```

You just need to re-apply it

```
R3
int f0/1
ip address 192.168.2.1 255.255.255.0
```

Now if we view the config on R3 int f0/1 you can see the VRF configured.

```
R3
```

```
R3#sh run int f0/1
Building configuration...


Current configuration : 119 bytes
!
interface FastEthernet0/1
ip vrf forwarding RED
ip address 192.168.2.1 255.255.255.0
duplex auto
speed auto
end
```

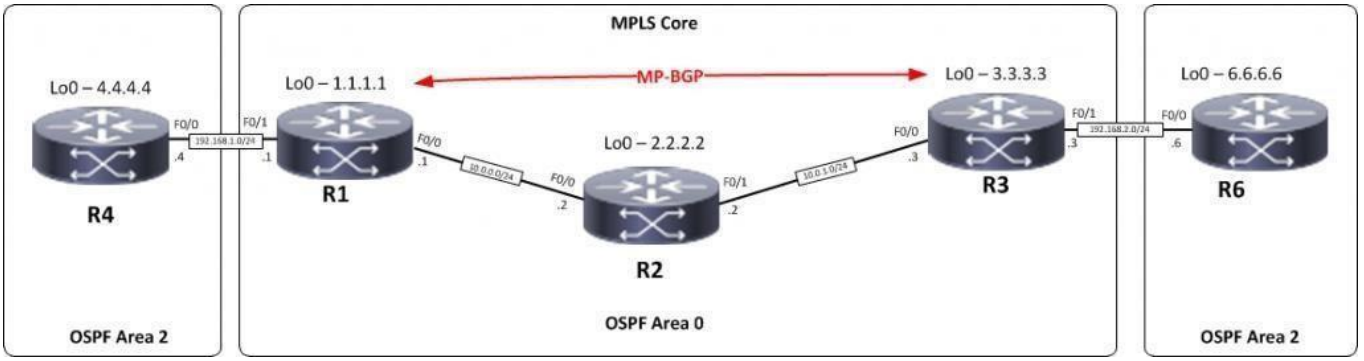
Finally we just need to enable OSPF on that interface and verify the routes are in the RED routing table.

```
R3
int f0/1
ip ospf 2 area 2
```

Check the routes in vrf RED

```
R3
R3#sh ip route vrf RED


Routing Table: RED
Codes: C - connected, S - static, R - RIP, M - mobile, B
- BGP


Gateway of last resort is not set
```

Ok so we have come a long way now let's review the current situation. We now have this setup



R1,R2,R3 form the MPLS Core and are running OSPF with all loopbacks running a /32 address and all have full connectivity. R1 and R3 are peeringwith MP-BGP. LDP is enabled on all the internal interfaces. The external interfaces of the MPLS core have been placed into a VRF called RED and then a site router has been joined to that VRF on each side of the MPLS core – (These represent a small office)

The final step to get full connectivity across the MPLS core is to redistributethe routes in OSPF on R1 and R3 into MP-BGP and MP-BGP into OSPF, this is what we are going to do now.

We need to redistribute the OSPF routes from R4 into BGP in the VRF on R1, the OSPF routes from R6 into MP-BGP in the VRF on R3 and then theroutes in MP-BGP in R1 and R3 back out to OSPF

Before we start lets do some verifications

Check the routes on R4

```
R4#sh ip route

4.0.0.0/32 is subnetted, 1 subnets

C 4.4.4.4 is directly connected, Loopback0
```

```
C 192.168.1.0/24 is directly connected, FastEthernet0/0
```

As expected we have the local interface and the loopback address.

When we are done we want to see 6.6.6.6 in there so we can ping acrossthe MPLS

Check the routes on R1

```
R1#sh ip route



1.0.0.0/32 is subnetted, 1 subnets

C 1.1.1.1 is directly connected, Loopback0

 2.0.0.0/32 is subnetted, 1 subnets

O 2.2.2.2 [110/11] via 10.0.0.2, 00:01:04,
FastEthernet0/0

 3.0.0.0/32 is subnetted, 1 subnets

O 3.3.3.3 [110/21] via 10.0.0.2, 00:00:54,
FastEthernet0/0

 10.0.0.0/24 is subnetted, 2 subnets

C 10.0.0.0 is directly connected, FastEthernet0/0

O 10.0.1.0 [110/20] via 10.0.0.2, 00:00:54,
FastEthernet0/0
```

Remember we have a VRF configured on this router so this command will show routes in the global routing table (the MPLS Core) and it will not showthe 192.168.1.0/24 route as that is in VRF RED – to see that we run the following command

```
R1#sh ip route vrf RED



Routing Table: RED



4.0.0.0/32 is subnetted, 1 subnets
```

```
O 4.4.4.4 [110/11] via 192.168.1.4, 00:02:32,
FastEthernet0/1

C 192.168.1.0/24 is directly connected, FastEthernet0/1
```

Here you can see Routing Table: RED is shown and the routes to R4 are now visible with 4.4.4.4 being in OSPF.So

we need to do the following;

- Redistribute OSPF into MP-BGP on R1
- Redistribute MP-BGP into OSPF on R1
- Redistribute OSPF into MP-BGP on R3
- Redistribute MP-BGP into OSPF on R3

Redistribute OSPF into MP-BGP on R1

```
R1

router bgp 1

address-family ipv4 vrf RED

redistribute ospf 2
```

Redistribute OSPF into MP-BGP on R3

```
R3

router bgp 1

address-family ipv4 vrf RED

redistribute ospf 2
```

This has enabled redistribution of the OSPF routes into BGP. We can check the routes from R4 and R6 are now showing in the BGP table for their VRFwith this command

sh ip bgp vpnv4 vrf RED

```
R1#sh ip bgp vpnv4 vrf RED

BGP table version is 9, local router ID is 1.1.1.1

Status codes: s suppressed, d damped, h history, *
valid, > best,
```

```
 r RIB-failure, S Stale

Origin codes: i - IGP, e - EGP, ? - incomplete


 Network Next Hop Metric LocPrf Weight Path

Route Distinguisher: 4:4 (default for vrf RED)

*> 4.4.4.4/32 192.168.1.4 11 32768 ?

*>i6.6.6.6/32 3.3.3.3 11  100 0 ?

*> 192.168.1.0 0.0.0.0 0  32768 ?

*>i192.168.2.0 3.3.3.3 0  100 0 ?
```

Here we can see that 4.4.4.4 is now in the BGP table in VRF RED on R1 with a next hop of 192.168.1.4 (R4) and also 6.6.6.6 is in there as wellwith a next hop of 3.3.3.3 (which is the loopback of R3 – showing that it isgoing over the MPLS and R1 is not in the picture)

The same should be true on R3

```
R3#sh ip bgp vpnv4 vrf RED

BGP table version is 9, local router ID is 3.3.3.3

Status codes: s suppressed, d damped, h history, *
valid, > best, i - internal,

r RIB-failure, S Stale

Origin codes: i - IGP, e - EGP, ? - incomplete


Network Next Hop Metric LocPrf Weight Path

Route Distinguisher: 4:4 (default for vrf RED)

*>i4.4.4.4/32 1.1.1.1 11 100 0 ?

*> 6.6.6.6/32 192.168.2.6 11 32768 ?

*>i192.168.1.0 1.1.1.1 0 100 0 ?
```

```
*> 192.168.2.0 0.0.0.0 0 32768 ?
```

Which it is! 6.6.6.6 is now in the BGP table in VRF RED on R3 with a next hop of 192.168.2.6 (R6) and also 4.4.4 is in there as well with a next hop of 1.1.1.1 (which is the loopback of R1 – showing that it is going over the MPLSand R2 is not in the picture)
The final step is to get the routes that have come across the MPLS back intoOSPF and then we can get end to end connectivity

```
R1


router ospf 2

redistribute bgp 1 subnets


R3


router ospf 2

redistribute bgp 1 subnets
```

If all has worked we should be now able to ping 6.6.6.6 from R4

Before we do let's see what the routing table looks like on R4

```
R4#sh ip route




4.0.0.0/32 is subnetted, 1 subnets

C 4.4.4.4 is directly connected, Loopback0

 6.0.0.0/32 is subnetted, 1 subnets

O IA 6.6.6.6 [110/21] via 192.168.1.1, 00:01:31,
FastEthernet0/0

C 192.168.1.0/24 is directly connected, FastEthernet0/0
```

```
O E2 192.168.2.0/24 [110/1] via 192.168.1.1, 00:01:31,
FastEthernet0/0
```

Great we have 6.6.6.6 in there

Also check the routing table on R6

```
R6#sh ip route



4.0.0.0/32 is subnetted, 1 subnets

O IA 4.4.4.4 [110/21] via 192.168.2.1, 00:01:22,
FastEthernet0/0

6.0.0.0/32 is subnetted, 1 subnets

C 6.6.6.6 is directly connected, Loopback0

O IA 192.168.1.0/24 [110/11] via
192.168.2.1,00:01:22,FastEthernet0/0

C 192.168.2.0/24 is directly connected, FastEthernet0/0
```

Brilliant we have 4.4.4.4 in there so we should be able to ping across the MPLS

```
R4#ping 6.6.6.6



Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 6.6.6.6, timeout is 2
seconds:

!!!!!

Success rate is 100 percent (5/5), round-trip
min/avg/max= 40/48/52ms
```

Which we can – to prove this is going over the MPLS and be label switched and not routed, lets do a trace

```
R4#trace 6.6.6.6
```

```
Type escape sequence to abort.

Tracing the route to 6.6.6.6



1 192.168.1.1 20 msec 8 msec 8 msec

 2 10.0.0.2 [MPLS: Labels 17/20 Exp 0] 36 msec 40 msec
36 msec

 3 192.168.2.1 [MPLS: Label 20 Exp 0] 16 msec 40 msec 16
msec

 4 192.168.2.6 44 msec 40 msec 56 msec

R4#
```

What is MPLS network and how does it work?

Multiprotocol Label Switching (**MPLS**) is a way of routing traffic within a telecommunications **network** that directs data from one node to the next based path labels rather than long **network** addresses, It also allows the sharing of address space for clients as it is labels that are being routed notprefixes.

Is MPLS a routing protocol?

No, MPLS is a method to route networks across a service provider network,routing protocols like OSPF and BGP are used to make MPLS work. MPLSoperates using BGP and typically uses OSPF to exchange routes with the customer.

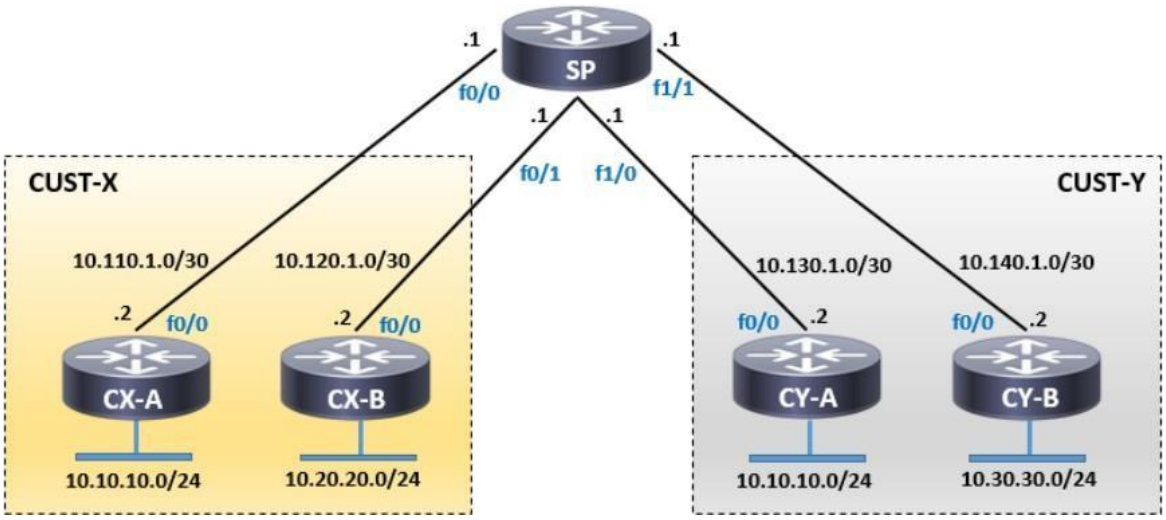
Why is MPLS used?

MPLS was designed to work in a multiple protocol environment.
Today, **MPLS** is **used** to support metro-Ethernet services & mobile communications back-haul it's main benefit is the ability to have two clients using the same address space and routing over the service provider networkas they are routing using labels and not prefixes.

Practical 9

Simulating VRF

Topology



Initial Configuration

Here's how the SP's global routing table looks like. As a side note, I'm using IOS 15 which is why I'm excluding the Local routes to minimize the entries in each output.

```
SP#show ip route | exclude L

Gateway of last resort is not set

10.0.0.0/8 is variably subnetted, 8 subnets, 2 masks
C 10.110.1.0/30 is directly connected, FastEthernet0/0
C 10.120.1.0/30 is directly connected, FastEthernet0/1
C 10.130.1.0/30 is directly connected, FastEthernet1/0
C 10.140.1.0/30 is directly connected, FastEthernet1/1
```

All customer routers have a default route pointing to the SP. Here is the routing table for one of them.

```
CX-A#show ip route | exclude L

Gateway of last resort is 10.110.1.1 to network 0.0.0.0
S* 0.0.0.0/0 [1/0] via 10.110.1.1
10.0.0.0/8 is variably subnetted, 4 subnets, 3 masks
C 10.110.1.0/30 is directly connected, FastEthernet0/0
```

Now, let's create the VRFs for each customer.

```
ISP(config)#ip vrf CUST-X
ISP(config-vrf)#exit
ISP(config)#ip vrf CUST-Y
ISP(config-vrf)#exit
ISP(config)#interface range FastEthernet 0/0 - 1
ISP(config-if-range)#ip vrf forwarding CUST-X
% Interface FastEthernet0/0 IPv4 disabled and address(es) removed
due to disabling VRF CUST-X
% Interface FastEthernet0/1 IPv4 disabled and address(es) removed
due to disabling VRF CUST-X
ISP(config-if-range)#interface range FastEthernet 1/0 - 1
ISP(config-if-range)#ip vrf forwarding CUST-Y
% Interface FastEthernet1/0 IPv4 disabled and address(es) removed
due to disabling VRF CUST-Y
% Interface FastEthernet1/1 IPv4 disabled and address(es) removed
due to disabling VRF CUST-Y
```

```
ISP#show ip interface brief

Interface IP-Address OK? Method Status Protocol

FastEthernet0/0 unassigned YES manual up up

FastEthernet0/1 unassigned YES manual up up

FastEthernet1/0 unassigned YES manual up up

FastEthernet1/1 unassigned YES manual up up


ISP#show ip route

Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B
- BGP

D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area

N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2

E1 - OSPF external type 1, E2 - OSPF external type 2

i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS
level-2

ia - IS-IS inter area, * - candidate default, U - per-user static
route

o - ODR, P - periodic downloaded static route, + - replicated route

Gateway of last resort is not set

ISP#
```

Notice the messages regarding the deletion of the IPv4 address on each interface where a VRF was associated. Since we've moved all interfaces to the respective VRFs, the global routing table of the SP router is now empty. All those addresses have been deleted and must be recreated again.

Now, with the creation of VRFs, normal verification commands become "VRF Aware". This means that each verification or testing syntax whether it be ping, traceroute, telnet, show, etc., that pertains to a specific VRF must include the *vrf <vrf name>* keyword.

The addresses have now been reconfigured to the appropriate interfaces to populate each table. Shown below are the routing tables of each customer.

```
ISP#show ip route vrf CUST-X | exclude L

Routing Table: CUST-X

Gateway of last resort is not set

10.0.0.0/8 is variably subnetted, 4 subnets, 2
masks

C 10.110.1.0/30 is directly connected,
FastEthernet0/0

C 10.120.1.0/30 is directly connected,
FastEthernet0/1

ISP#show ip route vrf CUST-Y | exclude L

Routing Table: CUST-Y


Gateway of last resort is not set

10.0.0.0/8 is variably subnetted, 4 subnets, 2
masks

C 10.130.1.0/30 is directly connected,
FastEthernet1/0

C 10.140.1.0/30 is directly connected,
FastEthernet1/1
```

Static Routing

The SP can't reach each customers' LAN at the moment. Since the setup only requires static routes, lets go ahead and create them.

```
ISP(config)#ip route vrf CUST-X 10.10.10.0
255.255.255.0 10.110.1.2

ISP(config)#ip route vrf CUST-X 10.20.20.0
255.255.255.0 10.120.1.2

ISP(config)#ip route vrf CUST-Y 10.10.10.0
255.255.255.0 10.130.1.2

ISP(config)#ip route vrf CUST-Y 10.30.30.0
255.255.255.0 10.140.1.2
```

Lets look at those tables again. Notice the static entries, specially the ones for 10.10.10.0/24 for each customer. If those were in the same routing table, those would show as the traffic will be load balanced between the next-hops.

```
ISP#show ip route vrf CUST-X | exclude L


Routing Table: CUST-X


Gateway of last resort is not set


10.0.0.0/8 is variably subnetted, 6 subnets, 3
masks

S 10.10.10.0/24 [1/0] via 10.110.1.2

S 10.20.20.0/24 [1/0] via 10.120.1.2

C 10.110.1.0/30 is directly connected,
FastEthernet0/0

C 10.120.1.0/30 is directly connected,
FastEthernet0/1
```

```
ISP#show ip route vrf CUST-Y | exclude L

Routing Table: CUST-Y

Gateway of last resort is not set

10.0.0.0/8 is variably subnetted, 6 subnets, 3
masks
S 10.10.10.0/24 [1/0] via 10.130.1.2
S 10.30.30.0/24 [1/0] via 10.140.1.2
C 10.130.1.0/30 is directly connected,
FastEthernet1/0
C 10.140.1.0/30 is directly connected,
FastEthernet1/1
```

Verification

As mentioned previously, when testing or verifying anything related to a specific VRF, the normal commands become VRF Aware. Hence, commands that don't specify a VRF will make the router refer to the global IP routing table, which is currently empty in this case.

```
ISP#ping 10.10.10.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.10.10.1,
timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)

ISP#ping 10.20.20.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.20.20.1,
timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)

ISP#ping 10.30.30.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.30.30.1,
timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
```

Shown below are pings to addresses within each customers' VRF. Also shown below is the TCL shell to achieve the same.

```
ISP#ping vrf CUST-X 10.10.10.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.10.10.1, timeout
is 2 seconds:
!!!!!

ISP#ping vrf CUST-X 10.20.20.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.20.20.1, timeout
is 2 seconds:
!!!!!

ISP#ping vrf CUST-Y 10.10.10.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.10.10.1, timeout
is 2 seconds:
!!!!!

ISP#ping vrf CUST-Y 10.30.30.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.30.30.1, timeout
is 2 seconds:
!!!!!
```

```
ISP#tclsh
ISP(tcl)#foreach IP {
+>10.10.10.1
+>10.20.20.1
+>} {puts "[exec ping vrf CUST-X $IP]"}

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.10.10.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max =
52/156/252 ms

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.20.20.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max =
4/156/348 ms

ISP#tclsh
ISP(tcl)#foreach IP {
+>10.10.10.1
+>10.30.30.1
+>} {puts "[exec ping vrf CUST-Y $IP]"}

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.10.10.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max =
140/168/244 ms

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.30.30.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max =
4/102/220 ms
```

Finally, let's test the connectivity between the customers.

```
CX-A#tclsh
CX-A(tcl)#foreach IP {
+>10.120.1.2
+>10.20.20.1
+>10.130.1.2
+>10.140.1.2
+>10.30.30.1
+>} {ping $IP}


Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.120.1.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max =
116/188/268 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.20.20.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max =
88/179/328 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.130.1.2, timeout is 2 seconds:
UUUUU
Success rate is 0 percent (0/5)
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.140.1.2, timeout is 2 seconds:
UUUUU
Success rate is 0 percent (0/5)
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.30.30.1, timeout is 2 seconds:
UUUUU
Success rate is 0 percent (0/5)
```

```
CY-A#tclsh
CY-A(tcl)#foreach IP {
+>10.140.1.2
+>10.30.30.1
+>10.110.1.2
+>10.120.1.2
+>10.20.20.1
+>} {ping $IP}

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.140.1.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max =
56/169/256 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.30.30.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max =
140/196/288 ms

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.110.1.2, timeout is 2 seconds:
UUUUU
Success rate is 0 percent (0/5)
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.120.1.2, timeout is 2 seconds:
UUUUU
Success rate is 0 percent (0/5)
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.20.20.1, timeout is 2 seconds:
UUUUU
Success rate is 0 percent (0/5)
```

Evidently, the SP router is sending host unreachable messages for traffic sourced from one customer that is destined to another customer's premise showing that traffic is isolated on each VRF.

Extra show Commands

Now, what if we have several VRFs? Do we have to memorize each name to check each routing table? Apparently, you can view all routing tables for each VRF at the same time. The syntax is **show ip route vrf *** such as shown below.

```
ISP#show ip route vrf *

Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B
- BGP

D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area

N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2

E1 - OSPF external type 1, E2 - OSPF external type 2

i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS
level-2

ia - IS-IS inter area, * - candidate default, U - per-user static
route

o - ODR, P - periodic downloaded static route, + - replicated route


Gateway of last resort is not set
```

Routing Table: CUST-X

```
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B
- BGP

D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area

N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2

E1 - OSPF external type 1, E2 - OSPF external type 2

i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS
level-2

ia - IS-IS inter area, * - candidate default, U - per-user static
route

o - ODR, P - periodic downloaded static route, + - replicated route


Gateway of last resort is not set


10.0.0.0/8 is variably subnetted, 6 subnets, 3 masks

S 10.10.10.0/24 [1/0] via 10.110.1.2

S 10.20.20.0/24 [1/0] via 10.120.1.2

C 10.110.1.0/30 is directly connected, FastEthernet0/0

L 10.110.1.1/32 is directly connected, FastEthernet0/0

C 10.120.1.0/30 is directly connected, FastEthernet0/1

L 10.120.1.1/32 is directly connected, FastEthernet0/1
```

Routing Table: CUST-Y

```
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B
- BGP

D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area

N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2

E1 - OSPF external type 1, E2 - OSPF external type 2

i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS
level-2

ia - IS-IS inter area, * - candidate default, U - per-user static
route

o - ODR, P - periodic downloaded static route, + - replicated route


Gateway of last resort is not set


10.0.0.0/8 is variably subnetted, 6 subnets, 3 masks

S 10.10.10.0/24 [1/0] via 10.130.1.2

S 10.30.30.0/24 [1/0] via 10.140.1.2

C 10.130.1.0/30 is directly connected, FastEthernet1/0

L 10.130.1.1/32 is directly connected, FastEthernet1/0

C 10.140.1.0/30 is directly connected, FastEthernet1/1

L 10.140.1.1/32 is directly connected, FastEthernet1/1
```

What if we want to quickly check the configuration entered in a VRF? You can do a **show run vrf <vrf name>** to view the configuration on a specific VRF or just plain **show run vrf** to display relevant configuration for all VRFs.

```
ISP#show run vrf
Building configuration...

Current configuration : 749 bytes
ip vrf CUST-X
!
!
interface FastEthernet0/0
ip vrf forwarding CUST-X
ip address 10.110.1.1 255.255.255.252
!
!
interface FastEthernet0/1
ip vrf forwarding CUST-X
ip address 10.120.1.1 255.255.255.252
!
!
ip route vrf CUST-X 10.10.10.0 255.255.255.0 10.110.1.2
ip route vrf CUST-X 10.20.20.0 255.255.255.0 10.120.1.2
!
```

ip vrf CUST-Y

```
!
!
interface FastEthernet1/0
ip vrf forwarding CUST-Y
ip address 10.130.1.1 255.255.255.252
!
!
interface FastEthernet1/1
ip vrf forwarding CUST-Y
ip address 10.140.1.1 255.255.255.252
!
!
ip route vrf CUST-Y 10.10.10.0 255.255.255.0 10.130.1.2
ip route vrf CUST-Y 10.30.30.0 255.255.255.0 10.140.1.2
!
end
```

Practical 10

Simulating SDN with OpenDaylight SDN Controller and Mininet

Part 1 : Installation

Step 1 :

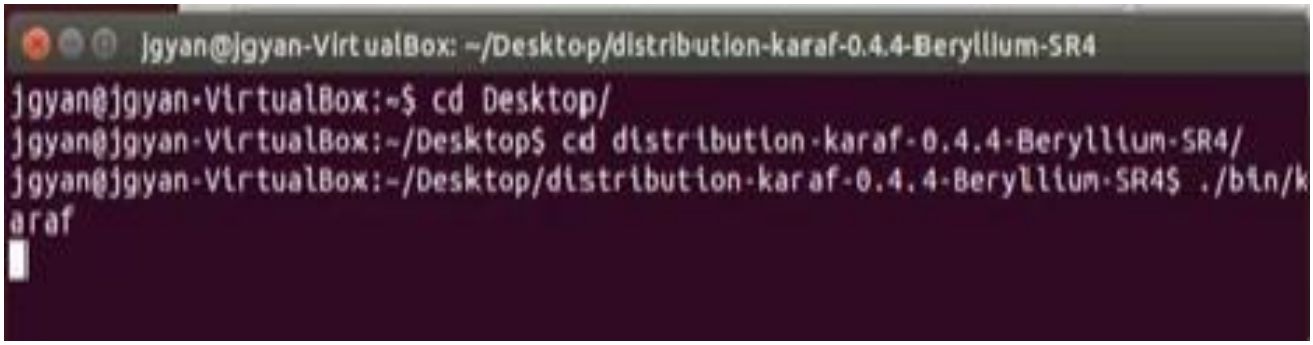
Download OpenDayLight and then Extract it, on your Desktop or any file location.

Step 2 :

Make sure that you have java on your system, after installing java you need to set JAVA_HOME .

Step 3 :

Now Start OpenDayLight.



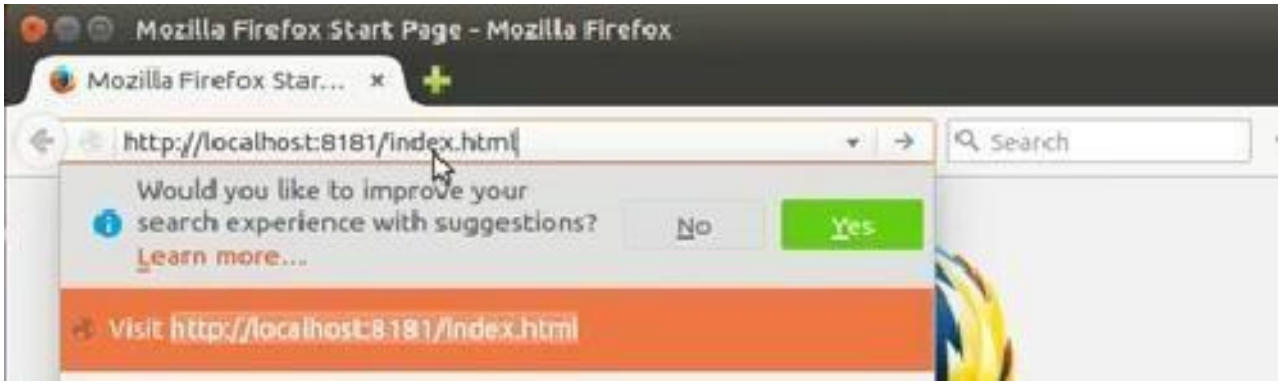
Step 4 :

Next you need to install Features , write the following command :

feature:install odl-restconf odl-l2switch-switch odl-mdsal-apidocs odl-dlux-all

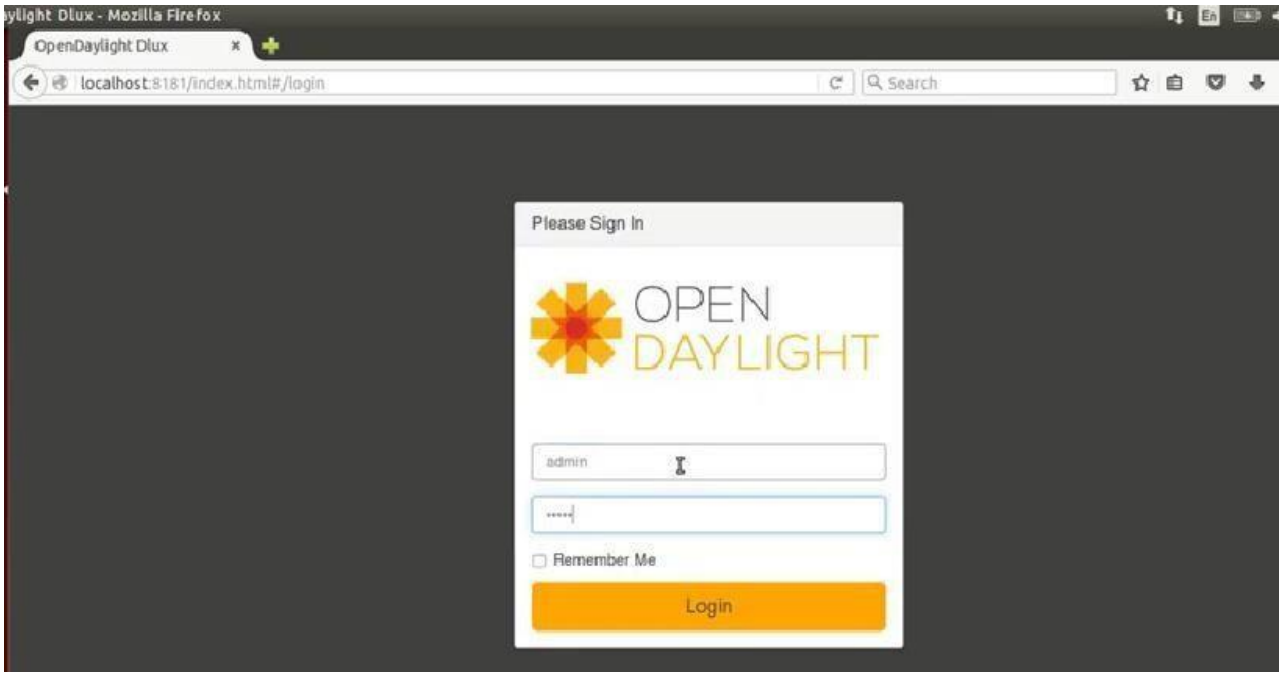
Step 5:

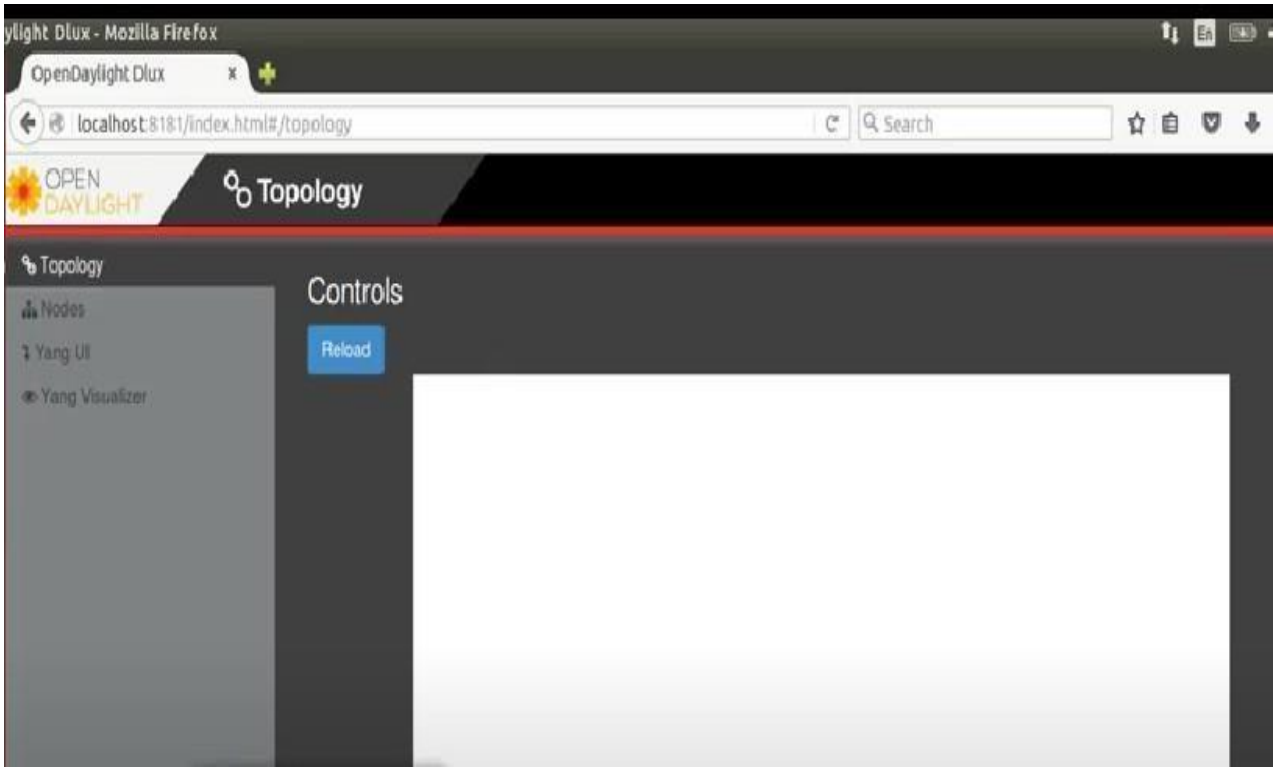
Now we will see whether it is running or not, open browser and search http://localhost:8181/index.html.



Step 6 :

Give Username and Password as admin.





OpenDayLight is working properly in browser.

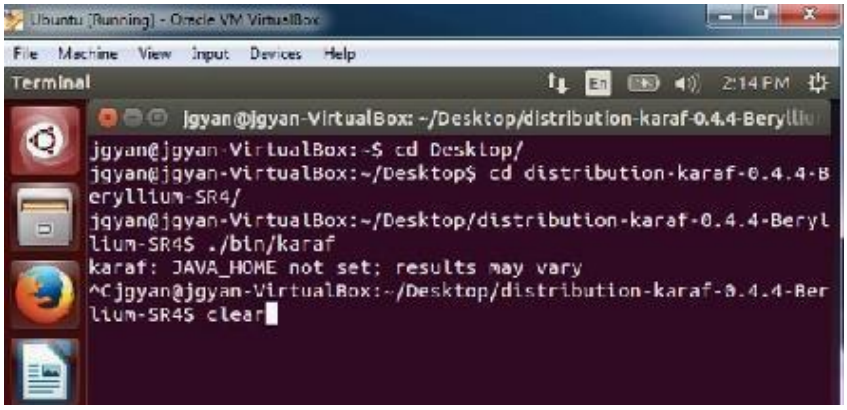
Part 2 :

Step 1:

I've installed two VMs in VirtualBox: one contains Mininet, and the other has Ubuntu with OpenDaylight installed.

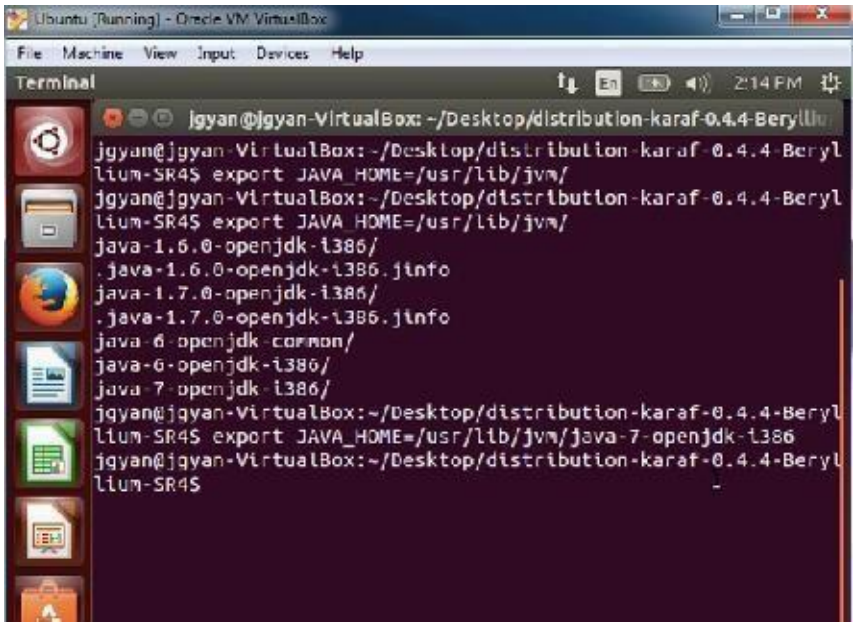
Step 2:

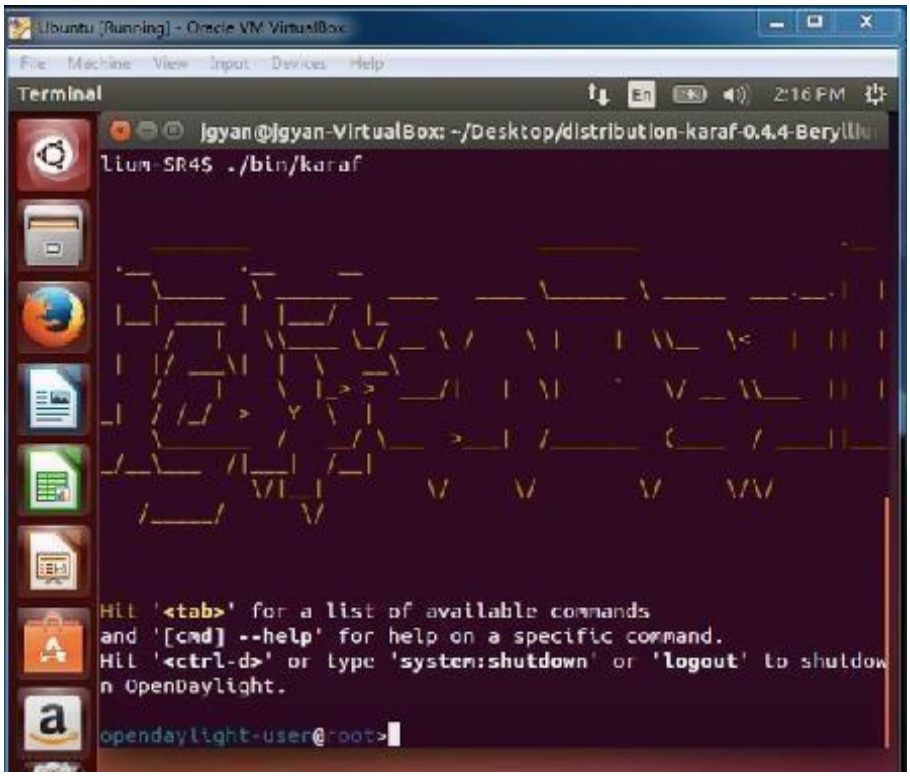
Firstly, start the Opendaylight Controller



Step 3:

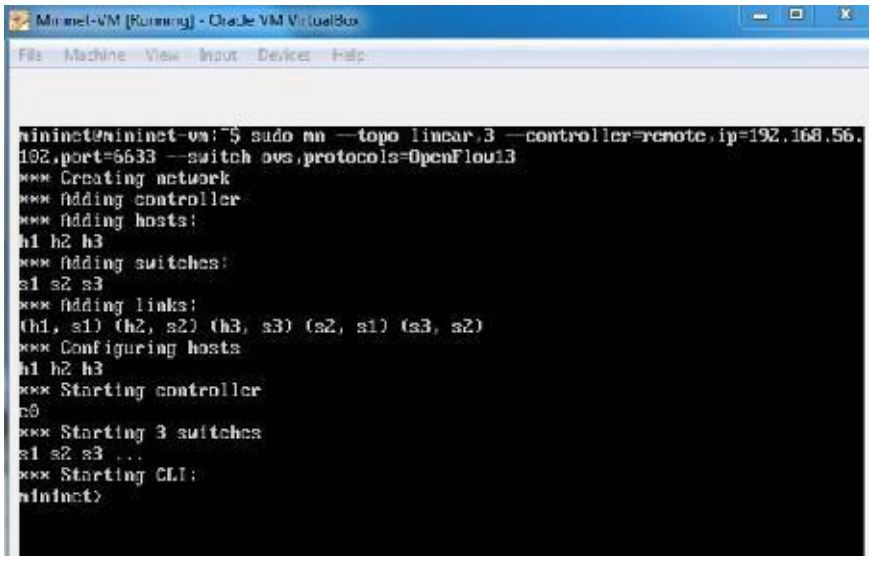
Now we have to set the JAVA HOME



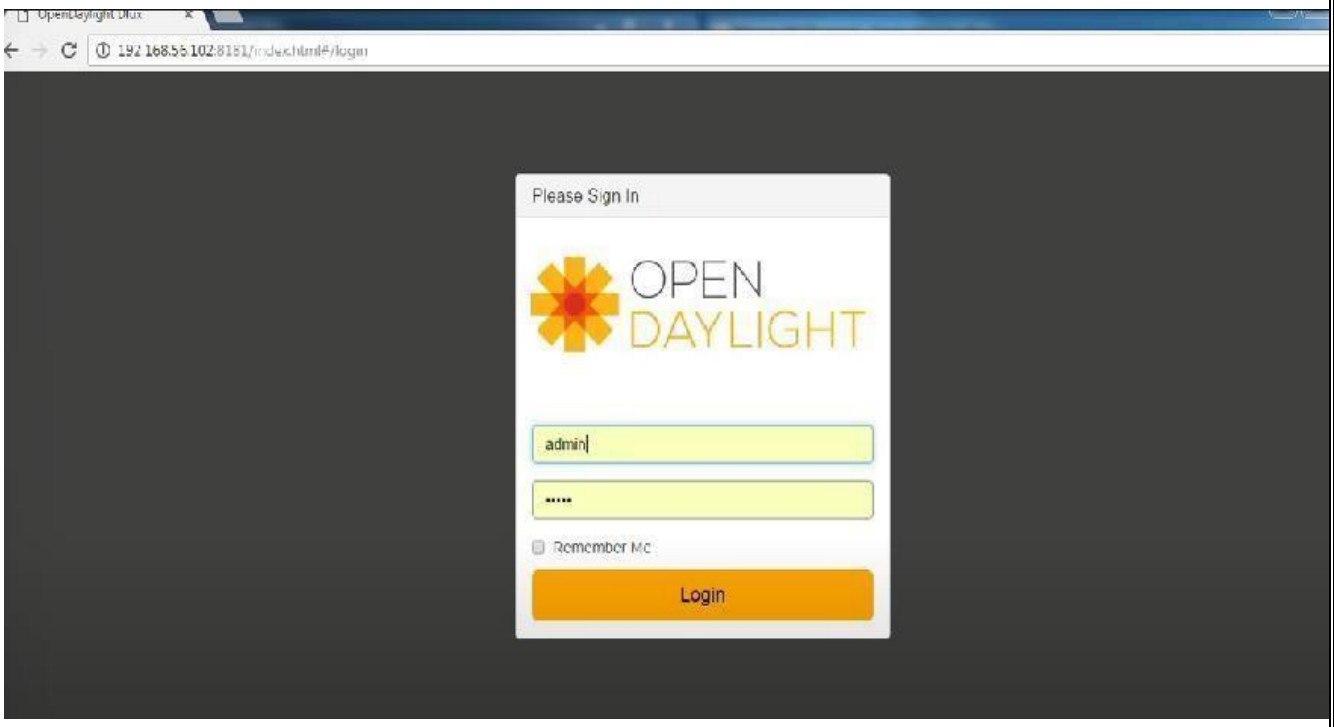


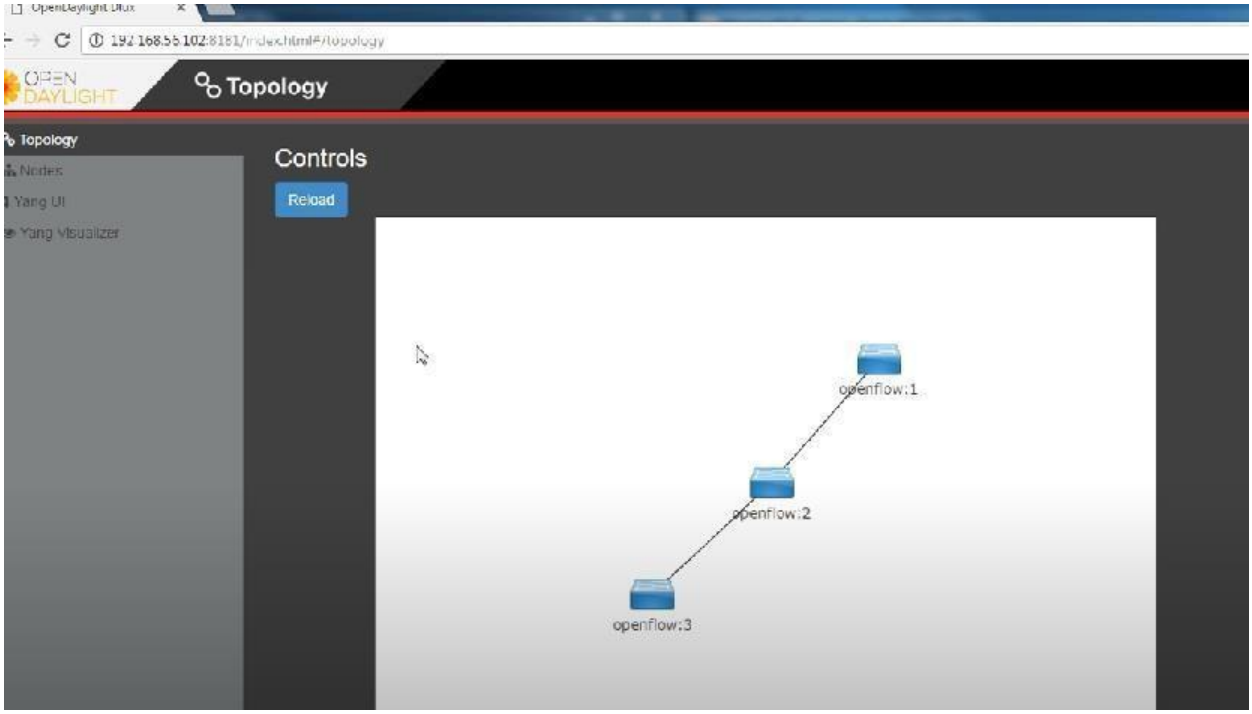
Step 4 :

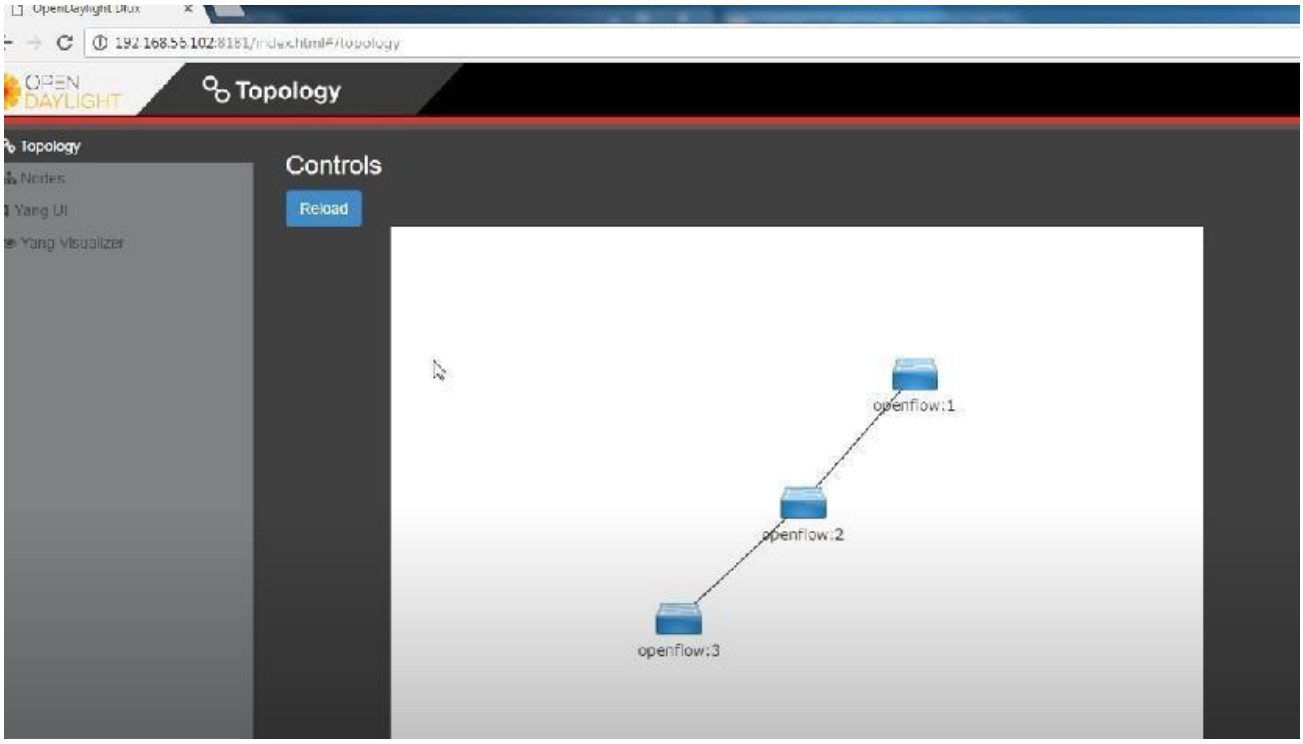
Creating the topology in Mininet.

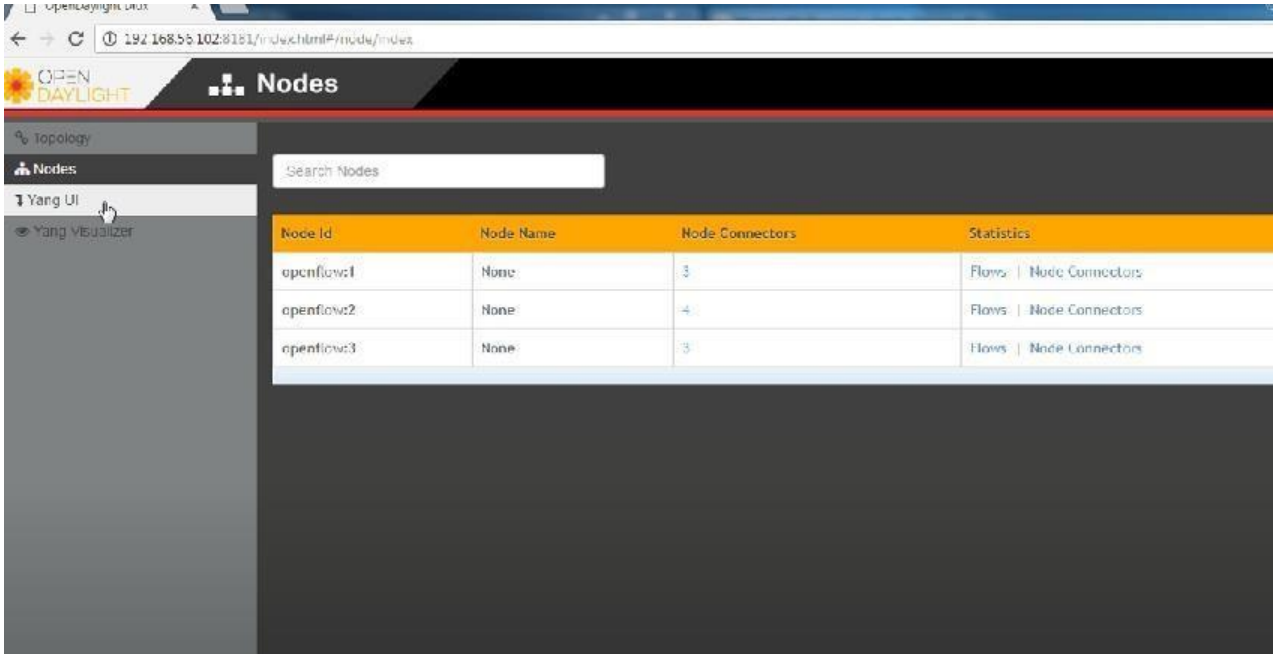


Now we can access Opendaylight web UI.


Please Sign In
OPEN
DAYLIGHT
admin
Remember Me
Login


Topology
Topology
Nodes
Yang UI
Yang Visualizer
Controls
Reload
openflow:1
openflow:2
openflow:3


Topology
Topology
Nodes
Yang UI
Yang Visualizer
Controls
Reload
openflow:1
openflow:2
openflow:3


Nodes
Topology
Nodes
Yang UI
Yang Visualizer
Search Nodes
Node Id
Node Name
Node Connectors
Statistics
openflow:1
None
3
Flows | Node Connectors
openflow:2
None
4
Flows | Node Connectors
openflow:3
None
3
Flows | Node Connectors

Practical 11

Simulating OpenFlow using MININET.

Simulating OpenFlow using Mininet is a fascinating practical task that provides insights into Software-Defined Networking (SDN) and its application in network management. Mininet is a popular network emulator that allows you to create virtual network topologies and test OpenFlow and other SDN protocols.

Objectives:

a . Simulate a network topology using Mininet.

b . Understand OpenFlow and its role in SDN.

c . Configure an OpenFlow controller.

Tools Required:

a .GNS3

b . Mininet VM or Mininet in Docker

c. OpenFlow Controller

Step 1 :

Download Mininet from the official website mininet.org and download the Mininet VM Image**,** you can also use VirtualBox, VMware ESXi,VMware Workstation,etc.

Step 2 :

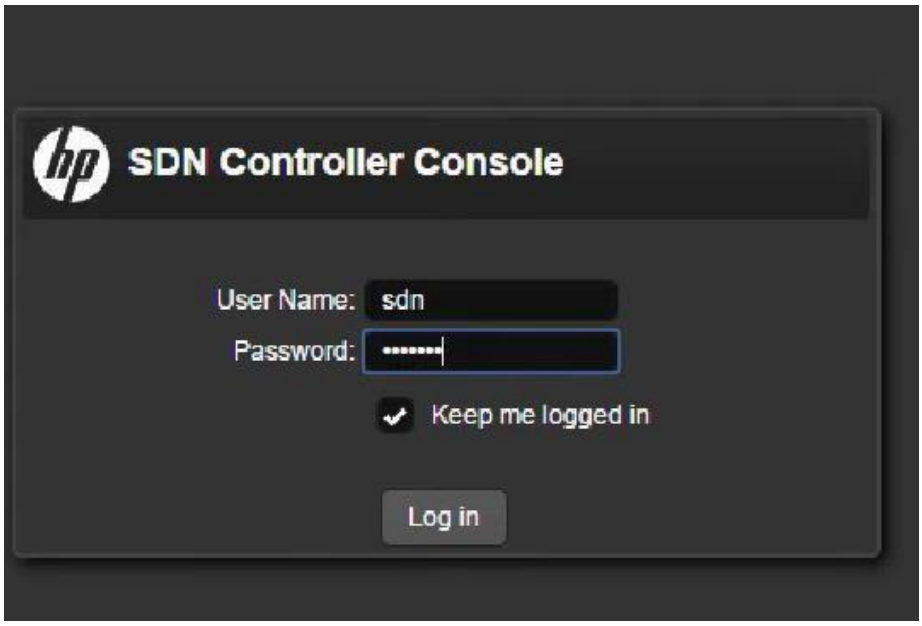
Creating Network Topology using Mininet.

Type ‘net’ to see the Connections.

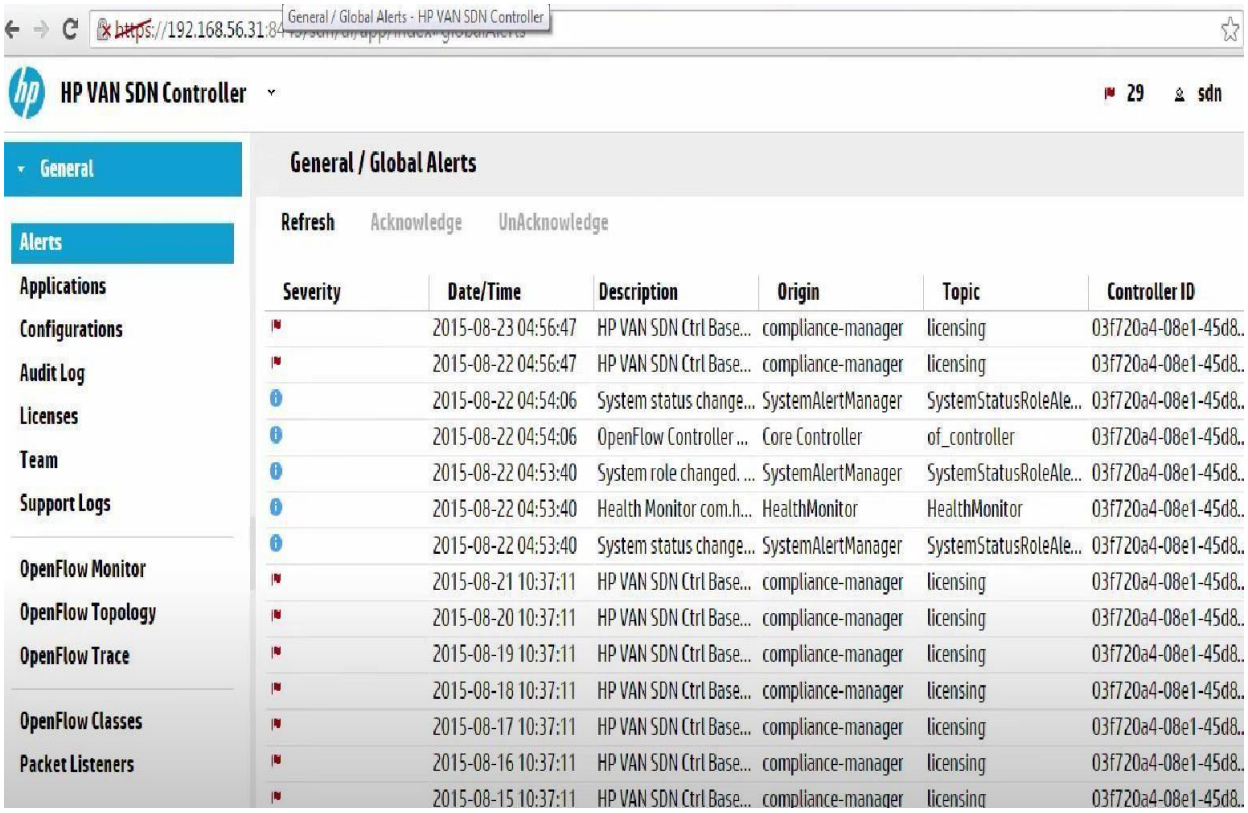
```
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0
c0
mininet>
```

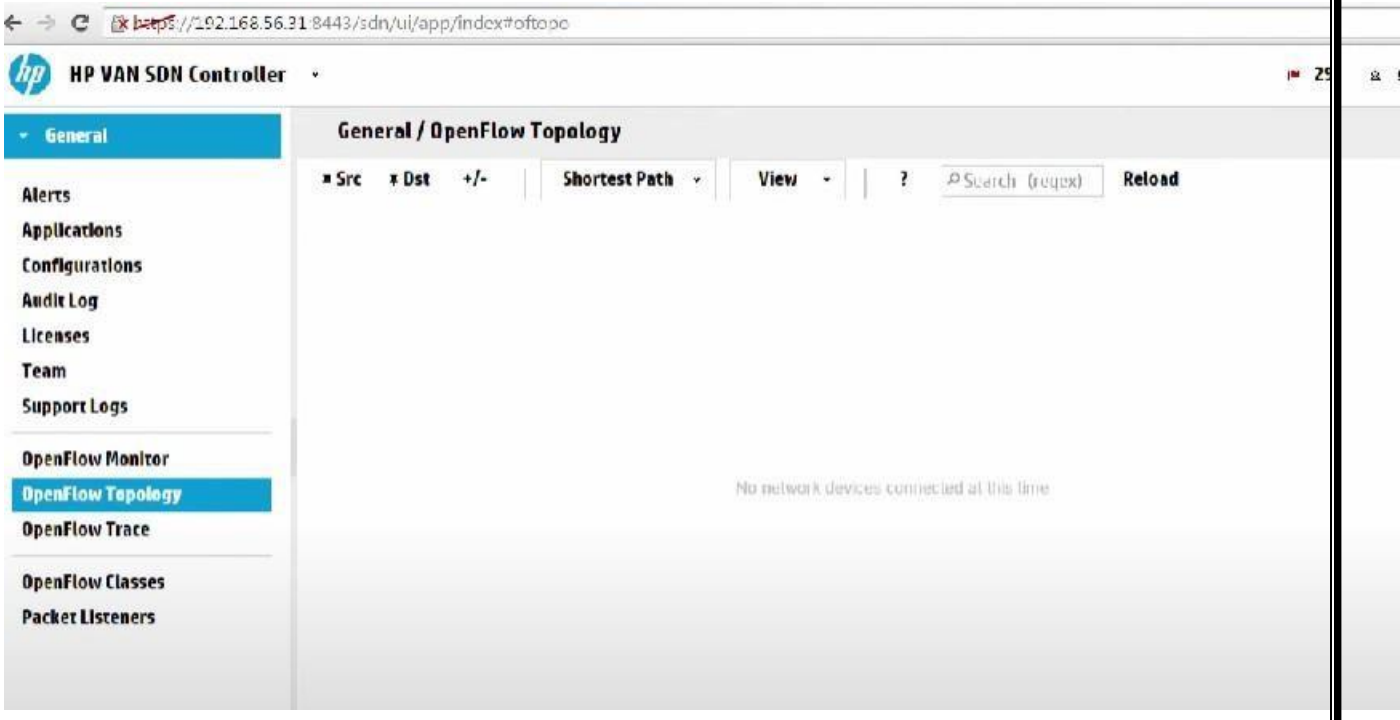
Step 3 :

Using SDN controller to to see the Topology in the Graphical User Interface.



You will be directed here:





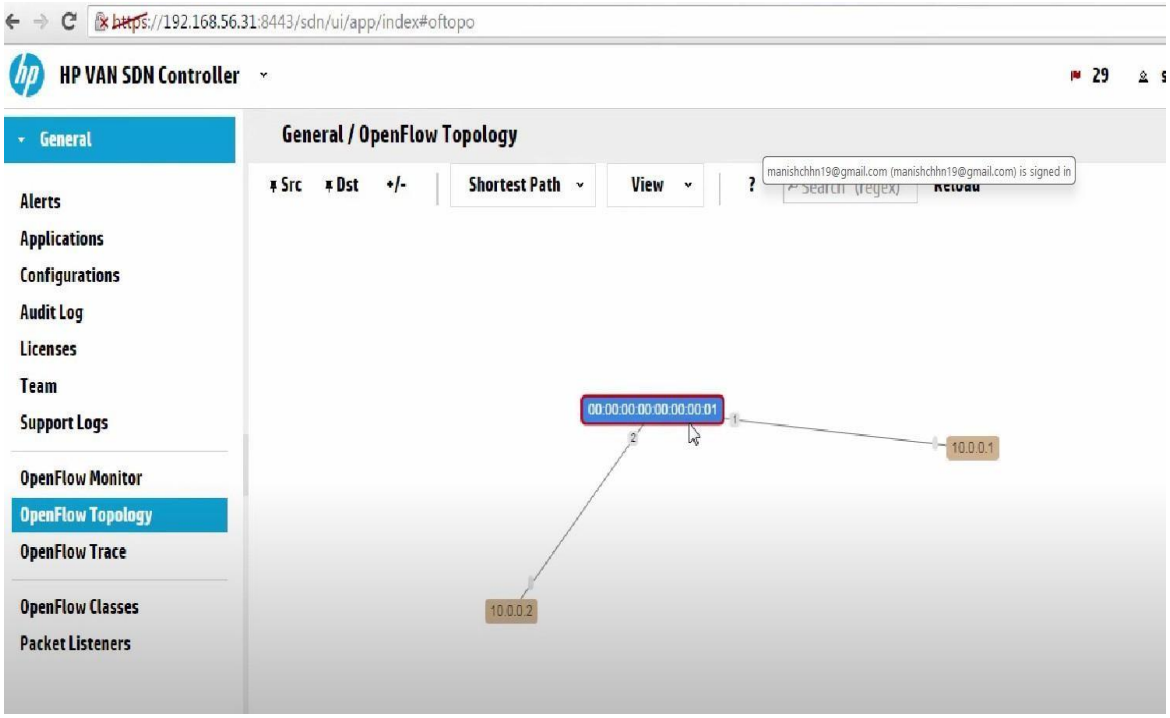
When you open OpenFlow Topology, you can see no network devices are connected at this time.

Step 4 :

Using Mininet to connect through the SDN controller OpenFlow Topology.

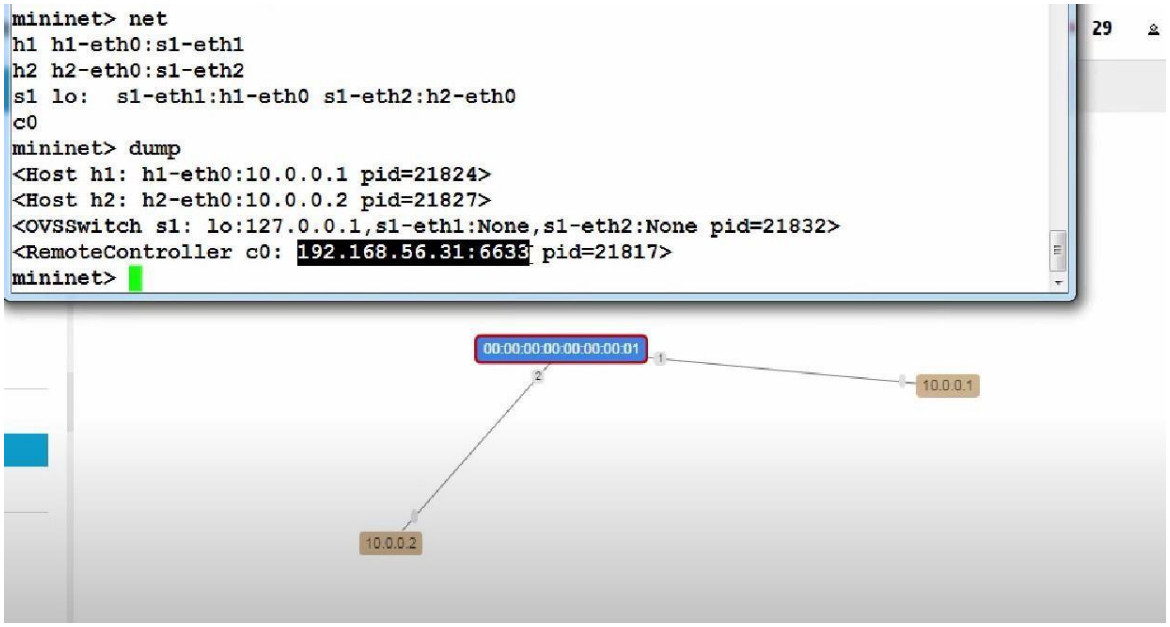
```
mininet@mininet-vm:~$ sudo mn --controller=remote,ip=192.168.56.31
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet>
```

Now open OpenFlow Topology, you will be able to see the connected devices.



Step 5 :

You can also check the network through Mininet



Step 6 :

You can see the Networks information in the OpenFlow Monitor.

Smt. Durgadevi Sharma Charitable Trust's

Chandrabhan Sharma College

of Arts, Commerce & Science

(Hindi Linguistic Minority Institution)

(Affiliated to the University of Mumbai)

NAAC Re-Accredited 'A' Grade (CGPA 3.10)

MASTER OF SCIENCE (INFORMATION TECHNOLOGY)

Subject Name : Computer Vision

Name : Yash Chandrakant Saundalkar

Seat No : 1312229

Teaching Faculty : Miss. Sailaja Tiwari

DEPARTMENT OF INFORMATION TECHNOLOGY

CHANDRABHAN SHARMA COLLEGE OF ARTS,

COMMERCE & SCIENCE

NAAC RE-ACCREDITED 'A' Grade (CGPA 3.10)

(Affiliated to the University of Mumbai)

(Autonomous)

MUMBAI, 400076

MAHARASHTRA

2024-2025

Smt. Durgadevi Sharma Charitable Trust's

Chandrabhan Sharma College

of Arts, Commerce & Science

(Hindi Linguistic Minority Institution)

(Affiliated to the University of Mumbai)

NAAC Re-Accredited 'A' Grade (CGPA 3.10)

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that Mr./~~Miss~~ **Yash Chandrakant Saundalkar** having Exam Seat No. **1312229** of M.Sc.IT **(Semester II)** has completed the Practical work in the subject of "**Computer Vision**" during the **Academic year 2024-25** under the guidance of **Miss. Sailaja Tiwari** being the Partial requirement for The fulfilment of the curriculum of Degree of Master of Science in Information Technology, University of Mumbai.

Internal Examiner **Co-ordinator**

Date: **College Seal**

External Examiner

INDEX

Practical Units.	Name of Practical	Date	Sign
1	A) Perform Geometric transformations B) Perform Image Stitching C) Perform Camera Calibration		
2	A) Face detection B) Object detection C) Pedestrian detection D) Face recognition D) Construct 3D model from images E) Implement object detection and tracking from video		
3	A) Perform Feature extraction using RANSAC B) Perform Colorization		
4	A) Perform Text detection and recognition B) Perform Image matting and Composting		

Practical 1

(A) Perform Geometric Transformations

```
import numpy as np
import cv2
def translate(image, x, y):
  M = np.float32([[1, 0, x], [0, 1, y]])
  translated_image = cv2.warpAffine(image, M, (image.shape[1], image.shape[0]))
  return translated_image
def rotate(image, angle, center=None, scale=1.0):
  (h, w) = image.shape[:2]
  if center is None:
    center = (w // 2, h // 2)
  M = cv2.getRotationMatrix2D(center, angle, scale)
  rotated_image = cv2.warpAffine(image, M, (w, h))
  return rotated_image
def scale(image, scale_x, scale_y):
  scaled_image = cv2.resize(image, None, fx=scale_x, fy=scale_y)
  return scaled_image
def shear(image, shear_x, shear_y):
  M = np.float32([[1, shear_x, 0], [shear_y, 1, 0]])
  sheared_image = cv2.warpAffine(image, M, (image.shape[1], image.shape[0]))
  return sheared_image
image = cv2.imread('img.jpg')
translated_image = translate(image, 100, 100)
rotated_image = rotate(image, 45)
```

scaled_image = scale(image, 0.5, 0.5)

sheared_image = shear(image, 0.2, 0.3)

cv2.imshow('Original',image)

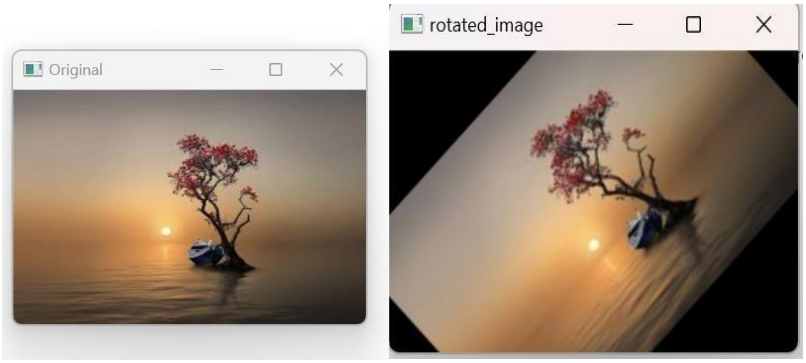
cv2.imshow('translated_image',translated_image)

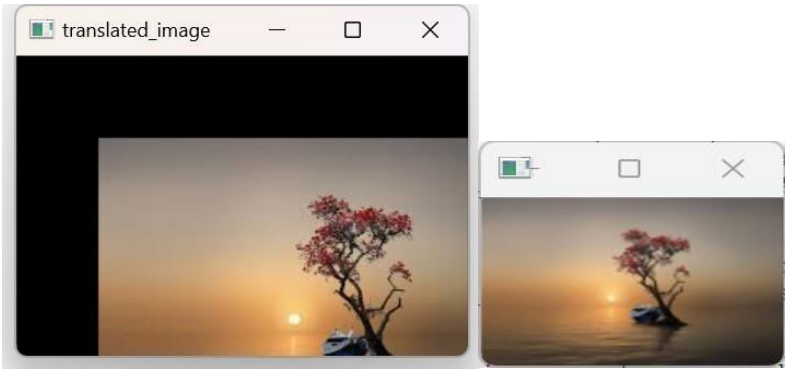
cv2.imshow('rotated_image',rotated_image)

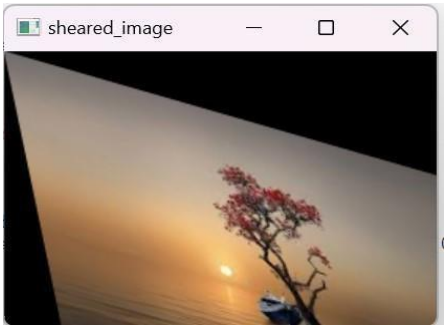
cv2.imshow('scaled_image',scaled_image)

cv2.imshow('sheared_image',sheared_image)

Output:







(B) Perform Image Stitching

```
from imutils import paths
import numpy as np
import argparse
import imutils
import cv2
imagePaths = sorted(list(paths.list_images('p2')))  #p2 is folder containing images
images=[]
for imagePath in imagePaths:
        image = cv2.imread(imagePath)
        images.append(image)
print("[INFO] stitching images...")
stitcher = cv2.createStitcher() if imutils.is_cv3() else cv2.Stitcher_create()
(status, stitched) = stitcher.stitch(images)
if status == 0:
        # write the output stitched image to disk
        cv2.imwrite("p2output.jpg", stitched)
        # display the output stitched image to our screen
        cv2.imshow("Stitched", stitched)
        cv2.waitKey(0)
else:
        print("[INFO] image stitching failed ({})".format(status))
```

Output:

```
[INFO] stitching images…
```

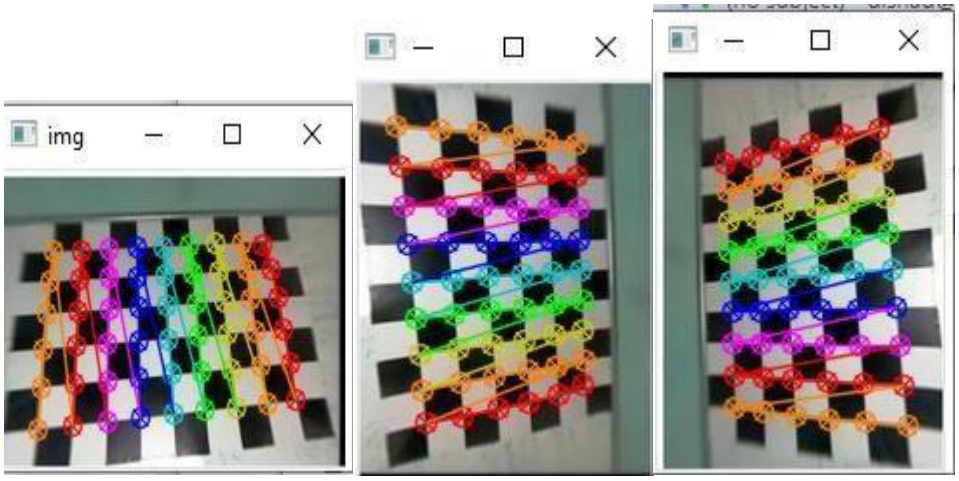

Stitched

(C) Perform Camera Caliberation

```
import cv2
import numpy as np
import os
import glob
chb = (6, 9)
criteria = (cv2.TERM_CRITERIA_EPS +
cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001)
threedpoints = []
twodpoints = []
objectp3d = np.zeros((1, chb[0] * chb[1], 3), np.float32)
objectp3d[0, :, :2] = np.mgrid[0:chb[0], 0:chb[1]].T.reshape(-1, 2)
prev_img_shape = None
images = glob.glob('*.jpg')
for filename in images:
    image = cv2.imread(filename)
    grayColor = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    ret, corners = cv2.findChessboardCorners(grayColor, chb,
                                             cv2.CALIB_CB_ADAPTIVE_THRESH
                                             + cv2.CALIB_CB_FAST_CHECK +
                                             cv2.CALIB_CB_NORMALIZE_IMAGE)
    if ret == True:
        threedpoints.append(objectp3d)
        corners2 = cv2.cornerSubPix(
            grayColor, corners, (11, 11), (-1, -1), criteria)
        twodpoints.append(corners2)
```

```
                image = cv2.drawChessboardCorners(image, chb, corners2, ret)
        cv2.imshow('img', image)
        cv2.waitKey(0)
cv2.destroyAllWindows()
h, w = image.shape[:2]
ret, matrix, distortion, r_vecs, t_vecs = cv2.calibrateCamera(
        threedpoints, twodpoints, grayColor.shape[::-1], None, None)
print(" Camera matrix:")
print(matrix)
print("\n Distortion coefficient:")
print(distortion)
print("\n Rotation Vectors:")
print(r_vecs)
print("\n Translation Vectors:")
print(t_vecs)
```

Output:




```
Camera matrix:
[[40.8618761   0.         83.80597971]
 [ 0.         41.29134045 94.37718569]
```

[0. 0. 1.]]

Distortion coefficient:

[[0.00051989 -0.00105068 -0.00512204 -0.00037702 0.00015534]]

Rotation Vectors:

(array([[-0.09670096],
[0.06446519],
[1.50927077]]), array([[-0.15910316],
[-0.0178901],
[3.07539003]]), array([[-0.03866621],
[0.09626499],
[-0.07417175]]))

Translation Vectors:

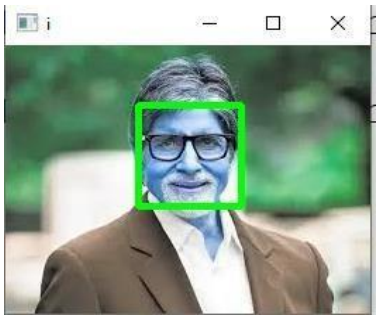
(array([[4.65717678],
[-3.68545603],
[2.75350073]]), array([[2.34304791],
[4.05578132],
[2.76539468]]), array([[-3.12854143],
[-3.38972333],
[2.83149017]]))

Practical 2

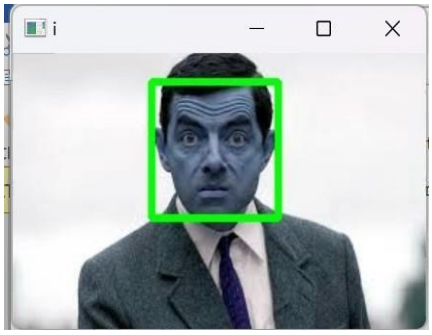
(A) Face Detection

```
import cv2
img = cv2.imread('input1.jpg')
gray_image = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
face_classifier = cv2.CascadeClassifier(
    cv2.data.haarcascades + "haarcascade_frontalface_default.xml")
face = face_classifier.detectMultiScale(
    gray_image, scaleFactor=1.1, minNeighbors=5, minSize=(40, 40))
for (x, y, w, h) in face:
    cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 4)
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
cv2.imshow('i',img_rgb)
```

Output:



Changing another image : img = cv2.imread('input3.jpg')



(B) Object detection

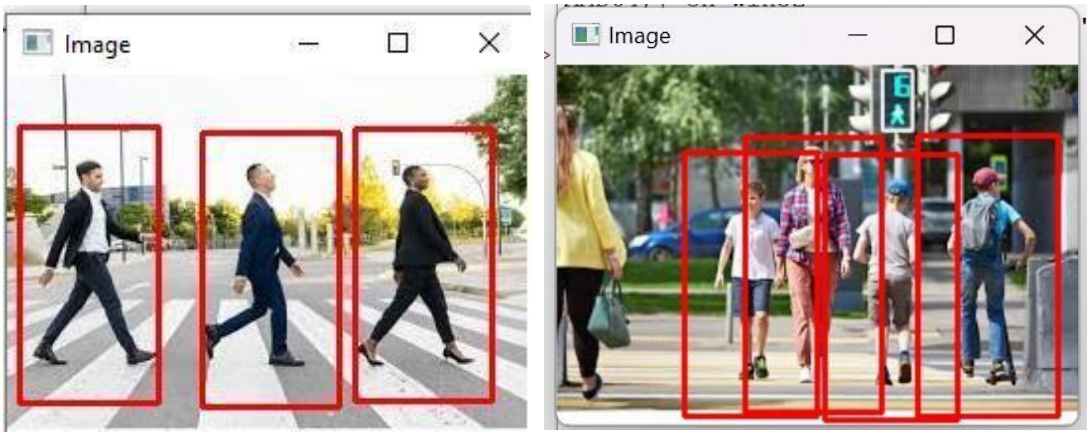
```
import cv2
# Opening image
img = cv2.imread("stop.png")
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
stop_data = cv2.CascadeClassifier('stop_data.xml')
found = stop_data.detectMultiScale(img_gray,minSize =(20, 20))
amount_found = len(found)
if amount_found != 0:
    for (x, y, width, height) in found:
        cv2.rectangle(img_rgb, (x, y),(x + height, y + width),(0, 255, 0), 5)
cv2.imshow('imgobject',img_rgb)
```

Output:

(C) Pedestrain Detection

```
import cv2
import imutils
hog = cv2.HOGDescriptor()
hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())
image = cv2.imread('pedestrian1.png')
image = imutils.resize(image,width=min(400, image.shape[1]))
(regions, _) = hog.detectMultiScale(image,winStride=(4, 4),padding=(4,
4),scale=1.05)
for (x, y, w, h) in regions:
        cv2.rectangle(image,(x, y),(x + w, y + h),(0, 0, 255), 2)
cv2.imshow("Image", image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output :



(D) Face Recognition

Write command in Command Prompt:

pip install dlib-19.24.99-cp312-cp312-win_amd64.whl (need the file to be downloaded)

pip install cmake

pip install face_recognition

pip install setuptools

Coding:

```
import cv2
import face_recognition
imgmain = face_recognition.load_image_file('random11.jpg')
imgmain = cv2.cvtColor(imgmain, cv2.COLOR_BGR2RGB)
imgTest = face_recognition.load_image_file('random12.jpg')
imgTest = cv2.cvtColor(imgTest, cv2.COLOR_BGR2RGB)
faceLoc = face_recognition.face_locations(imgmain)[0]
encodeElon = face_recognition.face_encodings(imgmain)[0]
cv2.rectangle(imgmain,(faceLoc[3],faceLoc[0]),(faceLoc[1],faceLoc[2]),(255, 0, 255), 2)
faceLocTest = face_recognition.face_locations(imgTest)[0]
encodeTest = face_recognition.face_encodings(imgTest)[0]
cv2.rectangle(imgTest, (faceLocTest[3],faceLocTest[0]),(faceLocTest[1],faceLocTest[2]),
        (255, 0, 255), 2)
results = face_recognition.compare_faces([encodeElon],encodeTest)
faceDis = face_recognition.face_distance([encodeElon],encodeTest)
print(results, faceDis)
```

```
cv2.putText(imgTest, f'{results} {round(faceDis[0], 2)}', (20, 20),
            cv2.FONT_HERSHEY_COMPLEX, 1, (0, 0, 255), 2)
cv2.imshow('Main Image', imgmain)
cv2.imshow('Test Image', imgTest)
cv2.waitKey(0)
```

Output:

[False] [0.72084828]

Changes

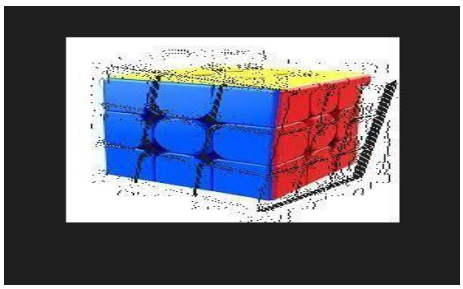
```
imgmain = face_recognition.load_image_file('random11.jpg')
imgmain = cv2.cvtColor(imgmain, cv2.COLOR_BGR2RGB)
imgTest = face_recognition.load_image_file('random2.jpg')
imgTest = cv2.cvtColor(imgTest, cv2.COLOR_BGR2RGB)
```

[True] [0.41407696]

(E) Construct 3D model from images

```
from PIL import Image
import numpy as np
def shift_image(img, depth_img, shift_amount=10):
    img = img.convert("RGBA")
    data = np.array(img)
    depth_img = depth_img.convert("L")
    depth_data = np.array(depth_img)
    deltas = ((depth_data / 255.0) * float(shift_amount)).astype(int)
    shifted_data = np.zeros_like(data)
    height, width, _ = data.shape
    for y, row in enumerate(deltas):
        for x, dx in enumerate(row):
            if x + dx < width and x + dx >= 0:
                shifted_data[y, x + dx] = data[y, x]
    shifted_image = Image.fromarray(shifted_data.astype(np.uint8))
    return shifted_image
img = Image.open("cube1.jpeg")
depth_img = Image.open("cube2.jpeg")
shifted_img = shift_image(img, depth_img, shift_amount=10)
shifted_img.show()
```

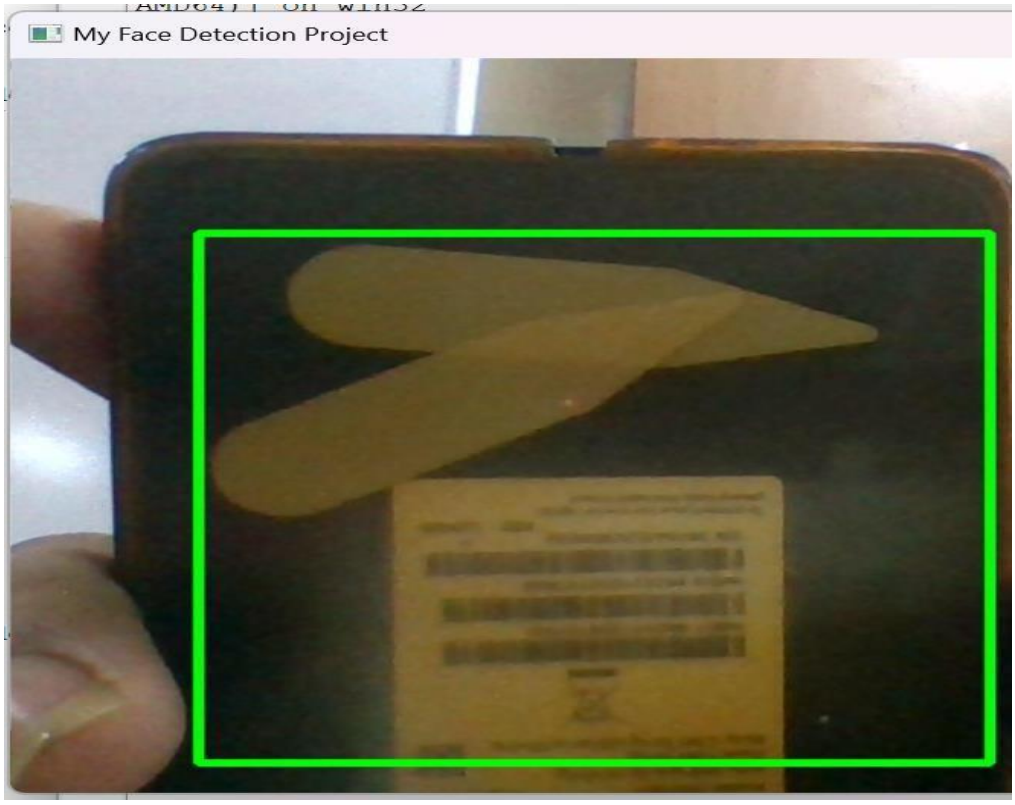
Output:



(F) Object detection from video

```
import cv2
face_classifier = cv2.CascadeClassifier(
    cv2.data.haarcascades + "haarcascade_frontalface_default.xml")
video_capture = cv2.VideoCapture('a.mp4')
def detect_bounding_box(vid):
    gray_image = cv2.cvtColor(vid, cv2.COLOR_BGR2GRAY)
    faces = face_classifier.detectMultiScale(gray_image, 1.1, 5, minSize=(40,
40))
    for (x, y, w, h) in faces:
        cv2.rectangle(vid, (x, y), (x+w+100, y+h+100), (0,0,255), 4)
    return faces
while True:
    result, video_frame = video_capture.read()
    if result is False:
        break
    faces = detect_bounding_box(video_frame)
    cv2.imshow("My Object Detection Project", video_frame)
    if cv2.waitKey(1) & 0xFF == ord("q"):
        break
video_capture.release()
cv2.destroyAllWindows()
```

Output:



Practical 3

(A) RANSAC Implementation

linearleastsqaure.py

```
import numpy as np
class LinearLeastSqaureModel:
    def fit(self, A, Y):
        A_transpose = A.transpose()
        ATA = A_transpose.dot(A)
        ATY = A_transpose.dot(Y)
        model = (np.linalg.inv(ATA)).dot(ATY)
        return model
```

ransac.py:

```
import numpy as np
import math
class RansacModel:
    def __init__(self, curve_fitting_model):
        self.curve_fitting_model = curve_fitting_model
    def fit(self, A, Y, num_sample, threshold):
        num_iterations = math.inf
        iterations_done = 0
        num_sample = 3
        max_inlier_count = 0
        best_model = None
        prob_outlier = 0.5
        desired_prob = 0.95
```

```
        total_data = np.column_stack((A, Y)) ## [ A | Y]
        data_size = len(total_data)
        while num_iterations > iterations_done:
            np.random.shuffle(total_data)
            sample_data = total_data[:num_sample, :]
            estimated_model = self.curve_fitting_model.fit(sample_data[:,:-1],
                                                sample_data[:, -1:])
            y_cap = A.dot(estimated_model)
            err = np.abs(Y - y_cap.T)
            inlier_count = np.count_nonzero(err < threshold)
            if inlier_count > max_inlier_count:
                max_inlier_count = inlier_count
                best_model = estimated_model
            prob_outlier = 1 - inlier_count/data_size
            print('# inliers:', inlier_count)
            print('# prob_outlier:', prob_outlier)
            num_iterations = math.log(1 - desired_prob)/math.log(1 - (1 -
prob_outlier)**num_sample)
            iterations_done = iterations_done + 1
            print('# s:', iterations_done)
            print('# n:', num_iterations)
            print('# max_inlier_count: ', max_inlier_count)
        return best_model
```

Modelfitting.py:

Ransac Implementation Coding:

```
import numpy as np
import math
```

```
import pandas as pd
from matplotlib import pyplot as plt
from ransac import RansacModel
from linearleastsquare import LinearLeastSqaureModel
def fit_curve(data):
    x_values = np.array(data['x'])
    y_values = np.array(data['y'])
    x_sq = np.power(x_values, 2)
    A = np.stack((x_sq, x_values, np.ones((len(x_values)), dtype = int)), axis = 1)
    threshold = np.std(y_values)/2

    linear_ls_model = LinearLeastSqaureModel()
    linear_ls_model_estimate = linear_ls_model.fit(A, y_values)
    linear_model_y = A.dot(linear_ls_model_estimate)
    ransac_model = RansacModel(linear_ls_model)
    ransac_model_estimate = ransac_model.fit(A, y_values, 3, threshold)
    ransac_model_y = A.dot(ransac_model_estimate)
    return linear_model_y, ransac_model_y
if __name___== '__main__':
    df1 = pd.read_csv('data_1.csv')
    df2 = pd.read_csv('data_2.csv')
    ls_model_y1, ransac_model_y1 = fit_curve(df1)
    ls_model_y2, ransac_model_y2 = fit_curve(df2)
    fig, (ax1, ax2) = plt.subplots(1, 2)
    ax1.set_title('Dataset-1')
    ax1.scatter(df1['x'], df1['y'], marker='o', color = (0,1,0), label='data points')
    ax1.plot(df1['x'], ls_model_y1, color = 'red', label='Least sqaure model')
```

```
    ax1.plot(df1['x'], ransac_model_y1, color = 'blue', label='Ransac model')
    ax1.set(xlabel='x-axis', ylabel='y-axis')
    ax1.legend()
    ax2.set_title('Dataset-2')
    ax2.scatter(df2['x'], df2['y'], marker='o', color = (0,1,0), label='data points')
    ax2.plot(df2['x'], ls_model_y2, color = 'red', label='Least sqaure model')
    ax2.plot(df2['x'], ransac_model_y2, color = 'blue', label='Ransac model')
    ax2.set(xlabel='x-axis', ylabel='y-axis')
    ax2.legend()
    plt.show()
```

Output:

```
# inliers: 117
# prob_outlier: 0.532
# s: 1
# n: 27.700875935823316
# max_inlier_count:  117
# inliers: 175
# prob_outlier: 0.30000000000000004
# s: 2
# n: 7.131485905524426
# max_inlier_count:  175
# inliers: 76
# prob_outlier: 0.696
# s: 3
# n: 105.1257167640639
# max_inlier_count:  175
# inliers: 134
```

prob_outlier: 0.46399999999999997

s: 4

n: 17.91439390915469

max_inlier_count: 175

inliers: 187

prob_outlier: 0.252

s: 5

n: 5.525552495791251

max_inlier_count: 187

inliers: 68

prob_outlier: 0.728

s: 6

n: 147.36332180032895

max_inlier_count: 187

inliers: 64

prob_outlier: 0.744

s: 7

n: 177.05746802114584

max_inlier_count: 187

inliers: 87

prob_outlier: 0.652

s: 8

n: 69.57430607360442

max_inlier_count: 187

inliers: 72

prob_outlier: 0.712

s: 9

n: 123.90418270340892

max_inlier_count: 187

inliers: 145

prob_outlier: 0.42000000000000004

s: 10

n: 13.801901471212751

max_inlier_count: 187

inliers: 58

prob_outlier: 0.768

s: 11

n: 238.4038555094377

max_inlier_count: 187

inliers: 118

prob_outlier: 0.528

s: 12

n: 26.963389844145

max_inlier_count: 187

inliers: 63

prob_outlier: 0.748

s: 13

n: 185.69618036762273

max_inlier_count: 187

inliers: 168

prob_outlier: 0.32799999999999996

s: 14

n: 8.283822964377974

max_inlier_count: 187

inliers: 204

prob_outlier: 0.18400000000000005

s: 1

n: 3.821999420219228

max_inlier_count: 204

inliers: 115

prob_outlier: 0.54

s: 2

n: 29.25380161992817

max_inlier_count: 204

inliers: 143

prob_outlier: 0.42800000000000005

s: 3

n: 14.457625659547233

max_inlier_count: 204

inliers: 91

prob_outlier: 0.636

s: 4

n: 60.60513156707005

max_inlier_count: 204

inliers: 120

prob_outlier: 0.52

s: 5

n: 25.561028720085947

max_inlier_count: 204

inliers: 184

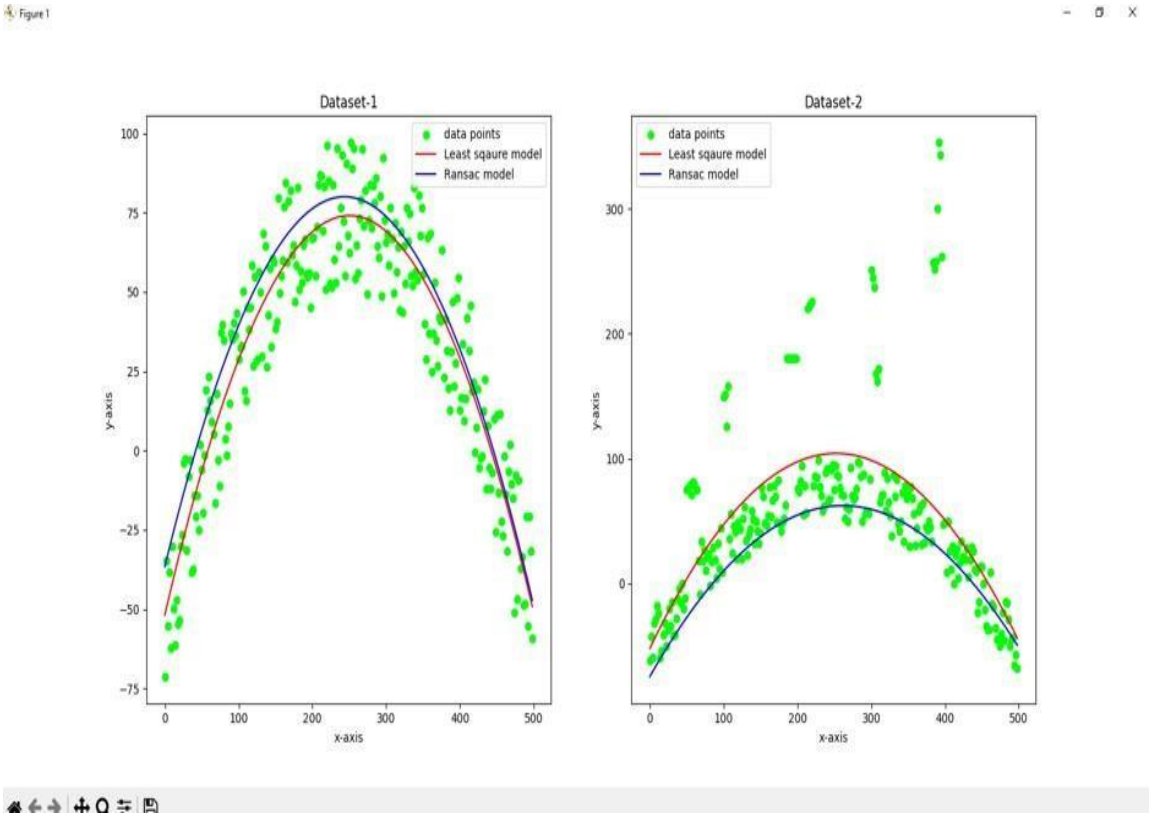
prob_outlier: 0.264

s: 6

n: 5.889670195389181

max_inlier_count: 204

Graph:



(B) Colorization

```
import argparse
import matplotlib.pyplot as plt
from colorizers import * #local package
c1 = eccv16(pretrained=True).eval()
c2 = siggraph17(pretrained=True).eval()
img = load_img('testbw1.jpg')
(orig,rs) = preprocess_img(img, HW=(256,256))
img_bw = postprocess_tens(orig, torch.cat((0*orig,0*orig),dim=1))
img1 = postprocess_tens(orig, c1(rs).cpu())
img2 = postprocess_tens(orig, c2(rs).cpu())
plt.imsave('s1.png', img1)
plt.imsave('s2.png', img2)
plt.figure(figsize=(12,8))
plt.subplot(2,2,1)
plt.imshow(img)
plt.title('Original Image')
plt.axis('off')
plt.subplot(2,2,2)
plt.imshow(img_bw)
plt.title('Input Black and White')
plt.axis('off')
plt.subplot(2,2,3)
plt.imshow(img1)
plt.title('Colored Image 1')
plt.axis('off')
plt.subplot(2,2,4)
```

```
plt.imshow(img2)
plt.title('Colored Image 2')
plt.axis('off')
plt.show()
```

Output:

Practical 4

(A) Text Detection And Recognition from Image

```
import cv2
import pytesseract
pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-OCR\tesseract.exe'
img = cv2.imread("imgtest1.jpg")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
ret, thresh1 = cv2.threshold(gray, 0, 255, cv2.THRESH_OTSU | cv2.THRESH_BINARY_INV)
rect_kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (20, 20))
dilation = cv2.dilate(thresh1, rect_kernel, iterations = 1)
contours, hierarchy = cv2.findContours(dilation, cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_NONE)
im2 = img.copy()
for cnt in contours:
    x, y, w, h = cv2.boundingRect(cnt)
    rect = cv2.rectangle(im2, (x, y), (x + w, y + h), (0, 255, 0), 2)
    cropped = im2[y:y + h, x:x + w]
    text = pytesseract.image_to_string(cropped)
    print(text)
```

Output:

Output:

Test

Hello World

(B) Perform Image matting and Composting

```
from pymatting import *
import numpy as np
import cv2
scale = 1.0
image = load_image("4b1.png", "RGB", scale, "box")
trimap = load_image("4b2.png", "GRAY", scale, "nearest")
alpha = estimate_alpha_cf(image, trimap)
new_background = np.zeros(image.shape)
new_background[:, :] = [0.5, 0.5, 0.5]
foreground, background = estimate_foreground_ml(image, alpha,
return_background=True)
save_image("output1.png", alpha)
save_image("output2.png", foreground)
save_image("output3.png", background)
cutout = stack_images(foreground, alpha)
save_image("output4.png", cutout)
cutout = cv2.imread('output4.png', cv2.IMREAD_UNCHANGED)
new_background = cv2.imread('4b3.png')
foreground = cutout[:, :, :3]
am = cutout[:, :, 3] / 255.0
newbg_resized = cv2.resize(new_background, (foreground.shape[1],
foreground.shape[0]))
composite = np.zeros_like(foreground, dtype=np.uint8)
for c in range(3):
    composite[:, :, c]=foreground[:,:,c]*am+newbg_resized[:,:,c]*(1-am)
cv2.imwrite('composite_image.png', composite)
```

Output: