

# **PRACTICAL JOURNAL**

In

## **Big Data Analytics**

&

## **Modern Networking**

Submitted to

**Laxman Devram Sonawane College, Kalyan (W) 421301**

*in partial fulfilment for the award of the degree of*

**Master of Science in Information Technology**



(Affiliated to Mumbai University)

*Submitted by*

**Vrushabh Ravindra Pawar**

Under the guidance of

**Mrs. Snehal Borole (Paper I - Big Data Analytics)**

&

**Mrs. Pradnya Mahajan (Paper II - Modern Networking)**

Department of Information  
Technology Kalyan, Maharashtra

Academic Year 2023-24

# **PRACTICAL JOURNAL**

in

## **Big Data Analytics**

Submitted to

**Laxman Devram Sonawane College, Kalyan (W) 421301**

*in partial fulfilment for the award of the degree of*

**Master of Science in Information Technology**



(Affiliated to Mumbai University)

*Submitted by*

**Vrushabh Ravindra Pawar**

Under the guidance of

**Mrs. Snehal Borole**

Department of Information  
Technology Kalyan, Maharashtra

Academic Year 2023-24



The Kalyan Wholesale Merchants Education Society's  
**Laxman Devram Sonawane College,**  
**Kalyan (W) 421301**

**Department of Information Technology  
Masters of Science – Part II**

**Certificate**

This is to certify that **Mr.Vrushabh Ravindra Pawar**, Seat number \_\_\_\_\_, studying in Masters of Science in Information Technology Part I , Semester II has satisfactorily completed the practical of "**Big Data Analytics**" as prescribed by University of Mumbai, during the academic year 2023-24.

---

Subject In-charge

---

Coordinator In-charge

---

External Examiner

---

College Seal

# **BIG DATA ANALYTICS**

## **INDEX**

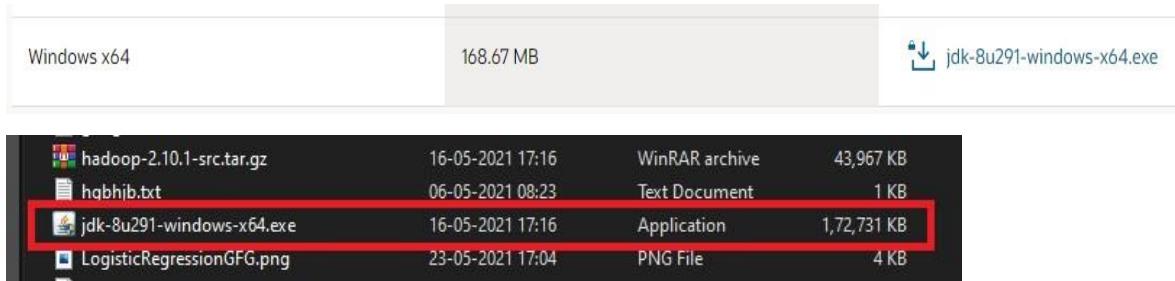
<u>Sr. No.</u>	<u>Practical Aim</u>	<u>Signature</u>
1.	<b>Install, configure and run Hadoop and HDFS</b>	
2.	<b>Implement Decision tree classification techniques</b>	
3.	<b>Classification using SVM</b>	
4.	<b>Implement an application that stores big data in Hbase / MongoDB and manipulate it using R / Python</b>	
5.	<b>Write Program Naive baye's theorem's</b>	
6.	<b>Write a Program showing implementation of Regression model.</b>	
7.	<b>Write a Program showing clustering.</b>	

# PRACTICAL NO : 1

## Aim: Install, configure and run Hadoop and HDFS Description:

Hadoop Installation.

Step 1: download java jdk first .the package size 168.67MB



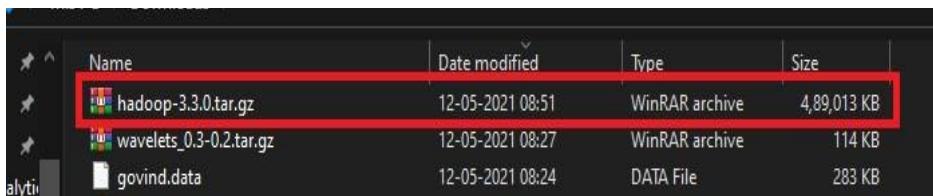
Step 2: download Hadoop binaries from the official website. The binary package size is about 342 MB.

### Download

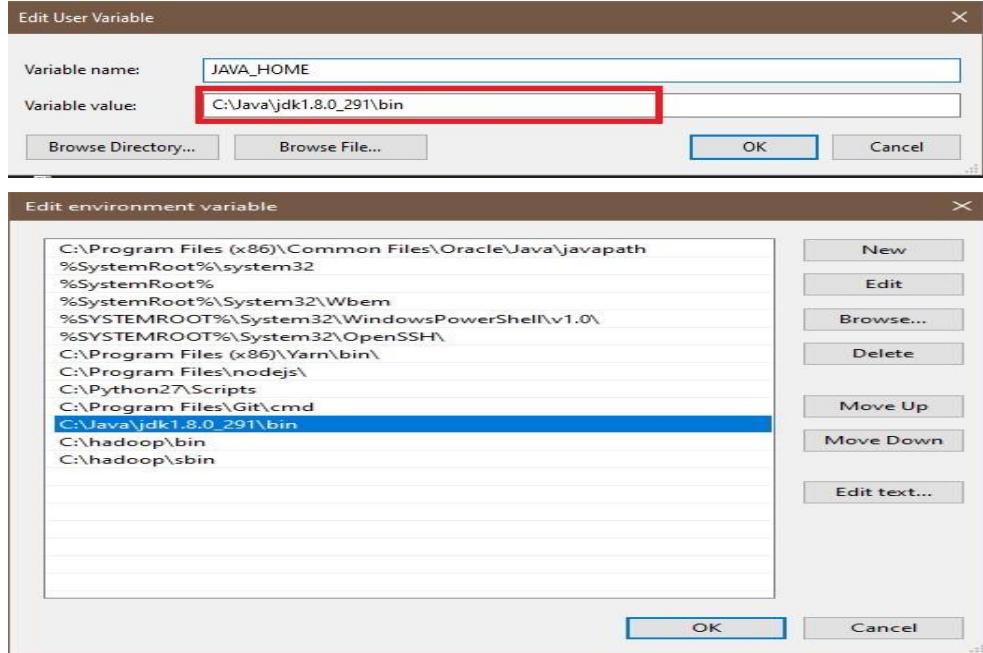
Hadoop is released as source code tarballs with corresponding binary tarballs for convenience. The downloads are distributed via mirror sites and should be checked for tampering using GPG or SHA-512.

Version	Release date	Source download	Binary download	Release notes
3.2.2	2021 Jan 9	source (checksum signature)	binary (checksum signature)	Announcement
2.10.1	2020 Sep 21	source (checksum signature)	binary (checksum signature)	Announcement
3.1.4	2020 Aug 3	source (checksum signature)	binary (checksum signature)	Announcement
3.3.0	2020 Jul 14	source (checksum signature)	binary (checksum signature) binary-aarch64 (checksum signature)	Announcement

Step 3: After finishing the file download, we should unpack the package using 7zip int two steps. First, we should extract the hadoop-3.2.1.tar.gz library, and then, we should unpack the extracted tar file:



Step 4: When the “Advanced system settings” dialog appears, go to the “Advanced” tab and click on the “Environment variables” button located on the bottom of the dialog.



## Step 5: Check the version of java

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19041.928]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hp>javac
Usage: javac <options> <source files>
where possible options include:
  -g                           Generate all debugging info
  -g:none                      Generate no debugging info
  -g:{lines,vars,source}        Generate only some debugging info
  -nowarn                      Generate no warnings
  -verbose                     Output messages about what the compiler is doing
  -deprecation                 Output source locations where deprecated APIs are used
  -classpath <path>            Specify where to find user class files and annotation process
  -cp <path>                   Specify where to find user class files and annotation process
  -sourcepath <path>           Specify where to find input source files
  -bootclasspath <path>        Override location of bootstrap class files
  -extdirs <dirs>              Override location of installed extensions
  -endorseddirs <dirs>         Override location of endorsed standards path
  -proc:{none,only}             Control whether annotation processing and/or compilation is o
  -processor <class1>[,<class2>,<class3>...] Names of the annotation processors to run; by
ss

C:\Users\hp>java -version
java version "1.8.0_291"
Java(TM) SE Runtime Environment (build 1.8.0_291-b10)
Java HotSpot(TM) 64-Bit Server VM (build 25.291-b10, mixed mode)
```

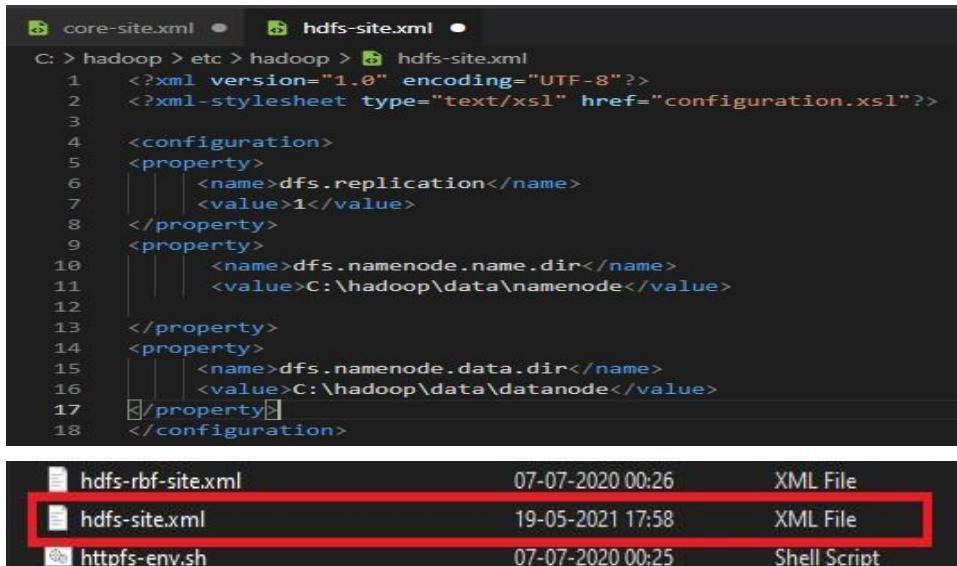
## Step 6: Configuration core-site.xml

container-executor.cfg	07-07-2020 01:03	CFG File
<b>core-site.xml</b>	19-05-2021 17:57	XML File
hadoop-env.cmd	19-05-2021 17:57	Windows Comma...

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

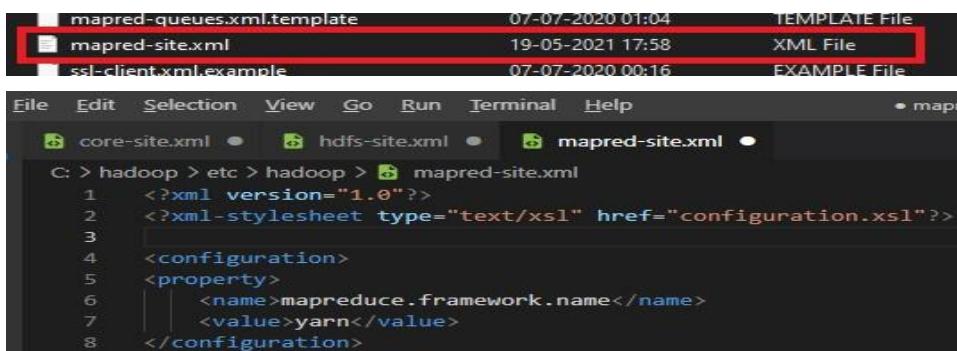
## Step 7: Configuration core-site.xml



```
C: > hadoop > etc > hadoop > core-site.xml
1   <?xml version="1.0" encoding="UTF-8"?>
2   <?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
3
4   <configuration>
5     <property>
6       |   <name>dfs.replication</name>
7       |   <value>1</value>
8     </property>
9     <property>
10    |   <name>dfs.namenode.name.dir</name>
11    |   <value>C:\hadoop\data\namenode</value>
12
13   </property>
14   <property>
15     |   <name>dfs.datanode.data.dir</name>
16     |   <value>C:\hadoop\data\datanode</value>
17   </property>
18 </configuration>
```

hdfs-rbf-site.xml	07-07-2020 00:26	XML File
hdfs-site.xml	19-05-2021 17:58	XML File
httpfs-env.sh	07-07-2020 00:25	Shell Script

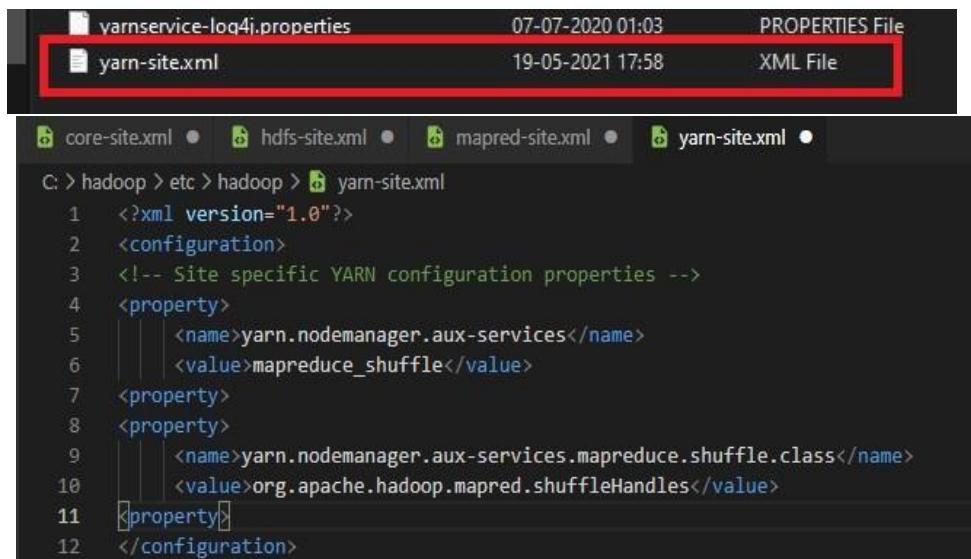
## Step 8: Configuration core-site.xml



mapred-queues.xml.template	07-07-2020 01:04	TEMPLATE File
mapred-site.xml	19-05-2021 17:58	XML File
ssl-client.xml.example	07-07-2020 00:16	EXAMPLE File

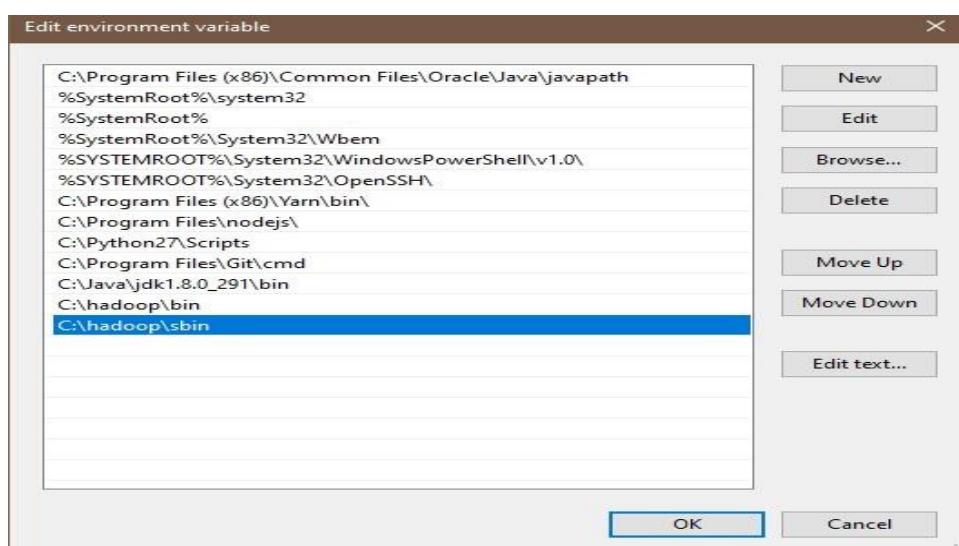
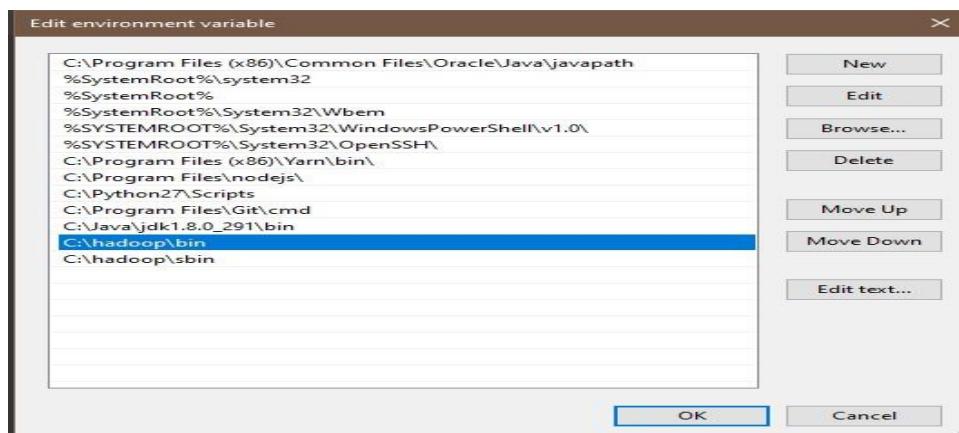
```
File Edit Selection View Go Run Terminal Help • mapred
core-site.xml ● hdfs-site.xml ● mapred-site.xml ●
C: > hadoop > etc > hadoop > mapred-site.xml
1   <?xml version="1.0"?>
2   <?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
3
4   <configuration>
5     <property>
6       |   <name>mapreduce.framework.name</name>
7       |   <value>yarn</value>
8     </configuration>
```

## Step 9: Configuration core-site.xml



```
C: > hadoop > etc > hadoop > yarn-site.xml
1  <?xml version="1.0"?>
2  <configuration>
3  <!-- Site specific YARN configuration properties -->
4  <property>
5      <name>yarn.nodemanager.aux-services</name>
6      <value>mapreduce_shuffle</value>
7  <property>
8  <property>
9      <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
10     <value>org.apache.hadoop.mapred.shuffleHandles</value>
11 </property>
12 </configuration>
```

Step 10: When the “Advanced system settings” dialog appears, go to the “Advanced” tab and click on the “Environment variables” button located on the bottom of the dialog.



Step 11: let's check Hadoop install Successfully

```
[INFO] Apache Hadoop Distribution

DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

2021-05-23 17:19:33,116 INFO namenode.NameNode: STARTUP_MSG:
*****STARTUP_MSG: Starting NameNode
STARTUP_MSG: host = DESKTOP-VUUFK2Q/192.168.0.104
STARTUP_MSG: args = []
STARTUP_MSG: version = 3.3.0
STARTUP_MSG: classpath = C:\hadoop\etc\hadoop;C:\hadoop\share\hadoop\common;C:\hadoop\share\hadoop\common\lib\accessors-smart-1.2.jar;C:\hadoop\share\hadoop\common\lib\animal-sniffer-annotations-1.17.jar;C:\hadoop\share\hadoop\common\lib\asm-5.0.4.jar;C:\hadoop\share\hadoop\common\lib\audience-annotations-0.5.0.jar;C:\hadoop\share\hadoop\common\lib\avro-1.7.7.jar;C:\hadoop\share\hadoop\common\lib\checker-qual-2.5.2.jar;C:\hadoop\share\hadoop\common\lib\commons-beanutils-1.9.4.jar;C:\hadoop\share\hadoop\common\lib\commons-cli-1.2.jar;C:\hadoop\share\hadoop\common\lib\commons-codec-1.11.jar;C:\hadoop\share\hadoop\common\lib\commons-collections-3.2.2.jar;C:\hadoop\share\hadoop\common\lib\commons-compress-1.19.jar;C:\hadoop\share\hadoop\common\lib\commons-configuration2-2.1.1.jar;C:\hadoop\share\hadoop\common\lib\commons-daemon-1.0.13.jar;C:\hadoop\share\hadoop\common\lib\commons-io-2.5.jar;C:\hadoop\share\hadoop\common\lib\commons-lang3-3.7.jar;C:\hadoop\share\hadoop\common\lib\commons-logging-1.1.3.jar;C:\hadoop\share\hadoop\common\lib\commons-math3-3.1.1.jar;C:\hadoop\share\hadoop\common\lib\commons-net-3.6.jar;C:\hadoop\share\hadoop\common\lib\commons-text-1.4.jar;C:\hadoop\share\hadoop\common\lib\curator-client-4.2.0.jar;C:\hadoop\share\hadoop\common\lib\curator-framework-4.2.0.jar;C:\hadoop\share\hadoop\common\lib\curator-recipes-4.2.0.jar;C:\hadoop\share\hadoop\common\lib\dnsjava-2.1.7.jar;C:\hadoop\share\hadoop
```

```
Apache Hadoop Distribution
at com.ctc.wstx.sr.StreamScanner.throwParseError(StreamScanner.java:491)
at com.ctc.wstx.sr.StreamScanner.throwParseError(StreamScanner.java:475)
at com.ctc.wstx.sr.BasicStreamReader.reportWrongEndElem(BasicStreamReader.java:3365)
at com.ctc.wstx.sr.BasicStreamReader.readEndElement(BasicStreamReader.java:3292)
at com.ctc.wstx.sr.BasicStreamReader.nextFromTree(BasicStreamReader.java:2911)
at com.ctc.wstx.sr.BasicStreamReader.next(BasicStreamReader.java:1123)
at org.apache.hadoop.conf.Configuration$Parser.parseNext(Configuration.java:3347)
at org.apache.hadoop.conf.Configuration$Parser.parse(Configuration.java:3141)
at org.apache.hadoop.conf.Configuration.loadResource(Configuration.java:3034)
...
9 more
```

### Step 12: Let check bin

```
C:\Windows\system32\cmd.exe
C:\Users\hp>cd C:\hadoop\sbin
C:\hadoop\sbin>start-all.cmd
This script is Deprecated. Instead use start-dfs.cmd and start-yarn.cmd
starting yarn daemons
C:\hadoop\sbin>
```

# PRACTICAL NO : 2

**Aim:** Implement Decision tree classification techniques

## Description:

Decision tree builds classification or regression models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with **decision nodes** and **leaf nodes**

**Step 1: The package "party" has the function ctree() which is used to create and analyze decision tree.**

```
> install.packages("party")
```

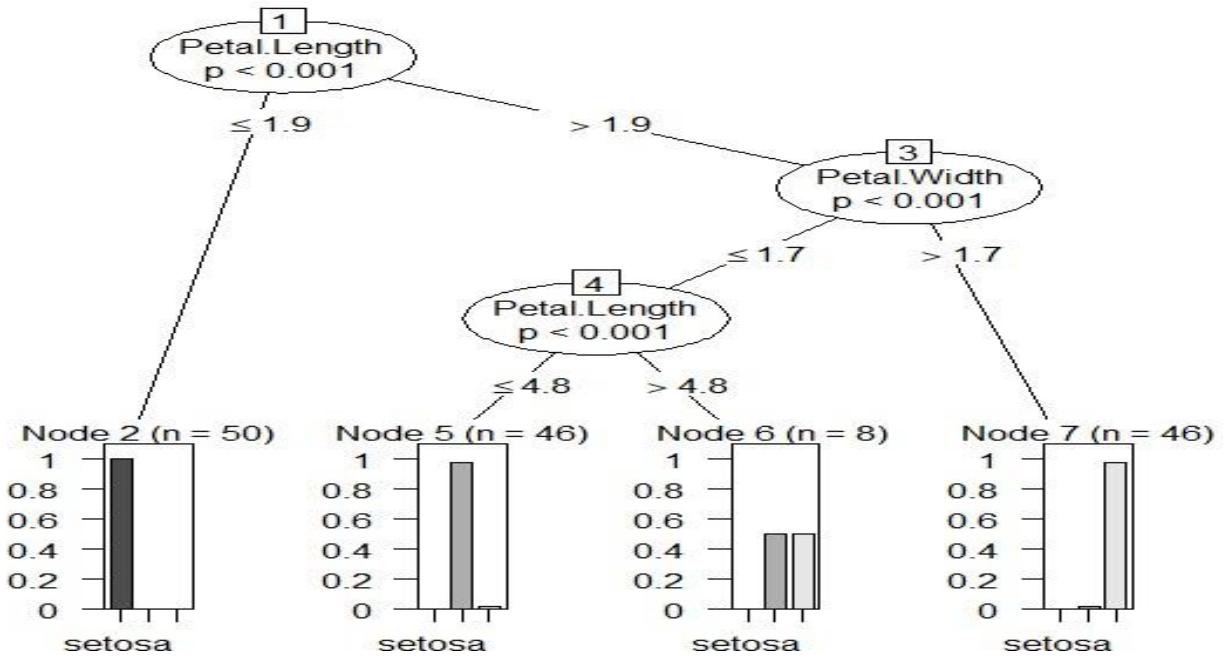
**Step 2: Load the party package. It will automatically load other# dependent packages Print some records from data set readingSkills.**

```
> library("party")
> print(head(readingskills))
  nativeSpeaker age shoesize      score
1           yes   5 24.83189 32.29385
2           yes   6 25.95238 36.63105
3          no    11 30.42170 49.60593
4          yes    7 28.66450 40.28456
5          yes   11 31.88207 55.46085
6          yes   10 30.07843 52.83124
>
```

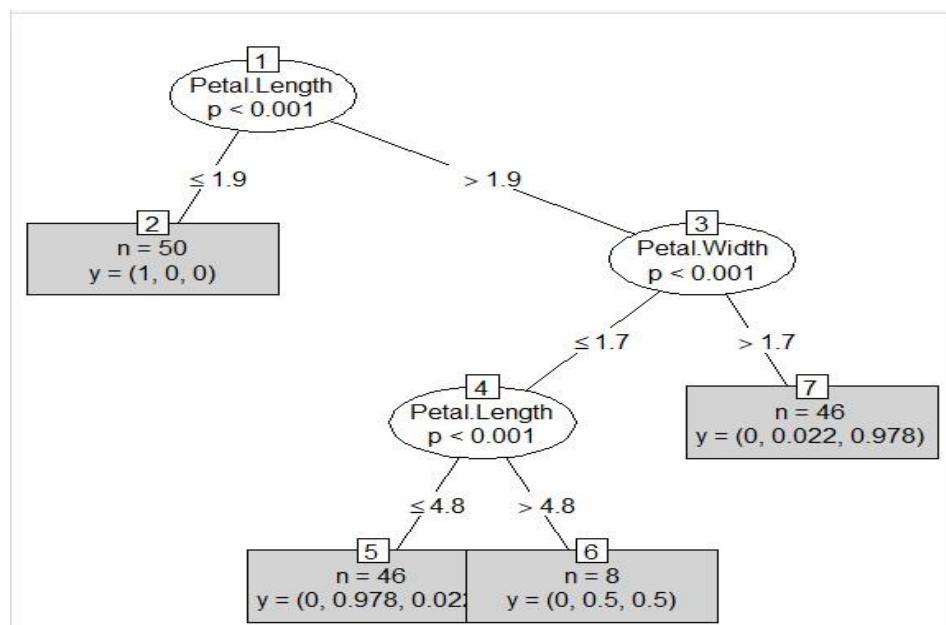
**Step 3 : Call function ctree to build a decision tree. The first parameter is a formula, which defines a target variable and a list of independent variables.**

```
> library("party")
> str(iris)
'data.frame': 150 obs. of 5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species      : Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1 1 1 1 1 1 1
>
> iris_ctree <- ctree(species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width, data=iris)
> print(iris_ctree)
  Conditional inference tree with 4 terminal nodes
Response: Species
Inputs: Sepal.Length, Sepal.Width, Petal.Length, Petal.Width
Number of observations: 150
1) Petal.Length <= 1.9; criterion = 1, statistic = 140.264
  2)* weights = 50
1) Petal.Length > 1.9
  3) Petal.Width <= 1.7; criterion = 1, statistic = 67.894
    4) Petal.Length <= 4.8; criterion = 0.999, statistic = 13.865
      5)* weights = 46
      4) Petal.Length > 4.8
        6)* weights = 8
      3) Petal.Width > 1.7
        7)* weights = 46
> plot(iris_ctree)
```

## Output :



```
> plot(iris_ctree, type="simple")
```



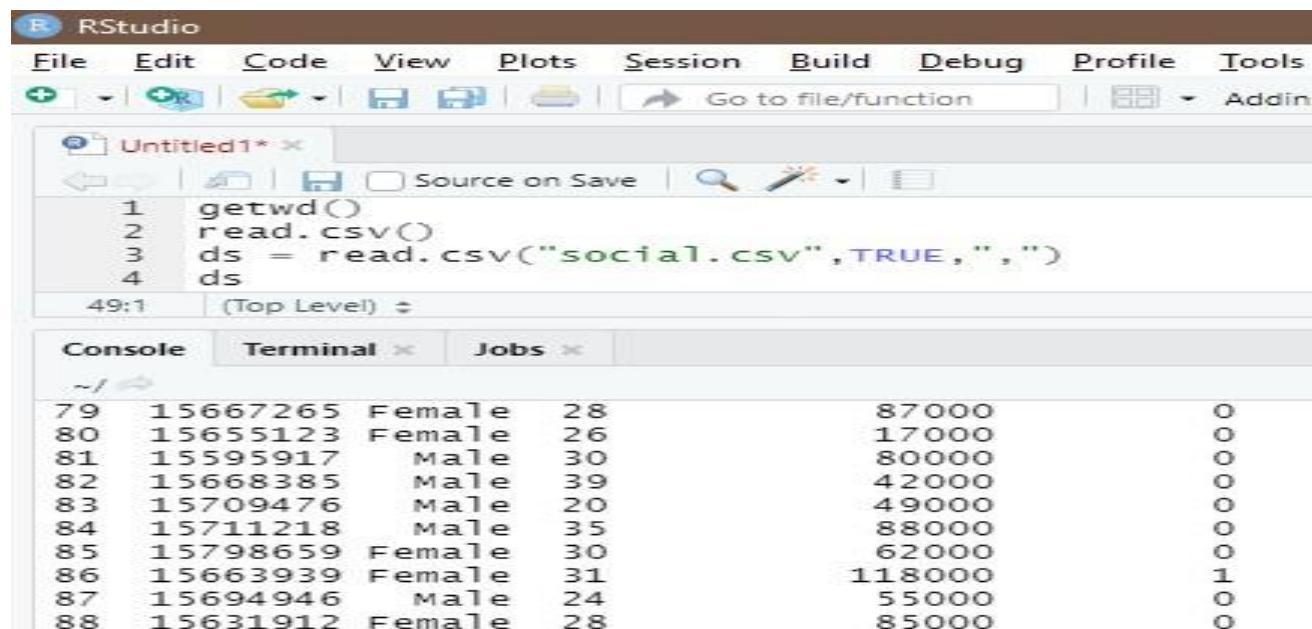
# PRACTICAL NO : 3

## Aim: Classification using SVM Description:

A support vector machine (SVM) is a supervised machine learning model that uses classification algorithms for two-group classification problems. After giving an SVM model sets of labeled training data for each category, they're able to categorize new text

The implementation is explained in the following steps:

### Step 1: Importing the dataset



The screenshot shows the RStudio interface. The code editor tab 'Untitled1' contains the following R code:

```
1 getwd()
2 read.csv()
3 ds = read.csv("social.csv", TRUE, ",")
4 ds
```

The console tab displays the following data table:

79	15667265	Female	28	87000	0
80	15655123	Female	26	17000	0
81	15595917	Male	30	80000	0
82	15668385	Male	39	42000	0
83	15709476	Male	20	49000	0
84	15711218	Male	35	88000	0
85	15798659	Female	30	62000	0
86	15663939	Female	31	118000	1
87	15694946	Male	24	55000	0
88	15631912	Female	28	85000	0

### Step 2: Selecting columns 3-5

```
> ds = ds[3:5]
> ds[3:5]
Error in ` [.data.frame` (ds, 3:5) : undefined
> ds
  Age EstimatedSalary Purchased
1 19      19000        0
2 35      20000        0
3 26      43000        0
4 27      57000        0
5 19      76000        0
6 27      58000        0
7 27      84000        0
8 32     150000        1
9 25      33000        0
10 35      65000        0
11 26      80000        0
12 26      52000        0
```

### Step 3: install package

```
> install.packages("caTools")
```

## Step 4: Splitting the dataset

```
> library(catools)
> set.seed(123)
> split = sample.split(ds$Purchased, splitRatio = 0.75)
> training_set = subset(ds, split == TRUE)
> test_set = subset(ds, split == FALSE)
> ds
   Age EstimatedSalary Purchased
1    19        19000       0
2    35        20000       0
3    26        43000       0
4    27        57000       0
5    19        76000       0
6    27        58000       0
7    27        84000       0
8    32        150000      1
9    25        33000       0
10   35        65000       0
```

## Step 5: Feature Scaling

```
332 48        119000      1
333 42        65000       0
[ reached 'max' / getoption("max.print") -- omitted 67 rows ]
> test_set[-3] = scale(test_set[-3])
> training_set[-3] = scale(training_set[-3])
> test_set[-3] = scale(test_set[-3])
> test_set[-3]
   Age EstimatedSalary
2   -0.30419063    -1.51354339
4   -1.05994374    -0.32456026
5   -1.81569686     0.28599864
9   -1.24888202    -1.09579256
12  -1.15441288    -0.48523366
18   0.64050076    -1.32073531
19   0.73496990    -1.25646596
20   0.92390818    -1.22433128
22   0.82943904    -0.58163769
29   -0.87100546   -0.77444577
32   -1.05994374    2.24621408
34   -0.96547460   -0.74231109
35   -1.05994374    0.73588415
38   -0.77653633   -0.58163769
45   -0.96547460    0.54307608
46   -1.43782030   -1.51354339
```

## Step 6: Fitting SVM to the training set

```
> install.packages('e1071')
> library(e1071)
> classifier = svm(formula = Purchased ~.,
+                     data = training_set,
+                     type = 'C-classification',
+                     kernel = 'linear')
> classifier
Call:
svm(formula = Purchased ~ ., data = training_set, type = "C-classification",
     kernel = "linear")

Parameters:
  SVM-Type:  C-classification
  SVM-Kernel: linear
        cost:  1

Number of support vectors: 116
```

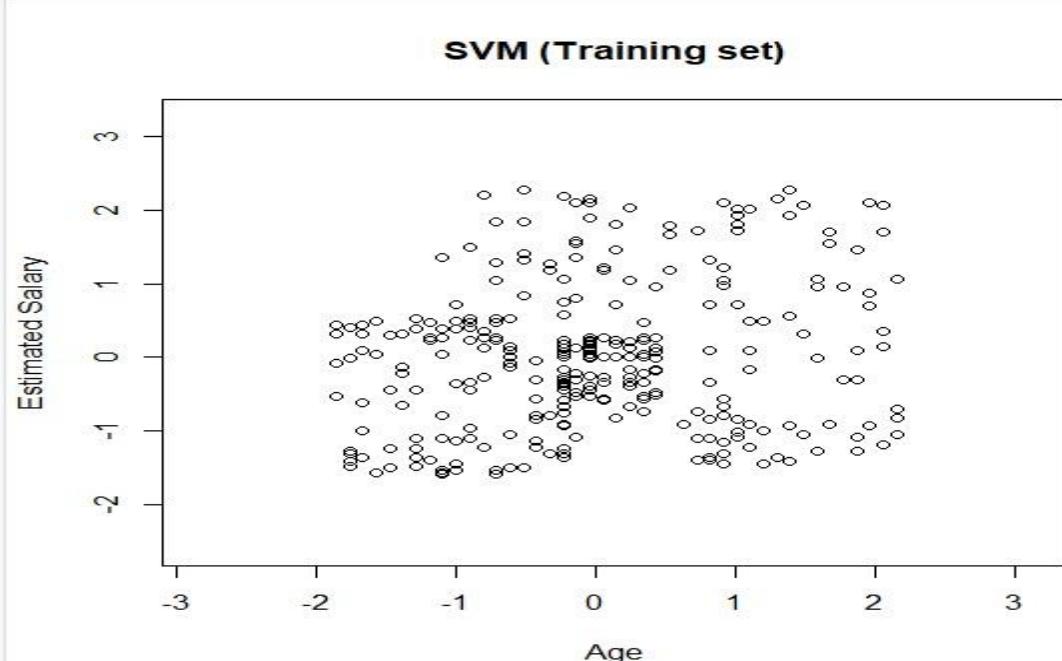
## Step 7: Predicting the test set result

```
> y_pred = predict(classifier, newdata = test_set[-3])
> y_pred
 2 4 5 9 12 18 19 20 22 29 32 34 35 38 45 46 48 52 66
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 69 74 75 82 84 85 86 87 89 103 104 107 108 109 117 124 126 127 131
 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
134 139 148 154 156 159 162 163 170 175 176 193 199 200 208 213 224 226 228
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1
229 230 234 236 237 239 241 255 264 265 266 273 274 281 286 292 299 302 305
 0 1 1 1 0 1 1 0 1 1 1 1 1 1 0 1 1 1 0 1
307 310 316 324 326 332 339 341 343 347 353 363 364 367 368 369 372 373 380
 1 0 0 0 0 1 0 1 0 1 1 0 1 1 1 0 1 0 1
383 389 392 395 400
 1 0 0 0 0
Levels: 0 1
```

```
> cm = table(test_set[, 3], y_pred)
> cm
y_pred
 0 1
 0 57 7
 1 13 23
```

## Step 8: Visualizing the Training set results

```
> set = training_set
> x1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)
> x2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)
```

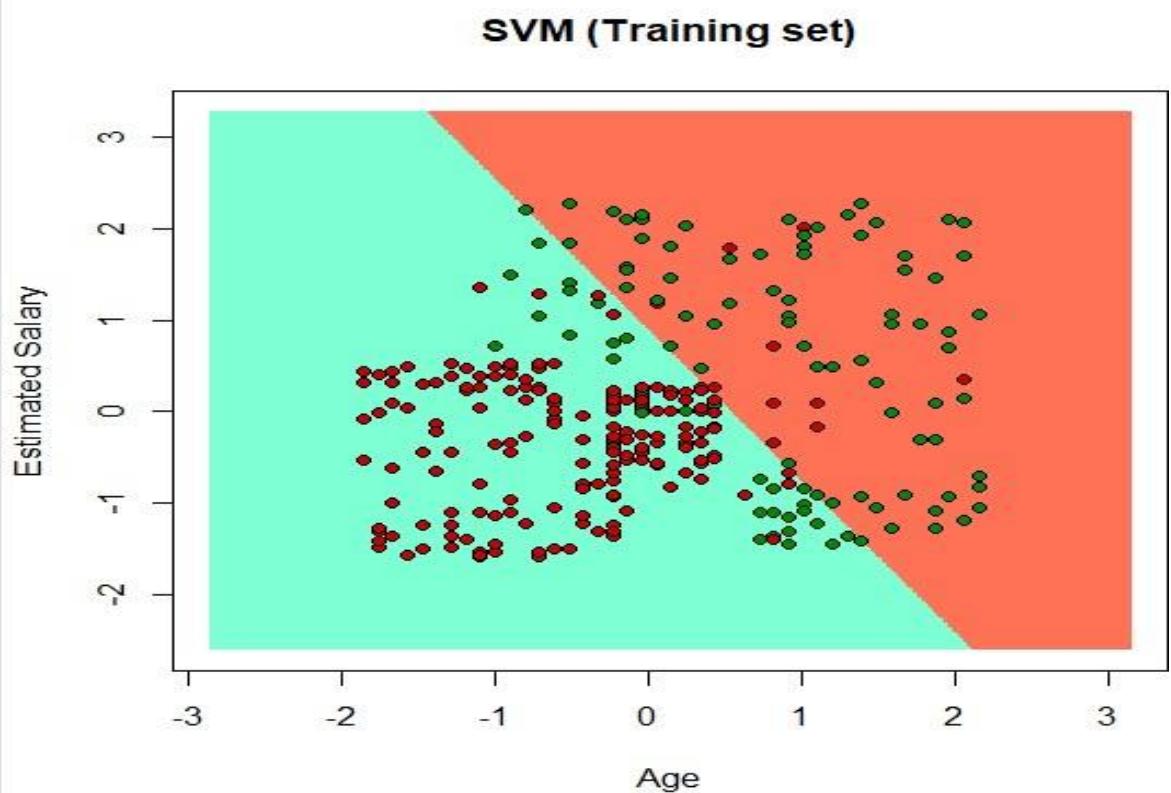


```
> grid_set = expand.grid(x1, x2)
> colnames(grid_set) = c('Age', 'Estimatedsalary')
> y_grid = predict(classifier, newdata = grid_set)
> plot(set[, -3],
+       main = 'SVM (Training set)',
+       xlab = 'Age', ylab = 'Estimated salary',
+       xlim = range(x1), ylim = range(x2))
```



```
> contour(x1, x2, matrix(as.numeric(y_grid), length(x1), length(x2)), add = TRUE)
> points(grid_set, pch = '.', col = ifelse(y_grid == 1, 'coral1', 'aquamarine'))
> points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))
```

## Output:



# PRACTICAL NO : 4

**Aim:** Implement an application that stores big data in Hbase / MongoDB and manipulate it using R / Python

## Description:

MongoDB is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas. MongoDB is developed by MongoDB Inc. and licensed under the Server Side Public License

**Name your organization and project**

**Organization**  
Your organization can be a business, team, or an individual

**Project Name**  
Use projects to isolate different environments (development/testing/production)

**What is your preferred language?**  
We'll use this to customize code samples and content we share with you. You can always change this later.

**JavaScript**     **C++**     **C# / .NET**     **Go**  
 **Java**     **C**     **Perl**     **PHP**  
 **Python**     **Ruby**     **Scala**     **Other**

[Skip](#) [Continue](#)

## Step 1 : Sign up and create a cluster.

CLUSTERS > CREATE A SHARED CLUSTER

### Create a Shared Cluster

Welcome to MongoDB Atlas! We've recommended some of our most popular options, but feel free to customize your cluster to your needs. For more information, check our [documentation](#).

**Cloud Provider & Region**

**AWS**     Google Cloud     Azure

**AWS, Mumbai (ap-south-1)**

**★ Recommended region**

**NORTH AMERICA**    **ASIA**    **EUROPE**    **AUSTRALIA**

**N. Virginia (us-east-1) ★**     **Singapore (ap-southeast-1) ★**     **Frankfurt (eu-central-1) ★**  
 **Oregon (us-west-2) ★**     **Mumbai (ap-south-1)**     **Ireland (eu-west-1) ★**  
 **Sydney (ap-southeast-2) ★**

This is the home page of mongoDB Atlas.

The screenshot shows the MongoDB Atlas web interface. At the top, there's a navigation bar with tabs for 'Clusters' and 'Atlas'. Below the navigation, there's a sidebar with sections for 'DATA STORAGE' (Clusters, Triggers, Data Lake), 'SECURITY' (Database Access, Network Access, Advanced), and a 'Connect to Atlas' checklist. The main area displays a cluster named 'Cluster0' (Version 4.4.6) in a 'SANDBOX' environment. It shows metrics like Logical Size (0.0 B), Operations (R: 0 W: 0), and Connections (0). A green 'Upgrade' button is visible. On the right, there's a 'Create a New Cluster' button. The bottom of the page includes a 'Get Started' button and copyright information.

**Step 2 : Click on collections to create and view existing databases.**

## Explore Your Data

- **Find:** run queries and interact with documents
- **Indexes:** build and manage indexes
- **Aggregation:** test aggregation pipelines
- **Search:** build search indexes

[Load a Sample Dataset](#)

[Add My Own Data](#)

[Learn more in Docs and Tutorials](#)

**Step 3 : Click on 'Add My Own Data' to create a database.**

## Create Database

**DATABASE NAME** ?

govind\_db

**COLLECTION NAME** ?

govind

**Capped Collection**

Before MongoDB can save your new database, a collection name must be specified at the time of creation.

[Cancel](#)

[Create](#)

## Step 4 : Click on insert document to add records.

The screenshot shows the MongoDB Compass interface. On the left, there's a sidebar with 'Databases: 1' and 'Collections: 2'. Below that, a 'NAMESPACES' dropdown is open, showing 'govind\_db' which is expanded, revealing '7\_govind' and 'govind'. The main area shows the 'govind\_db.govind' collection with a collection size of 144B, total documents at 2, and no indexes. There are tabs for 'Find', 'Indexes', and 'Schema Anti-Patterns'. A 'FILTER' button with the value '{ "filter": "example" }' is visible. At the bottom of the main area, there's a 'FIND' button.

Since MongoDB is a No-SQL database, so you can add 'n' number of columns for any row/record.

The screenshot shows the 'Insert to Collection' dialog in MongoDB Compass. It has a 'VIEW' section with three options: 'ObjectID' (selected), 'String', 'String', and 'String'. Below this, there's a code editor with the following JSON document:

```
1 _id : ObjectId("60a9f2437254d5ec231d1f06")
2 name : "Govind"
3 id : "7"
4 city : "Mumbai"
```

At the bottom right of the dialog are 'Cancel' and 'Insert' buttons. Below the dialog, there are navigation tabs: 'Find', 'Indexes', 'Schema Anti-Patterns', 'Aggregation', and 'Search Indexes'.

Perform updating data

#### QUERY RESULTS 1-2 OF 2

```
1   _id: ObjectId("60a9f2437254d5ec231d1f06")
2   name : "Govind Saini "
3   id : "7"
4   city : "Mumbai "
```

Document Updated.

```
_id: ObjectId("60a9f4917254d5ec231d1f07")
name: "Sayali Mam"
id: "8"
city: "Mumbai"
```

#### Performing deleting data

```
_id: ObjectId("60a9f2437254d5ec231d1f06")
name: "Govind Saini"
id: "7"
city: "Mumbai"
```

```
_id: ObjectId("60a9f4917254d5ec231d1f07")
name: "Sayali Mam"
id: "8"
city: "Mumbai"
```

Deleting Document.

#### Performing Insert data

```
_id: ObjectId("60a9ff027254d5ec231d1f0b")
name: "Govind Saini"
id: " 7"
city: "Mumbai"
```

```
_id: ObjectId("60a9ff3a7254d5ec231d1f0c")
name: "Sohrab Sir"
id: " 5"
city: "Mumbai"
```

#### Step 5 : To start with the connection click on Overview, and then click on Connect.

The screenshot shows the MongoDB Atlas Cluster Overview page. The cluster is named 'Cluster0' and is described as a 'Shared Tier Cluster'. It is connected to a 'MO Sandbox (General)' region in 'AWS / Mumbai (ap-south-1)'. The logical size is listed as 2.4 KB, with a maximum of 512.0 MB. The last 6 hours show 0.0 B of data. There are buttons for 'CONNECT', 'METRICS', 'COLLECTIONS', and '...'. A green 'Upgrade' button is visible on the right.

## Step 6 : Select on add your current IP and create a MongoDB user.

Connect to Cluster0

Setup connection security > Choose a connection method > Connect

You need to secure your MongoDB Atlas cluster before you can use it. Set which users and IP addresses can access your cluster now. [Read more](#)

You're ready to connect. Choose how you want to connect in the next step.

1 Add a connection IP address ✓ An IP address has been added to the IP Access List. [Add another address in the IP Access List tab.](#)

2 Create a Database User ✓ A MongoDB user has been added to this project. [Not yours? Create one in the MongoDB Users tab.](#)  
You'll need your MongoDB user's credentials in the next step.

[Close](#) [Choose a connection method](#)

## Step 7 : Click on 'Connect your application'.

Connect to Cluster0

✓ Setup connection security > Choose a connection method > Connect

Choose a connection method [View documentation](#)

Get your pre-formatted connection string by selecting your tool below.

Connect with the mongo shell  
Interact with your cluster using MongoDB's interactive Javascript interface

Connect your application  
Connect your application to your cluster using MongoDB's native drivers

Connect using MongoDB Compass  
Explore, modify, and visualize your data with MongoDB's GUI

[Go Back](#) [Close](#)

## Step 8 : Select the driver as 'Python' and version as '3.6 or later'. (Select the version as 3.6 or later only if your Python's version is 3.6 or later.)

Connect to Cluster0

✓ Setup connection security > ✓ Choose a connection method > Connect

1 Select your driver and version

DRIVER VERSION

Python 3.6 or later

2 Add your connection string into your application code

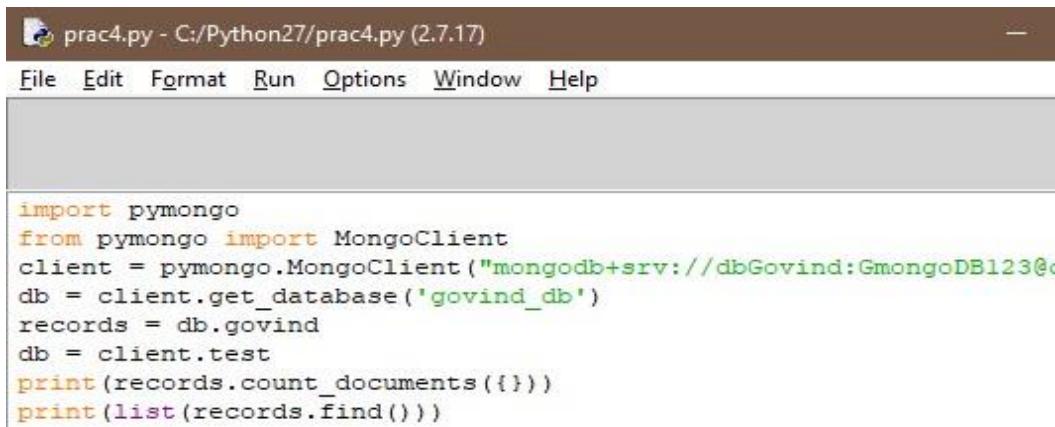
Include full driver code example

```
mongodb+srv://dbGovind:<password>@cluster0.m6shm.mongodb.net/myFirstDatabase?  
retryWrites=true&w=majority
```

Replace `<password>` with the password for the `dbGovind` user. Replace `myFirstDatabase` with the name of the database that connections will use by default. Ensure any option params are URL encoded.

Having trouble connecting? [View our troubleshooting documentation](#)

**Step 9 : Write the code given below in a Python file.**



The screenshot shows a Python code editor window titled "prac4.py - C:/Python27/prac4.py (2.7.17)". The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The code in the editor is as follows:

```
import pymongo
from pymongo import MongoClient
client = pymongo.MongoClient("mongodb+srv://dbGovind:GmongoDB123@c
db = client.get_database('govind_db')
records = db.govind
db = client.test
print(records.count_documents({}))
print(list(records.find()))
```

**Output :**

```
=====
RESTART: C:/Python27/prac.py =====
2[{"_id":{"$oid":"60a9ff027254d5ec231dlf0b"}, "name": "Govind Saini", "id": " 7", "city": "Mumbai"}, {"_id":{"$oid":"60a9ff3a7254d5ec231dlf0c"}, "name": "Sohrab Sir", "id": " 5", "city": "Mumbai"}]
>>> |
```

# PRACTICAL NO : 5

**Aim:** write program in R of Naive baye's theorem

## Description:

Naive Bayes is a Supervised Non-linear classification algorithm in R Programming.

Naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Baye's theorem with strong(Naive) independence assumptions between the features or variables

### # Loading data

```
> data(iris)
> str(iris)
'data.frame': 150 obs. of 5 variables:
 $ Sepal.Length: num 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1
```

### # Installing Packages

```
> install.packages("e1071")
> install.packages("caTools")
> install.packages("caret")
```

### # Loading package

```
> library(e1071)
> library(caTools)
> library(caret)
Loading required package: lattice
Loading required package: ggplot2
```

### # Splitting data into train and test data

```
> split <- sample.split(iris, splitRatio = 0.7)
> train_c1 <- subset(iris, split == "TRUE")
> test_c1 <- subset(iris, split == "FALSE")
>
> train_scale <- scale(train_c1[, 1:4])
> test_scale <- scale(test_c1[, 1:4])
>
> set.seed(120) # setting seed
> classifier_c1 <- naiveBayes(Species ~ ., data = train_c1)
> classifier_c1
```

```

Naive Bayes classifier for discrete Predictors
Call:
naiveBayes.default(x = x, y = Y, laplace = laplace)
A-priori probabilities:
Y
  setosa versicolor virginica
0.3333333 0.3333333 0.3333333
Conditional probabilities:
  Sepal.Length
Y [,1] [,2]
  setosa 5.046667 0.3848272
  versicolor 5.963333 0.5268536
  virginica 6.553333 0.6693967
  Sepal.Width
Y [,1] [,2]
  setosa 3.413333 0.4256705
  versicolor 2.823333 0.3470897
  virginica 2.956667 0.3136914
  Petal.Length
Y [,1] [,2]
  setosa 1.466667 0.1561019
  versicolor 4.320000 0.4759020
  virginica 5.496667 0.5738457
  Petal.Width
Y [,1] [,2]
  setosa 0.2766667 0.1135124
  versicolor 1.3533333 0.1960530
  virginica 2.0433333 0.2568823

```

### # Predicting on test data'

```

> y_pred <- predict(classifier_cl, newdata = test_cl)
> cm <- table(test_cl$Species, y_pred)
> cm
      y_pred
      setosa versicolor virginica
setosa     20        0        0
versicolor    0       19        1
virginica     0        2       18
>

```

### # Model Evaluation

```

> confusionMatrix(cm)
Confusion Matrix and Statistics

      y_pred
      setosa versicolor virginica
setosa     20        0        0
versicolor    0       19        1
virginica     0        2       18

Overall statistics

  Accuracy : 0.95
  95% CI   : (0.8608, 0.9896)
  No Information Rate : 0.35
  P-Value [Acc > NIR] : < 2.2e-16
  Kappa   : 0.925

  Mcnemar's Test P-Value : NA

Statistics by class:

      Class: setosa Class: versicolor Class: virginica
Sensitivity           1.0000          0.9048          0.9474
Specificity            1.0000          0.9744          0.9512
Pos Pred Value         1.0000          0.9500          0.9000
Neg Pred Value         1.0000          0.9500          0.9750
Prevalence              0.3333          0.3500          0.3167
Detection Rate          0.3333          0.3167          0.3000
Detection Prevalence      0.3333          0.3333          0.3333
Balanced Accuracy        1.0000          0.9396          0.9493

```

# PRACTICAL NO : 6

**Aim:** Write a Program showing implementation of Regression model.

## Description:

Regression is a method to mathematically formulate relationship between variables that in due course can be used to estimate, interpolate and extrapolate. Suppose we want to estimate the weight of individuals, which is influenced by height, diet, workout, etc. Here, *Weight* is the **predicted** variable

Lets implementation of Regression Model some Example:

```
> IQ <- rnorm(40, 30, 2)

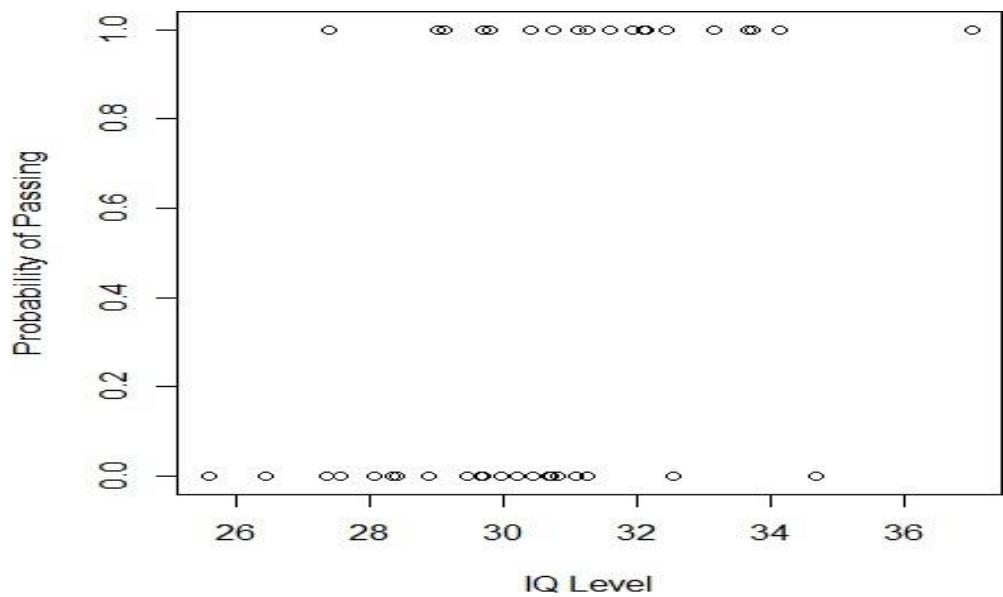
| > IQ <- sort(iQ)

> result <- c(0, 0, 0, 1, 0, 0, 0, 0, 0, 1,
+ 1, 0, 0, 0, 1, 1, 0, 0, 1, 0,
+ 0, 0, 1, 0, 0, 1, 1, 0, 1, 1,
+ 1, 1, 1, 0, 1, 1, 1, 0, 1)

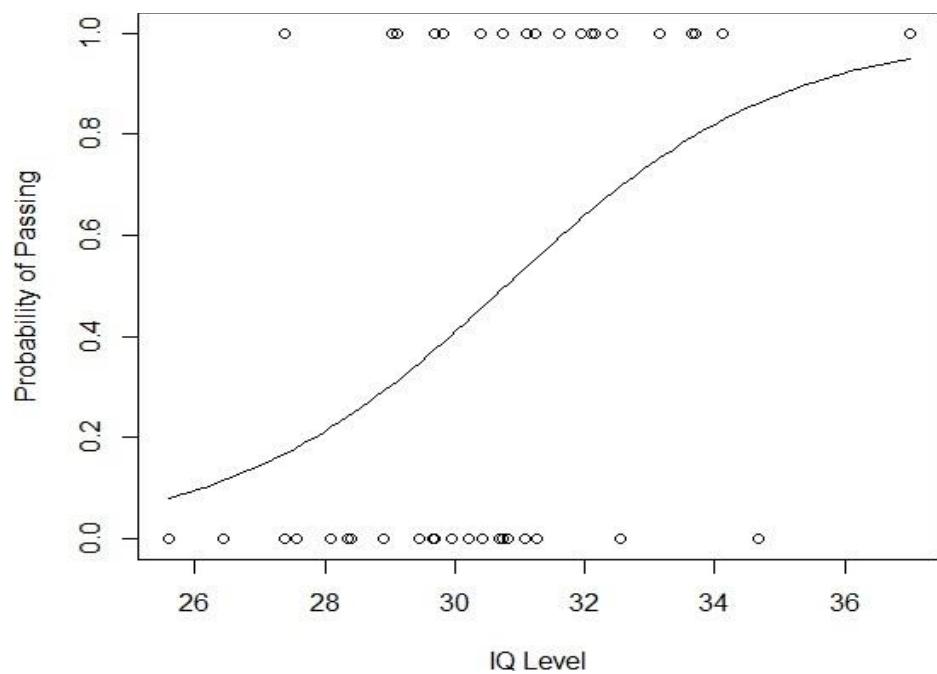
> df <- as.data.frame(cbind(iQ, result))
> print(df)
   iQ  result
1 25.58824     0
2 26.43200     0
3 27.37083     0
4 27.37898     1
5 27.56671     0
6 28.08275     0
7 28.35637     0
8 28.41538     0
9 28.89752     0
10 29.03158    1
11 29.12386    1
12 29.46181    0
13 29.66945    0
14 29.68934    0
15 29.69886    1
16 29.80735    1
17 29.95326    0
18 30.21428    0
19 30.39298    1
20 30.43421    0
21 30.67802    0
22 30.72653    0
23 30.74974    1
24 30.82265    0
25 31.07116    0
26 31.11633    1
27 31.24740    1
28 31.25662    0
29 31.60194    1
30 31.93038    1

| > png(file="LogisticRegressionGFG.png")

> plot(iQ, result, xlab = "IQ Level",
+       ylab = "Probability of Passing")
> g = glm(result~iQ, family=binomial, df)
```



```
> curve(predict(g, data.frame(IQ=x), type="resp"), add=TRUE)
> points(IQ, fitted(g), pch=30)
```



```
> summary(g)

Call:
glm(formula = result ~ IQ, family = binomial, data = df)

Deviance Residuals:
    Min      1Q  Median      3Q     Max 
-1.9877 -0.9804 -0.4502  0.9731  1.8898 

Coefficients:
            Estimate Std. Error z value Pr(>|z|)    
(Intercept) -14.4934    5.8835  -2.463   0.0138 *  
IQ           0.4708    0.1922   2.450   0.0143 *  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 55.352 on 39 degrees of freedom
Residual deviance: 47.090 on 38 degrees of freedom
AIC: 51.09

Number of Fisher Scoring iterations: 4
```

```
> dev.off()
null device
      1
```

# PRACTICAL NO : 7

**Aim:** Write a Program showing clustering.

## Description:

## # In this Program we understand about K-Mean Clustering #

## What Does K-Means Clustering Mean?

- K-means clustering is a simple unsupervised learning algorithm that is used to solve clustering problems.
  - It follows a simple procedure of classifying a given data set into a number of clusters, defined by the letter "k," which is fixed beforehand.
  - The clusters are then positioned as points and all observations or data points are associated with the nearest cluster, computed, adjusted and then the process starts over using the new adjustments until a desired result is reached.

### We Understand in different Steps :

**Step 1:** Apply kmeans to *newiris*, and store the clustering result in *kc*. The cluster number is set to 3.

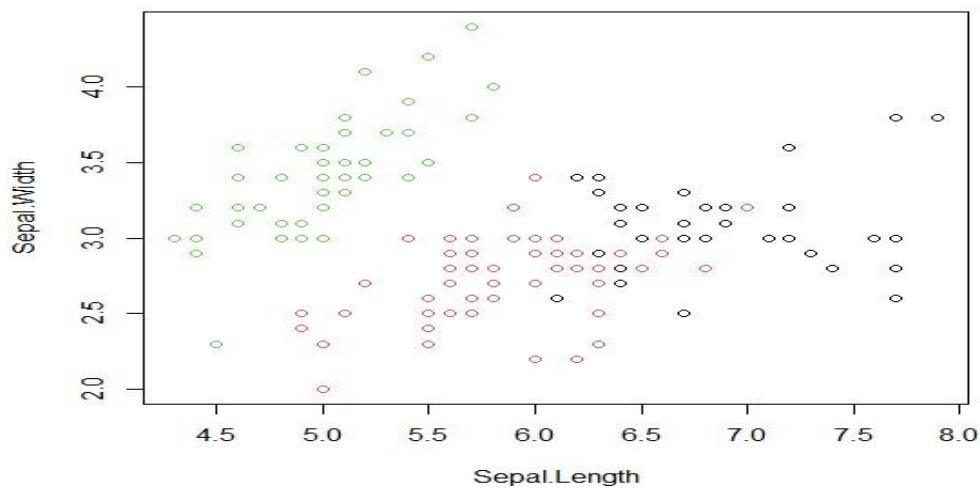
### **Step 2: Compare the Species label with the clustering result**

```
> table(iris$species, kc$cluster)

      1   2   3
setosa 0  0 50
versicolor 2 48 0
virginica 36 14 0
```

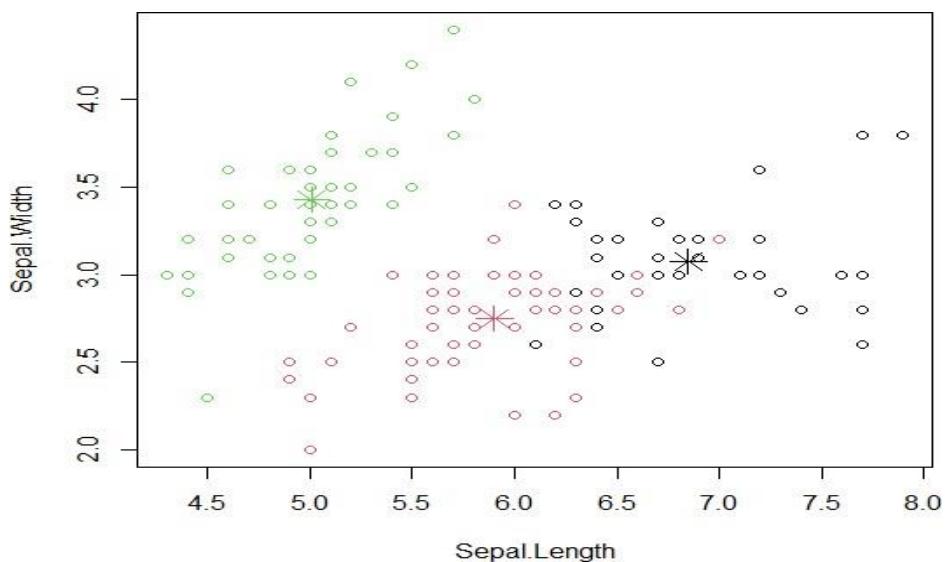
**Step 3 : Plot the clusters and their centres. Note that there are four dimensions in the data and that only the first two dimensions are used to draw the plot below.**

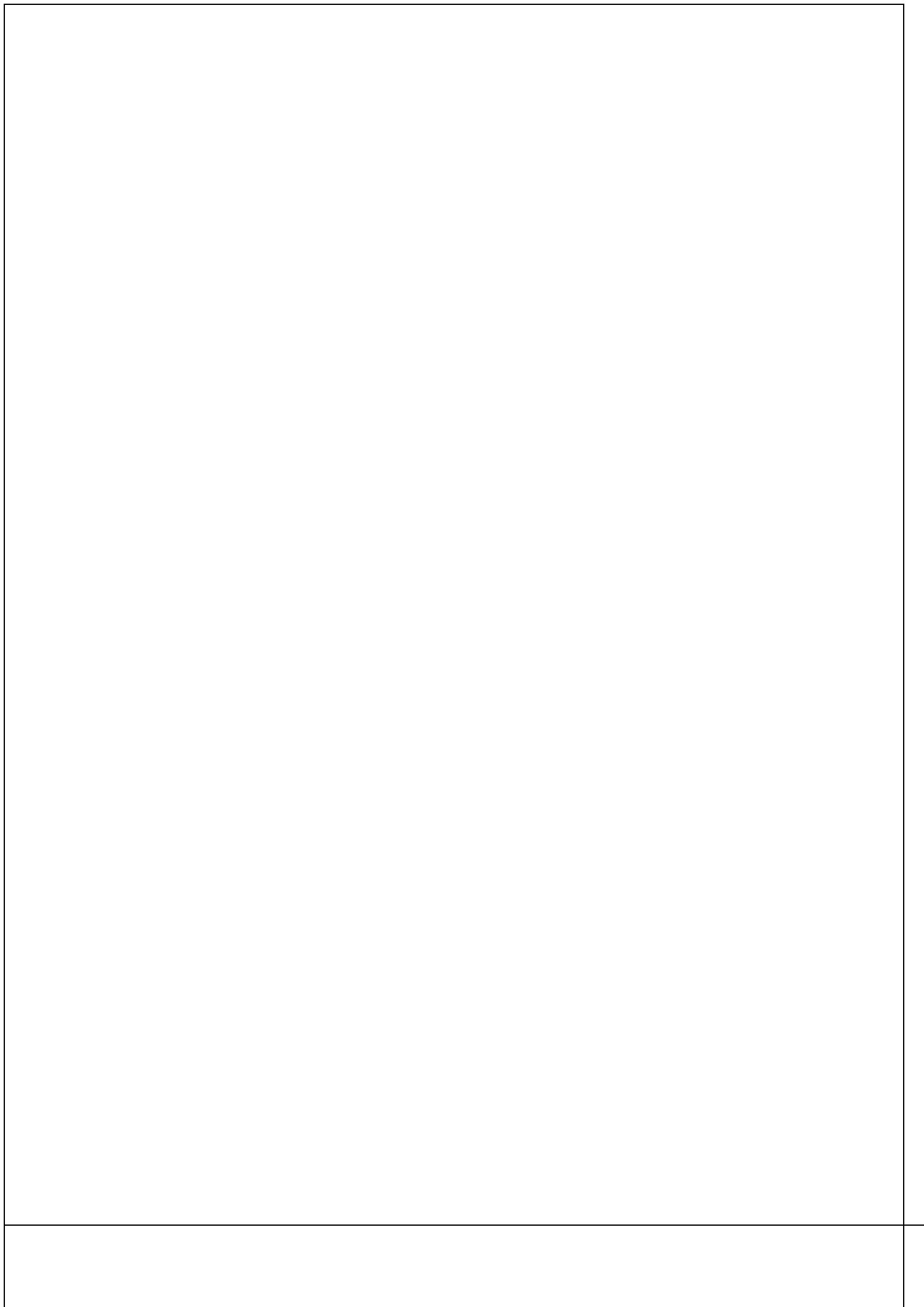
```
> plot(newiris[c("Sepal.Length", "Sepal.Width")], col=kc$cluster)
```



**Step 4: Some black points close to the green centre (asterisk) are actually closer to the black centre in the four dimensional space.**

```
> points(kc$centers[,c("Sepal.Length", "Sepal.Width")], col=1:3, pch=8, cex=2)
```





PRACTICAL JOURNAL

in

**Modern Networking**

Submitted to

**Laxman Devram Sonawane College, Kalyan (W) 421301**

*in partial fulfilment for the award of the degree of*

**Master of Science in Information Technology**



(Affiliated to Mumbai University)

*Submitted by*

**Vrushabh Ravindra Pawar**

Under the guidance of

**Mrs. Pradnya Mahajan**

Department of Information  
Technology Kalyan, Maharashtra

Academic Year 2023-24



The Kalyan Wholesale Merchants Education Society's  
**Laxman Devram Sonawane College,**  
**Kalyan (W) 421301**

**Department of Information Technology**  
**Masters of Science – Part II**

**Certificate**

This is to certify that **Mr. Vrushabh Ravindra Pawar**, Seat number \_\_\_\_\_, studying in Masters of Science in Information Technology Part I , Semester II has satisfactorily completed the practical of “**Modern Networking**” as prescribed by University of Mumbai, during the academic year 2023-24.

---

Subject In-charge

---

Coordinator In-charge

---

External Examiner

---

College Seal

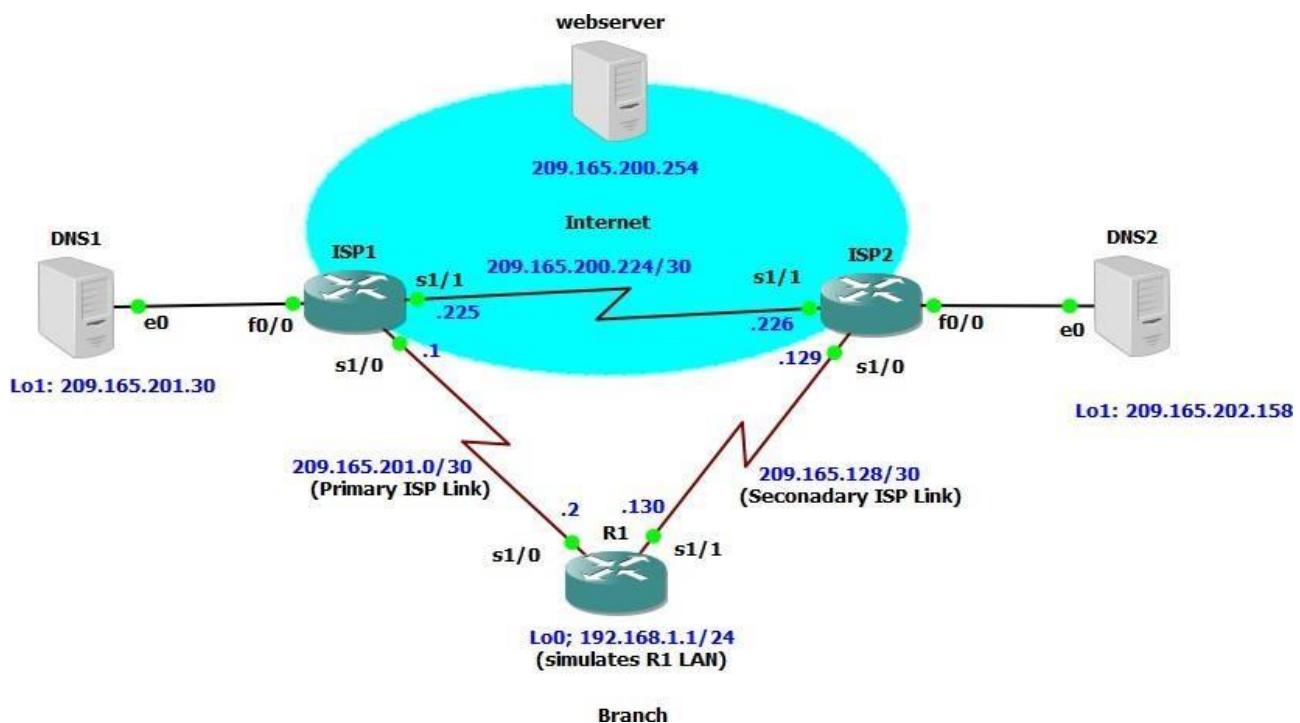
# **MODERN NETWORKING**

## **INDEX**

<b>Prac. No.</b>	<b>Practical</b>	<b>Date</b>
<b>1</b>	<b>Configure IP SLA Tracking and Path Control Topology</b>	
<b>2</b>	<b>Using the AS_PATH Attribute</b>	
<b>3</b>	<b>Configuring IBGP and EBGP Sessions, Local Preference, and MED</b>	
<b>4</b>	<b>Secure the Management Plane</b>	
<b>5</b>	<b>Configure and Verify Path Control Using PBR s</b>	
<b>6</b>	<b>Simulating MPLS environment</b>	

## Practical No - 1

**Aim:** Configure IP SLA Tracking and Path Control Topology **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.

**Step 1: Prepare the routers and configure the router hostname and interface addresses.**

#### Router R1

```
Interface Loopback 0 ip address 192.168.1.1
255.255.255.0
interface Serial 0/0/0 ip
address 209.165.201.2 255.255.255.252 no
shutdown
interface Serial 0/0/1 ip address
```

209.165.202.130 255.255.255.252 no

shutdown

```
R1(config)#interface Loopback 0
```

```
R1(config-if)#ip address 192.168.1.1 255.255.255.0
R1(config-if)#
R1(config-if)#int s1/0
R1(config-if)#ip address 209.165.201.2 255.255.255.252
R1(config-if)#no shutdown
```

```
R1(config-if)#int s1/1
```

```
R1(config-if)#ip address 209.165.202.130 255.255.255.252
R1(config-if)#no shutdown
```

**Router ISP1 (R2)** Interface Loopback 0 ip

address 209.165.200.254 255.255.255.255

Interface Loopback 1 ip address

209.165.201.30 255.255.255.255 interface

Serial 0/0/0 ip address 209.165.201.1

255.255.255.252 no shutdown interface Serial

0/0/1 ip address 209.165.200.255

255.255.255.252 no shutdown

```
ISP1(config)#interface Loopback0
ISP1(config-if)#
*May 18 15:24:24.315: %LINEPROTO-5-UPDOWN: Line protocol on
ISP1(config-if)#ip address 209.165.200.254 255.255.255.255
ISP1(config-if)#interface Loopback1
ISP1(config-if)#
*May 18 15:24:36.915: %LINEPROTO-5-UPDOWN: Line protocol on
ISP1(config-if)#ip address 209.165.201.30 255.255.255.255
ISP1(config-if)#int s1/0
ISP1(config-if)#ip address 209.165.201.1 255.255.255.252
ISP1(config-if)#no shutdown
ISP1(config-if)#
*May 18 15:25:03.695: %LINK-3-UPDOWN: Interface Serial1/0,
ISP1(config-if)#
ISP1(config-if)#i
*May 18 15:25:04.699: %LINEPROTO-5-UPDOWN: Line protocol on
ISP1(config-if)#int s1/1
ISP1(config-if)#ip address 209.165.200.225 255.255.255.252
ISP1(config-if)#no shutdown
```

**Router ISP2 (R3) Interface**

Loopback 0

ip address 209.165.200.254 255.255.255.255

Interface Loopback 1 ip address

209.165.202.158 255.255.255.255 interface

Serial 0/0/0 description ISP2→R1 ip address

209.165.202.129 255.255.255.252 no shutdown

interface Serial 0/0/1 ip address

209.165.200.256 255.255.255.252 no shutdown

```

ISP2(config)#interface Loopback0
ISP2(config-if)#
*May 18 15:25:22.219: %LINEPROTO-5-UPDOWN: Line protocol on
ISP2(config-if)#ip address 209.165.200.254 255.255.255.255
ISP2(config-if)#interface Loopback1
ISP2(config-if)#
*May 18 15:25:34.595: %LINEPROTO-5-UPDOWN: Line protocol on
ISP2(config-if)#ip address 209.165.202.158 255.255.255.255
ISP2(config-if)#int s1/0
ISP2(config-if)#ip address 209.165.202.129 255.255.255.252
ISP2(config-if)#no shutdown
ISP2(config-if)#
ISP2(config-if)#
*May 18 15:26:01.299: %LINK-3-UPDOWN: Interface Serial1/0,
ISP2(config-if)#i
*May 18 15:26:02.303: %LINEPROTO-5-UPDOWN: Line protocol on
ISP2(config-if)#int s1/1
ISP2(config-if)#ip address 209.165.200.226 255.255.255.252
ISP2(config-if)#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 interface description

```

R1#show interfaces description
Interface                  Status      Protocol Description
Fa0/0                     admin down   down
Se1/0                     up          up
Se1/1                     up          up
Se1/2                     admin down   down
Se1/3                     admin down   down
Lo0                      up          up

```

c. 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. **Router R1**

```
ip route 0.0.0.0 0.0.0.0 209.165.201.1
```

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

#### **Router ISP1 (R2) router eigrp 1**

```
network 209.165.200.224 0.0.0.3
```

```
network 209.165.201.0 0.0.0.31
```

```
no auto-summary ip route
```

```
192.168.1.0 255.255.255.0
```

```
209.165.201.2
```

```
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)#ip route 192.168.1.0 255.255.255.0 209.165.201.2
```

#### **Router ISP2 (R3) router eigrp 1**

```
network 209.165.200.224 0.0.0.3
```

```
network 209.165.202.128 0.0.0.31 no
```

```
auto-summary
```

```
ip route 192.168.1.0 255.255.255.0 209.165.202.130
```

```
ISP2(config)#router eigrp 1  
ISP2(config-router)#network 209.165.200.224 0.0.0.3  
ISP2(config-router)#  
*May 18 15:30:14.515: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 1: Neighbor 209.1  
ISP2(config-router)#network 209.165.202.128 0.0.0.31  
ISP2(config-router)#no auto-summary  
ISP2(config-router)#  
*May 18 15:30:28.971: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 1: Neighbor 209.1  
ISP2(config-router)#ip route 192.168.1.0 255.255.255.0 209.165.202.130
```

#### **Step 2: Verify server reachability.**

- 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.

```
R1(tcl)#foreach addess {
```

```
+>(tcl)# 209.165.200.254  
+>(tcl)#209.165.201.30  
+>(tcl)# 209.165.202.158  
+>(tcl)#} {ping $address source 192.168.1.1}
```

```
R1#tclsh  
R1(tcl)#foreach address {  
+>(tcl)#209.165.200.254  
+>(tcl)#209.165.201.30  
+>(tcl)#209.165.202.158  
+>(tcl)#} {  
+>(tcl)#ping $address source 192.168.1.1  
+>(tcl)#{
```

```
Type escape sequence to abort.  
Sending 5, 100-byte ICMP Echos to 209.165.200.254, timeout is 2 seconds:  
Packet sent with a source address of 192.168.1.1  
!!!!  
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/78/96 ms  
Type escape sequence to abort.  
Sending 5, 100-byte ICMP Echos to 209.165.201.30, timeout is 2 seconds:  
Packet sent with a source address of 192.168.1.1  
!!!!  
Success rate is 100 percent (5/5), round-trip min/avg/max = 20/31/48 ms  
Type escape sequence to abort.  
Sending 5, 100-byte ICMP Echos to 209.165.202.158, timeout is 2 seconds:  
Packet sent with a source address of 192.168.1.1  
!!!!  
Success rate is 100 percent (5/5), round-trip min/avg/max = 16/37/60 ms
```

- b. 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.

```
R1(tcl)#foreach addess {  
+>(tcl)# 209.165.200.254  
+>(tcl)#209.165.201.30  
+>(tcl)# 209.165.202.158  
+>(tcl)#} {trace $address source 192.168.1.1}
```

```
R1(tcl)#foreach address {  
+>(tcl)#209.165.200.254  
+>(tcl)#209.165.201.30  
+>(tcl)#209.165.202.158  
+>(tcl)#} {  
+>(tcl)#trace $address source 192.168.1.1  
+>(tcl)#{
```

```
Type escape sequence to abort.  
Tracing the route to 209.165.200.254  
  
    1 209.165.201.1 24 msec 24 msec 16 msec  
Type escape sequence to abort.  
Tracing the route to 209.165.201.30  
  
    1 209.165.201.1 16 msec 24 msec 24 msec  
Type escape sequence to abort.  
Tracing the route to 209.165.202.158  
  
    1 209.165.201.1 24 msec 24 msec 32 msec  
    2 209.165.200.226 28 msec 44 msec 72 msec
```

**Step 3: Configure IP SLA probes.**

- Create an ICMP echo probe on R1 to the primary DNS server on ISP1 using the ip sla command. the previous ip sla monitor command. In addition, the icmp-echo command has replaced the type echo protocol ipIcmpEcho 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)#ip sla schedule 11 life forever stat-time now
```

```
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)#ip sla schedule 11 life forever start-time now  
R1(config)#exit
```

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

```
R1#show ip sla configuration 11
```

```
R1(tcl)#show ip sla configuration 11
IP SLAs, Infrastructure Engine-II.
Entry number: 11
Owner:
Tag:
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
Operation timeout (milliseconds): 5000
Verify data: No
Vrf Name:
Schedule:
  Operation frequency (seconds): 10 (not considered if
  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
```

- 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

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

IPSLA operation id: 22
Type of operation: icmp-echo
  Latest RTT: 64 milliseconds
Latest operation start time: *15:40:40.823 UTC Tue May 18 2021
Latest operation return code: OK
Number of successes: 6
Number of failures: 0
Operation time to live: Forever
```

- 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. You can copy and paste the following commands on R1.

```
R1(config)#
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)#ip sla schedule 22 life forever start-time now
```

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

R1#show ip sla configuration 22

```
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: 209.165.202.158/0.0.0.0
Type Of Service parameter: 0x0
Request size (ARR data portion): 28
Operation timeout (milliseconds): 5000
Verify data: No
```

R1# show ip sla statistics 22

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

IPSLA operation id: 22
Type of operation: icmp-echo
    Latest RTT: 64 milliseconds
Latest operation start time: *15:40:40.823 UTC Tue May 18 2021
Latest operation return code: OK
Number of successes: 6
Number of failures: 0
Operation time to live: Forever
```

#### Step 4: Configure tracking options.

- Remove the current default route on R1, 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

```
R1(config)#
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
```

- Verify the routing table.

R1# show ip route

```
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 209.165.201.1 to network 0.0.0.0
```

- c. 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-tack)# delay down 10 up 1
```

```
R1(config-tack)# exit
```

```
R1(config)#
R1(config)#track 1 ip sla 11 reachability
R1(config-track)#delay down 10 up 1
R1(config-track)#exit
```

- d. Configure the floating static route that will be implemented when tracking object 1 is active. To view routing table changes as they happen, first enable the debug ip routing command. Next, 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# debug ip routing
```

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

```
R1(config)#ip route 0.0.0.0 0.0.0.0 209.165.201.1 2 track 1
```

```
R1(config)#ip route 0.0.0.0 0.0.0.0 209.165.201.1 2 track 1
R1(config)#
*May 18 15:43:00.035: RT: closer admin distance for 0.0.0.0, flushing 1 routes
*May 18 15:43:00.035: RT: NET-RED 0.0.0.0/0
*May 18 15:43:00.035: RT: add 0.0.0.0/0 via 209.165.201.1, static metric [2/0]
*May 18 15:43:00.039: RT: NET-RED 0.0.0.0/0
*May 18 15:43:00.039: RT: default path is now 0.0.0.0 via 209.165.201.1
*May 18 15:43:00.039: RT: new default network 0.0.0.0
*May 18 15:43:00.043: RT: NET-RED 0.0.0.0/0
```

- e. 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. track 2 ip sla 22 reachability delay down 10 up 1 exit

```
ip route 0.0.0.0 0.0.0.0 209.165.202.129 3 track 2
```

```
R1(config)#track 1 ip sla 22 reachability
R1(config-track)#delay down 10 up 1
R1(config-track)#exit
*May 18 15:43:56.339: RT: NET-RED 0.0.0.0/0
R1(config-track)#exit
R1(config)##ip route 0.0.0.0 0.0.0.0 209.165.201.1 2 track 1
R1(config)##ip route 0.0.0.0 0.0.0.0 209.165.202.129 3 track 2
...
```

- f. Verify the routing table again.

R1# show ip route

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

Gateway of last resort is 209.165.201.1 to network 0.0.0.0

      209.165.201.0/30 is subnetted, 1 subnets
C        209.165.201.0 is directly connected, Serial1/0
      209.165.202.0/30 is subnetted, 1 subnets
C        209.165.202.128 is directly connected, Serial1/1
C        192.168.1.0/24 is directly connected, Loopback0
S*    0.0.0.0 [2/0] via 209.165.201.1
```

### Step 5: Verify IP SLA operation.

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.

ISP1(config)#interface loopback 1

ISP1(config-if)# shutdown

```
ISP1(config)#
ISP1(config)#interface loopback 1
ISP1(config-if)#shutdown
```

- b. Verify the routing table.

R1#show ip route

```
R1#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B
      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
      ia - IS-IS inter area, * - candidate default, U - no route
      o - ODR, P - periodic downloaded static route

Gateway of last resort is 209.165.201.1 to network 0.0.0

      209.165.201.0/30 is subnetted, 1 subnets
C       209.165.201.0 is directly connected, Serial1/0
      209.165.202.0/30 is subnetted, 1 subnets
C       209.165.202.128 is directly connected, Serial1/1
C       192.168.1.0/24 is directly connected, Loopback0
S*   0.0.0.0/0 [2/0] via 209.165.201.1
```

c. Verify the IP SLA statistics.

R1# show ip sla statistics

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

IPSLA operation id: 11
Type of operation: icmp-echo
    Latest RTT: NoConnection/Busy/Timeout
Latest operation start time: *15:47:41.887 UTC Tue May 18 2021
Latest operation return code: No connection
Number of successes: 51
Number of failures: 13
Operation time to live: Forever
```

d. Initiate a trace to the web server from the internal LAN IP address.

R1# trace 209.165.200.254 source 192.168.1.1

```
R1#trace 209.165.200.254 source 192.168.1.1

Type escape sequence to abort.
Tracing the route to 209.165.200.254

 1 209.165.201.1 4 msec 32 msec 32 msec
```

e. Again examine the IP SLA statistics.

R1# show ip sla statistics

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

IPSLA operation id: 11
Type of operation: icmp-echo
    Latest RTT: 57 milliseconds
Latest operation start time: *15:50:01.887 UTC Tue May 18 2021
Latest operation return code: OK
Number of successes: 61
Number of failures: 17
Operation time to live: Forever
```

f. Verify the routing table.

```
R1# show ip route
```

```
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
      N1 - OSPF NSSA external type 1, N2 - OSPF NSSA externa
      E1 - OSPF external type 1, E2 - OSPF external type 2
      i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2
      ia - IS-IS inter area, * - candidate default, U - per-
          o - ODR, P - periodic downloaded static route

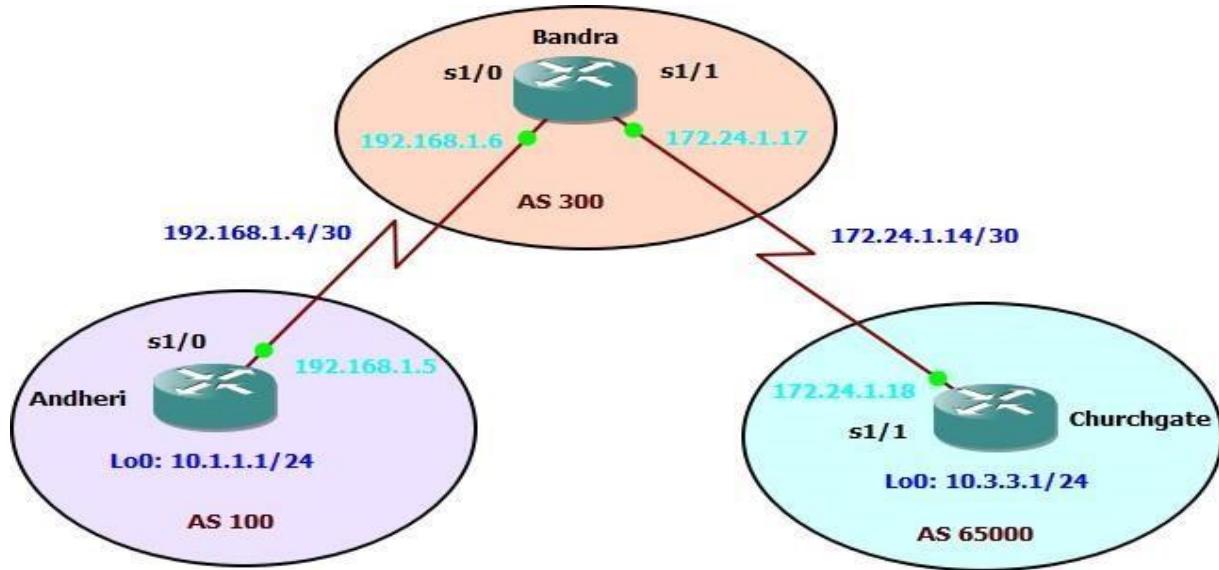
Gateway of last resort is 209.165.201.1 to network 0.0.0.0

      209.165.201.0/30 is subnetted, 1 subnets
C        209.165.201.0 is directly connected, Serial1/0
      209.165.202.0/30 is subnetted, 1 subnets
C        209.165.202.128 is directly connected, Serial1/1
C        192.168.1.0/24 is directly connected, Loopback0
S*   0.0.0.0/0 [2/0] via 209.165.201.1
```

## Practical No - 2

**Aim:** Using the AS\_PATH Attribute

**Topolgy :**



## Objective :

- 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 number

## 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 the hostname and interface addresses.

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

### Router R1 (hostname Andheri)

```

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

Andheri#

```
R1#
R1#conf t
Enter configuration commands, one per line. End with CNTL/Z.
R1(config)#hostname Andheri
Andheri(config)#int loopback 0
Andheri(config-if)#
*May  7 09:30:42.867: %LINEPROTO-5-UPDOWN: Line protocol on Interface Loopback0
Andheri(config-if)#ip address 10.1.1.1 255.255.255.0
Andheri(config-if)#exit
Andheri(config)#int s1/0
Andheri(config-if)#ip address 192.168.1.5 255.255.255.252
Andheri(config-if)#no shutdown
Andheri(config-if)#
*May  7 09:31:41.315: %LINK-3-UPDOWN: Interface Serial1/0, changed state to up
```

**Router R2 (hostname Bandra)**

```
Bandra(config)# interface Loopback0
Bandra(config-if)# ip address 10.2.2.1 255.255.255.0
Bandra(config-if)# interface Serial0/0/0
Bandra(config-if)# ip address 192.168.1.6 255.255.255.252
Bandra(config-if)# no shutdown
Bandra(config-if)# exit
Bandra(config)# interface Serial0/0/1
Bandra(config-if)# ip address 172.24.1.17 255.255.255.252
Bandra(config-if)# no shutdown
Bandra(config-if)# end
Bandra#
```

```
R2#
R2#conf t
Enter configuration commands, one per line. End with CNT
R2(config)#hostname Bandra
Bandra(config)#int loopback 0
Bandra(config-if)#ip addr
*May  7 09:31:30.407: %LINEPROTO-5-UPDOWN: Line protocol
Bandra(config-if)#ip address 10.2.2.1 255.255.255.0
Bandra(config-if)#exit
Bandra(config)#int s1/0
Bandra(config-if)#ip address 192.168.1.6 255.255.255.252
Bandra(config-if)#no shutdown
Bandra(config-if)#exit
```

**Router R3 (hostname ChurchGate)**

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

```
Bandra(config)#int s1/1
Bandra(config-if)#ip address 172.24.1.17 255.255.255.252
Bandra(config-if)#no shutdown
Bandra(config-if)#
*May  7 09:33:39.591: %LINK-3-UPDOWN: Interface Serial1/1,
Bandra(config-if)#

```

```
R3#
R3#conf t
Enter configuration commands, one per line. End with CNTL/Z.
R3(config)#hostname Churchgate
Churchgate(config)#int loopback 0
Churchgate(config-if)#ip[
*May  7 09:33:31.243: %LINEPROTO-5-UPDOWN: Line protocol on interface
Churchgate(config-if)#ip address 10.3.3.1 255.255.255.0
Churchgate(config-if)#exit
Churchgate(config)#int s1/1
Churchgate(config-if)#ip address 172.24.1.18 255.255.255.252
Churchgate(config-if)#no shutdown
Churchgate(config-if)#
*May  7 09:34:39.795: %LINK-3-UPDOWN: Interface Serial1/1, ch
```

- b. Use ping to test the connectivity between the directly connected routers.

```
Bandra#ping 192.168.1.5
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.1.5, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 20/32/40 ms
Bandra#
Bandra#
Bandra#ping 172.24.1.18
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.24.1.18, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/36/48 ms
Bandra#
```

### Step 3 : 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.

```
Andheri(config)# router bgp 100
```

```
Andheri(config-router)# neighbor 192.168.1.6 remote-as 300
```

```
Andheri(config-router)# network 10.1.1.0 mask 255.255.255.0
```

```
Andheri#conf t
Enter configuration commands, one per line. End with CNTL/D
Andheri(config)#router bgp 100
Andheri(config-router)#neighbor 192.168.1.6 remote-as 300
Andheri(config-router)#network 10.1.1.0 mask 255.255.255.0
```

```
Bandra(config)# router bgp 300
```

```
Bandra(config-router)# neighbor 192.168.1.5 remote-as 100
```

```
Bandra(config-router)# neighbor 172.24.1.18 remote-as 65000 Bandra(config-router)#

```

```
network 10.2.2.0 mask 255.255.255.0
```

```
Bandra#conf t
Enter configuration commands, one per line. End with CNTL/D
Bandra(config)#router bgp 300
Bandra(config-router)#neighbor 192.168.1.5 remote-as 100
Bandra(config-router)#
*May  7 10:04:59.051: %BGP-5-ADJCHANGE: neighbor 192.168.1.
Bandra(config-router)#neighbor 172.24.1.18 remote-as 65000
Bandra(config-router)#network 10.2.2.0 mask 255.255.255.0
```

```
Churchgate(config)# router bgp 65000
```

```
Churchgate(config-router)# neighbor 172.24.1.17 remote-as 300
```

```
Churchgate(config-router)# network 10.3.3.0 mask 255.255.255.0
```

```
Churchgate#conf t
Enter configuration commands, one per line. End with CNTL/Z.
Churchgate(config)#router bgp 65000
Churchgate(config-router)#neighbor 172.24.1.17 remote-as 300
Churchgate(config-router)#
*May 7 10:04:44.195: %BGP-5-ADJCHANGE: neighbor 172.24.1.17
Churchgate(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.

Bandra# show ip bgp neighbors

```
Bandra#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 00:01:30
  Last read 00:00:13, last write 00:00:44, hold time is 180, keepalive
  seconds
  Neighbor capabilities:
    Route refresh: advertised and received(new)
    New ASN Capability: advertised and received
    Address family IPv4 Unicast: advertised and received
  Message statistics:
    InQ depth is 0
    OutQ depth is 0

              Sent          Rcvd
  Opens:            1            1
  Notifications:   0            0
  Updates:          3            1
  Keepalives:       2            3
  Route Refresh:    0            0
  Total:           6            5
Default minimum time between advertisement runs is 30 seconds
```

#### Step 4 : Remove the private AS.

- a. DBandralay the Andheri routing table using the show ip route command. Andheri should have a route to both 10.2.2.0 and 10.3.3.0. Troubleshoot if necessary.

Andheri#show ip route

```
Andheri#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

  10.0.0.0/24 is subnetted, 3 subnets
B        10.3.3.0 [20/0] via 192.168.1.6, 00:04:51
B        10.2.2.0 [20/0] via 192.168.1.6, 00:05:22
C        10.1.1.0 is directly connected, Loopback0
  192.168.1.0/30 is subnetted, 1 subnets
C        192.168.1.4 is directly connected, Serial1/0
```

b . Ping again, this time as an extended ping, sourcing from the Loopback0 interface address. **ping 10.3.3.1 source 10.1.1.1 or ping 10.3.3.1 source Lo0**

```
Andheri#ping 10.3.3.1 source 10.1.1.1
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 = 36/52/64 ms
Andheri#
Andheri#show ip bgp
BGP table version is 4, local router ID is 10.1.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - in
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

      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            0 300 65000 i
```

c. Now check the BGP table on Andheri. The AS\_PATH to the 10.3.3.0 network should be AS 300. It no longer has the private AS in the path.

```
Andheri# show ip bgp
```

```
Andheri#ping 10.3.3.1 source 10.1.1.1
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 = 20/60/9
Andheri#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
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

      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        0 300 i
```

### **Step 5 : Use the AS\_PATH attribute to filter routes.**

- Configure a special kind of access list to match BGP routes with an AS\_PATH attribute that both begins and ends with the number 100. Enter the following commands on Bandra.

```
Bandra(config)# ip as-path access-list 1 deny ^100$
```

```
Bandra(config)# ip as-path access-list 1 permit *
```

```
Bandra#
Bandra#conf t
Enter configuration commands, one per line. End with CNTL/Z.
Bandra(config)#ip as-path access-list 1 deny ^100$
Bandra(config)#ip as-path access-list 1 permit .*
```

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

```
Bandra(config)# router bgp 300
```

```
Bandra (config-router)# neighbor 192.168.1.5 remove-private-as
```

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

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

```
Andheri# show ip route
```

```
Bandra#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

      172.24.0.0/30 is subnetted, 1 subnets
C        172.24.1.16 is directly connected, Serial1/1
      10.0.0.0/24 is subnetted, 3 subnets
B          10.3.3.0 [20/0] via 172.24.1.18, 00:13:19
C        10.2.2.0 is directly connected, Loopback0
B          10.1.1.0 [20/0] via 192.168.1.5, 00:13:20
      192.168.1.0/30 is subnetted, 1 subnets
C        192.168.1.4 is directly connected, Serial1/0
```

- d. Return to BANDRA and verify that the filter is working as intended.

```
Bandra# show ip bgp regexp ^100$
```

```
Bandra#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 - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

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

- e. Run the following Tcl script on all routers to verify whether there is connectivity. All pings from BANDRA should be successful. Andheri should not be able to ping the Churchgate loopback 10.3.3.1 or the WAN link 172.24.1.16/30. Churchgate should not be able to ping the Andheri loopback 10.1.1.1 or the WAN link 192.168.1.4/30.

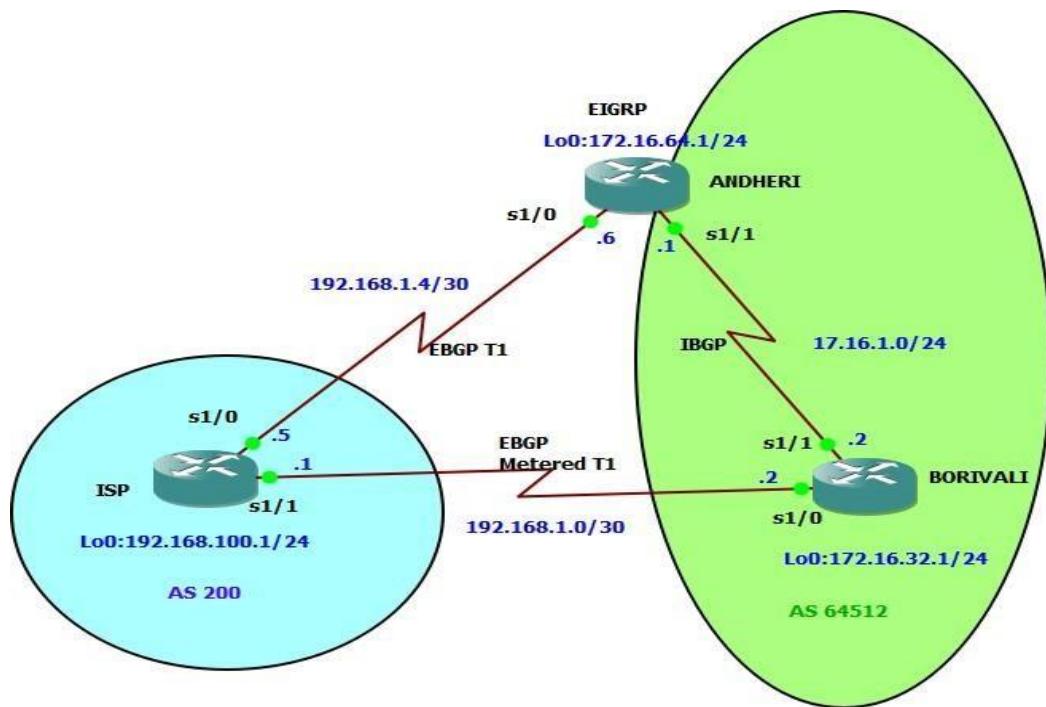
```
Bandra#tclsh

Bandra(tcl)#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 }
```

```
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 = 28/40/64 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 = 4/4/4 ms  
Type escape sequence to abort.  
Sending 5, 100-byte ICMP Echos to 10.3.3.1, timeout is 2 seconds:  
!!!!  
Success rate is 100 percent (5/5), round-trip min/avg/max = 16/32/48 ms  
Type escape sequence to abort.  
Sending 5, 100-byte ICMP Echos to 192.168.1.5, timeout is 2 seconds:  
!!!!  
Success rate is 100 percent (5/5), round-trip min/avg/max = 16/28/40 ms  
Type escape sequence to abort.  
Sending 5, 100-byte ICMP Echos to 192.168.1.6, timeout is 2 seconds:  
!!!!  
Success rate is 100 percent (5/5), round-trip min/avg/max = 48/59/64 ms  
Type escape sequence to abort.  
Sending 5, 100-byte ICMP Echos to 172.24.1.17, timeout is 2 seconds:  
!!!!  
Success rate is 100 percent (5/5), round-trip min/avg/max = 48/56/68 ms  
Type escape sequence to abort.  
Sending 5, 100-byte ICMP Echos to 172.24.1.18, timeout is 2 seconds:  
!!!!  
Success rate is 100 percent (5/5), round-trip min/avg/max = 16/31/48 ms
```

## Practical No - 3

**Aim:** Configuring IBGP and EBGP Sessions, Local Preference, and MED **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.

#### Step 1: Configure interface addresses. 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)# 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(config-if)#interface Loopback0
ISP(config-if)#ip address 192.168.100.1 255.255.255.0
ISP(config-if)#exit
ISP(config)#
ISP(config)#int s1/0
ISP(config-if)#ip address 192.168.1.5 255.255.255.252
ISP(config-if)#no shutdown
ISP(config-if)#exit
ISP(config)#
*May 18 17:42:51.491: %LINK-3-UPDOWN: Interface Serial1
ISP(config)#
*May 18 17:42:52.495: %LINEPROTO-5-UPDOWN: Line protocol
ISP(config)#
ISP(config)#int s1/1
ISP(config-if)#ip address 192.168.1.1 255.255.255.252
ISP(config-if)#no shutdown
```

### 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)# no shutdown
```

```
SanJose1(config-if)# end
```

```
ANDHERI(config)#interface Loopback0
ANDHERI(config-if)#
*May 18 17:42:40.167: %LINEPROTO-5-UPDOWN: Line protocol o
ANDHERI(config-if)#ip address 172.16.64.1 255.255.255.0
ANDHERI(config-if)#exit
ANDHERI(config)#
ANDHERI(config)#int s1/0
ANDHERI(config-if)#ip address 192.168.1.6 255.255.255.252
ANDHERI(config-if)#no shutdown
ANDHERI(config-if)#exit
ANDHERI(config)#
*May 18 17:43:11.899: %LINK-3-UPDOWN: Interface Serial1/0
ANDHERI(config)#
*May 18 17:43:12.903: %LINEPROTO-5-UPDOWN: Line protocol o
ANDHERI(config)#
ANDHERI(config)#int s1/1
ANDHERI(config-if)#ip address 172.16.1.1 255.255.255.0
ANDHERI(config-if)#no shutdown
```

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)# 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
```

```
BORIVALI(config)#interface Loopback0
BORIVALI(config-if)#
*May 18 17:43:25.783: %LINEPROTO-5-UPDOWN: Line protocol o
BORIVALI(config-if)#ip address 172.16.32.1 255.255.255.0
BORIVALI(config-if)#exit
BORIVALI(config)#
BORIVALI(config)#int s1/0
BORIVALI(config-if)#ip address 192.168.1.2 255.255.255.252
BORIVALI(config-if)#no shutdown
BORIVALI(config-if)#exit
BORIVALI(config)#
*May 18 17:43:54.311: %LINK-3-UPDOWN: Interface Serial1/0,
BORIVALI(config)#
BORIVALI(config)#
*May 18 17:43:55.315: %LINEPROTO-5-UPDOWN: Line protocol o
BORIVALI(config)#int s1/1
BORIVALI(config-if)#ip address 172.16.1.2 255.255.255.0
BORIVALI(config-if)#no shutdown
```

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

SanJose2(config)# **router eigrp 1**

SanJose2(config-router)# **network 172.16.0.0**

```
BORIVALI(config)#router eigrp 1  
BORIVALI(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
```

```
ANDHERI(config)#  
ANDHERI(config)#router bgp 64512  
ANDHERI(config-router)#neighbor 172.16.32.1 remote-as 64512  
ANDHERI(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
```

```
BORIVALI(config)#router bgp 64512
BORIVALI(config-router)#neighbor 172.16.64.1 remote-as 64512
BORIVALI(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**

```
BORIVALI#show ip bgp neighbors
BGP neighbor is 172.16.64.1, remote AS 64512,
  BGP version 4, remote router ID 172.16.64.1
  BGP state = Established, up for 00:00:47
  Last read 00:00:47, last write 00:00:47, hol
  Neighbor capabilities:
    Route refresh: advertised and received(new)
    New ASN Capability: advertised and receive
    Address family IPv4 Unicast: advertised an
  Message statistics:
    InQ depth is 0
    OutQ depth is 0

          Sent      Rcvd
  Opens:          1          1
  Notifications: 0          0
  Updates:        0          0
```

#### **Step 4: Configure EBGP and verify BGP neighbors.**

d. 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
```

```

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

```

- e. 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

```

ANDHERI(config)#ip route 172.16.0.0 255.255.0.0 null0
ANDHERI(config)#

```

- f. 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

```

ANDHERI(config)#router bgp 64512
ANDHERI(config-router)#neighbor 192.168.1.5 remote-as 200
ANDHERI(config-router)#network 172.16.0.0
ANDHERI(config-router)#exit

```

- g. 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

```

ANDHERI#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 00:02:49
  Last read 00:00:56, last write 00:00:21, hold time is 180
  Neighbor capabilities:
    Route refresh: advertised and received(new)
    New ASN Capability: advertised and received
    Address family IPv4 Unicast: advertised and received
  Message statistics:
    InQ depth is 0
    OutQ depth is 0

```

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
```

```
BORIVALI(config)#ip route 172.16.0.0 255.255.0.0 null0
BORIVALI(config)#router bgp 64512
BORIVALI(config-router)#neighbor 192.168.1.1 remote-as 200
BORIVALI(config-router)#
*May 18 18:00:01.031: %BGP-5-ADJCHANGE: neighbor 192.168.1.1 Up
BORIVALI(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
```

```
BORIVALI# show ip bgp summary
BGP router identifier 172.16.32.1, local AS number 64512
BGP table version is 5, main routing table version 5
2 network entries using 264 bytes of memory
4 path entries using 208 bytes of memory
5/2 BGP path/bestpath attribute entries using 840 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
Bitfield cache entries: current 2 (at peak 2) using 64 bytes of memory
BGP using 1400 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/Pf>
172.16.64.1    4     64512      7      8       5      0      0  00:04:18      2
192.168.1.1    4     200        5      4       3      0      0  00:00:26      1
```

### **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#clear ip bgp *
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**

```
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:
*May 18 18:02:42.575: %BGP-5-ADJCHANGE: neighbor 192.168.1.6 Up .....
Success rate is 0 percent (0/5)
```

ISP# ping 172.16.1.1

```
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)
```

- 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**

```
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 = 12/31/48 ms
```

ISP# ping 172.16.1.2

```
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 = 16/24/32 ms
```

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

```
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
Origin codes: i - IGP, e - EGP, ? - incomplete

      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
...  
...  
...
```

- i. 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**

```
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 = 16/38/48 ms
```

**ISP# ping 172.16.32.1 source 192.168.100.1**

```
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 = 4/32/48 ms
```

**ISP# ping 172.16.1.2 source 192.168.100.1**

```
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 = 16/28/48 ms
```

**ISP# ping 172.16.64.1 source 192.168.100.1**

```
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 = 24/35/56 ms
```

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

- j. 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
```

```
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
ISP(config-router)#exit
```

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

ISP# show ip 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 -
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

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

1. 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

```
BORIVALI#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
```

- m. 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)#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)#int s1/1
ISP(config-if)#shutdown
```

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

**route.** SanJose2#

**show ip bgp**

```
BORIVALI#show ip bgp
BGP table version is 14, local router ID is 172.16.32.1
Status codes: s suppressed, d damped, h history, * valid, > best, i -
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

      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    192.168.1.5             0          100      0 200 i
*>                   192.168.1.1             0          0       0 200 i
```

SanJose2# **show ip route**

```
BORIVALI#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
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

      172.16.0.0/16 is variably subnetted, 4 subnets, 2 masks
C       172.16.32.0/24 is directly connected, Loopback0
S       172.16.0.0/16 is directly connected, Null0
C       172.16.1.0/24 is directly connected, Serial1/1
D       172.16.64.0/24 [90/2297856] via 172.16.1.1, 00:11:47, Serial1/1
```

SanJose1(config)# **router bgp 64512**

SanJose1(config-router)# **neighbor 172.16.32.1 next-hop-self**

```
ANDHERI(config)#router bgp 64512
```

```
ANDHERI(config-router)#neighbor 172.16.32.1 next-hop-self
```

```
ANDHERI(config-router)#exit
```

SanJose2(config)# **router bgp 64512**

SanJose2(config-router)# **neighbor 172.16.64.1 next-hop-self**

```
BORIVALI(config-router)#router bgp 64512
```

```
BORIVALI(config-router)#neighbor 172.16.64.1 next-hop-self
```

- o. Reset BGP operation on either router with the **clear ip bgp \*** command. SanJose1# **clear**

```
ip bgp *
```

```
ANDHERI#clear ip bgp *
SanJose2# clear ip bgp *
BORIVALI#clear ip bgp *
```

- p. 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**

```
BORIVALI#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
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

      Network          Next Hop           Metric LocPrf Weight Path
* i172.16.0.0        172.16.64.1       0     100      0 i
* i192.168.100.0     172.16.64.1       0     100      0 200 i
```

- q. 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**

```
BORIVALI#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
       ia - IS-IS inter area, * - candidate default, U - per-user static r
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set
```

- r. 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)#int s1/1
ISP(config-if)#no shutdown
```

SanJose2# show ip route

```
BORIVALI#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
      D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF interarea
      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
      ia - IS-IS inter area, * - candidate default, U - per-
          o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

      172.16.0.0/16 is variably subnetted, 4 subnets, 2 masks
C        172.16.32.0/24 is directly connected, Loopback0
S        172.16.0.0/16 is directly connected, Null0
C        172.16.1.0/24 is directly connected, Serial1/1
D        172.16.64.0/24 [90/2297856] via 172.16.1.1, 00:15:05
      192.168.1.0/30 is subnetted, 1 subnets
C          192.168.1.0 is directly connected, Serial1/0
B        192.168.100.0/24 [20/0] via 192.168.1.1, 00:00:18
```

#### Step 8: Set BGP local preference.

- s. 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
ANDHERI(config)#route-map PRIMARY_T1_IN permit 10
ANDHERI(config-route-map)#set local-preference 150
ANDHERI(config-route-map)#exit
ANDHERI(config)#router bgp 64512
ANDHERI(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
```

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

- t. 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
```

```
ANDHERI#clear ip bgp * soft
```

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

```
BORIVALI#clear ip bgp * soft
```

```
SanJose1# show ip bgp
```

```
ANDHERI#show ip bgp
BGP table version is 6, local router ID is 172.16.64.1
Status codes: s suppressed, d damped, h history, * valid, > best, i
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

      Network          Next Hop            Metric LocPrf Weight Path
* i172.16.0.0        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
```

```
SanJose2# show ip bgp
```

```
BORIVALI#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
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

      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
* 192.168.100.0     192.168.1.1        0          0 200 i
*>i                172.16.64.1        0    150      0 200 i
```

### Step 9: Set BGP MED.

- u. 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

```
ISP#show ip bgp
BGP table version is 11, local router ID is 192.168.100.1
Status codes: s suppressed, d damped, h history, * valid, > best, i -
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

      Network          Next Hop           Metric LocPrf Weight Path
*  172.16.0.0        192.168.1.2       0          0 64512 i
*> 192.168.100.0    0.0.0.0          0          0 64512 i
*-> 192.168.100.0    0.0.0.0          0          32768 i
```

ISP# show ip route

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

Gateway of last resort is not set

B  172.16.0.0/16 [20/0] via 192.168.1.6, 00:04:56
    192.168.1.0/30 is subnetted, 2 subnets
C    192.168.1.0 is directly connected, Serial1/1
C    192.168.1.4 is directly connected, Serial1/0
C    192.168.100.0/24 is directly connected, Loopback0
```

- 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

```
BORIVALI#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)
```

```
Reply to request 4 (8 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 = 8/47/68 ms
```

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-
```

```
routemap)#exit
```

```
SanJose1(config)#router bgp 64512
```

```
SanJose1(config-router)#neighbor 192.168.1.5 route-map PRIMARY_T1_MED_OUT out
```

```
ANDHERI(config)#route-map PRIMARY_T1_MED_OUT permit 10
ANDHERI(config-route-map)#set Metric 50
ANDHERI(config-route-map)#exit
ANDHERI(config)#router bgp 64512
ANDHERI(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-
```

```
routemap)#exit
```

```
SanJose2(config)#router bgp 64512
```

```
SanJose2(config-router)#neighbor 192.168.1.1 route-map SECONDARY_T1_MED_OUT out
```

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

- v. 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
```

```
ANDHERI#clear ip bgp * soft
```

SanJose2# clear ip bgp \* soft

```
BORIVALI#clear ip bgp * soft
```

SanJose1# show ip bgp

```
ANDHERI#show ip bgp
BGP table version is 6, 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
Origin codes: i - IGP, e - EGP, ? - incomplete

      Network          Next Hop           Metric LocPrf Weight Path
* i172.16.0.0        172.16.32.1       0    100      0 i
*> 0.0.0.0           0.0.0.0          0            32768 i
*> 192.168.100.0     192.168.1.5       0    150      0 200 i
```

SanJose2# show ip bgp

```
BORIVALI#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
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

      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
* 192.168.100.0      192.168.1.1       0    125      0 200 i
*>i                 172.16.64.1       0    150      0 200 i
```

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.

```
BORIVALI#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 sec
Packet sent with a source address of 172.16.32.1
Packet has IP options: Total option bytes= 39, padded length=40
```

```
Reply to request 3 (60 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 (52 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 = 40/52/60 ms
```

```
ISP# show ip bgp
```

```
ISP#show ip bgp
BGP table version is 11, local router ID is 192.168.100.1
Status codes: s suppressed, d damped, h history, * valid, > best, i -
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

      Network          Next Hop            Metric LocPrf Weight Path
*  172.16.0.0        192.168.1.2          0          0 64512 i
*>                    192.168.1.6          0          0 64512 i
*-> 192.168.100.0    0.0.0.0            0          32768 i
```

### **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.

- 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)#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-if)#ip address 10.0.0.1 255.255.255.0
ISP(config-if)#exit
```

- 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
```

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

Gateway of last resort is 172.16.32.1 to network 0.0.0.0

  172.16.0.0/16 is variably subnetted, 4 subnets, 2 m
D    172.16.32.0/24 [90/2297856] via 172.16.1.2, 02:14
S    172.16.0.0/16 is directly connected, Null0
C    172.16.1.0/24 is directly connected, Serial1/1
C    172.16.64.0/24 is directly connected, Loopback0
B   192.168.100.0/24 [200/0] via 172.16.32.1, 01:44:43
B*  0.0.0.0/0 [200/0] via 172.16.32.1, 01:44:43
```

SanJose2# show ip route

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

Gateway of last resort is 192.168.1.1 to network 0.0.0.0

  172.16.0.0/16 is variably subnetted, 4 subnets, 2 m
C    172.16.32.0/24 is directly connected, Loopback0
S    172.16.0.0/16 is directly connected, Null0
C    172.16.1.0/24 is directly connected, Serial1/1
D   172.16.64.0/24 [90/2297856] via 172.16.1.1, 02:14
  192.168.1.0/30 is subnetted, 1 subnets
C    192.168.1.0 is directly connected, Serial1/0
B   192.168.100.0/24 [20/0] via 192.168.1.1, 01:44:59
B*  0.0.0.0/0 [20/0] via 192.168.1.1, 01:45:00
```

- 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

```
BORIVALI# show ip bgp
BGP table version is 9, local router ID is 172.16.32.1
Status codes: s suppressed, d damped, h history, * valid, > best, i -
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

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

- 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

```
BORIVALI#traceroute 10.0.0.1
Type escape sequence to abort.
Tracing the route to 10.0.0.1
 1 192.168.1.1 [AS 200] 28 msec 32 msec 32 msec
 2 192.168.1.1 [AS 200] !H !H !H
```

- 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)#int s1/0
ISP(config-if)#shutdown
```

- 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

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

Gateway of last resort is 172.16.32.1 to network 0.0.0.0

      172.16.0.0/16 is variably subnetted, 4 subnets, 2 ma
D        172.16.32.0/24 [90/2297856] via 172.16.1.2, 02:15
S        172.16.0.0/16 is directly connected, Null0
C        172.16.1.0/24 is directly connected, Serial1/1
C        172.16.64.0/24 is directly connected, Loopback0
B        192.168.100.0/24 [200/0] via 172.16.32.1, 01:45:58
B*       0.0.0.0/0 [200/0] via 172.16.32.1, 01:45:58
```

SanJose2# show ip route

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

Gateway of last resort is 192.168.1.1 to network 0.0.0.0

      172.16.0.0/16 is variably subnetted, 4 subnets, 2 mas
C        172.16.32.0/24 is directly connected, Loopback0
S        172.16.0.0/16 is directly connected, Null0
C        172.16.1.0/24 is directly connected, Serial1/1
D        172.16.64.0/24 [90/2297856] via 172.16.1.1, 02:15:
      192.168.1.0/30 is subnetted, 1 subnets
C          192.168.1.0 is directly connected, Serial1/0
B        192.168.100.0/24 [20/0] via 192.168.1.1, 01:46:10
B*       0.0.0.0/0 [20/0] via 192.168.1.1, 01:46:10
```

- 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

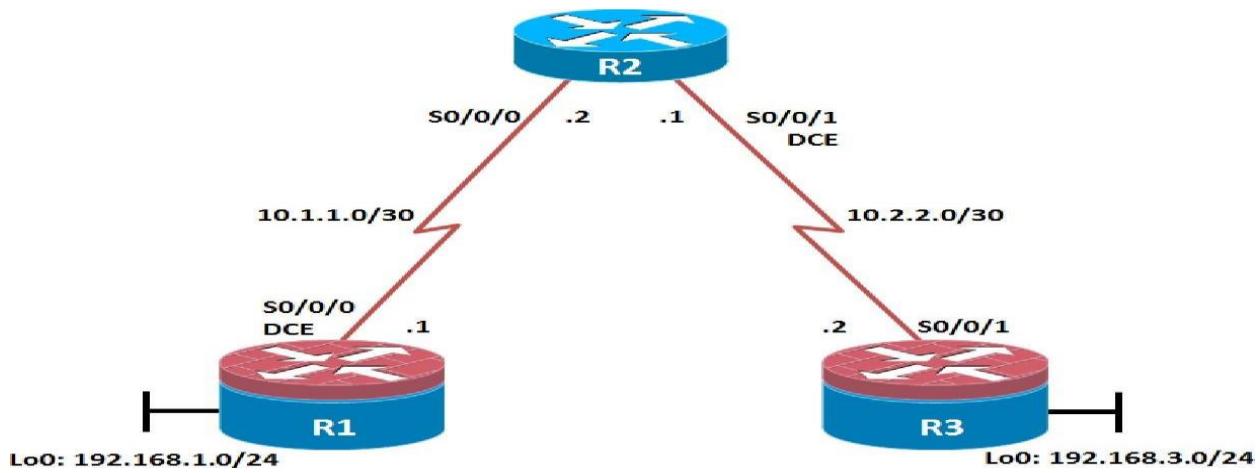
```
ANDHERI#trace 10.0.0.1
Type escape sequence to abort.
Tracing the route to 10.0.0.1

 1 172.16.1.2 64 msec 32 msec 32 msec
 2 192.168.1.1 [AS 200] 56 msec 28 msec 64 msec
 3 192.168.1.1 [AS 200] !H !H !H
ANDHERI#
```

## Practical No - 4

**Aim:** Secure the Management Plane

**Topology:**



## Objectives :

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

### **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. **Router R1 interface Loopback 0**

```
ip address 192.168.1.1 255.255.255.0
exit           interface
```

Serial0/0/0

```
ip address 10.1.1.1 255.255.255.252 no
```

shutdown

exit end

### **Router R2**

```
interface Serial0/0/0
```

```
ip address 10.1.1.2 255.255.255.252 no
```

shutdown

```
exit           interface
Serial0/0/1
ip address 10.2.2.1 255.255.255.252 no
shutdown
exit end Router R3
interface Loopback0
ip address 192.168.3.1 255.255.255.0
exit           interface
Serial0/0/1
ip address 10.2.2.2 255.255.255.252 no
shutdown
exit end
```

**Step 2:****Configure****static****routes.**

```
R1(config)# ip route 0.0.0.0 0.0.0.0 10.1.1.2
R3(config)# ip route 0.0.0.0 0.0.0.0 10.2.2.1
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
```

**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:

!!!!

**Step 3: Secure management access.**

1. On R1, use the **security passwords** command to set a minimum password length of 10 characters.  
R1(config)# **security passwords min-length 10**

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

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

3. 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.

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

```
R1(config-line)# exit
```

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
```

4. 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
```

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

---

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

```
R1(config)#
```

6. 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
```

#### **Step 4: Configure enhanced username password security.**

1. 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
```

2. 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
```

3. 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
```

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

w. 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:

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

Configure the specifics for the first RADIUS server located at 192.168.1.101. Use **RADIUS-1pa55w0rd** 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-radiusserver)# exit
```

1. 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-radiusserver)# exit
```

2. 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
```

3. 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
```

4. 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
```

Alter the VTY lines to use the TELNET-LOGIN AAA authentication method. **R1(config)# line vty 0 4**

```
R1(config-line)# login authentication TELNET-LGIN
```

```
R1(config-line)# exit
```

Repeat the steps 5a to 5g on R3.

5. 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:
```

#### **Step 6: Enabling secure remote management using SSH.**

1. 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
```

2. 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
```

3. 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 nonexportable...[OK]
```

```
R1(config)#
```

Jan 10 13:44:44.711: %SSH-5-ENABLED: SSH 1.99 has been enabled

4. 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).

Configure SSH version 2 on R1.

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

```
R1(config)#
```

5. 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
```

6. 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(sshrsa, base64

encoded):

ssh-rsa

AAAAB3NzaC1yc2EAAAQABAAQQC3Lehh7ReYlgyDzls6wq+mFzx qzoaZFr9XGx+Q/ yio

dFYw00hQo80tZy1W1Ff3Pz6q7Qi0y00urwddHZ0kBZceZK9EzJ6wZ+9a87KKDETCWr

GSLi6c8lE/y4K+ Z/oVrMMZk7bpTM1MFdP41YgkTf35utYv+TcqbsYo++KJiYk+xw==

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

8. 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

RADIUSGROUP server name

RADIUS-1 server name RADIUS-2

aaa authentication login default group RADIUS-GROUP local aaa

authentication login TELNET-LOGIN group RADIUSGROUP 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
TELNETLOGIN transport input ssh
end Router R2 hostname R2 enable
secret 5
$1$DJS7$xvJDW87zLs8pSJDFUICPB1
interface Serial0/0/0
```

```
ip address 10.1.1.2 255.255.255.252 no
fair-queue
interface Serial0/0/1 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$4J6VF1vGNKjwQ8XtajgUk1 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$pVgSEbinvXQ7f7uyxeKBj
```

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

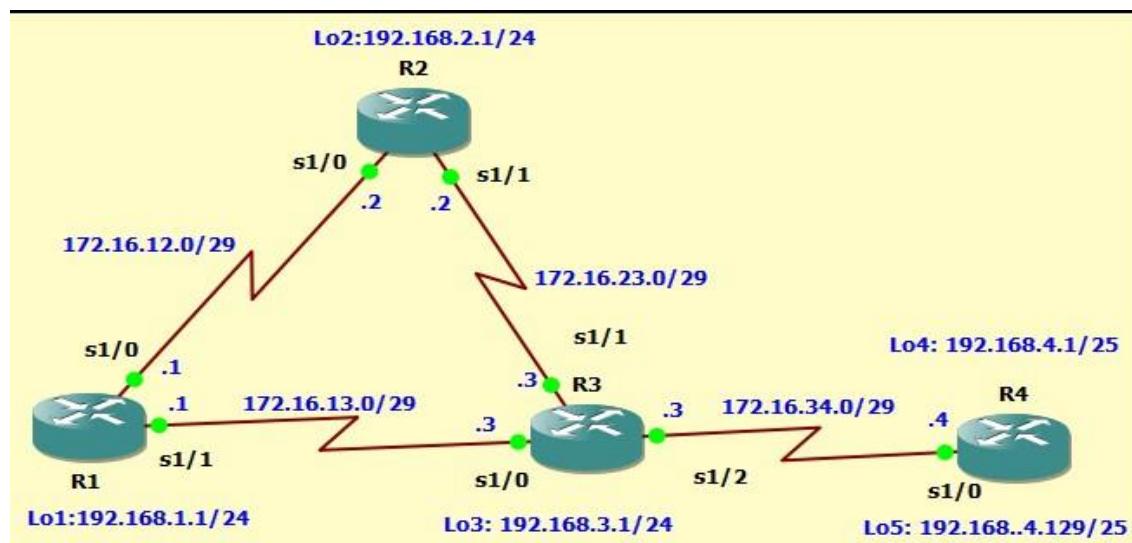
no exec

line vty 0 4

```
exec-timeout 5 0
password 7 070C285F4D060F110E020A1F17
login authentication TELNETLOGIN transport
input ssh
end
```

## Practical No - 5

**Aim :** 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.

#### Step 1: Configure loopbacks and assign addresses.

- x. Cable the network as shown in the topology diagram. Erase the startup configuration, and reload each router to clear previous configurations.
- y. 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** interface Lo1

```
ip address 192.168.1.1 255.255.255.0
interface Serial0/0/0
ip address 172.16.12.1 255.255.255.248
no shutdown
```

```
interface Serial0/0/1  
ip address 172.16.13.1 255.255.255.248 no  
shutdown
```

End

```
R1(config)#int Lo1  
R1(config-if)#ip address 192.168.1.1 255.255.255.0  
R1(config-if)#int s1/0  
R1(config-if)#ip address 172.16.12.1 255.255.255.248  
R1(config-if)#no shutdown  
  
R1(config-if)#int s1/1  
R1(config-if)#ip address 172.16.13.1 255.255.255.248  
R1(config-if)#no shutdown  
*May 19 23:06:21.987: %LINEPROTO-5-UPDOWN: Line proto  
R1(config-if)#no shutdown
```

**Router R2 interface**

```
Lo2  
ip address 192.168.2.1 255.255.255.0  
interface Serial0/0/0  
ip address 172.16.12.2 255.255.255.248  
no shutdown  
interface Serial0/0/1  
ip address 172.16.23.2 255.255.255.248  
no  
shutdown
```

End

```
R2(config)#int Lo2  
R2(config-if)#  
*May 19 23:06:13.083: %LINEPROTO-5-UPDOWN: Line proto  
R2(config-if)#ip address 192.168.2.1 255.255.255.0  
R2(config-if)#int s1/0  
R2(config-if)#ip address 172.16.12.2 255.255.255.248  
R2(config-if)#no shutdown  
  
R2(config)#int s1/1  
R2(config-if)#ip address 172.16.23.2 255.255.255.248  
R2(config-if)#no shutdown
```

**Router R3 interface**

Lo3

ip address 192.168.3.1 255.255.255.0  
interface Serial0/0/0

ip address 172.16.13.3 255.255.255.248

no shutdown interface

Serial0/0/1 ip address 172.16.23.3

255.255.255.248

no shutdown interface

Serial0/1/0 ip address 172.16.34.3

255.255.255.248 no

shutdown

End

```
R3(config)#int Lo3
R3(config-if)#
*May 19 23:07:08.351: %LINEPROTO-5-UPDOWN: Line proto
R3(config-if)#ip address 192.168.3.1 255.255.255.0
R3(config-if)#int s1/0
R3(config-if)#ip address 172.16.13.3 255.255.255.248
R3(config-if)#no shutdown
```

```
R3(config-if)#int s1/1
R3(config-if)#ip address 172.16.23.3 255.255.255.248
R3(config-if)#no shutdown
```

```
R3(config-if)#int s1/2
R3(config-if)#ip address 172.16.34.3 255.255.255.248
R3(config-if)#no shutdown
R3(config-if)#exit
```

**Router R4 interface Lo4 ip address**

192.168.4.1 255.255.255.128

interface Lo5

ip address 192.168.4.129 255.255.255.128

interface Serial0/0/0

ip address 172.16.34.4 255.255.255.248

no

shutdown

End

```
R4(config)#int lo4
R4(config-if)#
*May 19 23:08:16.239: %LINEPROTO-5-UPDOWN: Line protocol on interface Lo4 changed state to up
R4(config-if)#ip address 192.168.4.1 255.255.255.128
R4(config-if)#interface Lo5
R4(config-if)#
*May 19 23:08:32.527: %LINEPROTO-5-UPDOWN: Line protocol on interface Lo5 changed state to up
R4(config-if)#ip address 192.168.4.129 255.255.255.128
R4(config-if)#int s1/0
R4(config-if)#ip address 172.16.34.4 255.255.255.248
R4(config-if)#no shutdown
```

- z. 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	unset	administratively down	down
Serial1/0	172.16.13.3	YES	manual	up	up
Serial1/1	172.16.23.3	YES	manual	up	up
Serial1/2	172.16.34.3	YES	manual	up	up
Serial1/3	unassigned	YES	unset	administratively down	down
Loopback3	192.168.3.1	YES	manual	up	up

R3# show protocols

```
R3#show protocols
Global values:
  Internet Protocol routing is enabled
FastEthernet0/0 is administratively down, line protocol is down
  Internet address is 172.16.13.3/29
Serial1/0 is up, line protocol is up
  Internet address is 172.16.13.3/29
Serial1/1 is up, line protocol is up
  Internet address is 172.16.23.3/29
Serial1/2 is up, line protocol is up
  Internet address is 172.16.34.3/29
Serial1/3 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

```
R3#show interfaces description
Interface          Status      Protocol Description
Fa0/0              admin down  down
Se1/0              up         up
Se1/1              up         up
Se1/2              up         up
Se1/3              admin down down
```

### Step 3: Configure basic EIGRP.

- aa. Implement EIGRP AS 1 over the serial and loopback interfaces as you have configured it for the other EIGRP labs.
- bb. 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
```

```
R1(config)#router eigrp 1
R1(config-router)#network 192.168.1.0
R1(config-router)#network 172.16.12.0 0.0.0.7
R1(config-router)#network 172.16.13.0 0.0.0.7
R1(config-router)#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
```

```
R2(config)#router eigrp 1
R2(config-router)#network 192.168.2.0
R2(config-router)#network 172.16.12.0 0.0.0.7
R2(config-router)#network 172.16.23.0 0.0.0.7
R2(config-router)#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
```

```
R3(config)#router eigrp 1
R3(config-router)#network 192.168.3.0
R3(config-router)#network 172.16.13.0 0.0.0.7
R3(config-router)#network 172.16.23.0 0.0.0.7
R3(config-router)#network 172.16.34.0 0.0.0.7
R3(config-router)#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
```

```
R4(config)#router eigrp 1
R4(config-router)#network 192.168.4.0
R4(config-router)#network 172.16.34.0 0.0.0.7
R4(config-router)#no auto-summary
```

You should see EIGRP neighbor relationship messages being generated.

**Step 4: Verify EIGRP connectivity.**

- cc. Verify the configuration by using the **show ip eigrp neighbors** command to check which routers have EIGRP adjacencies. R1# **show ip eigrp neighbors**

```
R1#show ip eigrp neighbors
IP-EIGRP neighbors for process 1
          =   =
H  Address           Interface      Hold Uptime    SRTT     RTO  Q  Seq
   (sec)
1  172.16.13.3       Se1/1        11 00:00:31   26    200  0  18
0  172.16.12.2       Se1/0        12 00:00:44   37    222  0  13
```

R2# **show ip eigrp neighbors**

```
R2#show ip eigrp neighbors
*May 19 23:13:42.783: %SYS-5-CONFIG_I: Configured from console by console
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        Se1/1            11 00:00:50   41    246  0  20
  0  172.16.12.1        Se1/0            11 00:01:04   30    200  0  18
R2#
```

R3# show ip eigrp neighbors

```
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        Se1/2            12 00:00:44   48    288  0  6
  1  172.16.23.2        Se1/1            11 00:00:58   26    200  0  19
  0  172.16.13.1        Se1/0            12 00:00:58   281   1686 0  20
R3#
```

R4# show ip eigrp neighbors

```
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        Se1/0            10 00:00:55   23    200  0  26
R4#
```

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

R1# tclsh

```
R1#tclsh
R1(tcl)#foreach address {
+>(tcl)#172.16.12.1
+>(tcl)#172.16.12.2
+>(tcl)#172.16.13.1
+>(tcl)#172.16.13.3
+>(tcl)#172.16.23.2
+>(tcl)#172.16.23.3
+>(tcl)#172.16.34.3
+>(tcl)#172.16.34.4
+>(tcl)#192.168.1.1
+>(tcl)#192.168.2.1
+>(tcl)#192.168.3.1
+>(tcl)#192.168.4.1
+>(tcl)#192.168.4.129
+>(tcl)#} { ping $address }
```

```
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.12.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 44/61/76 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.12.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 16/27/40 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.13.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 40/58/80 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.13.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 20/31/44 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.23.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 20/28/32 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.23.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 24/40/48 ms
Type escape sequence to abort.
```

### **Step 5: Verify the current path.**

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

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

```
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 le
      ia - IS-IS inter area, * - candidate default, U - per-user stati
      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 [90/2681856] via 172.16.13.3, 00:01:50, Serial1/1
D        172.16.23.0 [90/2681856] via 172.16.13.3, 00:01:50, Serial1/1
                  [90/2681856] via 172.16.12.2, 00:01:50, Serial1/0
C        172.16.12.0 is directly connected, Serial1/0
C        172.16.13.0 is directly connected, Serial1/1
D        192.168.4.0/24 [90/2809856] via 172.16.13.3, 00:01:38, Serial1/1
C        192.168.1.0/24 is directly connected, Loopback1
D        192.168.2.0/24 [90/2297856] via 172.16.12.2, 00:01:50, Serial1/0
D        192.168.3.0/24 [90/2297856] via 172.16.13.3, 00:01:50, Serial1/1
```

R4# traceroute 192.168.1.1 source 192.168.4.1

```
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 36 msec 32 msec 32 msec
 2 172.16.13.1 28 msec 56 msec 84 msec
```

R4# traceroute 192.168.1.1 source 192.168.4.129

```
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 44 msec 28 msec 28 msec
 2 172.16.13.1 64 msec 28 msec 64 msec
```

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**

```
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
      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, Serial1/2
C    172.16.23.0 is directly connected, Serial1/1
D    172.16.12.0 [90/2681856] via 172.16.23.2, 00:02:29, Serial1/1
                  [90/2681856] via 172.16.13.1, 00:02:29, Serial1/0
C    172.16.13.0 is directly connected, Serial1/0
D    192.168.4.0/24 [90/2297856] via 172.16.34.4, 00:02:17, Serial1/2
D    192.168.1.0/24 [90/2297856] via 172.16.13.1, 00:02:29, Serial1/0
D    192.168.2.0/24 [90/2297856] via 172.16.23.2, 00:02:29, Serial1/1
C    192.168.3.0/24 is directly connected, Loopback3
```

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

**show interfaces serial0/0/0**

```
R3#show int s1/0
Serial1/0 is up, line protocol is up
  Hardware is M4T
  Internet address is 172.16.13.3/29
  MTU 1500 bytes, BW 1544 Kbit/sec, DLY 20000 usec,
```

```

Routing Descriptor Blocks:
 172.16.13.1 (Serial1/0), from 172.16.13.1, Send flag is 0x0
   Composite metric is (2297856/128256), Route is Internal
   Vector metric:
     Minimum bandwidth is 1544 Kbit
     Total delay is 25000 microseconds
     Reliability is 255/255
     Load is 1/255
     Minimum MTU is 1500
     Hop count is 1
 172.16.23.2 (Serial1/1), from 172.16.23.2, Send flag is 0x0

```

gg. 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**

```

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
  Routing Descriptor Blocks:
    172.16.13.1 (Serial1/0), from 172.16.13.1, Send flag is 0x0
      Composite metric is (2297856/128256), Route is Internal
      Vector metric:
        Minimum bandwidth is 1544 Kbit
        Total delay is 25000 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 1
    172.16.23.2 (Serial1/1), from 172.16.23.2, Send flag is 0x0

```

#### **Step 6: Configure PBR to provide path control.**

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.

hh. 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**

```
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
```

R3(config)#

- ii. 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)# **description RM to forward LAN B traffic to R1**

R3(config-route-map)# **match ip address PBR-**

**ACL** R3(config-route-map)# **set ip nexthop**

**172.16.13.1** R3(config-route-map)# **exit**

```
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
```

- jj. 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**

```
R3(config)#int s1/2
R3(config-if)#ip policy route-map R3-to-R1
R3(config-if)#end
```

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

```
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
```

### Step 7: Test the policy.

- ll. 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**

```
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
```

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

R3# **debug ip policy ?**

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

R3# **debug ip policy 1**

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

nn. 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**

```
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 40 msec 12 msec 32 msec
 2 172.16.13.1 60 msec 48 msec 88 msec
```

```
R3#
*May 19 23:17:36.819: IP: s=192.168.4.1 (Serial1/2), d=192.168.1.1,
*May 19 23:17:36.851: IP: s=192.168.4.1 (Serial1/2), d=192.168.1.1,
*May 19 23:17:36.879: IP: s=192.168.4.1 (Serial1/2), d=192.168.1.1,
*May 19 23:17:36.915: IP: s=192.168.4.1 (Serial1/2), d=192.168.1.1,
g
*May 19 23:17:36.971: IP: s=192.168.4.1 (Serial1/2), d=192.168.1.1,
g
*May 19 23:17:37.031: IP: s=192.168.4.1 (Serial1/2), d=192.168.1.1
R3#, len 28, FIB policy rejected(no match) - normal forwarding
R3#
```

oo. 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**

```
R3# .168.4.129 (Serial1/2), d=192.168.1.1, len 28, FIB policy match
*May 19 23:17:55.763: IP: s=192.168.4.129 (Serial1/2), d=192.168.1.1,
*May 19 23:17:55.763: IP: s=192.168.4.129 (Serial1/2), d=192.168.1.1,
*May 19 23:17:55.823: IP: s=192.168.4.129 (Serial1/2), d=192.168.1.1,
*May 19 23:17:55.823: IP: s=192.168.4.129 (Serial1/2), d=192.168.1.1,
*May 19 23:17:55.827: IP: s=192.168.4.129 (Serial1/2), d=192.168.1.1,
*May 19 23:17:55.883: IP: s=192.168.4.129 (Serial1/2), d=192.168.1.1,
*May 19 23:17:55.883: IP: s=192.168.4.129 (Serial1/2), d=192.168.1.1,
*May 19 23:17:55.887: IP: s=192.168.4.129 (Serial1/2), d=192.168.1.1,
```

```
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 40 msec 28 msec 32 msec
 2 172.16.13.1 60 msec 64 msec 32 msec
```

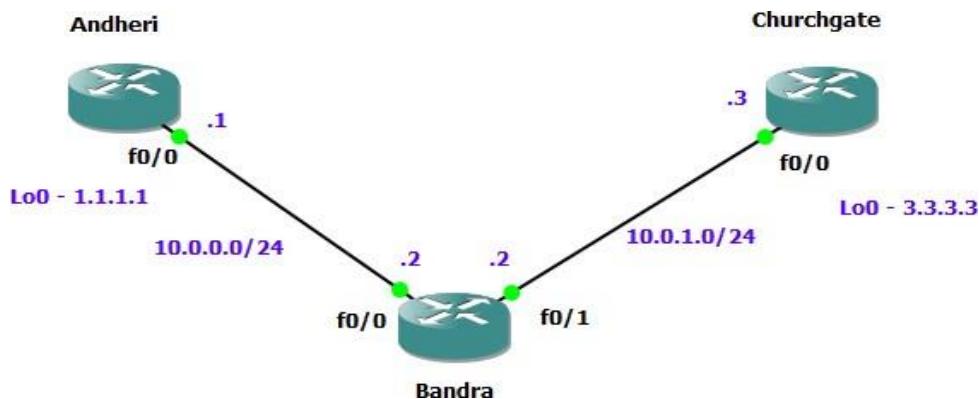
pp. On R3, display the policy and matches using the **show route-map** command. R3# **show route-map**

```
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: 6 packets, 192 bytes
```

## Practical No - 6

**Aim:** Cisco MPLS Configuration

**Topology :**



### 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 loopback to loopback connectivity between R1 and R3

```
Andheri(config)#int lo0
Andheri(config-if)#ip add 1.1.1.1 255.255.255.255
Andheri(config-if)#ip ospf 1 area 0
Andheri(config-if)#
Andheri(config-if)#int f0/0
Andheri(config-if)#ip add 10.0.0.1 255.255.255.0
Andheri(config-if)#no shut
Andheri(config-if)#ip ospf 1 area 0
```

```
Bandra(config)#int lo0
Bandra(config-if)#
Bandra(config-if)#ip add 2.2.2.2 255.255.255.255
Bandra(config-if)#ip ospf 1 area 0
Bandra(config-if)#
Bandra(config-if)#int f0/0
Bandra(config-if)#ip add 10.0.0.2 255.255.255.0
Bandra(config-if)#no shut
Bandra(config-if)#ip ospf 1 area 0
Bandra(config-if)#
Bandra(config-if)#int f0/1
Bandra(config-if)#ip add 10.0.1.2 255.255.255.0
Bandra(config-if)#no shut
Bandra(config-if)#ip ospf 1 area 0
```

```
Churchgate(config)#int lo0
Churchgate(config-if)#ip add 3.3.3.3 255.255.255.255
Churchgate(config-if)#ip ospf 1 area 0
Churchgate(config-if)#
Churchgate(config-if)#int f0/0
Churchgate(config-if)#ip add 10.0.1.3 255.255.255.0
Churchgate(config-if)#no shut
Churchgate(config-if)#ip ospf 1 area 0
```

You should now have full ip connectivity between R1, R2, R3 to verify this we need to see if we can ping between the loopbacks of R1 and R3

```
Andheri#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 = 20/52/64 ms
```

## 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

```
Andheri(config)#router ospf 1
Andheri(config-router)#mpls ldp autoconfig
```

```
Bandra(config)#router ospf 1
Bandra(config-router)#mpls ldp autoconfig
```

```
Churchgate(config)#router ospf 1
Churchgate(config-router)#mpls ldp autoconfig
```

You should see log messages coming up showing the LDP neighbors are up.

```
Bandra#
*May 29 17:03:09.559: %SYS-5-CONFIG_I: Configured from console by console
Bandra#
*May 29 17:03:28.631: %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 mpls and using LDP

```
Bandra#sh mpls int
Interface          IP           Tunnel   BGP Static Operational
FastEthernet0/0    Yes (ldp)    No        No  No      Yes
FastEthernet0/1    Yes (ldp)    No        No  No      Yes
Bandra#
```

You can also verify the LDP neighbors with the sh mpls ldp neighbors command.

```
Bandra#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 - 2.2.2.2.25712
  State: Oper; Msgs sent/rcvd: 9/9; Downstream
  Up time: 00:01:23
  LDP discovery sources:
    FastEthernet0/0, Src IP addr: 10.0.0.1
    Addresses bound to peer LDP Ident:
      10.0.0.1      1.1.1.1
Peer LDP Ident: 3.3.3.3:0; Local LDP Ident 2.2.2.2:0
  TCP connection: 3.3.3.3.50470 - 2.2.2.2.646
  State: Oper; Msgs sent/rcvd: 8/8; Downstream
  Up time: 00:00:54
  LDP discovery sources:
    FastEthernet0/1, Src IP addr: 10.0.1.3
    Addresses bound to peer LDP Ident:
      10.0.1.3      3.3.3.3
```

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.

```
Andheri#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] 20 msec 60 msec 60 msec
2 10.0.1.3 60 msec 60 msec 60 msec
```

### Step 3 – MPLS BGP Configuration between R1 and R3

We need to establish a Multi Protocol BGP session between R1 and R3 this is done by configuring the vpng4 address family as below

```
Andheri(config)#router bgp 1
Andheri(config-router)#neighbor 3.3.3.3 remote-as 1
Andheri(config-router)#neighbor 3.3.3.3 update-source Loopback0
Andheri(config-router)#no auto-summary
Andheri(config-router)#
Andheri(config-router)#address-family vpng4
Andheri(config-router-af)#neighbor 3.3.3.3 activate
Churchgate(config)#router bgp 1
Churchgate(config-router)#neighbor 1.1.1.1 remote-as 1
Churchgate(config-router)#neighbor 1.1.1.1
*May 29 17:06:19.459: %BGP-5-ADJCHANGE: neighbor 1.1.1.1 Up
Churchgate(config-router)#neighbor 1.1.1.1 update-source loopback 0
Churchgate(config-router)#no auto-summary
Churchgate(config-router)#address-family vpng4
Churchgate(config-router-af)#neighbor 1.1.1.1 activate
```

To verify the BGP session between R1 and R3 issue the command sh bgp vpng4 unicast all summary

```
Andheri#sh bgp vpng4 unicast all summary
BGP router identifier 1.1.1.1, local AS number 1
BGP table version is 1, main routing table version 1

Neighbor      V      AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
3.3.3.3        4      1      5      6      1      0      0  00:00:30      0
```

### Step 4 – Add two more routers, create VRFs

We will add two more routers into the topology so it now looks like the final topology

```
Borivali(config)#int lo0
Borivali(config-if)#ip ad
*May 29 17:13:47.223: %LINEPROTO-5-UPDOWN: Line protocol o
Borivali(config-if)#ip address 4.4.4.4 255.255.255.255
Borivali(config-if)#ip ospf 2 area 2
Borivali(config-if)#int f0/0
Borivali(config-if)#ip addressss 192.168.1.4 255.255.255.0
^
% Invalid input detected at '^' marker.

Borivali(config-if)#ip address 192.168.1.4 255.255.255.0
Borivali(config-if)#ip ospf 2 area 2
Borivali(config-if)#no shut
```

```

Andheri(config)#int f0/1
Andheri(config-if)#no shut
Andheri(config-if)#ip address
*May 29 17:14:16.199: %LINK-3-UPDOWN: Interface FastEther
*May 29 17:14:17.199: %LINEPROTO-5-UPDOWN: Line protocol
Andheri(config-if)#ip address 192.168.1.1 255.255.255.0

Andheri(config-if)#ip vrf RED
Andheri(config-vrf)#rd 4:4
Andheri(config-vrf)#route-target both 4:4

Andheri(config-vrf)#int f0/1
Andheri(config-if)#ip vrf forwarding RED
% Interface FastEthernet0/1 IP address 192.168.1.1 removed due to enabling VRF RED

Andheri#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

```

If you issue the command sh ip route this shows the routes in the global table and you will notice that you do not see 192.168.1.0/24

```

Andheri#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
      ia - IS-IS inter area, * - candidate default, U - per-user
      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 [110/2] via 10.0.0.2, 00:19:39, FastEthernet0/0
      3.0.0.0/32 is subnetted, 1 subnets
O        3.3.3.3 [110/3] via 10.0.0.2, 00:18:35, 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 [110/2] via 10.0.0.2, 00:18:45, FastEthernet0/0

Andheri#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
      ia - IS-IS inter area, * - candidate default, U - per-user
      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

```

We just need to enable OSPF on this interface and get the loopback address for R4 in the VRF RED routing table before proceeding.

```
Andheri(config)#int f0/1
Andheri(config-if)#ip ospf 2 area 2
```

If we now check the routes in the VRF RED routing table you should see 4.4.4.4 in there as well.

```
Andheri#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
      ia - IS-IS inter area, * - candidate default, U - per-user static
      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 [110/2] via 192.168.1.4, 00:00:11, FastEthernet0/1
C      192.168.1.0/24 is directly connected, FastEthernet0/1
```

```
Andheri#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
      ia - IS-IS inter area, * - candidate default, U - per-user static
      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 [110/2] via 10.0.0.2, 00:28:18, FastEthernet0/0
  3.0.0.0/32 is subnetted, 1 subnets
O      3.3.3.3 [110/3] via 10.0.0.2, 00:27:14, 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 [110/2] via 10.0.0.2, 00:27:24, FastEthernet0/0
```

```
Andheri#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
      ia - IS-IS inter area, * - candidate default, U - per-user static
      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 [110/2] via 192.168.1.4, 00:07:42, FastEthernet0/1
C      192.168.1.0/24 is directly connected, FastEthernet0/1
```

We now need to repeat this process for R3 & R6 Router 6 will peer OSPF using process number 2 to a VRF configured on R3. It will use the local site addressing to 192.168.2.0/24

```
Mahim(config)#INT LO0
Mahim(config-if)#
*May 29 17:18:58.903: %LINEPROTO-5-UPDOWN: Line protocol
Mahim(config-if)#ip add 6.6.6.6 255.255.255.255
Mahim(config-if)#ip ospf 2 area 2
Mahim(config-if)#int f0/0
Mahim(config-if)#ip add 192.168.2.6 255.255.255.0
Mahim(config-if)#ip ospf 2 area 2
Mahim(config-if)#no shut
```

```
Churchgate(config)#int f0/1
Churchgate(config-if)#no shut
Churchgate(config-if)#ip add
*May 29 17:23:19.111: %LINK-3-UPDOWN: Interface FastEthernet0/1 is up
*May 29 17:23:20.111: %LINEPROTO-5-UPDOWN: Line protocol
Churchgate(config-if)#ip add 192.168.2.3 255.255.255.0
```

We also need to configure a VRF onto R3 as well.

```
Churchgate(config-if)#ip vrf RED
Churchgate(config-vrf)#rd 4:4
Churchgate(config-vrf)#route-target both 4:4

Churchgate(config-vrf)#int f0/1
Churchgate(config-if)#ip vrf forwarding RED
% Interface FastEthernet0/1 IP address 192.168.2.3 removed due to enabling VRF RED
Churchgate(config-if)#int f0/1
Churchgate(config-if)#ip add 192.168.2.1 255.255.255.0
```

```
Churchgate#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
```

Check the router in vrf RED

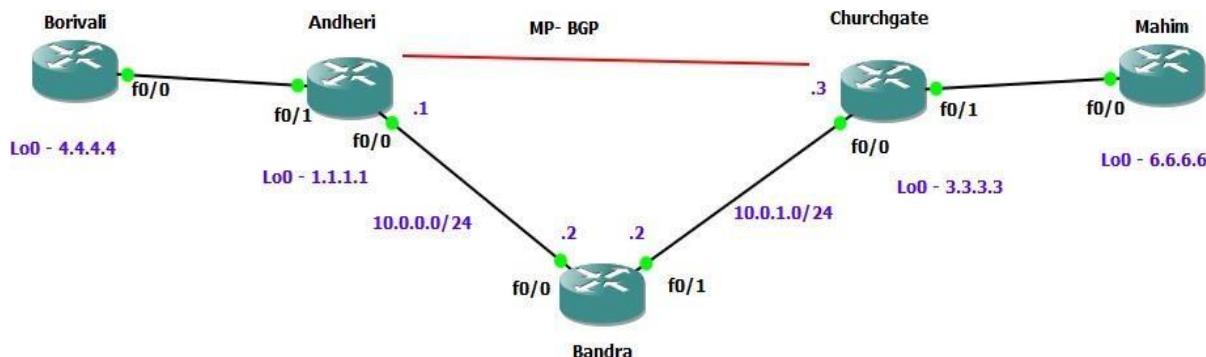
```
Churchgate#sh ip route vrf RED

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
      o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

  6.0.0.0/32 is subnetted, 1 subnets
O      6.6.6.6 [110/2] via 192.168.2.6, 00:01:10, FastEthernet0/1
C      192.168.2.0/24 is directly connected, FastEthernet0/1
```

Ok so we have come a long way now let's review the current situation. We now have this setup



```

Borivali#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 subnet route
      o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

        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 across the MPLS Check the routes on R1

```

Andheri#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 subnet 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 [110/2] via 10.0.0.2, 00:28:18, FastEthernet0/0
        3.0.0.0/32 is subnetted, 1 subnets
          O        3.3.3.3 [110/3] via 10.0.0.2, 00:27:14, 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 [110/2] via 10.0.0.2, 00:27:24, FastEthernet0/0
  
```

```
Andheri#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  
ia - IS-IS inter area, \* - candidate default, U - per-user sta  
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 [110/2] via 192.168.1.4, 00:07:42, FastEthernet0/1
C      192.168.1.0/24 is directly connected, FastEthernet0/1
```

```
Andheri(config)#router bgp 1
Andheri(config-router)#address-family ipv4 vrf RED
Andheri(config-router-af)#redistribute ospf 2
Andheri(config-router-af)#exit
Andheri(config-router)#end
```

```
Churchgate(config)#router bgp 1
Churchgate(config-router)#address-family ipv4 vrf RED
Churchgate(config-router-af)#redistribute ospf 2
Churchgate(config-router-af)#end
```

```
Andheri#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, i
              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        2      32768 ??
*>i6.6.6.6/32       3.3.3.3           2      100    0 ??
*> 192.168.1.0       0.0.0.0           0      32768 ??
*>i192.168.2.0      3.3.3.3           0      100    0 ??
```

```
Churchgate#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,
              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           2      100    0 ??
*> 6.6.6.6/32        192.168.2.6        2      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 MPLS and R2 is not in the picture)

```
Andheri(config)#router ospf 2
Andheri(config-router)#redistribute bgp 1 subnets
Churchgate(config)#router ospf 2
Churchgate(config-router)#redistribute bgp 1 subnets
```

Before we do let's see what the routing table look like on R4

```
Borivali#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
      o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

        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 [110/3] via 192.168.1.1, 00:00:50, FastEthernet0/0
C            192.168.1.0/24 is directly connected, FastEthernet0/0
O IA 192.168.2.0/24 [110/2] via 192.168.1.1, 00:00:50, FastEthernet0/0
```

Do the same step of on R6

```
Mahim#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
      o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

        4.0.0.0/32 is subnetted, 1 subnets
O IA      4.4.4.4 [110/3] via 192.168.2.1, 00:00:22, 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 [110/2] via 192.168.2.1, 00:00:22, FastEthernet0/0
C            192.168.2.0/24 is directly connected, FastEthernet0/0
```

Lets chevk ping command

```
Borivali#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 = 112/120/128 ms
```

Which we can – to prove this is going over the mpls and be label switched and not routed, lets do a trace

```
Borivali#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 32 msec 24 msec
2 10.0.0.2 [MPLS: Labels 17/19 Exp 0] 112 msec 136 msec 124 msec
3 192.168.2.1 [MPLS: Label 19 Exp 0] 72 msec 92 msec 92 msec
4 192.168.2.6 140 msec 124 msec 124 msec
```