NURTURING POTENTIAL

SAKET GYANPEETH'S

SAKET COLLEGE OF ARTS, SCIENCE AND COMMERCE

KALYAN (EAST)

ACADEMIC YEAR [2022 – 2023]

MSC. INFORMATION TECHNOLOGY

SEMESTER – II

1. BIG DATA ANALYTICS
2. MODERN NETWORKING
3. MICROSERVICES ARCHITECTURE
4. IMAGE PROCESSING

SUBMITTED BY

NA

AS PRESCRIPED BY

UNIVERSITY OF MUMBAI

BIG DATA ANALYTICS

NURTURING POTENTIAL

SAKET GYANPEETH'S

SAKET COLLEGE OF ARTS, SCIENCE, AND COMMERCE

(Permanently Affiliated to University of Mumbai)

NAAC Accredited B Grade

Saket Vidyanagari Marg, Chinchpada Road, Katemanivali, Kalyan (East) – 421306, Dist. Thane (MAH)

Department of Information Technology

CERTIFICATE

This is to certify that
NA of **MSc Information Technology Part-I**
Class has satisfactory carried out the required practical in the subject.

BIG DATA ANALYTICS

For the Academic year 2022-2023

Practical in Charge **Head of Department** **External Examiner**

INDEX

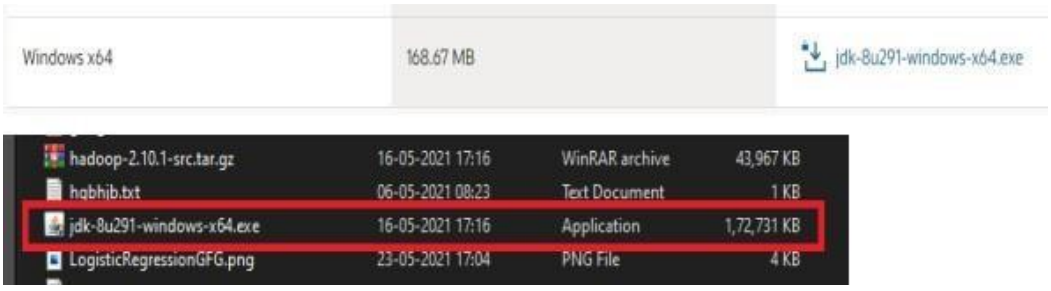
PRACT NO:	PRACTICAL	SIGN
1.	Install, configure and run Hadoop and HDFS	
2.	Implement Decision tree classification techniques	
3.	Classification using SVM	
4.	Implement an application that stores big data in Hbase / MongoDB and manipulates it using R / Python.	
5.	Write program in R of Naive baye's theorem	
6.	Write a program showing implementation of Regression model.	
7.	Write a Program showing clustering.	

PRACTICAL 1

Aim: Install, configure and run Hadoop and HDFS

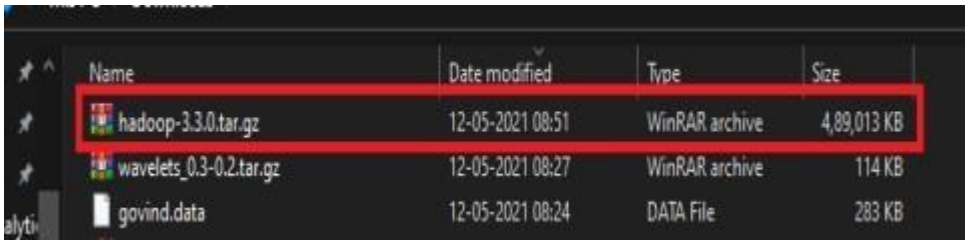
Description: Hadoop Installation.

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

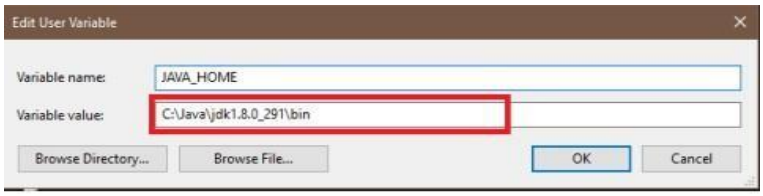


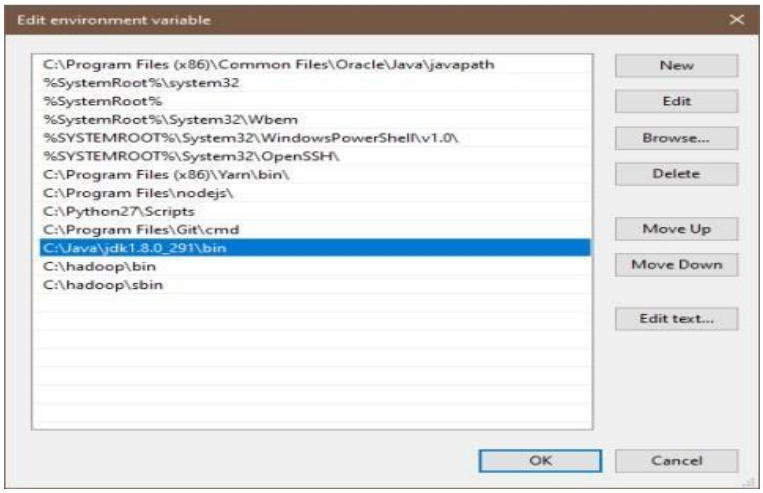
Step 2: Download Hadoop binaries from the official website. The binary package size is about 342 MB.

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

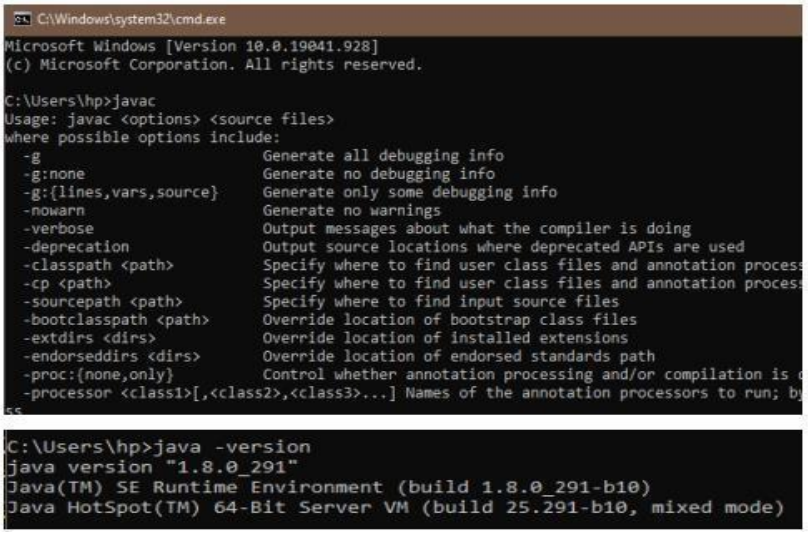


Step 4: When the "Advanced system settings" dialog appears, go to the "Advanced" tab and click on the "Environment variables" button located on the bottom of the dialog

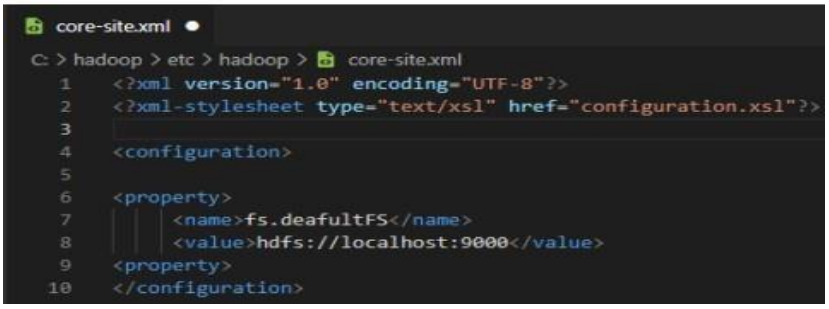




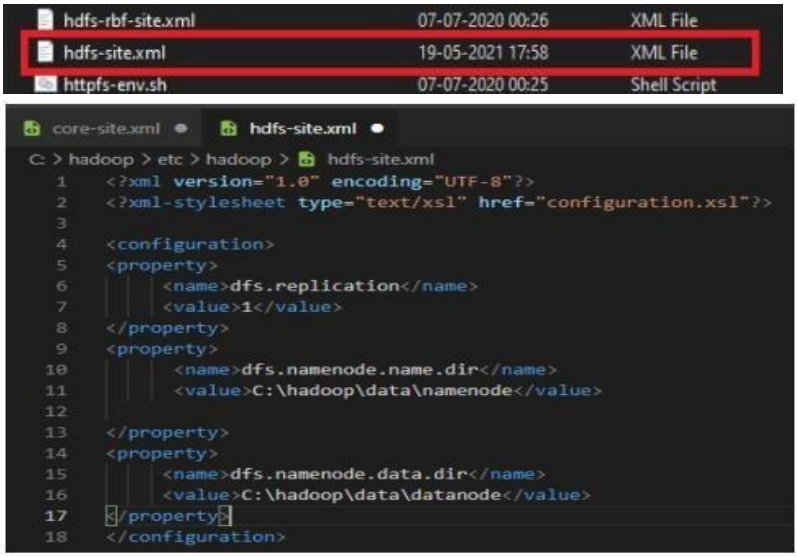
Step 5: Check the version of java



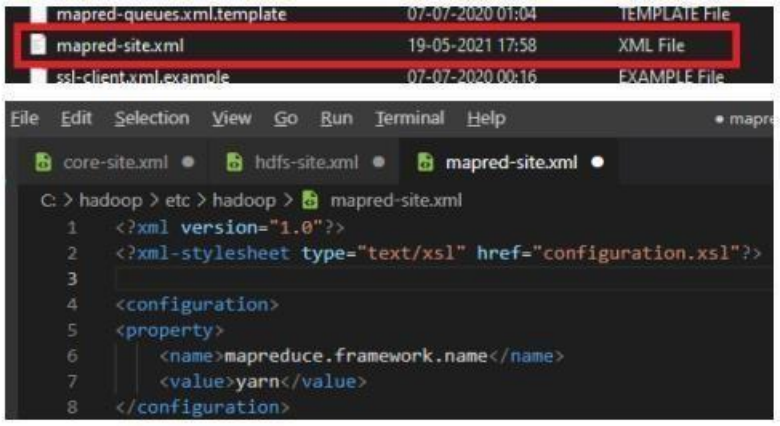
Step 6: Configuration core-site.xml



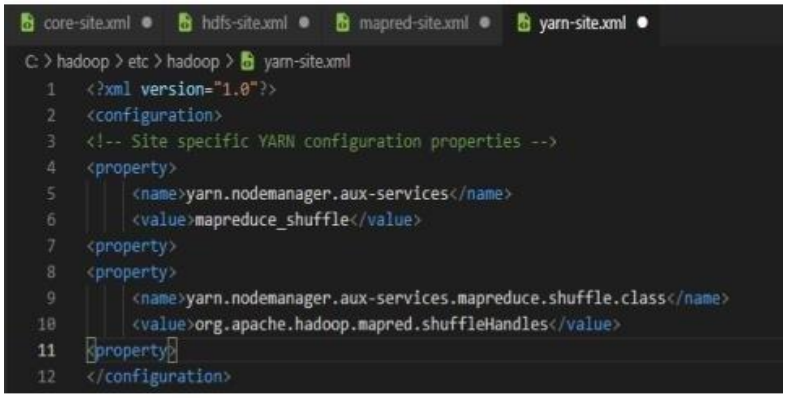
Step 7: Configuration core-site.xml



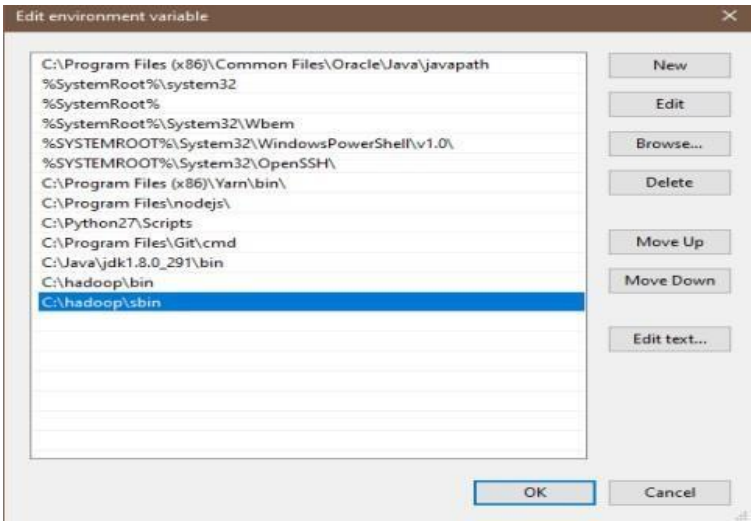
Step 8: Configuration core-site.xml



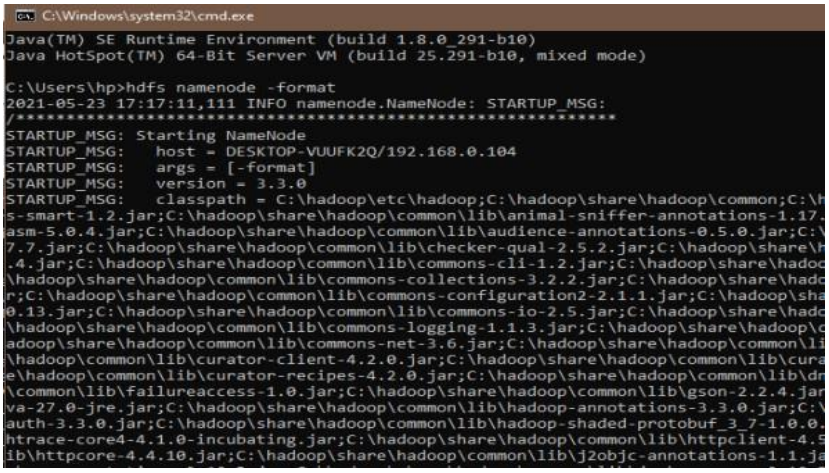
Step 9: Configuration core-site.xml



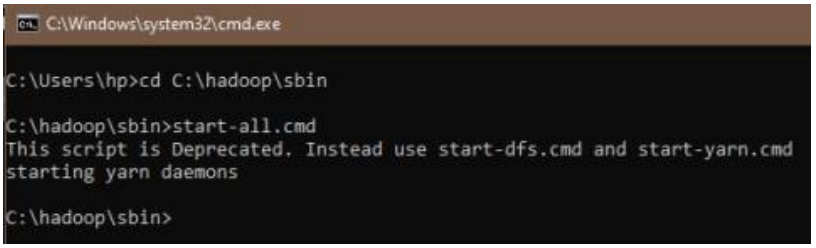
Step 10: When the "Advanced system settings" dialog appears, go to the "Advanced" tab and click on the "Environment variables" button located on the bottom of the dialog.



Step 11: let's check Hadoop install Successfully



Step 12: Let check bin



PRACTICAL 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 tree() which is used to create and analyse decision tree.

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

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

```
> 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)
```

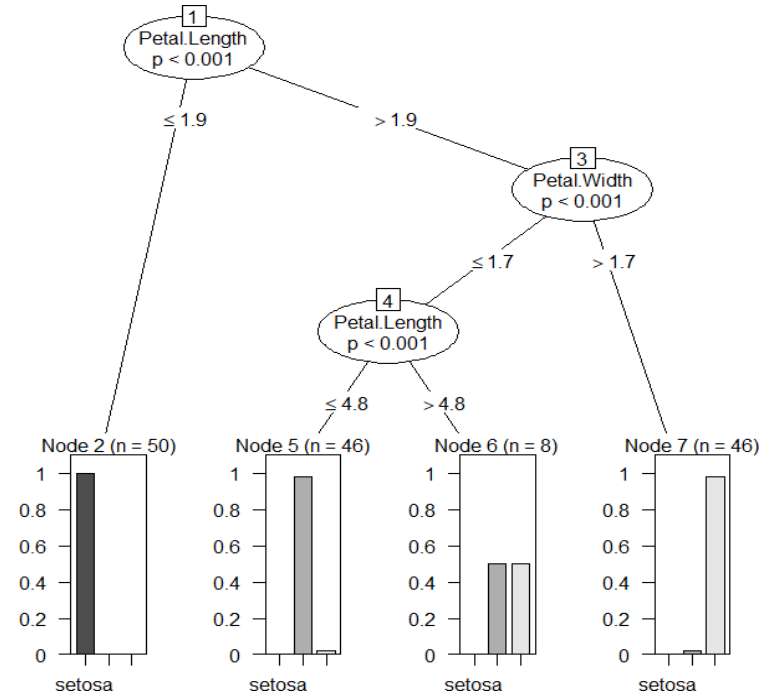
```
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ 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)
```

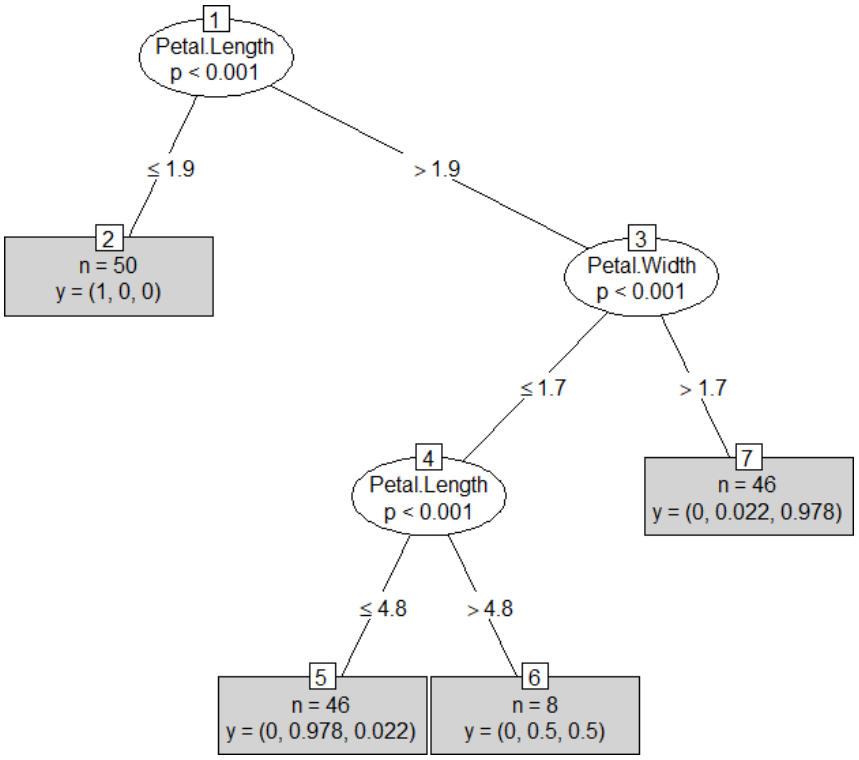
```
          Conditional inference tree with 4 terminal nodes

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 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:

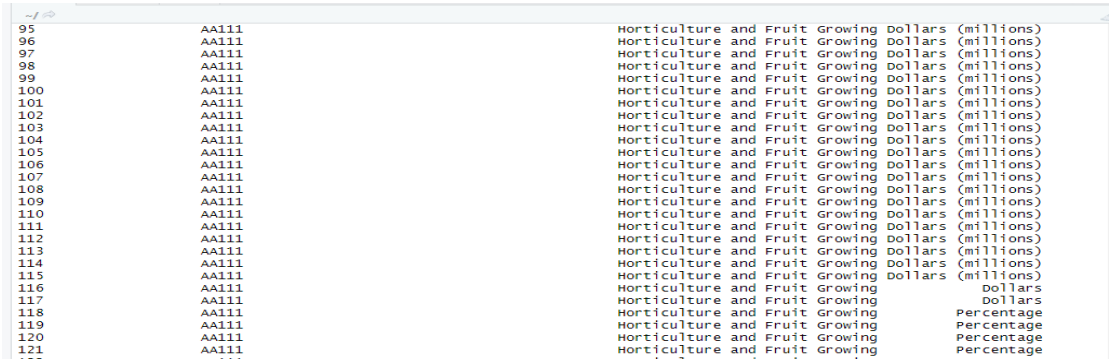
Step 1: Importing the dataset

getwd()

file<-'social.csv'

ds<- read.csv(file)

ds



95	AA111	Horticulture and Fruit Growing Dollars (millions)
96	AA111	Horticulture and Fruit Growing Dollars (millions)
97	AA111	Horticulture and Fruit Growing Dollars (millions)
98	AA111	Horticulture and Fruit Growing Dollars (millions)
99	AA111	Horticulture and Fruit Growing Dollars (millions)
100	AA111	Horticulture and Fruit Growing Dollars (millions)
101	AA111	Horticulture and Fruit Growing Dollars (millions)
102	AA111	Horticulture and Fruit Growing Dollars (millions)
103	AA111	Horticulture and Fruit Growing Dollars (millions)
104	AA111	Horticulture and Fruit Growing Dollars (millions)
105	AA111	Horticulture and Fruit Growing Dollars (millions)
106	AA111	Horticulture and Fruit Growing Dollars (millions)
107	AA111	Horticulture and Fruit Growing Dollars (millions)
108	AA111	Horticulture and Fruit Growing Dollars (millions)
109	AA111	Horticulture and Fruit Growing Dollars (millions)
110	AA111	Horticulture and Fruit Growing Dollars (millions)
111	AA111	Horticulture and Fruit Growing Dollars (millions)
112	AA111	Horticulture and Fruit Growing Dollars (millions)
113	AA111	Horticulture and Fruit Growing Dollars (millions)
114	AA111	Horticulture and Fruit Growing Dollars (millions)
115	AA111	Horticulture and Fruit Growing Dollars (millions)
116	AA111	Horticulture and Fruit Growing Dollars
117	AA111	Horticulture and Fruit Growing Dollars
118	AA111	Horticulture and Fruit Growing Percentage
119	AA111	Horticulture and Fruit Growing Percentage
120	AA111	Horticulture and Fruit Growing Percentage
121	AA111	Horticulture and Fruit Growing Percentage

Step 2: Selecting columns 3-5

>ds=ds[3:5]

>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)
> srt.seed(123)
Error in srt.seed(123) : could not find function "srt.seed"
> set.seed(123)
> split = sample.split(ds$Purchased, SplitRatio = 0.75)
> training_set = subset(ds, split=TRUE)
Warning message:
In subset.data.frame(ds, split = TRUE) :
 extra argument 'split' will be disregarded
> 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
```

Step 5: Feature Scaling

```
> 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     -1.52370715
4   -1.05994374     -0.33467023
5   -1.81569686      0.27591629
9   -1.24888202     -1.10593742
12  -1.15441288     -0.49535090
18   0.64050076     -1.33089035
19   0.73496990     -1.26661809
20   0.92390818     -1.23448195
22   0.82943904     -0.59175930
29  -0.87100546     -0.78457609
32  -1.05994374      2.23622040
34  -0.96547460     -0.75243996
35  -1.05994374      0.72582215
38  -0.77653633     -0.59175930
45  -0.96547460      0.53300536
```

Step 6: Fitting SVM to the training set

```
395  0.07368593     -0.27039797
400  1.01837732             NA
> install.packages('e1071')
also installing the dependency 'proxy'

warning in install.packages :
  the 'wininet' method is deprecated for http:// and https:// URLs
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.3/proxy_0.4-27.zip'
Content type 'application/zip' length 179492 bytes (175 KB)
downloaded 175 KB

warning in install.packages :
  the 'wininet' method is deprecated for http:// and https:// URLs
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.3/e1071_1.7-13.zip'
Content type 'application/zip' length 653422 bytes (638 KB)
downloaded 638 KB

package 'proxy' successfully unpacked and MD5 sums checked
package 'e1071' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
```

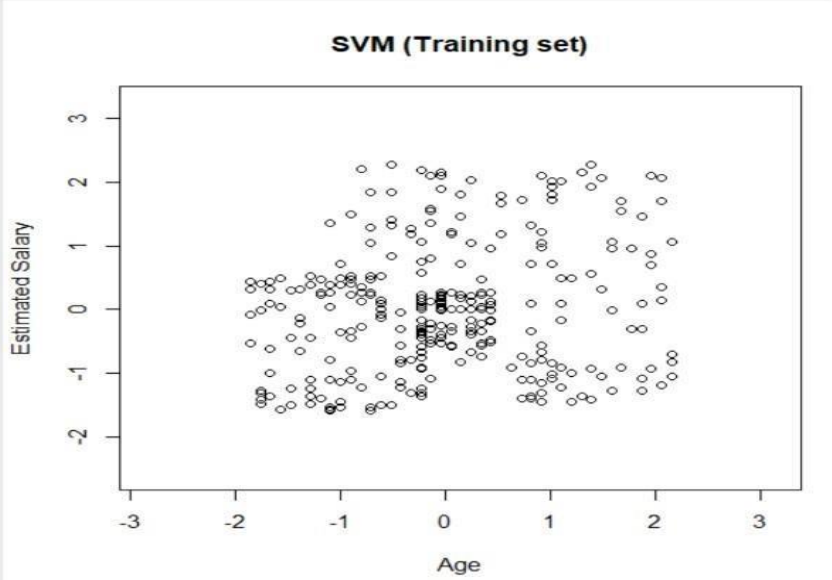
Step 7: Predicting the test set result

```
> 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
 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   0   1   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   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   1   0   1   1   1   1   1   0   1   1   1   0
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
```

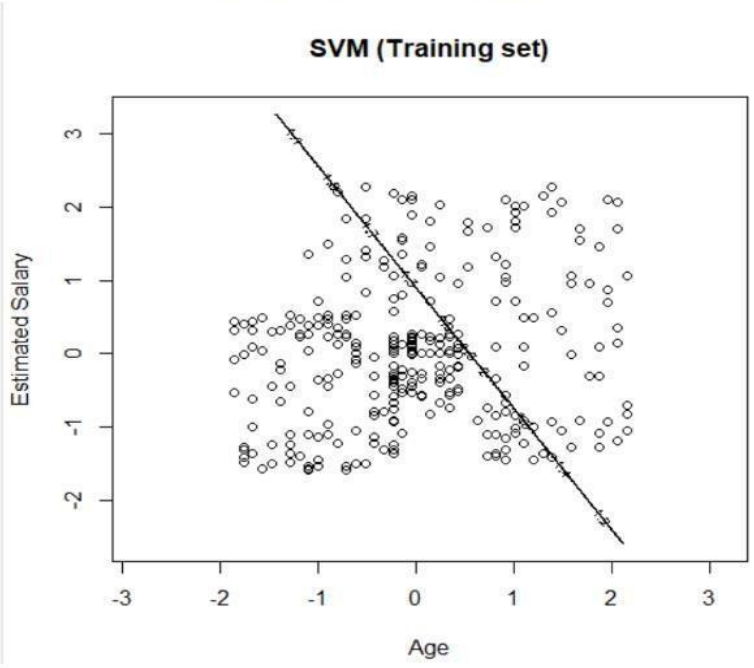
```
> cm = table(test_set[, 3], y_pred)
> cm
   y_pred
     0  1
  0 57  7
  1 13 23
```

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)
```

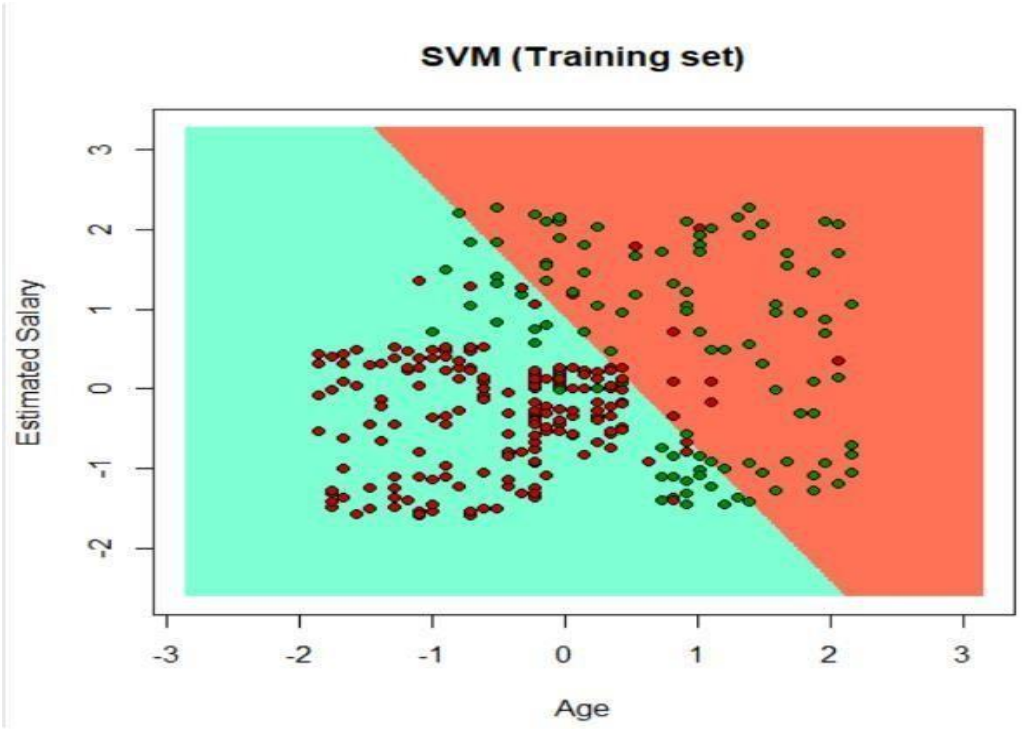


```
> grid_set = expand.grid(X1, X2)
> colnames(grid_set) = c('Age', 'EstimatedSalary')
> y_grid = predict(classifier, newdata = grid_set)
> plot(set[, -3],
+      main = 'SVM (Training set)',
+      xlab = 'Age', ylab = 'Estimated Salary',
+      xlim = range(X1), ylim = range(X2))
```




```
> 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 4

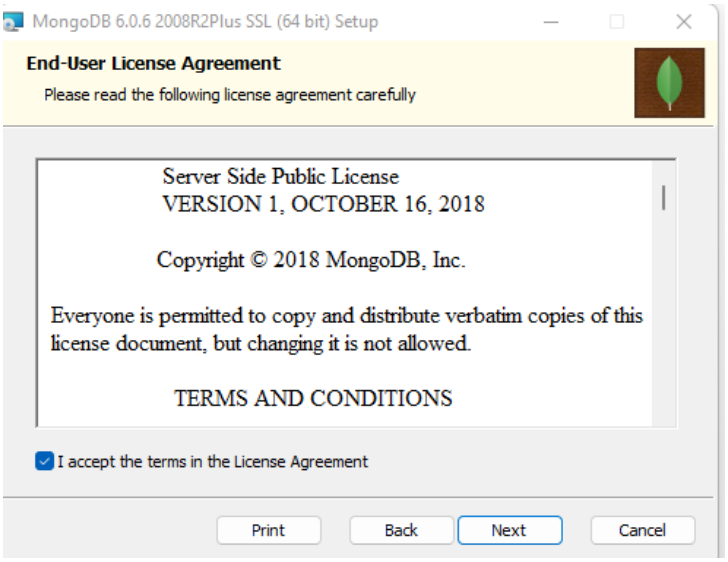
Aim: Implement an application that stores big data in Hbase / MongoDB and manipulates it using R / Python.

Part A

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

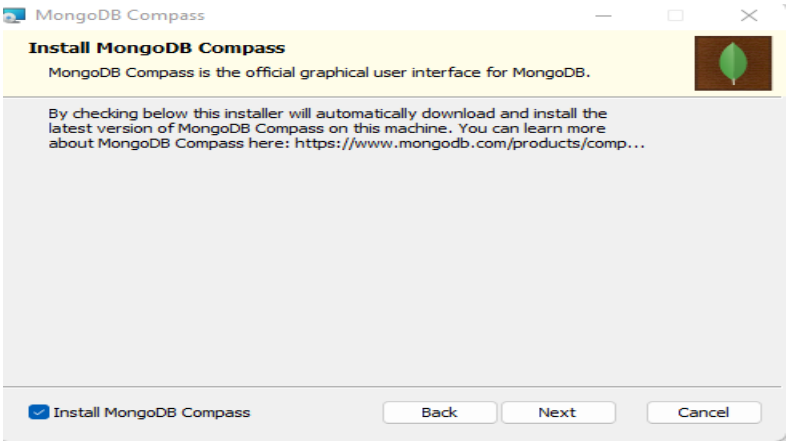
Step 1: **Download the and install Mongo database.**

Step 2: Once download is complete open the msi file. Click Next in the start up screen

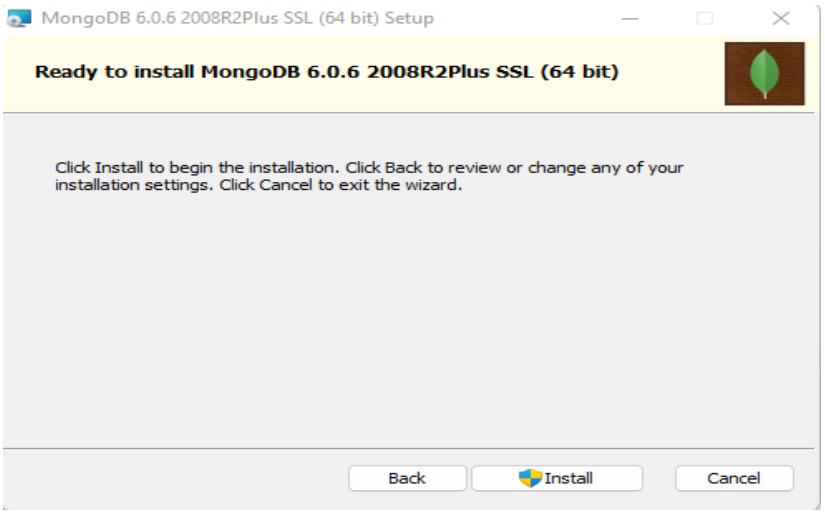


Step 3: 1. Accept the End-User License Agreement

2. Click Next



Step 4: Install



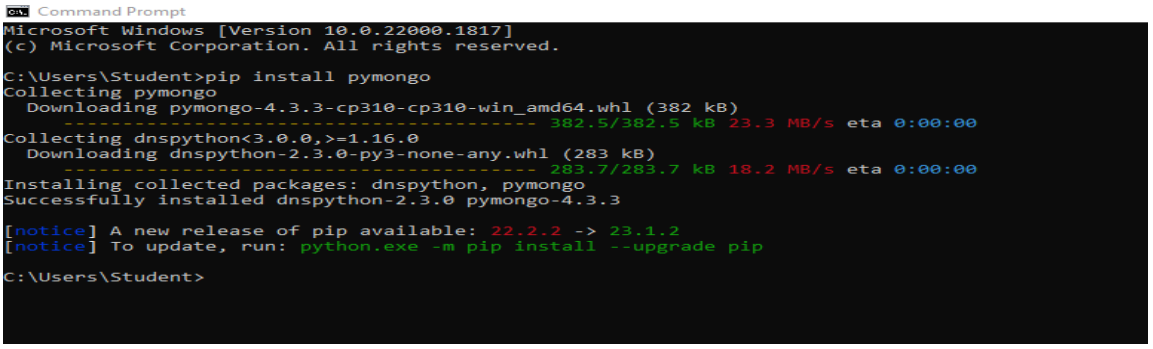
Test Mongodb

Step 1: Go to " C:\Program Files\MongoDB\Server\4.0\bin" and double-click on mongo.exe. Alternatively, you can also click on the MongoDB desktop item.

1. Create the directory where MongoDB will store it's files. From the command prompt run md \data\db . This is the default location. However, other locations can be specified using the --dbpath parameter. See the Mongo docs for more information.
    - C:\>md data
    - C:\md data\db
    - C:\Program Files\MongoDB\Server\4.05\bin>mongod.exe --dbpath "C:\data".
2. Start the mongodb daemon by running C:\mongodb\bin\mongod.exe in the Command Prompt. Or by running, C:\path\to\mongodb\bin\mongod.exe
3. Connect to MongoDB using the Mongo shell While the MongoDB daemon is running, from a different Command prompt window run C:\mongodb\bin\mongo.exe
    - C:\Program Files\MongoDB\Server\4.05\bin>mongod.exe --dbpath "C:\data"
    - C:\Program Files\MongoDB\Server\4.05\bin>mongo.exe

Part B

Step 1: Install PyMongo by below command. pip install pymongo.

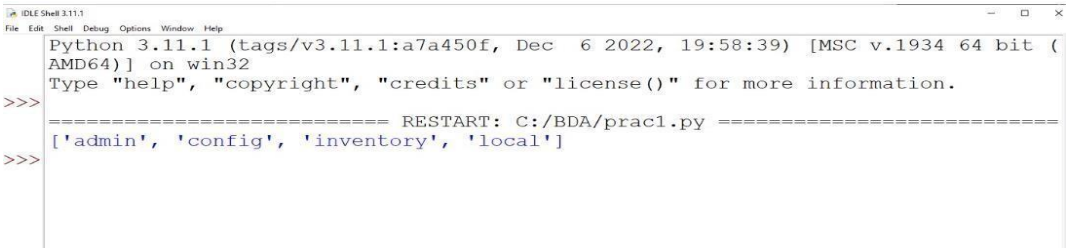


Now you have downloaded and installed a mongoDB driver.

Test PyMongo
demo_mongodb_test.py:
import pymongo
Program 1: **Creating a Database**

```
import pymongo
myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient["mybigdata"]
print(myclient.list_database_names())
```

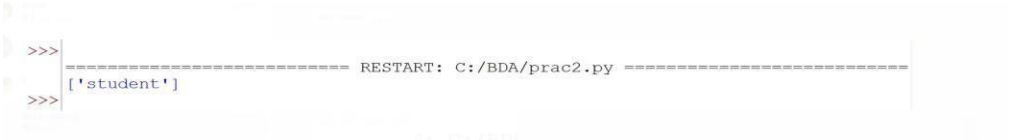
OUTPUT:



Progam 2: **Creating a Collection**

```
import pymongo
myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient["mybigdata"]
mycol=mydb[“student’]
print(mydb.list_collection_names())
```

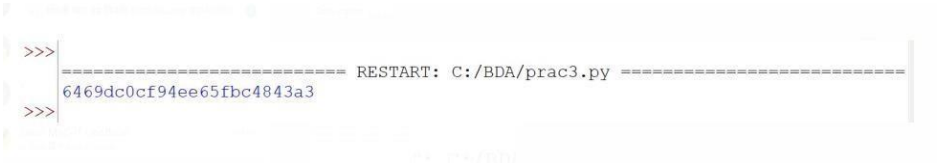
OUTPUT:



Progam 3: Insert into Collection

```
import pymongo
myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient["mybigdata"]
mycol=mydb[“student’]
mydict={“name”:”Kaushal”, “address”:”Mumbai”}
x=mycol.insert_one(mydict) # insert_one(containing the name(s) and value(s) of each
field
```

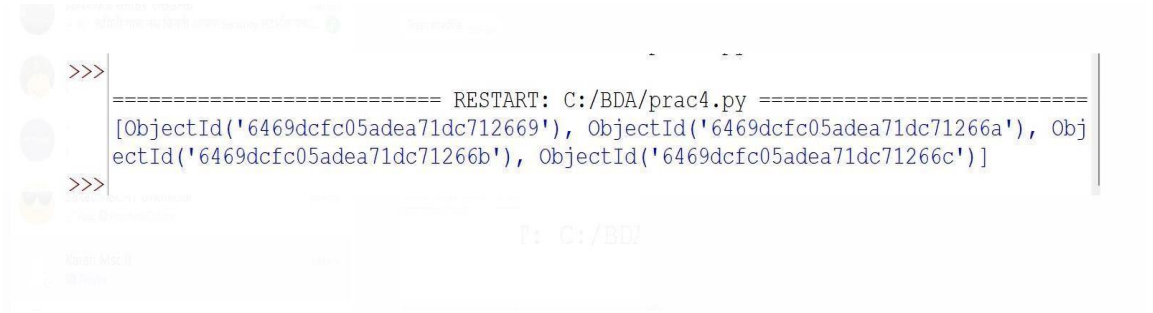
OUTPUT:



Program 4: Insert Multiple data into Collection

```
import pymongo
myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient["mybigdata"]
mycol=mydb[“student’]
mylist=[{“name”:”Kaushal”, “address”:”Mumbai”}, {“name”:”A”, “address”:”Mumbai”},
{“name”:”B”, “address”:”Pune”}, {“name”:”C”, “address”:”Pune”},]
```

OUTPUT:



PRACTICAL 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

Step 1: 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 1 1 1 1 1 1 ...
```

Step 2: Installing Packages:

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

Step 3: Loading Packages:

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

Step 4: Splitting data into train and test data:

```
> split<-sample.split(iris,SplitRatio=0.7)
> train_cl<-subset(iris,split=="Trus")
> train_cl<-subset(iris,split=="True")
> train_cl<-subset(iris,split=="False")

> train_scale<-scale(train_cl[,1:4])
> train_scale<-scale(test_cl[,1:4])

> set.seed(120)
> classifier_cl <-naiveBayes(Species ~ ., data=train_cl)
> classifier_cl
```

```
Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
     setosa versicolor  virginica

Conditional probabilities:
            Sepal.Length
Y            [,1] [,2]
  setosa       NA   NA
  versicolor   NA   NA
  virginica    NA   NA

            Sepal.Width
Y            [,1] [,2]
  setosa       NA   NA
  versicolor   NA   NA
  virginica    NA   NA

            Petal.Length
Y            [,1] [,2]
  setosa       NA   NA
  versicolor   NA   NA
  virginica    NA   NA

            Petal.Width
Y            [,1] [,2]
  setosa       NA   NA
  versicolor   NA   NA
  virginica    NA   NA
> 
```

Step 4: 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           0          0         0
  versicolor       0          0         0
  virginica        0          0         0
> 
```

Step 5: Model Evauation

> confusionMatrix(cm)

```
Confusion Matrix and Statistics

            y_pred
             setosa versicolor virginica
  setosa          0          0         0
  versicolor      0          0         0
  virginica       0          0         0

Overall Statistics

               Accuracy : NaN
                 95% CI : (NA, NA)
    No Information Rate : NA
    P-Value [Acc > NIR] : NA

                  Kappa : NaN

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: setosa Class: versicolor Class: virginica
Sensitivity                     NA                NA                NA
Specificity                     NA                NA                NA
Pos Pred Value                  NA                NA                NA
Neg Pred Value                  NA                NA                NA
Prevalence                     NaN               NaN               NaN
Detection Rate                 NaN               NaN               NaN
Detection Prevalence           NaN               NaN               NaN
Balanced Accuracy               NA                NA                NA
> 
```

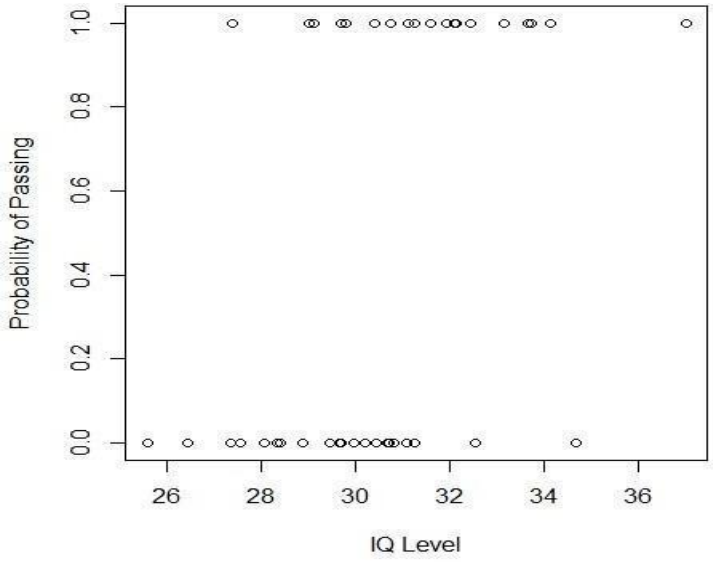
PRACTICAL 6

Aim: WAP showing implementation of Regression model.

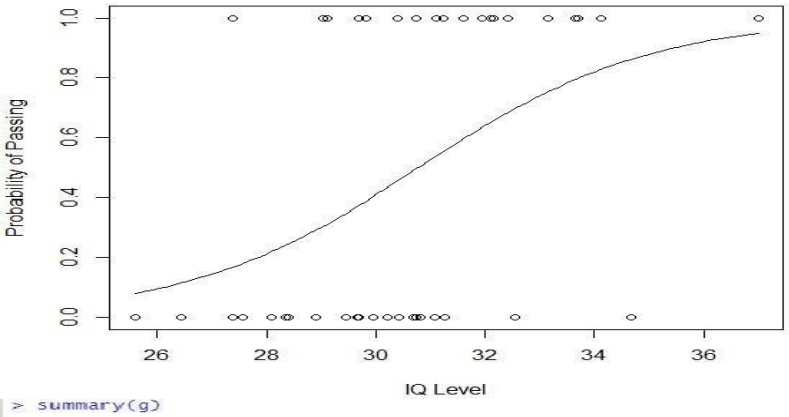
Description: Regression is a method to mathematically formulate relationship between variables that in due course can be used to estimate, interpolate and extrapolate. Suppose we want to estimate the weight of individuals, which is influenced by height, diet, workout, etc. Here, *Weight* is the **predicted** 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, 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 7

Aim: Write a Program showing clustering.

Description:

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.

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

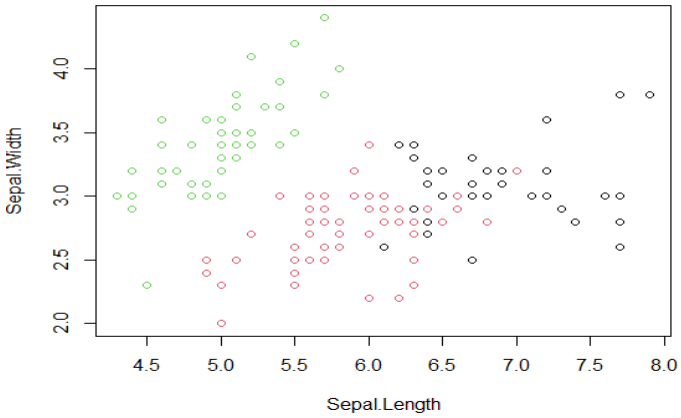
```
> newiris <- iris
> newiris$Species <- NULL
> (kc <- kmeans(newiris))
Error in kmeans(newiris) : 'centers' must be a number or a matrix
> (kc <- kmeans(newiris, 3))
K-means clustering with 3 clusters of sizes 38, 62, 50

Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1     6.850000    3.073684     5.742105    2.071053
2     5.901613    2.748387     4.393548    1.433871
3     5.006000    3.428000     1.462000    0.246000

Clustering vector:
  [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [27] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2
 [53] 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1
 [79] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 1 1
[105] 1 1 2 1 1 1 1 1 1 2 2 1 1 1 1 2 1 2 1 2 1 1 2 2 1 1
[131] 1 1 1 2 1 1 1 1 2 1 1 1 2 1 1 1 2 1 1 2

Within cluster sum of squares by cluster:
[1] 23.87947 39.82097 15.15100
 (between_SS / total_SS =  88.4 %)

Available components:

[1] "cluster"      "centers"      "totss"
[4] "withinss"     "tot.withinss" "betweenss"
[7] "size"         "iter"         "ifault"
>
```

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
>
```

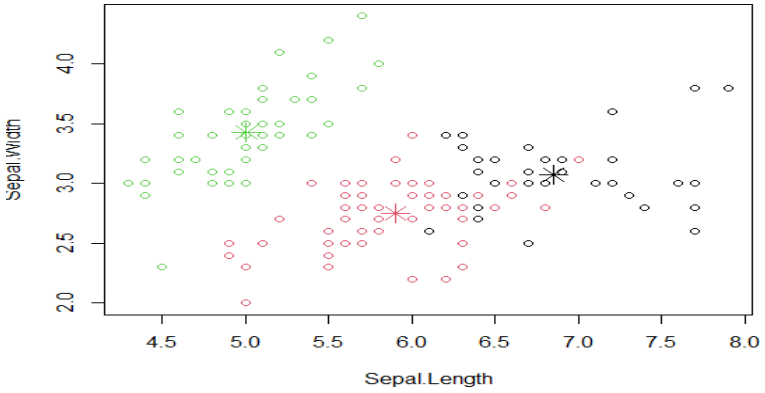
Step 3 : Plot the clusters and their centres. Note that there are four dimensions in the data and that only the first two dimensions are used to draw the plot below.

```
> 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)
```



MODERN NETWORKING

NURTURING POTENTIAL

SAKET GYANPEETH'S

SAKET COLLEGE OF ARTS, SCIENCE, AND COMMERCE

(Permanently Affiliated to University of Mumbai)

NAAC Accredited B Grade

Saket Vidyanagari Marg, Chinchpada Road, Katemanivali, Kalyan (East) – 421306, Dist. Thane (MAH)

Department of Information Technology

CERTIFICATE

This is to certify that
NA of **MSc Information Technology Part-I**
Class has satisfactorily carried out the required practical in the subject.

MODERN NETWORKING
For the Academic year 2022-2023

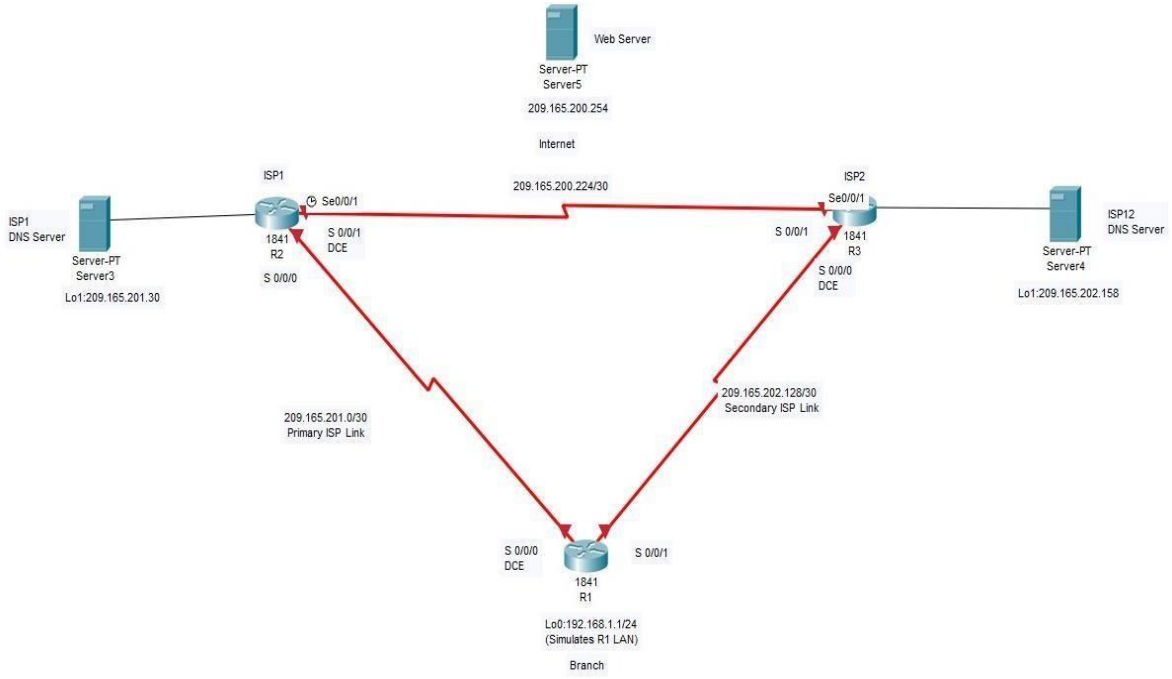
Practical in Charge **Head of Department** **External Examiner**

INDEX

PRACTICAL 1

Aim: Configure ip sla Tracking and Path Control

Topology



Objectives:

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

Background:

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

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

Required Resources:

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

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

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

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

Router R1

```
hostname R1
interface Loopback 0
description R1 LAN
 ip address 192.168.1.1 255.255.255.0
interface Serial0/0/0
 description R1 --> ISP1
 ip address 209.165.201.2 255.255.255.252
 clock rate 128000
 bandwidth 128
 no shutdown
interface Serial0/0/1
 description R1 --> ISP2
 ip address 209.165.202.130 255.255.255.252
 bandwidth 128
 no shutdown
```

Router ISP1 (R2)

```
hostname ISP1
interface Loopback0
 description Simulated Internet Web Server
 ip address 209.165.200.254 255.255.255.255
interface Loopback1
 description ISP1 DNS Server
 ip address 209.165.201.30 255.255.255.255
interface Serial0/0/0
 description ISP1 --> R1
 ip address 209.165.201.1 255.255.255.252
 bandwidth 128
 no shutdown
interface Serial0/0/1
```

```
description ISP1 --> ISP2

 ip address 209.165.200.225 255.255.255.252
 clock rate 128000
 bandwidth  128
no shutdown
```

Router ISP2 (R3)

```
hostname ISP2
interface Loopback0
 description Simulated Internet Web Server
 ip address 209.165.200.254 255.255.255.255
interface Loopback1
 description ISP2 DNS Server
 ip address 209.165.202.158 255.255.255.255
interface Serial0/0/0
 description ISP2 --> R1
 ip address 209.165.202.129 255.255.255.252
 clock rate 128000
 bandwidth 128
 no shutdown
interface Serial0/0/1
 description ISP2 --> ISP1
 ip address 209.165.200.226 255.255.255.252
 bandwidth 128
 no shutdown
```

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

R1# show interfaces description

Interface	Status	Protocol	Description
Fa0/0	admin down	down	
Fa0/1	admin down	down	
Se0/0/0	up	up	R1 --> ISP1
Se0/0/1	up	up	R1 --> ISP2
Lo0	up	up	R1 LAN

All three interfaces should be active. Troubleshoot if necessary.

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

Translation (NAT) would be configured for all traffic exiting the branch LAN. Therefore, the static routes on the ISP routers would be pointing to the provided public pool of the branch office. This is covered in Lab 7-1, "Configure Routing Facilities to the Branch Office."

configurations.

Router R1
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
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
EIGRP neighbor relationship messages on ISP1 and ISP2 should be generated. Troubleshoot if necessary.
%DUAL-5-NBRCHANGE: IP-EIGRP(0) 1: Neighbor 209.165.200.225 (Serial0/0/1) is up: new adjacency
Step 2: Verify server reachability.

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

a. 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..

foreach address {
209.165.200.254
209.165.201.30
209.165.202.158
} {
ping $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)# 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 = 12/15/16 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 = 12/14/16 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 = 20/21/24 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.

```
foreach address {
209.165.200.254
209.165.201.30
209.165.202.158
} {
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 20 msec 8 msec *
Type escape sequence to abort.
Tracing the route to 209.165.201.30
1 209.165.201.1 8 msec 8 msec *
Type escape sequence to abort.
Tracing the route to 209.165.202.158
```

1 209.165.201.1 8 msec 8 msec 4 msec
2 209.165.200.226 8 msec 8 msec *

Through which ISP is traffic flowing?

Step 3: Configure IP SLA probes.

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

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

Note: With Cisco IOS Release 12.4(4)T, 12.2(33)SB, and 12.2(33)SXI, the ip sla command has replaced 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 start-time now

The operation number of 11 is only locally significant to the router. The frequency 10 command schedules the connectivity test to repeat every 10 seconds.

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

Note: With Cisco IOS Release 12.4(4)T, 12.2(33)SB, and 12.2(33)SXI, the show ip sla configuration command has replaced the show ip sla monitor configuration command.

R1# 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 randomly scheduled)
Next Scheduled Start Time: Start Time already passed
Group Scheduled : FALSE
Randomly Scheduled : FALSE

Life (seconds): Forever
Entry Ageout (seconds): never
Recurring (Starting Everyday): FALSE
Status of entry (SNMP RowStatus): Active
Threshold (milliseconds): 5000 (not considered if react RTT is configured)
Distribution Statistics:
Number of statistic hours kept: 2
Number of statistic distribution buckets kept: 1
Statistic distribution interval (milliseconds): 20
History Statistics:
Number of history Lives kept: 0
Number of history Buckets kept: 15
History Filter Type: None
Enhanced History:
The output lists the details of the configuration of operation 11. The operation is an ICMP echo to 209.165.201.30, with a frequency of 10 seconds, and it has already started (the start time has already passed).

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

Note: With Cisco IOS Release 12.4(4)T, 12.2(33)SB, and 12.2(33)SXI, the show ip sla statistics command has replaced the show ip sla monitor statistics command.

R1# show ip sla statistics
IPSLAs Latest Operation Statistics
IPSLA operation id: 11
Latest operation start time: *21:22:29.707 UTC Fri Apr 2 2010
Latest operation return code: OK
Number of successes: 5
Number of failures: 0
Operation time to live: Forever You can see that operation 11 has already succeeded five times, has had no failures, and the last operation returned an OK result.

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

ip sla 22
icmp-echo 209.165.202.158
frequency 10
exit
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
IP SLAs, Infrastructure Engine-II.
Entry number: 22
Owner:

Tag:
Type of operation to perform: icmp-echo
Target address/Source address: 209.165.201.158/0.0.0.0
Type Of Service parameter: 0x0
Request size (ARR data portion): 28
Operation timeout (milliseconds): 5000
Verify data: No
Vrf Name:
Schedule:

Operation frequency (seconds): 10 (not considered if randomly scheduled)
Next Scheduled Start Time: Start Time already passed
Group Scheduled : FALSE
Randomly Scheduled : FALSE
Life (seconds): Forever
Entry Ageout (seconds): never
Recurring (Starting Everyday): FALSE
Status of entry (SNMP RowStatus): Active
Threshold (milliseconds): 5000 (not considered if react RTT is configured)
Distribution Statistics:
Number of statistic hours kept: 2
Number of statistic distribution buckets kept: 1
Statistic distribution interval (milliseconds): 20
History Statistics:
Number of history Lives kept: 0
Number of history Buckets kept: 15
History Filter Type: None
Enhanced History:
R1# show ip sla statistics 22
IPSLAs Latest Operation Statistics
IPSLA operation id: 22
Latest operation start time: *21:24:14.215 UTC Fri Apr 2 2010
Latest operation return code: OK
Number of successes: 4
Number of failures: 0

Operation time to live: Forever

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

Step 4: Configure tracking options.

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

a. 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
```

b. Verify the routing table.

```
R1# show ip route
*Apr 2 20:00:37.367: %SYS-5-CONFIG_I: Configured from console by console
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
     209.165.201.0/30 is subnetted, 1 subnets
C       209.165.201.0 is directly connected, Serial0/0/0
     209.165.202.0/30 is subnetted, 1 subnets
C       209.165.202.128 is directly connected, Serial0/0/1
C    192.168.1.0/24 is directly connected, FastEthernet0/0
S*  0.0.0.0/0 [5/0] via 209.165.201.1
```

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

c. Use the track 1 ip sla 11 reachability command to enter the config-track sub configuration mode.

Note: With Cisco IOS Release 12.4(20)T, 12.2(33)SXI1, and 12.2(33)SRE and Cisco IOS XE Release 2.4, the track ip sla command has replaced the track rtr command.

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

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

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

e. 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

IP routing debugging is on
R1#
*Apr 2 21:26:46.171: RT: NET-RED 0.0.0.0/0
R1# conf t
Enter configuration commands, one per line. End with CNTL/Z.
R1(config)# ip route 0.0.0.0 0.0.0.0 209.165.201.1 2 track 1
R1(config)#
*Apr 2 21:27:02.851: RT: closer admin distance for 0.0.0.0, flushing 1 routes
*Apr 2 21:27:02.851: RT: NET-RED 0.0.0.0/0
*Apr 2 21:27:02.851: RT: add 0.0.0.0/0 via 209.165.201.1, static metric [2/0]
*Apr 2 21:27:02.851: RT: NET-RED 0.0.0.0/0
*Apr 2 21:27:02.851: RT: default path is now 0.0.0.0 via 209.165.201.1
*Apr 2 21:27:02.855: RT: new default network 0.0.0.0
*Apr 2 21:27:02.855: RT: NET-RED 0.0.0.0/0
*Apr 2 21:27:07.851: RT: NET-RED 0.0.0.0/0
Notice that the default route with an administrative distance of 5 has been immediately flushed because of a route with a better admin distance. It then adds the new default route with the admin distance of 2.

f. 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

g. Verify the routing table again.

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
209.165.201.0/30 is subnetted, 1 subnets
C 209.165.201.0 is directly connected, Serial0/0/0
209.165.202.0/30 is subnetted, 1 subnets
C 209.165.202.128 is directly connected, Serial0/0/1
C 192.168.1.0/24 is directly connected, FastEthernet0/0
S* 0.0.0.0/0 [2/0] via 209.165.201.1
Although a new default route was entered, its administrative distance is not better than 2. Therefore, it does not replace the previously entered default route.

Step 5: Verify IP SLA operation.

In this step you observe and verify the dynamic operations and routing changes when tracked objects fail.
The following summarizes the process:
• Disable the DNS loopback interface on ISP1 (R2).
• Observe the output of the debug command on R1.
• Verify the static route entries in the routing table and the IP SLA statistics of R1.
• Re-enable the loopback interface on ISP1 (R2) and again observe the operation of the IP SLA tracking feature.

```
ISP1(config)# interface loopback 1
ISP1(config-if)# shutdown
ISP1(config-if)#
*Apr 2 15:53:14.307: %LINK-5-CHANGED: Interface Loopback1, changed state to administratively down
*Apr 2 15:53:15.307: %LINEPROTO-5-UPDOWN: Line protocol on Interface Loopback1, changed state to down
```

a. Shortly after the loopback interface is administratively down, observe the debug output being generated on R1.

```
R1#
*Apr  2 21:32:33.323: %TRACKING-5-STATE: 1 ip sla 11 reachability Up->Down
*Apr  2 21:32:33.323: RT: del 0.0.0.0 via 209.165.201.1, static metric [2/0]
*Apr  2 21:32:33.323: RT: delete network route to 0.0.0.0
*Apr  2 21:32:33.323: RT: NET-RED 0.0.0.0/0
*Apr  2 21:32:33.323: RT: NET-RED 0.0.0.0/0
*Apr  2 21:32:33.323: RT: add 0.0.0.0/0 via 209.165.202.129, static metric [3/0]
*Apr  2 21:32:33.323: RT: NET-RED 0.0.0.0/0
*Apr  2 21:32:33.323: RT: default path is now 0.0.0.0 via 209.165.202.129
*Apr  2 21:32:33.323: RT: new default network 0.0.0.0
*Apr  2 21:32:33.327: RT: NET-RED 0.0.0.0/0
*Apr  2 21:32:46.171: RT: NET-RED 0.0.0.0/0
```

The tracking state of track 1 changes from up to down. This is the object that tracked reachability for IP SLA object 11, with an ICMP echo to the ISP1 DNS server at 209.165.201.30. R1 then proceeds to delete the default route with the administrative distance of 2 and installs the next highest default route to ISP2 with the administrative distance of 3.

b. Verify the routing table.

```
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 209.165.202.129 to network 0.0.0.0

    209.165.201.0/30 is subnetted, 1 subnets
```

C 209.165.201.0 is directly connected, Serial0/0/0
209.165.202.0/30 is subnetted, 1 subnets
C 209.165.202.128 is directly connected, Serial0/0/1
C 192.168.1.0/24 is directly connected, FastEthernet0/0
S* 0.0.0.0/0 [3/0] via 209.165.202.129
The new static route has an administrative distance of 3 and is being forwarded to ISP2 as it should.

c. Verify the IP SLA statistics.

R1# show ip sla statistics

IPSLAs Latest Operation Statistics

PSLA operation id: 11
Type of operation: icmp-echo
Latest RTT: NoConnection/Busy/Timeout
Latest operation start time: *15:36:42.871 UTC Fri Apr 2 2010
Latest operation return code: No connection
Number of successes: 84
Number of failures: 13
Operation time to live: Forever
IPSLA operation id: 22
Type of operation: icmp-echo
Latest RTT: 8 milliseconds
Latest operation start time: *15:36:46.335 UTC Fri Apr 2 2010
Latest operation return code: OK
Number of successes: 81
Number of failures: 1
Operation time to live: Forever
Notice that the latest return code is No connection and there have been 12 failures on IP SLA object 11.

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

Type escape sequence to abort.
Tracing the route to 209.165.200.254
1 209.165.202.129 8 msec 8 msec *
This confirms that traffic is leaving router R1 and being forwarded to the ISP2 router.

e. To examine the routing behavior when connectivity to the ISP1 DNS is restored, re-enable the DNS

address on ISP1 (R2) by issuing the no shutdown command on the loopback 1 interface on ISP2.

ISP1(config-if)# no shutdown

*Apr 2 15:56:24.655: %LINK-3-UPDOWN: Interface Loopback1, changed state to up
*Apr 2 15:56:25.655: %LINEPROTO-5-UPDOWN: Line protocol on Interface

Loopback1, changed state to up
Notice the output of the debug ip routing command on R1.

R1#

*Apr 2 21:35:34.327: %TRACKING-5-STATE: 1 ip sla 11 reachability Down->Up
*Apr 2 21:35:34.327: RT: closer admin distance for 0.0.0.0, flushing 1,routes
*Apr 2 21:35:34.327: RT: NET-RED 0.0.0.0/0
*Apr 2 21:35:34.327: RT: add 0.0.0.0/0 via 209.165.201.1, static metric [2/0]
*Apr 2 21:35:34.327: RT: NET-RED 0.0.0.0/0
*Apr 2 21:35:34.327: RT: default path is now 0.0.0.0 via 209.165.201.1
*Apr 2 21:35:34.327: RT: new default network 0.0.0.0
*Apr 2 21:35:34.327: RT: NET-RED 0.0.0.0/0
*Apr 2 21:35:39.327: RT: NET-RED 0.0.0.0/0
*Apr 2 21:35:46.171: RT: NET-RED 0.0.0.0/0
Now the IP SLA 11 operation transitions back to an up state and reestablishes the default static route to ISP1 with an administrative distance of 2.

f. Again examine the IP SLA statistics.

R1# show ip sla statistics
IPSLAs Latest Operation Statistics
Type of operation: icmp-echo
Latest RTT: 8 milliseconds
Latest operation start time: *15:40:42.871 UTC Fri Apr 2 2010
Latest operation return code: OK
Number of successes: 88
Number of failures: 35
Operation time to live: Forever
IPSLA operation id: 22
Type of operation: icmp-echo
Latest RTT: 16 milliseconds
Latest operation start time: *15:40:46.335 UTC Fri Apr 2 2010
Latest operation return code: OK
Number of successes: 105
Number of failures: 1
Operation time to live: Forever

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

g. Verify the routing table.

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

    209.165.201.0/30 is subnetted, 1 subnets
C       209.165.201.0 is directly connected, Serial0/0/0
    209.165.202.0/30 is subnetted, 1 subnets
C       209.165.202.128 is directly connected, Serial0/0/1
C    192.168.1.0/24 is directly connected, FastEthernet0/0
S*  0.0.0.0/0 [2/0] via 209.165.201.1
```

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

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

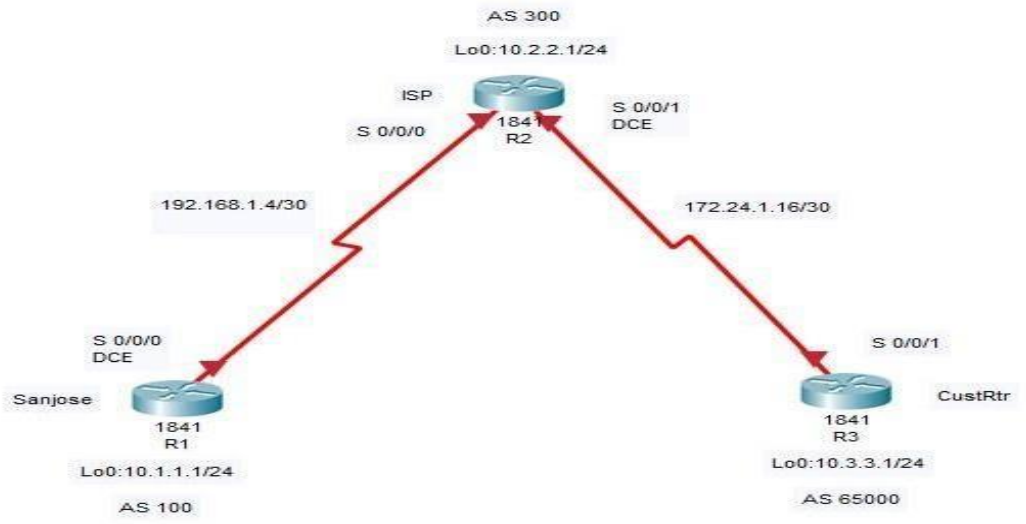
Before deploying a Cisco IOS IP SLA solution, the impact of the additional probe traffic being generated should be considered, including how that traffic affects bandwidth utilization, and congestion levels. Tuning the configuration (for example, with the delay and frequency commands) is critical to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

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

PRACTICAL 2

Aim: Using the AS_PATH Attribute

Topology



Objectives

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

Background: The International Travel Agency's ISP has been assigned an AS number of 300. This provider uses BGP to exchange routing information with several customer networks. Each customer network is assigned an AS number from the private range, such as AS 65000. Configure the ISP router to remove the private AS numbers from the AS Path information of CustRtr. route information from International Travel Agency's AS 100. Use the AS_PATH attribute to implement this policy.

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

Required Resources

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

Step 1: Prepare the routers for the lab.

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

Step 2: Configure the hostname and interface addresses.

a.. Router R1 (hostname SanJose)

```
hostname SanJose
!
interface Loopback0
ip address 10.1.1.1 255.255.255.0
!
interface Serial0/0/0
ip address 192.168.1.5 255.255.255.252
clock rate 128000
no shutdown
```

Router R2 (hostname ISP)

```
hostname ISP
!
interface Loopback0
ip address 10.2.2.1 255.255.255.0
!
interface Serial0/0/0
ip address 192.168.1.6 255.255.255.252
no shutdown
!
interface Serial0/0/1
ip address 172.24.1.17 255.255.255.252
clock rate 128000
no shutdown
```

Router R3 (hostname CustRtr)

```
hostname CustRtr
!
interface Loopback0
ip address 10.3.3.1 255.255.255.0
!
interface Serial0/0/1
ip address 172.24.1.18 255.255.255.252
no shutdown
```

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

Note: SanJose will not be able to reach either ISP's loopback (10.2.2.1) or CustRtr's loopback (10.3.3.1), nor will it be able to reach either end of the link joining ISP to CustRtr (172.24.1.17 and 172.24.1.18).

Step 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.

```
SanJose(config)# router bgp 100

SanJose(config-router)# neighbor 192.168.1.6 remote-as 300
SanJose(config-router)# network 10.1.1.0 mask 255.255.255.0
ISP(config)# router bgp 300
ISP(config-router)# neighbor 192.168.1.5 remote-as 100
ISP(config-router)# neighbor 172.24.1.18 remote-as 65000
ISP(config-router)# network 10.2.2.0 mask 255.255.255.0
CustRtr(config)# router bgp 65000
CustRtr(config-router)# neighbor 172.24.1.17 remote-as 300
CustRtr(config-router)# network 10.3.3.0 mask 255.255.255.0
```

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

```
ISP# show ip bgp neighbors
BGP neighbor is 172.24.1.18, remote AS 65000, external link
BGP version 4, remote router ID 10.3.3.1
BGP state = Established, up for 00:02:05
<output omitted>
BGP neighbor is 192.168.1.5, remote AS 100, external link
BGP version 4, remote router ID 10.1.1.1
BGP state = Established, up for 00:04:19
<output omitted>
```

Step 4: Remove the private AS.

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

```
SanJose# show ip route
Codes: 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, 00:01:11
B 10.2.2.0 [20/0] via 192.168.1.6, 00:02:16
C 10.1.1.0 is directly connected, Loopback0
192.168.1.0/30 is subnetted, 1 subnet
C 192.168.1.4 is directly connected, Serial0/0/0
```

b. **Ping the 10.3.3.1 address from SanJose.** Why does this fail?

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

SanJose# ping

Protocol [ip]:
Target IP address: 10.3.3.1
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface: 10.1.1.1
Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.3.3.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 64/64/68 ms

Note: You can bypass extended ping mode and specify a source address using one of these **commands:** SanJose# ping 10.3.3.1 source 10.1.1.1 or SanJose# ping 10.3.3.1 source Lo0

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

SanJose# show ip bgp

BGP table version is 5, local router ID is 10.1.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i – internal
Origin codes: i - IGP, e - EGP, ? - incomplete

Network	Next Hop	Metric LocPrf	Weight Path
*> 10.1.1.0	0.0.0.0	0	32768 i
*> 10.2.2.0	192.168.1.6	0	300 i
*> 10.3.3.0	192.168.1.6	0	300 65000 i

Why is this a problem?

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

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

f. After issuing these commands, use the clear ip bgp * command on ISP to reestablish the BGP relationship between the three routers. Wait several seconds and then return to SanJose to check its routing table.

Note: The clear ip bgp * soft command can also be used to force each router to resend its BGP table.Does SanJose still have a route to 10.3.3.0? SanJose should be able to ping 10.3.3.1 using its loopback 0 interface as the source of the ping.

SanJose# ping 10.3.3.1 source lo0

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

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

longer has the private AS in the path.
SanJose# show ip bgp
BGP table version is 8, local router ID is 10.1.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i – internal
Origin codes: i - IGP, e - EGP, ? - incomplete

Network	Next Hop	Metric LocPrf	Weight Path
*> 10.1.1.0	0.0.0.0	0	32768 i
*> 10.2.2.0	192.168.1.6	0	0 300 i
*> 10.3.3.0	192.168.1.6	0	0 300 i

Step 5: Use the AS_PATH attribute to filter routes.

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

a. 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 ISP.

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

The first command uses the ^ character to indicate that the AS path must begin with the given number 100. The $ character indicates that the AS_PATH attribute must also end with 100. Essentially, this statement matches only paths that are sourced from AS 100 .Together, .* matches any value of the AS_PATH attribute, which in effect permits any update that has not been denied by the previous access-list statement. For more details on configuring regular expressions on Cisco routers, see:
http://www.cisco.com/en/US/docs/ios/12_2/termserv/configuration/guide/tcfaapre_ps1835_TSD_Products _Configuration_Guide_Chapter.html

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

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

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

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

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

ISP# show ip route

<output omitted>
C 172.24.0.0/30 is subnetted, 1 subnets
B 172.24.1.16 is directly connected, Serial0/0/1
C 10.0.0.0/24 is subnetted, 3 subnets
B 10.3.3.0 [20/0] via 172.24.1.18, 00:07:34
C 10.2.2.0 is directly connected, Loopback0
10.1.1.0 [20/0] via 192.168.1.5, 00:10:53
192.168.1.0/30 is subnetted, 1 subnets
192.168.1.4 is directly connected, Serial0/0/0

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

CustRtr# show ip route

<output omitted>
C 172.24.0.0/30 is subnetted, 1 subnets
C 172.24.1.16 is directly connected, Serial0/0/1
B10.0.0.0/24 is subnetted, 2 subnets
10.3.3.0 is directly connected, Loopback0
10.2.2.0 [20/0] via 172.24.1.17, 00:11:57

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

ISP# show ip bgp regexp ^100$

BGP table version is 4, local router ID is 10.2.2.1
Status codes: s suppressed, d damped, h history, * valid, > best, i – internal
Origin codes: i - IGP, e - EGP, ? - incomplete

Network	Next Hop	Metric	Loc	Prf	Weight	Path
*> 10.1.1.0	192.168.1.5	0			0	100 i

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

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

192.168.1.4/30.

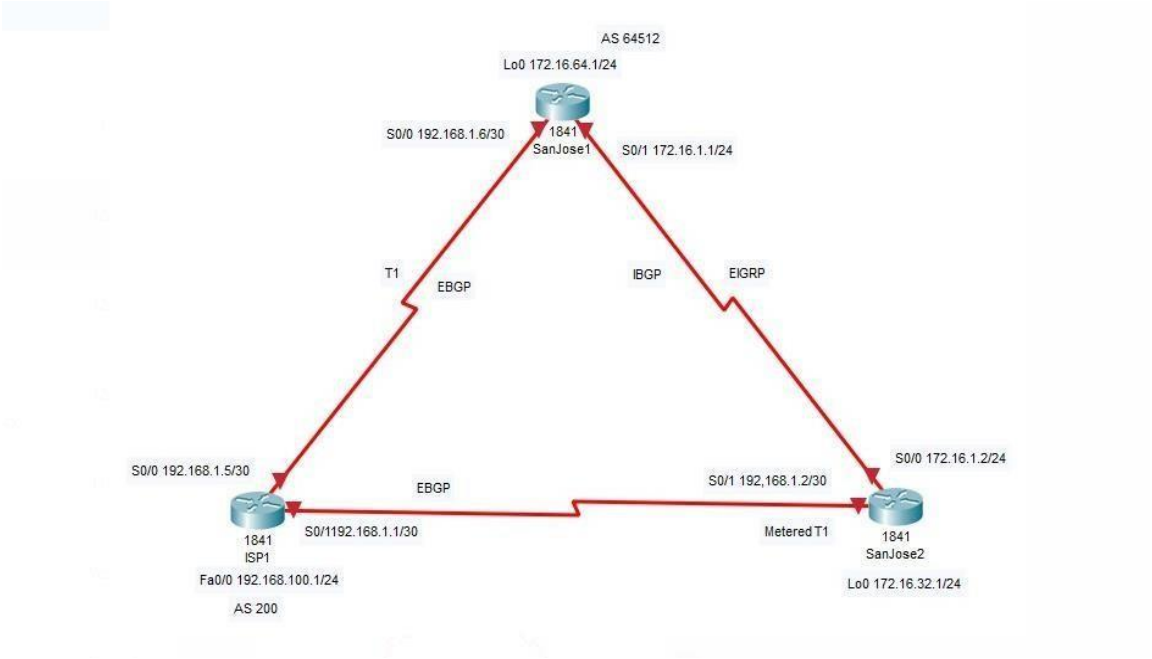
```
ISP# tclsh
foreach address {
10.1.1.1
10.2.2.1
10.3.3.1
192.168.1.5
192.168.1.6
172.24.1.17
172.24.1.18
} {
ping $address }
```

\

PRACTICAL 3

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

Topology



Objective: In this lab, the student will configure both IBGP and EBGP. In order for IBGP peers to correctly exchange routing information, the next-hop-self-command must be used. Use of Local Preference and MED attributes must also be used. This is to insure that the flat rate unlimited use. T1 link is used for sending and receiving data to and from the AS 200 on ISP. The metered T1 should only be used in the event that the primary T1 link has failed.

Scenario: The International Travel Agency runs BGP on its SanJose1 and SanJose2 routers externally with ISP1, AS 200. IBGP is run internally between SanJose1 and SanJose2. The job is to configure both EBGP and IBGP for this internetwork to allow for redundancy.

Step 1: Build and configure the network according to the diagram, but do not configure a routing protocol. Configure a loopback interface on the SanJose1 and SanJose2 routers as shown. These loopbacks will be used with BGP neighbor statements for increased stability.Use ping to test connectivity between the directly connected routers.

Note: The ISP1 router will not be able to reach the segment between SanJose1 and SanJose2. Both SanJose routers should be able to ping each other as well as their local ISP serial link IP address.

Step 2: Configure EIGRP between the SanJose1 and SanJose2 routers with the same commands as follows:

(config)#router eigrp 64512
(config-router)#network 172.16.0.0

Step 3: Configure IBGP between the SanJose1 and SanJose2 routers. On the SanJose1 router, enter the following configuration:

```
SanJose1(config)#router bgp 64512
SanJose1(config-router)#no auto-summary
SanJose1(config-router)#neighbor 172.16.32.1 remote-as 64512
SanJose1(config-router)#neighbor 172.16.32.1 update-source lo0
```

If multiple pathways to the neighbor exist, then the router can use any IP interface to communicate by way of BGP.. Because BGP will eventually advertise outside networks that are not part of the EIGRP cloud, the following command must be entered on SanJose1 and SanJose2:

```
SanJose1(config)#router bgp 64512
SanJose1(config-router)#no synchronization
SanJose2(config)#router bgp 64512
SanJose2(config-router)#no synchronization
```

The no synchronization command permits BGP to advertise networks regardless of whether EIGRP knows of the network.

Step 4: Complete the IBGP configuration on SanJose2 by entering the following commands:

```
SanJose2(config)#router bgp 64512
SanJose2(config-router)#no auto-summary
SanJose2(config-router)#neighbor 172.16.64.1 remote-as 64512
SanJose2(config-router)#neighbor 172.16.64.1 update-source lo0
```

View the following partial output. If the BGP state is not established, troubleshoot the connection. The link between SanJose1 and SanJose2 should indicate an internal link as shown in the following: SanJose

```
show ip bgp neighbors
BGP neighbor is 172.16.64.1, remote AS 64512, internal link
BGP version 4, remote router ID 172.16.64.1
BGP state = Established, up for 00:00:01
```

Step 5: Configure ISP1 to run EBGP with SanJose1 and SanJose2. Enter the following commands on ISP1 as shown in the following:

```
ISP1(config)#router bgp 200
ISP1(config-router)#no auto-summary
ISP1(config-router)#neighbor 192.168.1.6 remote-as 64512
ISP1(config-router)#neighbor 192.168.1.2 remote-as 64512
ISP1(config-router)#network 192.168.100.0
```

Step 6: Configure SanJose1 as an EBGP peer to ISP1 as shown in the following:

```
SanJose1(config)#ip route 172.16.0.0 255.255.0.0 null0
SanJose1(config)#router bgp 64512
SanJose1(config-router)#neighbor 192.168.1.5 remote-as 200
SanJose1(config-router)#network 172.16.0.0
```

Step 7 : Configure SanJose2 as an EBGP peer to ISP1:

```
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
```

In Step 6 the show ip bgp neighbors command was used to verify that SanJose1 and ISP1 had reached the Established state. A useful alternative command is the show ip bgp summary command. Output should be similar to the sample output displayed below:

```
SanJose2#

show ip bgp summary
BGP router identifier 172.16.32.1, local AS number 64512
BGP table version is 2, main routing table version 2
1 network entries and 1 paths using 137 bytes of memory
1 BGP path attribute entries using 60 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP activity 2/1 prefixes, 2/1 paths, scan interval 15 secs Neighbor
```

Step 8 : Test whether ISP1 can ping the Loopback 0 address of 172.16.64.1 from SanJose1, as well as the serial link between SanJose1 and SanJose2, 172.16.1.1. Now ping from ISP1 to the Loopback 0 address of 172.16.32.1 from SanJose2, as well as the serial link between SanJose1 and SanJose2. Ping attempts to the 172.16.64.1 and 172.16.1.1 should fail. Issue the show ip bgp command on ISP1 as follows to verify BGP routes and metrics:

```
ISP1#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
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop          Metric LocPrf Weight Path
*  172.16.0.0       192.168.1.6                    0 64512 i
*>                  192.168.1.2            0       0 64512 i
*> 192.168.100.0    0.0.0.0                0           32768 i
```

Notice that ISP1 has two valid routes to the 172.16.0.0 network, indicated by the *. However, the link to SanJose2, the metered T1, has been selected as the best path.

BGP operates differently than all other protocols. Unlike other routing protocols which may use complex algorithms involving factors such as bandwidth, delay, reliability, and load to formulate a metric, BGP is policy-based. BGP will determine the best path based upon variables such as, AS Path, Weight, Local Preference, MED, and so on. The SanJose2 router with address 192.168.1.2 was preferred to the higher IP address of the SanJose1 router, 192.168.1.6. At this point, the ISP1 router should be able to get to each network connected to SanJose1 and SanJose2 from the Fast Ethernet address 192.168.100.1.

```
ISP1#ping
Protocol [ip]:
Target IP address: 172.16.64.1
```

Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface: 192.168.100.1
Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.64.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 48/48/52 ms

ISP1#ping

Protocol [ip]:
Target IP address: 172.16.1.1
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface: 192.168.100.1
Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 48/48/48 ms

ISP1#ping

Protocol [ip]:
Target IP address: 172.16.32.1
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface: 192.168.100.1
Type of service [0]:
Set DF bit in IP header? [no]:

Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.32.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 32/33/36 ms

ISP1#ping
Protocol [ip]:
Target IP address: 172.16.1.2
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface: 192.168.100.1
Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.1.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 32/36/56 ms
Complete reachability was proven between the ISP1 router and both SanJose1 and SanJose2.

4. Why do the following ping requests fail?

ISP1#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)

ISP1#ping 172.16.64.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.64.1, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)

Step 9: Before the ISP can successfully ping the internal serial interfaces of AS 64512, two issues need to be resolved. First, SanJose1 does not know about the link between the ISP and SanJose2. This can be resolved by an advertisement of these serial links by way of BGP on the ISP router. If they are advertised and then, a BGP link is activated to another ISP in

addition to ISP1 at AS 200, then there is a risk of becoming a Transit AS. Issue the following commands on the ISP1 router:

```
ISP1(config)#router bgp 200
ISP1(config-router)#network 192.168.1.0 mask 255.255.255.252
ISP1(config-router)#network 192.168.1.4 mask 255.255.255.252
```

Clear the IP BGP conversation with the clear ip bgp*command on ISP1. Issue the show ip bgp command as follows to verify that the ISP1 router can its own WAN links through BGP:

```
ISP1#show ip bgp

BGP table version is 5, local router ID is 192.168.100.1
Status codes: s suppressed, d damped, h history, * valid, > best, i – internal Origin codes: i - IGP, e - EGP, ? - incomplete
 Network          Next Hop          Metric LocPrf Weight Path
* 172.16.0.0      192.168.1.6                    0 64512 i
*>                192.168.1.2            0       0 64512 i
*> 192.168.1.0/30  0.0.0.0               0           32768 i
*> 192.168.1.4/30  0.0.0.0               0           32768 i
*> 192.168.100.0   0.0.0.0               0           32768 i
```

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

```
Gateway of last resort is not set
     172.16.0.0/24 is subnetted, 3 subnets
C       172.16.32.0 is directly connected, Loopback0
172.16.1.0 is directly connected, Serial0/0
172.16.64.0 [90/20640000] via 172.16.1.1, 00:57:10, Serial0/0
     192.168.1.0/30 is subnetted, 2 subnets
C       192.168.1.0 is directly connected, Serial0/1
B       192.168.1.4 [20/0] via 192.168.1.1, 00:04:23
B    192.168.100.0/24 [20/0] via 192.168.1.1, 00:04:23
```

The next issue to consider is BGP policy routing between AS systems. The policy for routing from AS 64512 to AS 200 is to forward packets to the 192.168.1.1 interface. In the event that either WAN link fails, it is critical that the opposite router become a valid gateway Before the next-hop-self-command was issued:

```
SanJose2#show ip bgp

BGP table version is 11, local router ID is 172.16.32.1
Status codes: s suppressed, d damped, h history, * valid, > best, i – internal
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
i192.168.1.0/30  192.168.1.5          0      100      0 200 i
*>               192.168.1.1          0               0 200 i
i192.168.1.4/30  192.168.1.5          0      100      0 200 i
*>               192.168.1.1          0               0 200 i
i192.168.100.0    192.168.1.5         0      100      0 200 i
```

```
*>              192.168.1.1              0              0 200 i
```

```
SanJose1(config)#router bgp 64512
SanJose1(config-router)#neighbor 172.16.32.1 next-hop-self
SanJose2(config)#router bgp 64512
SanJose2(config-router)#neighbor 172.16.64.1 next-hop-self
```

After issuing these commands, reset BGP operation on either router by entering the command clear ip bgp *.

```
SanJose2#show ip bgp

BGP table version is 11, local router ID is 172.16.32.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
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
i192.168.1.0/30  172.16.64.1              0    100      0 200 i
*>               192.168.1.1             0                0 200 i
i192.168.1.4/30  172.16.64.1              0    100      0 200 i
*>               192.168.1.1             0                0 200 i
i192.168.100.0   172.16.64.1              0    100      0 200 i
*>               192.168.1.1             0                0 200 i
```

Step 10: At this point, everything looks good with the exception of default routes, the outbound flow of data, and inbound packet flow. Since the local preference value is shared between IBGP neighbors, configure a simple route-map that references local preference value on SanJose1 and SanJose2.

```
SanJose1(config)#route-map PRIMARY_T1_IN permit 10
SanJose1(config-route-map)#set local-preference 150
SanJose1(config-route-map)#exit
SanJose1(config)#router bgp 64512
SanJose1(config-router)#neighbor 192.168.1.5 route-map PRIMARY_T1_IN in
SanJose2(config)#route-map SECONDARY_T1_IN permit 10
SanJose2(config-route-map)#set local-preference 125
SanJose2(config-route-map)#router bgp 64512
SanJose2(config-router)# neighbor 192.168.1.1 route-map SECONDARY_T1_IN in
```

Do not forget to use the command clear ip bgp * after configuring this new policy. Once the conversations have been re-established, issue the show ip bgp command on SanJose1 and SanJose2 as follows:

```
SanJose1#show ip bgp

BGP table version is 8, local router ID is 172.16.64.1 Status codes: s suppressed, d damped, h history, * valid, > best, i – internal  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
*> 192.168.1.0/30   192.168.1.5              0    150      0 200 i
*> 192.168.1.4/30   192.168.1.5              0    150      0 200 i
```

*> 192.168.100.0 192.168.1.5 0 150 0 200 i

SanJose2#show ip bgp

BGP table version is 11, local router ID is 172.16.32.1 Status codes: s suppressed, d damped, h history, * valid, > best, i – internal 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
*>i192.168.1.0/30	172.16.64.1	0	150	0	200 i
	192.168.1.1	0	125	0	200 i
*>i192.168.1.4/30	172.16.64.1	0	150	0	200 i
	192.168.1.1	0	125	0	200 i
*>i192.168.100.0	172.16.64.1	0	150	0	200 i
	192.168.1.1	0	125	0	200 i

Step 11: How will traffic return from network 192.168.100.0 /24? Will it be routed through SanJose1 or SanJose2?Use an extended ping in this situation. Issue the following command and compare the output to that provided in the following:

SanJose2#ping

Protocol [ip]:
Target IP address: 192.168.100.1
Repeat count [5]: 2
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 has IP options: Total option bytes= 39, padded length=40
Record route: <*>
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)

(0.0.0.0)
Reply to request 0 (48 ms). Received packet has options Total option bytes= 40, padded length=40
Record route:
(172.16.1.2)
(192.168.1.6)
(192.168.100.1)
(192.168.1.1)
(172.16.32.1) <*>
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
End of list
Reply to request 1 (48 ms). Received packet has options Total option bytes= 40, padded length=40
Record route:
(172.16.1.2)
(192.168.1.6)
(192.168.100.1)
(192.168.1.1)
(172.16.32.1) <*>
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
End of list
The output can be interpreted as follows:

A ping that is sourced from 172.16.32.1 will exit SanJose2 through s0/0, 172.16.1.2. It will then arrive at the S0/1 interface for SanJose1. SanJose1 S0/0, 192.168.1.6, routes the packet out to arrive at the S0/0 interface of ISP1. The target of 192.168.100.1 is reached, 192.168.100.1. The packet is next forwarded out the S0/1, 192.168.1.1, interface for ISP1 and arrives at the S0/1 interface for SanJose2. 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, the ISP prefers the link from SanJose2 for all return traffic. The next step is to create a new policy to force the ISP to return all traffic via SanJose1. Create a second route-map utilizing the MED (metric) which is shared between EBGP neighbors.

SanJose1(config)#route-map PRIMARY_T1_MED_OUT permit 10
SanJose1(config-route-map)#set Metric 50
SanJose1(config-route-map)#exit
SanJose1(config)#router bgp 64512 SanJose1(config-router)#neighbor 192.168.1.5 route-map PRIMARY_T1_MED_OUT out
SanJose2(config)#route-map SECONDARY_T1_MED_OUT permit 10
SanJose2(config-route-map)#set Metric 75
SanJose2(config-route-map)#exit

SanJose2(config)#router bgp 64512 SanJose2(config-router)#neighbor 192.168.1.1 route-map SECONDARY_T1_MED_OUT out

As before, do not forget to clear ip bgp * after issuing this new policy. Issuing the show ip bgp command as follows on SanJose1 or SanJose2 will not indicate anything about this newly defined policy:

SanJose1#show ip bgp

BGP table version is 10, local router ID is 172.16.64.1
Status codes: s suppressed, d damped, h history, * valid, > best, i – internal
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
*> 192.168.1.0/30	192.168.1.5	0	150	0	200 i
*> 192.168.1.4/30	192.168.1.5	0	150	0	200 i
*> 192.168.100.0	192.168.1.5	0	150	0	200 i

Now reissue an extended ping with a record command as follows:

SanJose2#ping Protocol [ip]:

Target IP address: 192.168.100.1
Repeat count [5]: 2 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 has IP options: Total option bytes= 39, padded length=40
Record route: <*>
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
Reply to request 0 (64 ms). Received packet has options
Total option bytes= 40, padded length=40 Record route:
(172.16.1.2)
(192.168.1.6)

(192.168.100.1)
(192.168.1.5)
(172.16.1.1)
(172.16.32.1) <*>
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
End of list
Reply to request 1 (64 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

Does the output look correct? Does the 192.168.1.5 above mean that the ISP1 will now prefer SanJose1 for return traffic?

There may not be a chance to telnet to the ISP router and to issue the show ip bgp command. However, the command on the opposite side of the newly configured policy MED is clear, showing that the lower value is considered best. The ISP now prefers the route with the lower MED value to AS 64512. This is just opposite from the local-preference knob configured earlier.

BGP table version is 12, local router ID is 192.168.100.1 Status codes: s suppressed, d damped, h history, * valid, > best, i – internal Origin codes: i - IGP, e - EGP, ? - incomplete

Network	Next Hop	Metric	LocPrf	Weight	Path
* 172.16.0.0	192.168.1.2	75		0	64512 i
*>	192.168.1.6	50		0	64512 i
*> 192.168.1.0/30	0.0.0.0	0		32768	i
*> 192.168.1.4/30	0.0.0.0	0		32768	i
*> 192.168.100.0	0.0.0.0	0		32768	i

Step 12: The final step is to establish a default route that uses a policy statement that will adjust to changes in the network. Configure both SanJose1 and SanJose2 to use the 192.168.100.0 /24 network as the default network. The output that follows includes the routing table before the command was issued, the actual command syntax, and then the routing table after the command was issued. Complete the same task on the SanJose2 router.

SanJose1#show ip route

****Note: Prior to Default-Network Statement

Gateway of last resort is not set

172.16.0.0/16 is variably subnetted, 4 subnets, 2 masks
D 172.16.32.0/24 [90/20640000] via 172.16.1.2, 02:43:46, Serial0/1
172.16.0.0/16 [200/0] via 172.16.32.1, 00:12:32
172.16.1.0/24 is directly connected, Serial0/1 C 172.16.64.0/24 is directly connected, Loopback0
192.168.1.0/30 is subnetted, 2 subnets
192.168.1.0 [20/0] via 192.168.1.5, 00:14:05
192.168.1.4 is directly connected, Serial0/0 B 192.168.100.0/24 [20/0] via 192.168.1.5, 00:14:05

SanJose1(config)#ip default-network 192.168.100.0

SanJose1#show ip route

Gateway of last resort is 192.168.1.5 to network 192.168.100.0
172.16.0.0/16 is variably subnetted, 4 subnets, 2 masks
172.16.32.0/24 [90/20640000] via 172.16.1.2, 02:44:09, Serial0/1
172.16.0.0/16 [200/0] via 172.16.32.1, 00:12:55
172.16.1.0/24 is directly connected, Serial0/1 C 172.16.64.0/24 is directly connected, Loopback0
192.168.1.0/30 is subnetted, 2 subnets
192.168.1.0 [20/0] via 192.168.1.5, 00:14:28
192.168.1.4 is directly connected, Serial0/0 B* 192.168.100.0/24 [20/0] via 192.168.1.5, 00:14:29

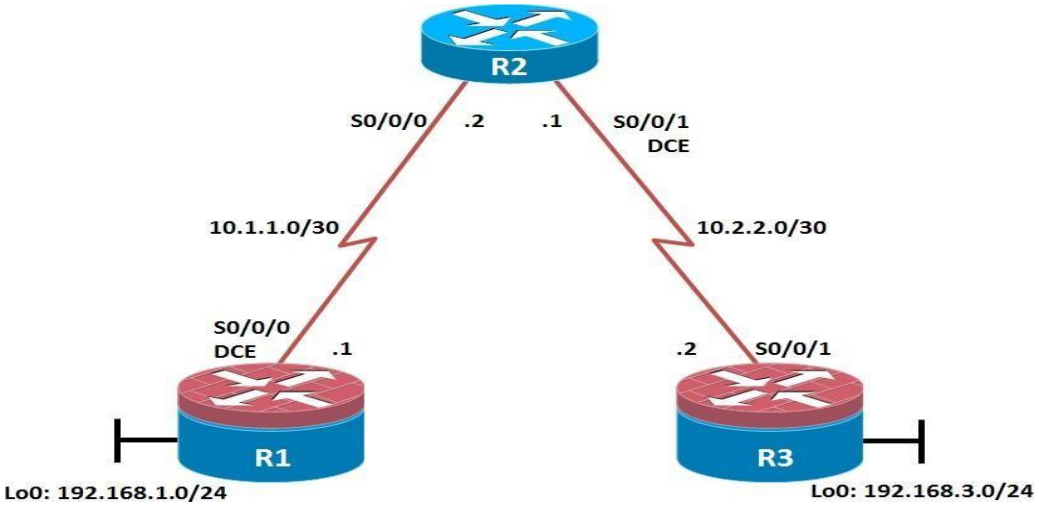
What would be required to add a future T3 link on SanJose2 and for this future link to have preference for incoming and outgoing traffic?

A newly added route would be as easy as adding another route-map for local-preference with a value of 175 and a route-map referencing a MED (metric) value of 35. Issue the clear ip bgp * command and this lab is then completed.

PRACTICAL 4

Aim: Secure the Management Plane

Topology



Objectives:

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

Background : The management plane of any infrastructure device should be protected as much as possible. Controlling access to routers and enabling reporting on routers are critical to network security and should be part of a comprehensive security policy. In this lab, you build a multi-router network and secure the management plane of routers R1 and R3.

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

Required Resources

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

Step 1: Configure loopbacks and assign addresses.

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

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

R1

hostname R1

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

R2

```
hostname R2
!
interface Serial0/0/0
 description R2 --> R1
 ip address 10.1.1.2 255.255.255.252
 no shutdown
exit
interface Serial0/0/1
 description R2 --> R3
 ip address 10.2.2.1 255.255.255.252
 clock rate 128000
 no shutdown
exit
!
End
```

R3

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

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

Step 2: Configure static routes.

On R1, configure a default static route to ISP.
R1(config)# ip route 0.0.0.0 0.0.0.0 10.1.1.2
On R3, configure a default static route to ISP.
R3(config)# ip route 0.0.0.0 0.0.0.0 10.2.2.1
On R2, configure two static routes.
R2(config)# ip route 192.168.1.0 255.255.255.0 10.1.1.1
R2(config)# ip route 192.168.3.0 255.255.255.0 10.2.2.2

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

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

```
R1#  tclsh
R1(tcl)#foreach address {
+>(tcl)#192.168.1.1
+>(tcl)#10.1.1.1
+>(tcl)#10.1.1.2
+>(tcl)#10.2.2.1
+>(tcl)#10.2.2.2
+>(tcl)#192.168.3.1
+>(tcl)#} { ping $address }

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.1.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.2.2.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
```

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.2.2.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 12/14/16 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.3.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 12/15/16 ms
R1(tcl)#
Are the pings now successful?
Yes. If not, troubleshoot.

Step 3: Secure management access.

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

R1(config)# security passwords min-length 10
Configure the enable secret encrypted password on both routers.
R1(config)# enable secret class12345
How does configuring an enable secret password help protect a router from being compromised by an attack?

The goal is to always prevent unauthorized users from accessing a device using Telnet, SSH, or via the console. Unless the enable secret password is known, a user cannot go into privileged EXEC mode where they can display the running config and enter various configuration commands to make changes to the router.

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

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

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

R1(config)# line console 0
R1(config-line)# password ciscoconpass
R1(config-line)# exec-timeout 5 0
R1(config-line)# login
R1(config-line)# logging synchronous
R1(config-line)# exit
R1(config)#
Configure the password on the vty lines for router R1.

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

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

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

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

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

Note : The enable secret password is encrypted automatically using the MD5 or SHA hash algorithm. . IOS 15.0(1)S and later default to SHA256 hashing algorithm. SHA256 which is considered to be a very strong hashing algorithm and is extremely difficult to reverse. Earlier IOS versions use the weaker MD5 hashing algorithm.

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

Can you read the console, aux, and vty passwords? Why or why not? Yes. They are all in clear text. Use the service password-encryption command to encrypt the line console and vty passwords.

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

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

Issue the show run command. Can you read the console, aux, and vty passwords? Why or why not? No. The passwords are now encrypted.

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

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

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

Issue the show run command. What does the $ convert to in the output? The $ is converted to ^C when the running-config is displayed.

Exit privileged EXEC mode using the disable or exit command and press Enter to get started. Does the MOTD banner look like what you created with the banner motd command? If the MOTD banner is not as you wanted it, recreate it using the banner motd command. Repeat the configuration portion of steps 3a through 3k on router R3

Step 4: Configure enhanced username password security.

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

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

R1(config)# username JR-ADMIN secret class12345
R1(config)# username ADMIN secret class54321
Note: An older method for creating local database entries is to use the username name password password command. Set the console line to use the locally defined login accounts.

R1(config)# line console 0
R1(config-line)# login local
R1(config-line)# exit
R1(config)#
Set the vty lines to use the locally defined login accounts.

R1(config)# line vty 0 4
R1(config-line)# login local
R1(config-line)# end
R1(config)#
Repeat the steps 4a to 4c on R3.

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

R1# telnet 10.2.2.2

Trying 10.2.2.2 ... Open
Unauthorized access strictly prohibited!
User Access Verification

Username: ADMIN
Password:
R3>

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

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

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

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

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

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

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

R1(config)# aaa new-model

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

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

R1(config)# radius server RADIUS-1
R1(config-radius-server)# address ipv4 192.168.1.101
R1(config-radius-server)# key RADIUS-1-pa55w0rd
R1(config-radius-server)# exit
R1(config)#
Configure the specifics for the second RADIUS server located at 192.168.1.102. Use RADIUS-2-pa55w0rd as the server password.

R1(config)# radius server RADIUS-2
R1(config-radius-server)# address ipv4 192.168.1.102
R1(config-radius-server)# key RADIUS-2-pa55w0rd
R1(config-radius-server)# exit
R1(config)#
Assign both RADIUS servers to a server group.

R1(config)# aaa group server radius RADIUS-GROUP
R1(config-sg-radius)# server name RADIUS-1
R1(config-sg-radius)# server name RADIUS-2
R1(config-sg-radius)# exit
R1(config)#
Enable the default AAA authentication login to attempt to validate against the server group. If they are not available, then authentication should be validated against the local database..

R1(config)# aaa authentication login default group RADIUS-GROUP local
R1(config)#
Note: Once this command is configured, all line access methods default to the default authentication method. The local option enables AAA to refer to the local database. Only the password is case sensitive. Enable the default AAA authentication Telnet login to attempt to validate against the server group. If they are not available, then authentication should be validated against a case sensitive local database.

R1(config)# aaa authentication login TELNET-LOGIN group RADIUS-GROUP local-case
R1(config)#
Note: Unlike the local option that makes the password is case sensitive, local-case makes the username and password case sensitive.

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

R1(config)# line vty 0 4
R1(config-line)# login authentication TELNET-LOGIN
R1(config-line)# exit
R1(config)#
Repeat the steps 5a to 5g on R3.

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

R1# telnet 10.2.2.2

Trying 10.2.2.2 ... Open
Unauthorized access strictly prohibited!
User Access Verification
Username: admin
Password:
% Authentication failed
Username: ADMIN
Password:
R3>

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

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

Step 6: Enabling secure remote management using SSH.

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

Secure Shell (SSH) is a network protocol that establishes a secure terminal emulation connection to a router or other networking device. SSH encrypts all information that passes over the network link and provides authentication of the remote computer.

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

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

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

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

R1(config)# crypto key zeroize rsa

Note: If no keys exist, you might receive this message: % No Signature RSA Keys found in configuration. Generate the RSA encryption key pair for the router. Configure the RSA keys with 1024 for the number of modulus bits. The default is 512, and the range is from 360 to 2048.

R1(config)# crypto key generate rsa general-keys modulus 1024

The name for the keys will be: R1.ccnasecurity.com
% The key modulus size is 1024 bits
% Generating 1024 bit RSA keys, keys will be non-exportable...[OK]

R1(config)#

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

R1(config)#

Cisco routers support two versions of SSH:

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

The default setting for SSH is SSH version 1.99. This is also known as compatibility mode and is merely an indication that the server supports both SSH version 2 and SSH version 1.

Configure SSH version 2 on R1.

R1(config)# ip ssh version 2
R1(config)#
Configure the vty lines to use only SSH connections.

R1(config)# line vty 0 4
R1(config-line)# transport input ssh
R1(config-line)# end
Note: SSH requires that the login local command be configured. However, in the previous step we enabled AAA authentication using the TELNET-LOGIN authentication method, therefore login local is not necessary.

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

Verify the SSH configuration using the show ip ssh command.

R1# show ip ssh

SSH Enabled - version 2.0

Authentication timeout: 120 secs; Authentication retries: 3
Minimum expected Diffie Hellman key size : 1024 bits
IOS Keys in SECSH format(ssh-rsa, base64 encoded):
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAAAgQC3Lehh7ReYlgyDzls6wq+mFzxqzoaZFr9X
Gx+Q/yiodFYw00hQo80tZy1W1Ff3Pz6q7Qi0y00urwddHZ0kBZceZK9EzJ6wZ+9a87KKD
ETCWrGSLi6c8lE/y4K+Z/oVrMMZk7bpTM1MFdP41YgkTf35utYv+TcqbsYo++KJiYk+x
w==
R1#

Repeat the steps 6a to 6f on R3.

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 (Instructor version)
Router R1
service password-encryption
!
hostname R1
!
security passwords min-length 10
enable secret 5 1t6eK$FZ.JdmMLj8QSgNkpChyZz.
!
aaa new-model
!
!
aaa group server radius RADIUS-GROUP
 server name RADIUS-1
 server name RADIUS-2
!
aaa authentication login default group RADIUS-GROUP local

aaa authentication login TELNET-LOGIN group RADIUS-GROUP local-case

!

ip domain name ccnasecurity.com

!

username JR-ADMIN secret 5 $1$0u0q$lwimCZIAuQtV4C1ezXL1S0

```
username ADMIN secret 5 $1$NSVD$/YjzB7Auyes1sAt4qMfpd.
!
ip ssh version 2
!
interface Loopback0
 description R1 LAN
 ip address 192.168.1.1 255.255.255.0
!
interface Serial0/0/0
 description R1 --> R2
 ip address 10.1.1.1 255.255.255.252
 no fair-queue
!
ip route 0.0.0.0 0.0.0.0 10.1.1.2
!
radius server RADIUS-1
 address ipv4 192.168.1.101 auth-port 1645 acct-port 1646
 key 7 107C283D2C2221465D493A2A717D24653017
!
radius server RADIUS-2
 address ipv4 192.168.1.102 auth-port 1645 acct-port 1646
 key 7 03367A2F2F3A12011C44090442471C5C162E
!
banner motd ^CUnauthorized access strictly prohibited!^C
!
line con 0
 exec-timeout 5 0
 password 7 070C285F4D061A0A19020A1F17
 logging synchronous
!
line aux 0
 no exec
!
password 7 060506324F411F0D1C0713181F
 login authentication TELNET-LOGIN
 transport input ssh
!
end
Router R2
hostname R2
!
enable secret 5 $1$DJS7$xvJDW87zLs8pSJDFUlCPB1
!
interface Serial0/0/0
 description R2 --> R1
 ip address 10.1.1.2 255.255.255.252
```

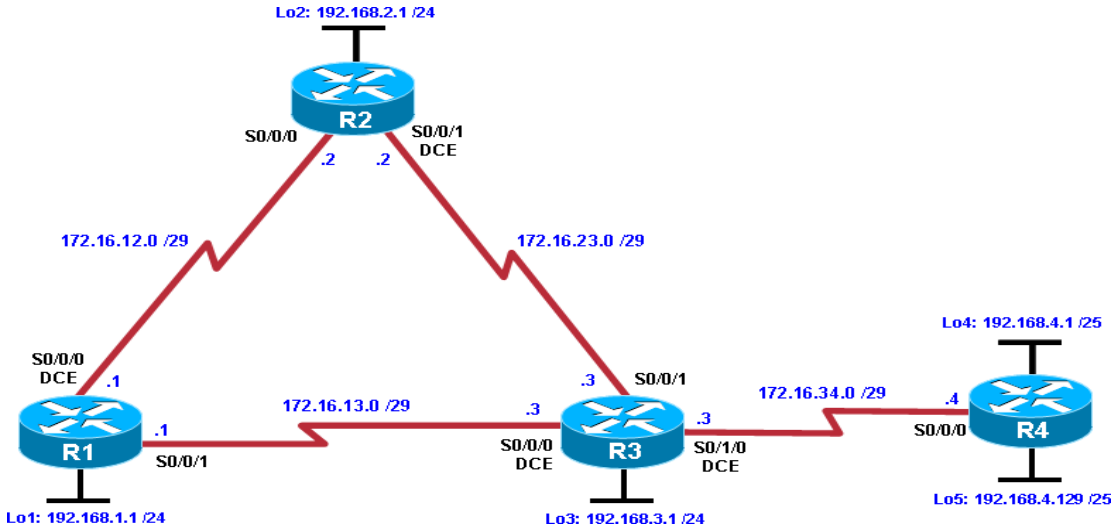
```
 no  fair-queue
 clock rate 2000000
!
interface Serial0/0/1
 description R2 --> R3
 ip address 10.2.2.1 255.255.255.252
 clock rate 128000
!
ip route 192.168.1.0 255.255.255.0 10.1.1.1
ip route 192.168.3.0 255.255.255.0 10.2.2.2
!
line con 0
 exec-timeout 0 0
 logging synchronous
!
line vty 0 4
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

service password-encryption
!
hostname R3
!
security passwords min-length 10
enable secret 5 $1$5OY4$4J6VFlvGNKjwQ8XtajgUk1
!
aaa new-model
!
!
aaa group server radius RADIUS-GROUP
 server name RADIUS-1
 server name RADIUS-2
!
aaa authentication login default group RADIUS-GROUP local
aaa authentication login TELNET-LOGIN group RADIUS-GROUP local-case
!
ip domain name ccnasecurity.com
!
username JR-ADMIN secret 5 $1$b4m1$RVmjL9S3gxKh1xr8qzNqr/
username ADMIN secret 5 $1$zGV7$pVgSEbinvXQ7f7uyxeKBj0
!
ip ssh version 2
!
```

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

PRACTICAL 5

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.

Background : You want to experiment with policy-based routing (PBR) to see how it is implemented and to study how it could be of value to your organization. All routers are exchanging routing information using EIGRP.

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

Required Resources

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

Step 1: Configure loopbacks and assign addresses.

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

b. 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.

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

Router R1

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

Router R2

```
hostname R2
!
interface Lo2
 description R2 LAN
 ip address 192.168.2.1 255.255.255.0
!
interface Serial0/0/0
 description R2 --> R1
 ip address 172.16.12.2 255.255.255.248
 bandwidth 128
 no shutdown
interface Serial0/0/1
description R2 --> R3
 ip address 172.16.23.2 255.255.255.248
 clock rate 128000
 bandwidth 128
 no shutdown
!
end
```

Router R3

```
hostname R3
```

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

Router R4

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

C.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 | include up
Serial0/0/0            172.16.13.3     YES manual up                    up
Serial0/0/1            172.16.23.3     YES manual up                    up
Serial0/1/0            172.16.34.3     YES manual up                    up
Loopback3              192.168.3.1     YES manual up                    up
R3#

R3# show protocols

Global values:
  Internet Protocol routing is enabled
Embedded-Service-Engine0/0 is administratively down, line protocol is down
GigabitEthernet0/0 is administratively down, line protocol is down
GigabitEthernet0/1 is administratively down, line protocol is down
Serial0/0/0 is up, line protocol is up
  Internet address is 172.16.13.3/29
Serial0/0/1 is up, line protocol is up
Internet address is 172.16.23.3/29
Serial0/1/0 is up, line protocol is up
Internet address is 172.16.34.3/29
Serial0/1/1 is administratively down, line protocol is down
Loopback3 is up, line protocol is up
  Internet address is 192.168.3.1/24
R3#

R3# show interfaces description | include up
Se0/0/0                        up              up      R3 --> R1
Se0/0/1                        up              up      R3 --> R2
Se0/1/0                        up              up      R3 --> R4
Lo3                            up              up      R3 LAN
R3#
```

Step 3: Configure basic EIGRP.

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

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

Router R1

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

Router R2

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

Router R3

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

Router R4

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

You should see EIGRP neighbor relationship messages being generated.

Step 4: Verify EIGRP connectivity.

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

R1# **show ip eigrp neighbors**

EIGRP-IPv4 Neighbors for AS(1)

H	Address	Interface	Hold (sec)	Uptime	SRTT (ms)	RTO	Q Cnt	Seq Num
1	172.16.13.3	Se0/0/1	10	00:01:55	27	2340	0	9
0	172.16.12.2	Se0/0/0	13	00:02:07	8	1170	0	11

R1#

R2# **show ip eigrp neighbors**

EIGRP-IPv4 Neighbors for AS(1)

H	Address	Interface	Hold (sec)	Uptime	SRTT (ms)	RTO	Q Cnt	Seq Num
1	172.16.23.3	Se0/0/1	12	00:02:15	12	1170	0	10
0	172.16.12.1	Se0/0/0	11	00:02:27	9	1170	0	13

R2#

R3# **show ip eigrp neighbors**

EIGRP-IPv4 Neighbors for AS(1)

H	Address	Interface	Hold (sec)	Uptime	SRTT (ms)	RTO	Q Cnt	Seq Num
2	172.16.34.4	Se0/1/0	12	00:02:14	44	2340	0	3
1	172.16.23.2	Se0/0/1	11	00:02:23	10	1170	0	10
0	172.16.13.1	Se0/0/0	10	00:02:23	1031	5000	0	12

R3#

R4# **show ip eigrp neighbors**

```
EIGRP-IPv4 Neighbors for AS(1)
H  Address                 Interface       Hold Uptime   SRTT   RTO  Q  Seq
                                           (sec)         (ms)       Cnt Num
0  172.16.34.3             Se0/0/0          10 00:02:22  37  2340  0  11

R4#
```

Did you receive the output you expected?

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

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

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

Step 5: Verify the current path.

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

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

```
R1# show ip route | begin Gateway
Gateway of last resort is not set
     172.16.0.0/16 is variably subnetted, 6 subnets, 2 masks
C       172.16.12.0/29 is directly connected, Serial0/0/0
L  172.16.12.1/32 is directly connected, Serial0/0/0C
172.16.13.0/29 is directly connected, Serial0/0/1 L
172.16.13.1/32 is directly connected, Serial0/0/1
D       172.16.23.0/29 [90/21024000] via 172.16.12.2, 00:07:22, Serial0/0/0
D  172.16.34.0/29 [90/41024000] via 172.16.13.3, 00:07:22, Serial0/0/1
     192.168.1.0/24 is variably subnetted, 2 subnets, 2 masks
C  192.168.1.0/24 is directly connected, Loopback1
L       192.168.1.1/32 is directly connected, Loopback1
D    192.168.2.0/24 [90/20640000] via 172.16.12.2, 00:07:22, Serial0/0/0
D  192.168.3.0/24 [90/21152000] via 172.16.12.2, 00:07:22, Serial0/0/0
     192.168.4.0/25 is subnetted, 2 subnets
D       192.168.4.0 [90/41152000] via 172.16.13.3, 00:07:14, Serial0/0/1
D       192.168.4.128 [90/41152000] via 172.16.13.3, 00:07:14, Serial0/0/1
```

R1#

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

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

R4# **traceroute 192.168.1.1 source 192.168.4.1**

```
Type escape sequence to abort.
Tracing the route to 192.168.1.1
VRF info: (vrf in name/id, vrf out name/id)
  1 172.16.34.3 12 msec 12 msec 16 msec
  2 172.16.23.2 20 msec 20 msec 20 msec
  3 172.16.12.1 24 msec * 24 msec
```

R4#

R4# **traceroute 192.168.1.1 source 192.168.4.129**

```
Type escape sequence to abort.
Tracing the route to 192.168.1.1
VRF info: (vrf in name/id, vrf out name/id)
  1 172.16.34.3 12 msec 16 msec 12 msec
  2 172.16.23.2 28 msec 20 msec 16 msec
  3 172.16.12.1 24 msec * 24 msec
```

R4#

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

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

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

R3# **show ip route | begin Gateway**

```
Gateway of last resort is not set
     172.16.0.0/16 is variably subnetted, 7 subnets, 2 masks
D       172.16.12.0/29 [90/21024000] via 172.16.23.2, 00:10:54, Serial0/0/1
C     172.16.13.0/29 is directly connected, Serial0/0/0 L
172.16.13.3/32 is directly connected, Serial0/0/0C
172.16.23.0/29 is directly connected, Serial0/0/1 L
172.16.23.3/32 is directly connected, Serial0/0/1 C
172.16.34.0/29 is directly connected, Serial0/1/0 L
172.16.34.3/32 is directly connected, Serial0/1/0
D       192.168.1.0/24 [90/21152000] via 172.16.23.2, 00:10:54, Serial0/0/1
D  192.168.2.0/24 [90/20640000] via 172.16.23.2, 00:10:54, Serial0/0/1
        192.168.3.0/24 is variably subnetted, 2 subnets, 2 masks
C  192.168.3.0/24 is directly connected, Loopback3
L       192.168.3.1/32 is directly connected, Loopback3
        192.168.4.0/25 is subnetted, 2 subnets
D       192.168.4.0 [90/40640000] via 172.16.34.4, 00:10:47, Serial0/1/0
D       192.168.4.128 [90/40640000] via 172.16.34.4, 00:10:47, Serial0/1/0
R3#
```

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

```
R3# show interfaces serial0/0/0
Serial0/0/0 is up, line protocol is up    D 192.168.1.0/24 [90/21152000] via
172.16.23.2, 00:10:54, Serial0/0/1
  Hardware is WIC MBRD Serial
  Description: R3 --> R1
  Internet address is 172.16.13.3/29
  MTU 1500 bytes, BW 64 Kbit/sec, DLY 20000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation HDLC, loopback not set
  Keepalive set (10 sec)
  Last input 00:00:01, output 00:00:00, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: fifo
  Output queue: 0/40 (size/max)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     399 packets input, 29561 bytes, 0 no buffer
     Received 186 broadcasts (0 IP multicasts)
     0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     393 packets output, 29567 bytes, 0 underruns
     0 output errors, 0 collisions, 3 interface resets
     0 unknown protocol drops
     0 output buffer failures, 0 output buffers swapped out
     0 carrier transitions
     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up

R3# show interfaces serial0/0/0 | include BW
  MTU 1500 bytes, BW 64 Kbit/sec, DLY 20000 usec,
R3# show interfaces serial0/0/1 | include BW
  MTU 1500 bytes, BW 128 Kbit/sec, DLY 20000 usec,
R3#
```

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

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

```
R3# show ip eigrp topology 192.168.1.0
EIGRP-IPv4 Topology Entry for AS(1)/ID(192.168.3.1) for 192.168.1.0/24
  State is Passive, Query origin flag is 1, 1 Successor(s), FD is 21152000
  Descriptor Blocks:
  172.16.23.2 (Serial0/0/1), from 172.16.23.2, Send flag is 0x0
     Composite metric is (21152000/20640000), route is Internal
     Vector metric:
       Minimum bandwidth is 128 Kbit
       Total delay is 45000 microseconds
       Reliability is 255/255
       Load is 1/255
```

```
      Minimum MTU is 1500
      Hop count is 2
      Originating router is 192.168.1.1
    172.16.13.1 (Serial0/0/0), from 172.16.13.1, Send flag is 0x0
      Composite metric is (40640000/128256), route is Internal
      Vector metric:
        Minimum bandwidth is 64 Kbit
        Total delay is 25000 microseconds
        Reliability is 255/255
        Load is  1/255
        Minimum MTU is 1500
        Hop count is 1
        Originating router is 192.168.1.1

R3#
```

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

Step 6: Configure PBR to provide path control.

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

Recall that routers normally forward packets to destination addresses based on information in their routing table. By using PBR, you can implement policies that selectively cause packets to take different paths based on source address, protocol type, or application type.Configuring PBR involves configuring a route map with **match** and **set** commands and then applying the route map to the interface.

The steps required to implement path control include the following:

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

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

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

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

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

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

```
R3(config)# route-map R3-to-R1 permit
R3(config-route-map)# description RM to forward LAN B traffic to R1
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
R3(config)#
```

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

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

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

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

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

Step 7: Test the policy.

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

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

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

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

LANs.R3# **debug ip policy ?**

```
  <1-199> Access list
  dynamic  dynamic PBR
  <cr>
```

R3# **debug ip policy 1**

```
  Policy routing debugging is on for access list 1
```

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

R4# **traceroute 192.168.1.1 source 192.168.4.1**

Type escape sequence to abort.
Tracing the route to 192.168.1.1 172.16.12.1 4
1 172.16.34.3 0 msec 0 msec 4 msec
2 172.16.34.3 0 msec 0 msec 4 msec
3 172.16.12.1 4 msec 0 msec *

Notice path taken packet sourced from R4 LAN A is still going through R3 --> R2 --> R1. As the traceroute was being executed, router R3 should be generating the following debug output.

R3#

Jan 10 10:49:48.411: IP: s=192.168.4.1 (Serial0/1/0), d=192.168.1.1, len 28, policy rejected—normal forwarding

Jan 10 10:49:48.427: IP: s=192.168.4.1 (Serial0/1/0), d=192.168.1.1, len 28, policy rejected—normal forwarding

Jan 10 10:49:48.439: IP: s=192.168.4.1 (Serial0/1/0), d=192.168.1.1, len 28, policy rejected—normal forwarding

Jan 10 10:49:48.451: IP: s=192.168.4.1 (Serial0/1/0), d=192.168.1.1, len 28, FIB policy rejected(no match) - normal forwarding

Jan 10 10:49:48.471: IP: s=192.168.4.1 (Serial0/1/0), d=192.168.1.1, len 28, FIB policy rejected(no match) - normal forwarding

Jan 10 10:49:48.491: IP: s=192.168.4.1 (Serial0/1/0), d=192.168.1.1, len 28, FIB policy rejected(no match) - normal forwarding

Jan 10 10:49:48.511: IP: s=192.168.4.1 (Serial0/1/0), d=192.168.1.1, len 28, FIB policy rejected(no match) - normal forwarding

Jan 10 10:49:48.539: IP: s=192.168.4.1 (Serial0/1/0), d=192.168.1.1, len 28, FIB policy rejected(no match) - normal forwarding

Jan 10 10:49:51.539: IP: s=192.168.4.1 (Serial0/1/0), d=192.168.1.1, len 28, FIB policy rejected(no match) - normal forwarding

R3#

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

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

R4# **traceroute 192.168.1.1 source 192.168.4.129**
Type escape sequence to abort.
Tracing the route to 192.168.1.1
1 172.16.34.3 12 msec 12 msec 16 msec
2 172.16.34.3 28 msec 28 msec *

Now the path taken for the packet sourced from R4 LAN B is R3 --> R1, as expected. The debug output on R3 also confirms that the traffic meets the criteria of the R3-to-R1 policy.

R3#
R3#
Jan 10 10:50:04.283: IP: s=192.168.4.129 (Serial0/1/0), d=192.168.1.1, len 28, policy match

Jan 10 10:50:04.283: IP: route map R3-to-R1, item 10, permit
Jan 10 10:50:04.283: IP: s=192.168.4.129 (Serial0/1/0), d=192.168.1.1 (Serial0/0/0), len 28, policy routed
Jan 10 10:50:04.283: IP: Serial0/1/0 to Serial0/0/0 172.16.13.1
Jan 10 10:50:04.295: IP: s=192.168.4.129 (Serial0/1/0), d=192.168.1.1, len 28, policy match
Jan 10 10:50:04.295: IP: route map R3-to-R1, item 10, permit
Jan 10 10:50:04.295: IP: s=192.168.4.129 (Serial0/1/0), d=192.168.1.1 (Serial0/0/0), len 28, policy routed
Jan 10 10:50:04.295: IP: Serial0/1/0 to Serial0/0/0 172.16.13.1
Jan 10 10:50:04.311: IP: s=192.168.4.129 (Serial0/1/0), d=192.168.1.1, len 28, policy match
Jan 10 10:50:04.311: IP: route map R3-to-R1, item 10, permit
Jan 10 10:50:04.311: IP: s=192.168.4.129 (Serial0/1/0), d=192.168.1.1 (Serial0/0/0), len 28, policy routed
Jan 10 10:50:04.311: IP: Serial0/1/0 to Serial0/0/0 172.16.13.1
Jan 10 10:50:04.323: IP: s=192.168.4.129 (Serial0/1/0), d=192.168.1.1, len 28, FIB policy match
Jan 10 10:50:04.323: IP: s=192.168.4.129 (Serial0/1/0), d=192.168.1.1, len 28, PBR Counted
Jan 10 10:50:04.323: IP: s=192.168.4.129 (Serial0/1/0), d=192.168.1.1, g=172.16.13.1, len 28, FIB policy routed
Jan 10 10:50:04.351: IP: s=192.168.4.129 (Serial0/1/0), d=192.168.1.1, len 28, FIB policy match
Jan 10 10:50:04.351: IP: s=192.168.4.129 (Serial0/1/0), d=192.168.1.1, len 28, PBR Counted
Jan 10 10:50:04.351: IP: s=192.168.4.129 (Serial0/1/0), d=192.168.1.1, g=172.16.13.1, len 28, FIB policy routed
Jan 10 10:50:07.347: IP: s=192.168.4.129 (Serial0/1/0), d=192.168.1.1, len 28, FIB policy match
Jan 10 10:50:07.347: IP: s=192.168.4.129 (Serial0/1/0), d=192.168.1.1, len 28, PBR Counted
Jan 10 10:50:07.347: IP: s=192.168.4.129 (Serial0/1/0), d=192.168.1.1, g=172.16.13.1, len 28, FIB policy routed
R3#

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

R3# **show route-map**
route-map R3-to-R1, permit, sequence 10
  Match clauses:
    ip address (access-lists): PBR-ACL
  Set clauses:
    ip next-hop 172.16.13.1
Nexthop tracking current: 0.0.0.0
172.16.13.1, fib_nh:0,oce:0,status:0
Policy routing matches : 12 packets, 384 bytes

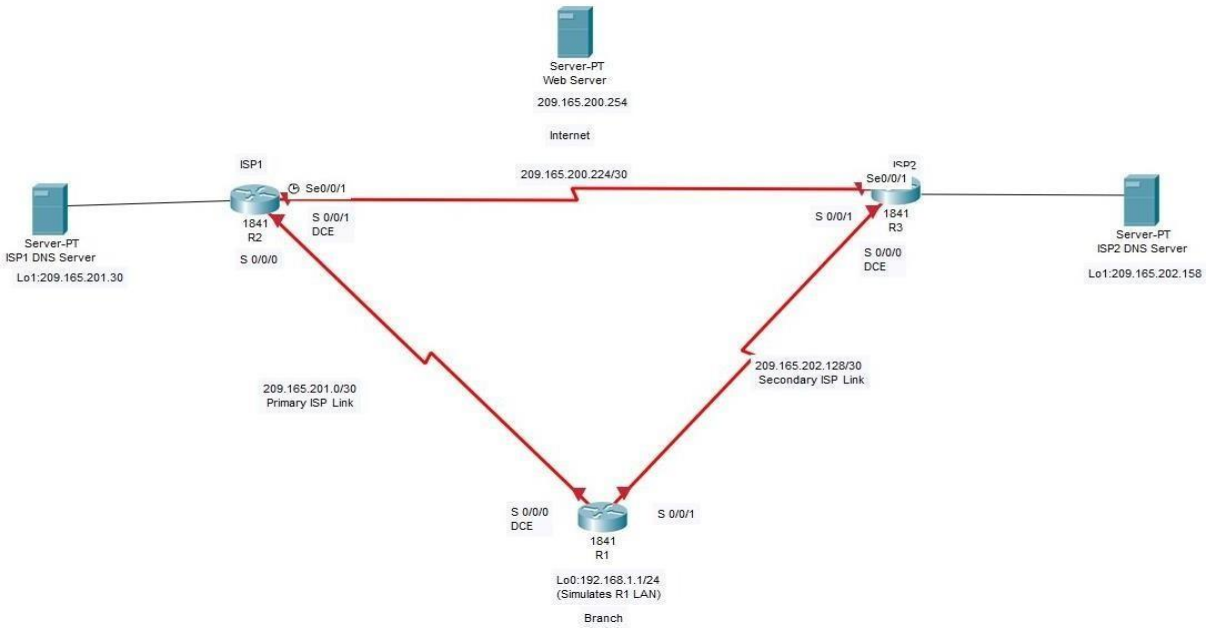
R3#

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

PRACTICAL 6

Aim: Configure IP SLA Tracking and Path Control

Topology



Objectives

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

Background

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

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

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

Required Resources

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

Step 1: Configure loopbacks and assign addresses.

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

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

Router R1

```
hostname R1
interface Loopback 0
description R1 LAN
 ip address 192.168.1.1 255.255.255.0
interface Serial0/0/0
 description R1 --> ISP1
 ip address 209.165.201.2 255.255.255.252
 clock rate 128000
 bandwidth 128
 no shutdown
interface Serial0/0/1
 description R1 --> ISP2
 ip address 209.165.202.130 255.255.255.252
 bandwidth 128
 no shutdown
```

Router ISP1 (R2)

```
hostname ISP1
interface Loopback0
 description Simulated Internet Web Server
 ip address 209.165.200.254 255.255.255.255
interface Loopback1
 description ISP1 DNS Server
 ip address 209.165.201.30 255.255.255.255
interface Serial0/0/0
description ISP1 --> R1
ip address 209.165.201.1 255.255.255.252
 bandwidth 128
 no shutdown
interface Serial0/0/1
 description ISP1 --> ISP2
 ip address 209.165.200.225 255.255.255.252
 clock rate 128000
 bandwidth 128
 no shutdown
```

Router ISP2 (R3)

```
hostname ISP2
interface Loopback0
 description Simulated Internet Web Server
 ip address 209.165.200.254 255.255.255.255
interface Loopback1
 description ISP2 DNS Server
 ip address 209.165.202.158 255.255.255.255
interface Serial0/0/0
 description ISP2 --> R1
 ip address 209.165.202.129 255.255.255.252
 clock rate 128000
 bandwidth 128
 no shutdown
interface Serial0/0/1
 description ISP2 --> ISP1
 ip address 209.165.200.226 255.255.255.252
 bandwidth 128
 no shutdown
```

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

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

All three interfaces should be active. Troubleshoot if necessary.

Step 2: Configure static routing.

The current routing policy in the topology is as follows:
Router R1 establishes connectivity to the Internet through ISP1 using a default static route.
ISP1 and ISP2 have dynamic routing enabled between them, advertising their respective public address pools.
ISP1 and ISP2 both have static routes back to the ISP LAN.
Note: For the purpose of this lab, the ISPs have a static route to an RFC 1918 private network address on the branch router R1. In an actual branch implementation, Network Address Translation (NAT) would be configured for all traffic exiting the branch LAN. Therefore, the static routes on the ISP routers would be pointing to the provided public pool of the branch office. .

Router R1

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

Router ISP1 (R2)

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

Router ISP2 (R3)

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

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

The Cisco IOS IP SLA feature enables an administrator to monitor network performance between Cisco devices (switches or routers) or from a Cisco device to a remote IP device. IP SLA probes continuously check the reachability of a specific destination, such as a provider edge router interface, the DNS server of the ISP, or any other specific destination, and can conditionally announce a default route only if the connectivity is verified. Before implementing the Cisco IOS SLA feature, you must verify reachability to the Internet servers. From router R1, ping the web server, ISP1 DNS server, and ISP2 DNS server to verify connectivity. You can copy the following Tcl script and paste it into R1.

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

All pings should be successful. Troubleshoot if necessary.
Trace the path taken to the web server, ISP1 DNS server, and ISP2 DNS server. You can copy the following Tcl script and paste it into R1.

```
foreach address {
209.165.200.254
209.165.201.30
209.165.202.158
```

} {
trace $address source 192.168.1.1
}
Through which ISP is traffic flowing?

Step 3: Configure IP SLA probes.

When the reachability tests are successful, you can configure the Cisco IOS IP SLAs probes. Different types of probes can be created, including FTP, HTTP, and jitter probes. In this scenario, you will configure ICMP echo probes.Create an ICMP echo probe on R1 to the primary DNS server on ISP1 using the ip sla command.

R1(config)# ip sla 11
R1(config-ip-sla)# icmp-echo 209.165.201.30
R1(config-ip-sla-echo)# frequency 10
R1(config-ip-sla-echo)# exit
R1(config)#
R1(config)# ip sla schedule 11 life forever start-time now
R1(config)#
The operation number of 11 is only locally significant to the router. The frequency 10 command schedules the connectivity test to repeat every 10 seconds. The probe is scheduled to start now and to run forever.

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

R1# show ip sla configuration 11

IP SLAs Infrastructure Engine-III
Entry number: 11
Owner:
Tag:
Operation timeout (milliseconds): 5000
Type of operation to perform: icmp-echo
Target address/Source address: 209.165.201.30/0.0.0.0
Type Of Service parameter: 0x0
Request size (ARR data portion): 28
Verify data: No
Vrf Name:
Schedule:
  Operation frequency (seconds): 10 (not considered if randomly scheduled)
  Next Scheduled Start Time: Start Time already passed
  Group Scheduled : FALSE
  Randomly Scheduled : FALSE
  Life (seconds): Forever
  Entry Ageout (seconds): never
  Recurring (Starting Everyday): FALSE
  Status of entry (SNMP RowStatus): Active

Threshold (milliseconds): 5000

```
Distribution Statistics:
  Number of statistic hours kept: 2
  Number of statistic distribution buckets kept: 1
  Statistic distribution interval (milliseconds): 20
Enhanced History:
History Statistics:
  Number of history Lives kept: 0
  Number of history Buckets kept: 15
  History Filter Type: None
R1#
```

The output lists the details of the configuration of operation 11. The operation is an ICMP echo to 209.165.201.30, with a frequency of 10 seconds, and it has already started (the start time has already passed). Issue the show ip sla statistics command to display the number of successes, failures, and results of the latest operations.

```
R1# show ip sla statistics

IPSLAs Latest Operation Statistics
IPSLA operation id: 11
        Latest RTT: 8 milliseconds
Latest operation start time: 10:33:18 UTC Sat Jan 10 2015
Latest operation return code: OK
Number of successes: 51
Number of failures: 0
Operation time to live: Forever
R1#
```

You can see that operation 11 has already succeeded five times, has had no failures, and the last operation returned an OK result. Although not actually required because IP SLA session 11 alone could provide the desired fault tolerance, create a second probe, 22, to test connectivity to the second DNS server located on router ISP2.

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

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

```
R1# show ip sla configuration 22

IP SLAs Infrastructure Engine-III
Entry number: 22
Owner:
Tag:
```

Operation timeout (milliseconds): 5000
Type of operation to perform: icmp-echo
Target address/Source address: 209.165.202.158/0.0.0.0
Type Of Service parameter: 0x0
Request size (ARR data portion): 28
Verify data: No
Vrf Name:
Schedule:
  Operation frequency (seconds): 10 (not considered if randomly scheduled)
  Next Scheduled Start Time: Start Time already passed
  Group Scheduled : FALSE
  Randomly Scheduled : FALSE
  Life (seconds): Forever
  Entry Ageout (seconds): never
  Recurring (Starting Everyday): FALSE
  Status of entry (SNMP RowStatus): Active
Threshold (milliseconds): 5000
Distribution Statistics:
  Number of statistic hours kept: 2
  Number of statistic distribution buckets kept: 1
  Statistic distribution interval (milliseconds): 20
Enhanced History:
History Statistics:
  Number of history Lives kept: 0
  Number of history Buckets kept: 15
  History Filter Type: None
R1#

R1# show ip sla configuration 22

IP SLAs, Infrastructure Engine-II.
Entry number: 22
Owner:
Tag:
Type of operation to perform: icmp-echo
Target address/Source address: 209.165.201.158/0.0.0.0
Type Of Service parameter: 0x0
Request size (ARR data portion): 28
Operation timeout (milliseconds): 5000
Verify data: No
Vrf Name:
Schedule:
Operation frequency (seconds): 10 (not considered if randomly scheduled)
  Next Scheduled Start Time: Start Time already passed
  Group Scheduled : FALSE
 Randomly Scheduled : FALSE
 Life (seconds): Forever

```
Entry Ageout (seconds): never
  Recurring (Starting Everyday): FALSE
  Status of entry (SNMP RowStatus): Active
Threshold (milliseconds): 5000 (not considered if react RTT is configured)
Distribution Statistics:
  Number of statistic hours kept: 2
  Number of statistic distribution buckets kept: 1
  Statistic distribution interval (milliseconds): 20
History Statistics:
  Number of history Lives kept: 0
  Number of history Buckets kept: 15
  History Filter Type: None
Enhanced History:
R1#
```

```
R1# show ip sla statistics 22
```

```
IPSLAs Latest Operation Statistics
IPSLA operation id: 22
        Latest RTT: 16 milliseconds
Latest operation start time: 10:38:29 UTC Sat Jan 10 2015
Latest operation return code: OK
Number of successes: 82
Number of failures: 0
Operation time to live: Forever
R1#
```

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

Step 4: Configure tracking options.

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

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

```
R1(config)# no ip route 0.0.0.0 0.0.0.0 209.165.201.1
R1(config)# ip route 0.0.0.0 0.0.0.0 209.165.201.1 5
R1(config)# exit
Verify the routing table.
```

```
R1# show ip route | begin Gateway
```

```
Gateway of last resort is 209.165.201.1 to network 0.0.0.0
S*    0.0.0.0/0 [5/0] via 209.165.201.1
     192.168.1.0/24 is variably subnetted, 2 subnets, 2 masks

C       192.168.1.0/24 is directly connected, Loopback0
```

```
L       192.168.1.1/32 is directly connected, Loopback0
     209.165.201.0/24 is variably subnetted, 2 subnets, 2 masks
C       209.165.201.0/30 is directly connected, Serial0/0/0
L       209.165.201.2/32 is directly connected, Serial0/0/0
     209.165.202.0/24 is variably subnetted, 2 subnets, 2 masks
C       209.165.202.128/30 is directly connected, Serial0/0/1
L       209.165.202.130/32 is directly connected, Serial0/0/1
R1#
```

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

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

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

R1(config-track)#
```

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

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

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

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

Configure the floating static route that will be implemented when tracking object 1 is active. Use the ip route 0.0.0.0 0.0.0.0 209.165.201.1 2 track 1 command to create a floating static default route via 209.165.201.1 (ISP1).

```
R1(config)# ip route 0.0.0.0 0.0.0.0 209.165.201.1 2 track 1
R1(config)#
Jan 10 10:45:39.119: RT: updating static 0.0.0.0/0 (0x0)  : via 209.165.201.1  0 1048578
Jan 10 10:45:39.119: RT: closer admin distance for 0.0.0.0, flushing 1 routes
Jan 10 10:45:39.119: RT: add 0.0.0.0/0 via 209.165.201.1, static metric [2/0]
Jan 10 10:45:39.119: RT: updating static 0.0.0.0/0 (0x0)  :via 209.165.201.1 0 1048578
Jan 10 10:45:39.119: RT: rib update return code: 17
Jan 10 10:45:39.119: RT: updating static 0.0.0.0/0 (0x0)  :via 209.165.201.1 0 1048578
Jan 10 10:45:39.119: RT: rib update return code: 17

R1(config)#
```

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

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

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

Verify the routing table again.

```
R1#show ip route | begin Gateway

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

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

Step 5: Verify IP SLA operation.

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

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

On ISP1, disable the loopback interface 1.

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

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

R1#

Jan 10 10:53:59.551: %TRACK-6-STATE: 1 ip sla 11 reachability Up -> Down
Jan 10 10:53:59.551: RT: del 0.0.0.0 via 209.165.201.1, static metric [2/0]
Jan 10 10:53:59.551: RT: delete network route to 0.0.0.0/0
Jan 10 10:53:59.551: RT: default path has been cleared
Jan 10 10:53:59.551: RT: updating static 0.0.0.0/0 (0x0) :via 209.165.202.129 0 1048578
Jan 10 10:53:59.551: RT: add 0.0.0.0/0 via 209.165.202.129, static metric [3/0]
Jan 10 10:53:59.551: RT: default path is now 0.0.0.0 via 209.165.202.129
Jan 10 10:53:59.551: RT: updating static 0.0.0.0/0 (0x0) : via 209.165.201.1 0 1048578
Jan 10 10:53:59.551: RT: rib update return code: 17
Jan 10 10:53:59.551: RT: updating static 0.0.0.0/0 (0x0) : via 209.165.202.129 0 1048578
Jan 10 10:53:59.551: RT: updating static 0.0.0.0/0 (0x0) : via 209.165.201.1 0 1048578
Jan 10 10:53:59.551: RT: rib update return code: 17
R1#

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

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

On R1, verify the routing table.

R1# show ip route | begin Gateway

Gateway of last resort is 209.165.202.129 to network 0.0.0.0

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

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

R1# show ip sla statistics

IPSLAs Latest Operation Statistics
IPSLA operation id: 11
Latest RTT: NoConnection/Busy/Timeout

Latest operation start time: 11:01:08 UTC Sat Jan 10 2015
Latest operation return code: Timeout
Number of successes: 173
Number of failures: 45
Operation time to live: Forever
IPSLA operation id: 22
        Latest RTT: 8 milliseconds
Latest operation start time: 11:01:09 UTC Sat Jan 10 2015
Latest operation return code: OK
Number of successes: 218
Number of failures: 0
Operation time to live: Forever
R1#

Notice that the latest return code is Timeout and there have been 45 failures on IP SLA object 11. On R1, initiate a trace to the web server from the internal LAN IP address.

R1# trace 209.165.200.254 source 192.168.1.1

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

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

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

ISP1(config-if)# no shutdown

Jan 10 11:05:45.847: %LINK-3-UPDOWN: Interface Loopback1, changed state to up
Jan 10 11:05:46.847: %LINEPROTO-5-UPDOWN: Line protocol on Interface Loopback1, changed state to up

ISP1(config-if)#

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

R1#

Jan 10 11:06:20.551: %TRACK-6-STATE: 1 ip sla 11 reachability Down -> Up
Jan 10 11:06:20.551: RT: updating static 0.0.0.0/0 (0x0) : via 209.165.201.1 0 1048578
Jan 10 11:06:20.551: RT: closer admin distance for 0.0.0.0, flushing 1 routes
Jan 10 11:06:20.551: RT: add 0.0.0.0/0 via 209.165.201.1, static metric [2/0]
Jan 10 11:06:20.551: RT: updating static 0.0.0.0/0 (0x0) : via 209.165.202.129 0 1048578
Jan 10 11:06:20.551: RT: rib update return code: 17
Jan 10 11:06:20.551: RT: u
R1#pdating static 0.0.0.0/0 (0x0) :

   via 209.165.202.129 0 1048578

```
Jan 10 11:06:20.551: RT: rib update return code: 17
Jan 10 11:06:20.551: RT: updating static 0.0.0.0/0 (0x0)  : via 209.165.201.1  0 1048578
Jan 10 11:06:20.551: RT: rib update return code: 17
R1#
```

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

```
R1# show ip sla statistics

IPSLAs Latest Operation Statistics
IPSLA operation id: 11
        Latest RTT: 8 milliseconds
Latest operation start time: 11:07:38 UTC Sat Jan 10 2015
Latest operation return code: OK
Number of successes: 182
Number of failures: 75
Operation time to live: Forever
IPSLA operation id: 22
        Latest RTT: 16 milliseconds
Latest operation start time: 11:07:39 UTC Sat Jan 10 2015
Latest operation return code: OK
Number of successes: 257
Number of failures: 0
Operation time to live: Forever
R1#
```

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

```
R1# show ip route | begin Gateway

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

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

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

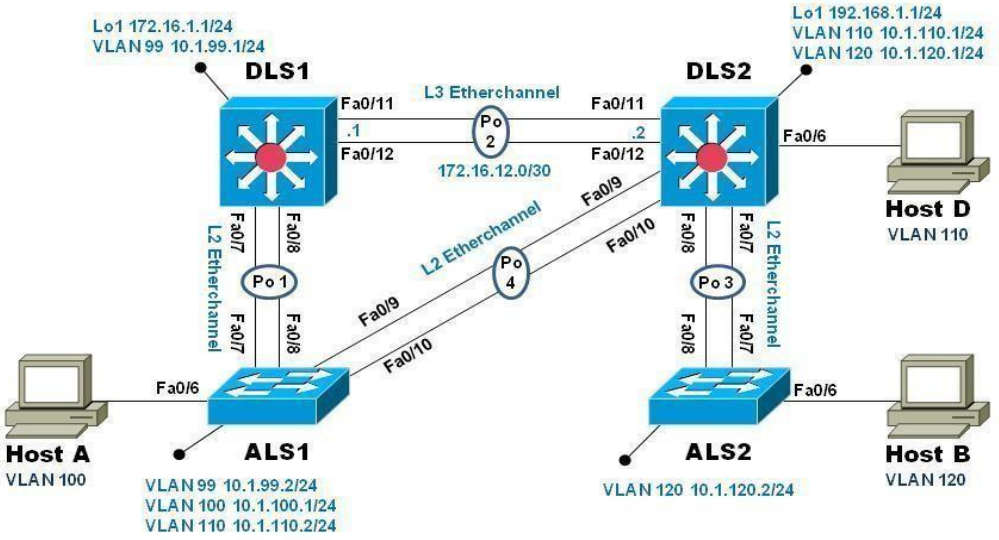
Before deploying a Cisco IOS IP SLA solution, the impact of the additional probe traffic being generated should be considered, including how that traffic affects bandwidth utilization, and congestion levels. Tuning the configuration (for example, with the delay and frequency commands) is critical to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

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

PRACTICAL 7

Aim: Inter-VLAN Routing

1. Topology



2.Objectives

- Implement a Layer 3 EtherChannel
- Implement Static Routing
- Implement Inter-VLAN Routing

3.Background

Cisco's switching product line offers robust support for IP routing. It is common practice to use only multi-layer switching in the distribution layer of the network, eliminating routers in all but special use cases, usually when a gateway interface is required. Doing so provides many benefits in terms of cost and manageability. In this lab you will configure Inter-VLAN routing on the multi-layer switches in your pod and then a Layer 3 EtherChannel link to interconnect them. You will further configure one of your access-layer switches to support basic routing, and apply static routes so that there is simple path control.

Note: This lab uses Cisco Catalyst 3560 and 2960 switches running Cisco IOS 15.0(2)SE6 IP Services and LAN Base images, respectively. The 3560 and 2960 switches are configured with the SDM templates “dual-ipv4-and-ipv6 routing” and “lanbase-routing”, respectively. Depending on the switch model and Cisco IOS Software version, the commands available and output produced might vary from what is shown in this lab. Catalyst 3650 switches (running any Cisco IOS XE release) and Catalyst 2960-Plus switches (running any comparable Cisco IOS image) can be used in place of the Catalyst 3560 switches and the Catalyst 2960 switches.

Required Resources

- 2 Cisco 2960 with the Cisco IOS Release 15.0(2)SE6 C2960-LANBASEK9-M or comparable
- 2 Cisco 3560v2 with the Cisco IOS Release 15.0(2)SE6 C3560-IPSERVICESK9-M or comparable
- Computer with terminal emulation software
- Ethernet and console cables
- 3 PCs with appropriate software

1. **Configure Multilayer Switching using Distribution Layer Switches**

1. Load base config

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

DLS1# **tclsh reset.tcl**

Erasing the nvram filesystem will remove all configuration files! Continue? [confirm]

[OK]

Erase of nvram: complete

Reloading the switch in 1 minute, type reload cancel to halt

Proceed with reload? [confirm]

*Mar 7 18:41:40.403: %SYS-7-NV_BLOCK_INIT: Initialized the geometry of nvram

*Mar 7 18:41:41.141: %SYS-5-RELOAD: Reload requested by console. Reload Reason: Reload command. <*switch reloads - output omitted*>

Would you like to enter the initial configuration dialog? [yes/no]: n

Switch> **en**

*Mar 1 00:01:30.915: %LINK-5-CHANGED: Interface Vlan1, changed state t administratively down

Switch# **copy BASE.CFG running-config**

Destination filename [running-config]?

184 bytes copied in 0.310 secs (594 bytes/sec)

DLS1#

2. Verify switch management database configuration

At each switch, use the show sdm prefer command to verify the appropriate template is chosen. The DLS switches should be using the "dual ipv4-and-ipv6 routing" template and the ALS switches should be using the "lanbase-routing" template. If any of the switches are using the wrong template, make the necessary change and reboot the switch with the **reload** command. An example from ALS1 is below:

ALS1# **sho sdm pref**

The current template is "default" template.

<output omitted>

ALS1# **conf t**

Enter configuration commands, one per line. End with CNTL/Z.

ALS1(config)# **sdm pref lanbase-routing**

Changes to the running SDM preferences have been stored, but cannot take effect until the next reload.

Use 'show sdm prefer' to see what SDM preference is currently active.

ALS1(config)# **end**

ALS1# **reload**

System configuration has been modified. Save? [yes/no]: **y**

*Mar 1 02:12:00.699: %SYS-5-CONFIG_I: Configured from console by console

Building configuration...

[OK]

Proceed with reload? [confirm]

3. Configure layer 3 interfaces on the DLS switches

Enable IP Routing, create broadcast domains (VLANs), and configure the DLS switches with the layer 3 interfaces and addresses shown:

Switch	Interface	Address/Mask
DLS1	VLAN 99	10.1.99.1/24
DLS1	Loopback 1	172.16.1.1/24
DLS2	VLAN 110	10.1.110.1/24
DLS2	VLAN 120	10.1.120.1/24
DLS2	Loopback 1	192.168.2.1/24

An example from DLS2:

DLS2(config)# **ip routing**

DLS2(config)# **vlan 110**

DLS2(config-vlan)# **name Management**

DLS2(config-vlan)# **exit**

DLS2(config)# **vlan 120**

DLS2(config-vlan)# **name Local**

DLS2(config-vlan)# **exit**

DLS2(config)# **int vlan 110**

DLS2(config-if)# **ip address 10.1.110.1 255.255.255.0**

DLS2(config-if)# **no shut**

DLS2(config-if)# **exit**

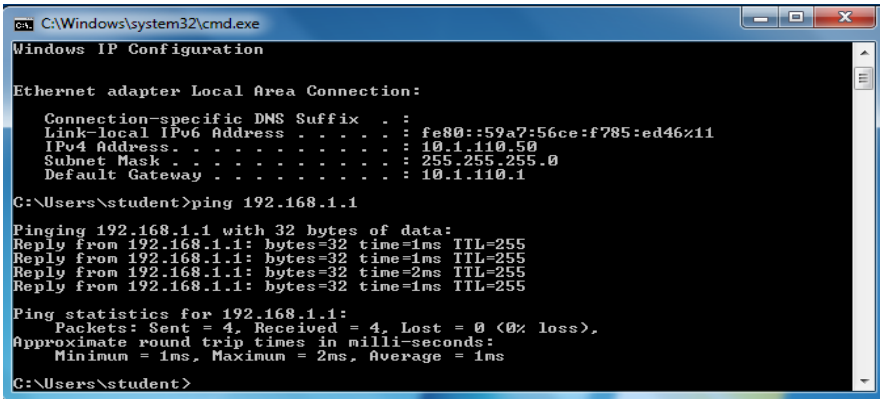
DLS2(config)# **int vlan 120**

DLS2(config-if)# **ip address 10.1.120.1 255.255.255.0**

DLS2(config-if)# **no shut**
DLS2(config-if)# **exit**
DLS2(config)# **int loopback 1**
DLS2(config-if)# **ip address 192.168.1.1 255.255.255.0**
DLS2(config-if)# **no shut**
DLS2(config-if)# **exit**
DLS2(config)#

At this point, basic inter vlan routing can be demonstrated using an attached host. Host D is attached to DLS2 via interface Fa0/6. On DLS2, assign interface Fa0/6 to VLAN 110 and configure the host with the address 10.1.110.50/24 and default gateway of 10.1.110.1. Once you have done that, try and ping Loopback 1's IP address (192.168.1.1). In the output below, the **switchport host** macro was used to quickly configure interface Fa0/6 with host-relative commands:

DLS2(config)# **int f0/6**
DLS2(config-if)# **switchport host**
switchport mode will be set to access
spanning-tree portfast will be enabled
channel group will be disabled
DLS2(config-if)# **switchport access vlan 110**
DLS2(config-if)# **no shut**
DLS2(config-if)# **exit**
DLS2(config)#



4. Configure a Layer 3 Etherchannel between DLS1 and DLS2

Now you will interconnect the multilayer switches in preparation to demonstrate other routing capabilities. Configure a layer 3 EtherChannel between the DLS switches. To convert the links from layer 2 to layer 3, issue the **no switchport** command. Then, combine interfaces F0/11 and F0/12 into a single PAgP EtherChannel and then assign an IP address as shown.

DLS1	172.16.12.1/30	DLS2	172.16.12.2/30

Example from DLS1:

DLS1(config)# **interface range f0/11-12**

```
DLS1(config-if-range)# no switchport
DLS1(config-if-range)# channel-group 2 mode desirable
Creating a port-channel interface Port-channel 2
DLS1(config-if-range)# no shut
DLS1(config-if-range)# exit
DLS1(config)# interface port-channel 2
DLS1(config-if)# ip address 172.16.12.1 255.255.255.252
DLS1(config-if)# no shut
DLS1(config-if)# exit
DLS1(config)#
```

Once you have configured both sides, verify that the EtherChannel link is up

```
DLS2# show etherchannel summary
Flags:  D - down        P - bundled in port-channel
        I - stand-alone s - suspended
        H - Hot-standby (LACP only)
        R - Layer3      S - Layer2
        U - in use      f - failed to allocate aggregator
        M - not in use, minimum links not met
        u - unsuitable for bundling
        w - waiting to be aggregated
        d - default port
Number of channel-groups in use: 1
Number of aggregators:           1
Group  Port-channel  Protocol    Ports
------+-------------+-----------+-----------------------------------------------
2      Po2(RU)           PAgP      Fa0/11(P) Fa0/12(P)
DLS2# ping 172.16.12.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.12.1, timeout is 2 seconds:
.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 1/3/9 ms
DLS2#
```

5. Configure default routing between DLS switches

At this point, local routing is support at each distribution layer switch. Now to provide reachability across the layer 3 EtherChannel trunk, configure fully qualified static default routes at DLS1 and DLS2 that point to each other. From DLS1:

```
DLS1(config)# ip route 0.0.0.0 0.0.0.0 port-channel 2
%Default route without gateway, if not a point-to-point interface, may impact performance
DLS1(config)# ip route 0.0.0.0 0.0.0.0 port-channel 2 172.16.12.2
DLS1(config)#
```

Once done at both ends, verify connectivity by pinging from one switch to the other. In the example below, DLS2 pings the Loopback 1 interface at DLS1.

DLS2# **show ip route**
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
<output omitted>
Gateway of last resort is 172.16.12.1 to network 0.0.0.0
S* 0.0.0.0/0 [1/0] via 172.16.12.1, Port-channel2
10.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C 10.1.110.0/24 is directly connected, Vlan110
L 10.1.110.1/32 is directly connected, Vlan110
172.16.0.0/16 is variably subnetted, 2 subnets, 2 masks
C 172.16.12.0/30 is directly connected, Port-channel2
L 172.16.12.2/32 is directly connected, Port-channel2
192.168.1.0/24 is variably subnetted, 2 subnets, 2 masks
C 192.168.1.0/24 is directly connected, Loopback1
L 192.168.1.1/32 is directly connected, Loopback1
DLS2# **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 100 percent (5/5), round-trip min/avg/max = 1/4/9 ms
DLS2#

6. Configure the remaining EtherChannels for the topology

Configure the remaining EtherChannel links as layer 2 PagP trunks using VLAN 1 as the native VLAN.

Endpoint 1	Channel number	Endpoint 2	VLANs Allowed
ALS1 F0/7-8	1	DLS1 F0/7-8	All except 110
ALS1 F0/9-10	4	DLS2 F0/9-10	110 Only
ALS2 F0/7-8	3	DLS2 F0/7-8	All

Example from ALS1:

ALS1(config)# **interface range f0/7-8**
ALS1(config-if-range)# **switchport mode trunk**
ALS1(config-if-range)# **switchport trunk allowed vlan except 110**
ALS1(config-if-range)# **channel-group 1 mode desirable**
Creating a port-channel interface Port-channel 1
ALS1(config-if-range)# **no shut**
ALS1(config-if-range)# **exit**
ALS1(config)# **interface range f0/9-10**
ALS1(config-if-range)# **switchport mode trunk**

```
ALS1(config-if-range)# switchport trunk allowed vlan 110
ALS1(config-if-range)# channel-group 4 mode desirable
Creating a port-channel interface Port-channel 4
ALS1(config-if-range)# no shut
ALS1(config-if-range)# exit
ALS1(config)#end
ALS1# show etherchannel summary
Flags:  D - down        P - bundled in port-channel
        I - stand-alone s - suspended
        H - Hot-standby (LACP only)
        R - Layer3      S - Layer2
        U - in use      f - failed to allocate aggregator
        M - not in use, minimum links not met
        u - unsuitable for bundling
        w - waiting to be aggregated
        d - default port
Number of channel-groups in use: 2
Number of aggregators:           2
Group  Port-channel  Protocol    Ports
------+-------------+-----------+-----------------------------------------------
1      Po1(SU)         PAgP      Fa0/7(P)    Fa0/8(P)
4      Po4(SU)         PAgP      Fa0/9(P)    Fa0/10(P)
ALS1# show interface trunk

Port        Mode         Encapsulation  Status        Native vlan
Po1         on           802.1q         trunking      1
Po4         on           802.1q         trunking      1

Port        Vlans allowed on trunk
Po1         1-109-,111-4049
Po4         110

<output omitted>
ALS1#
```

7. Enable and Verify Layer 3 connectivity across the network

In this step we will enable basic connectivity from the management VLANs on both sides of the network.

- Create the management VLANs (99 at ALS1, 120 at ALS2)
- Configure interface VLAN 99 at ALS1 and interface VLAN 120 at ALS2
- Assign addresses (refer to the diagram) and default gateways (at DLS1/DLS2 respectively).

Once that is all done, pings across the network should work, flowing across the layer 3 EtherChannel. An example from ALS2:

```
ALS2(config)# vlan 120
ALS2(config-vlan)# name Management
ALS2(config-vlan)# exit
ALS2(config)# int vlan 120
ALS2(config-if)# ip address 10.1.120.2 255.255.255.0
ALS2(config-if)# no shut
ALS2(config-if)# exit
ALS2(config)# ip default-gateway 10.1.120.1
ALS2(config)# end
ALS2# ping 10.1.99.2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.99.2, timeout is 2 seconds:
..!!!
Success rate is 60 percent (3/5), round-trip min/avg/max = 1/3/8 ms
ALS2#
ALS2# traceroute 10.1.99.2
Type escape sequence to abort.
Tracing the route to 10.1.99.2
VRF info: (vrf in name/id, vrf out name/id)
  1 10.1.120.1 0 msec 0 msec 8 msec
  2 172.16.12.1 0 msec 8 msec
  3 10.1.99.2 0 msec 0 msec *
ALS2#
```

2. Configure Multilayer Switching at ALS1

At this point all routing is going through the DLS switches, and the port channel between ALS1 and DLS2 is not passing anything but control traffic (BPDUs, etc).

The Cisco 2960 is able to support basic routing when it is using the LANBASE IOS. In this step you will configure ALS1 to support multiple SVIs and configure it for basic static routing. The objectives of this step are:

- Enable intervlan routing between two VLANs locally at ALS1
- Enable IP Routing
- Configure a static route for DLS2's Lo1 network travel via Port-Channel 4.

1. Configure additional VLANs and VLAN interfaces

At ALS1, create VLAN 100 and VLAN 110 and then create SVIs for those VLANs:

```
ALS1(config)# ip routing
ALS1(config)# vlan 100
ALS1(config-vlan)# name Local
ALS1(config-vlan)# exit
ALS1(config)# vlan 110
```

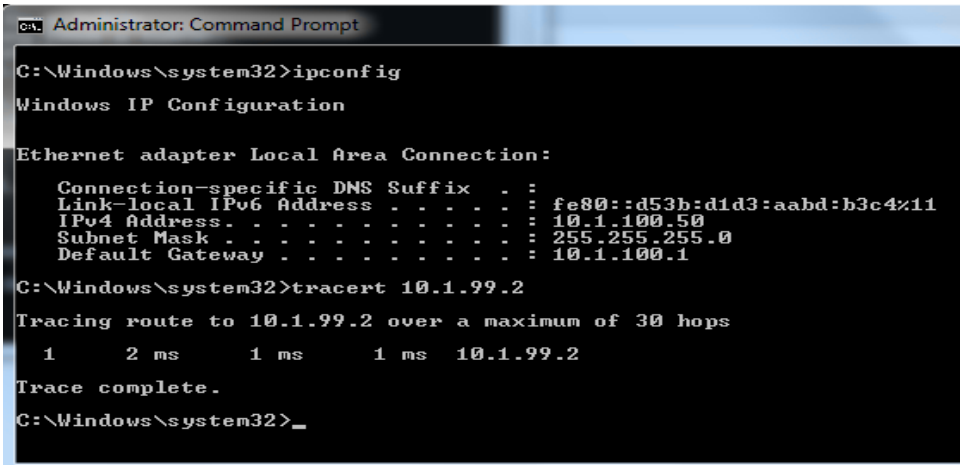
ALS1(config-vlan)# **name InterNode**
ALS1(config-vlan)# **exit**
ALS1(config)# **int vlan 100**
ALS1(config-if)# **ip address 10.1.100.1 255.255.255.0**
ALS1(config-if)# **no shut**
ALS1(config-if)# **exit**
ALS1(config)# **int vlan 110**
ALS1(config-if)# **ip address 10.1.110.2 255.255.255.0**
ALS1(config-if)# **no shut**
ALS1(config-if)# **exit**
ALS1(config)#

2. Configure and test Host Access

Assign interface Fa0/6 to VLAN 100. On the attached host (Host A) configure the IP address 10.1.100.50/24 with a default gateway of 10.1.100.1. Once configured, try a traceroute from the host to 10.1.99.2 and observe the results.

In the output below, the **switchport host** macro was used to quickly configure interface Fa0/6 with host-relative commands.

ALS1(config)# **interface f0/6**
ALS1(config-if)# **switchport host**
switchport mode will be set to access
spanning-tree portfast will be enabled
channel group will be disabled
ALS1(config-if)# **switchport access vlan 100**
ALS1(config-if)# **no shut**
ALS1(config-if)# **exit**



The output from the host shows that attempts to communicate with interface VLAN 99 at ALS1 were fulfilled locally, and not sent to DLS1 for routing.

3. Configure and verify static routing across the network

At this point, local routing (at ALS1) works, and off-net routing (outside of ALS1) will not work, because DLS1 doesn't have any knowledge of the 10.1.100.0 subnet. In this step you will configure routing on several different switches:

- At DLS1, configure:
  - a static route to the 10.1.100.0/24 network via VLAN 99
- At DLS2, configure
  - a static route to the 10.1.100.0/24 network via VLAN 110
- At ALS1, configure
  - a static route to the 192.168.1.0/24 network via VLAN 110
  - a default static route to use 10.1.99.1

Here is an example from ALS1:

ALS1(config)# **ip route 192.168.1.0 255.255.255.0 vlan 110**
ALS1(config)# **ip route 0.0.0.0 0.0.0.0 10.1.99.1**
ALS1(config)# **end**
ALS1# **show ip route**
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2
i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
ia - IS-IS inter area, * - candidate default, U - per-user static route
o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
+ - replicated route, % - next hop override
Gateway of last resort is 10.1.99.1 to network 0.0.0.0
S*0.0.0.0/0 [1/0] via 10.1.99.1
10.0.0.0/8 is variably subnetted, 6 subnets, 2 masks
C 10.1.99.0/24 is directly connected, Vlan99
L 10.1.99.2/32 is directly connected, Vlan99
C 10.1.100.0/24 is directly connected, Vlan100L
10.1.100.1/32 is directly connected, Vlan100 C
10.1.110.0/24 is directly connected, Vlan110 L
10.1.110.2/32 is directly connected, Vlan110S
192.168.1.0/24 is directly connected Vlan110
After configuring all of the required routes, test to see that the network behaves as expected.

From ALS1, a traceroute to 10.1.120.2 should take three hops:

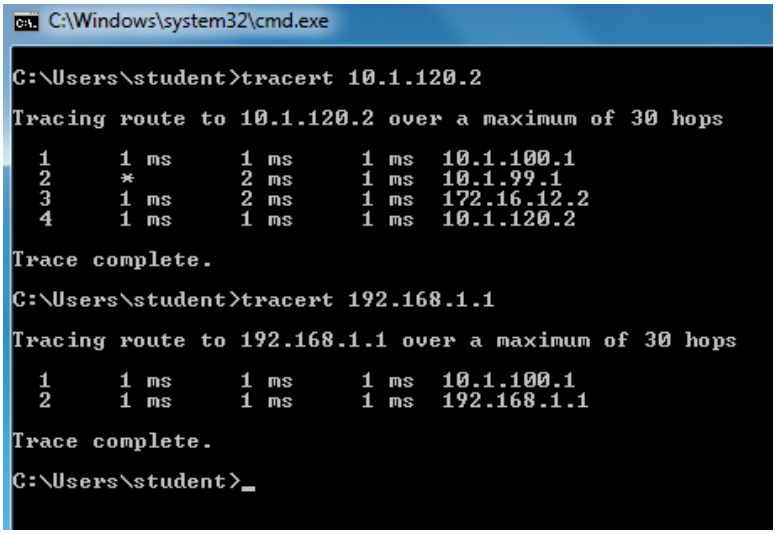
ALS1# **traceroute 10.1.120.2**
Type escape sequence to abort.
Tracing the route to 10.1.120.2
VRF info: (vrf in name/id, vrf out name/id)

```
  1 10.1.99.1 0 msec 0 msec 0 msec
  2 172.16.12.2 9 msec 0 msec 0 msec
  3 10.1.120.2 0 msec 8 msec *
ALS1#
```

From ALS1, a traceroute to 192.168.1.1 should take one hop:

```
ALS1# traceroute 192.168.1.1
Type escape sequence to abort.
Tracing the route to 192.168.1.1
VRF info: (vrf in name/id, vrf out name/id)
  1 10.1.110.1 0 msec 0 msec *
ALS1#
```

Traces from Host A show an additional hop, but follow the appointed path:



MICROSERVICES ARCHITECTURE

NURTURING POTENTIAL

SAKET GYANPEETH'S

SAKET COLLEGE OF ARTS, SCIENCE, AND COMMERCE

(Permanently Affiliated to University of Mumbai)

NAAC Accredited B Grade

Saket Vidyanagari Marg, Chinchpada Road, Katemanivali, Kalyan (East) – 421306, Dist. Thane (MAH)

Department of Information Technology

CERTIFICATE

This is to certify that
NA of **MSc Information Technology Part-I**
Class has satisfactorily carried out the required practical in the subject.

MICROSERVICES ARCHITECTURE
For the Academic year 2022-2023

Practical in Charge **Head of Department** **External Examiner**

INDEX

PRACTICAL 1

Aim: Asp.Net Core MVC

1) Install .Net Core Sdk (Link: https://dotnet.microsoft.com/learn/dotnet/hello-world-tutorial/install)

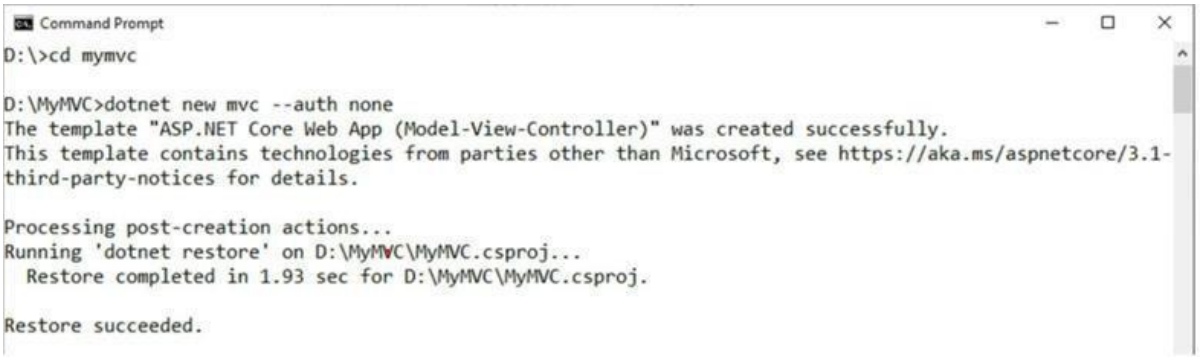
2) Create folder MyMVC folder in D: drive or any other drive

3) Open command prompt and perform following operations

Command: to create mvc project

dotnet new mvc --auth none

OUTPUT:



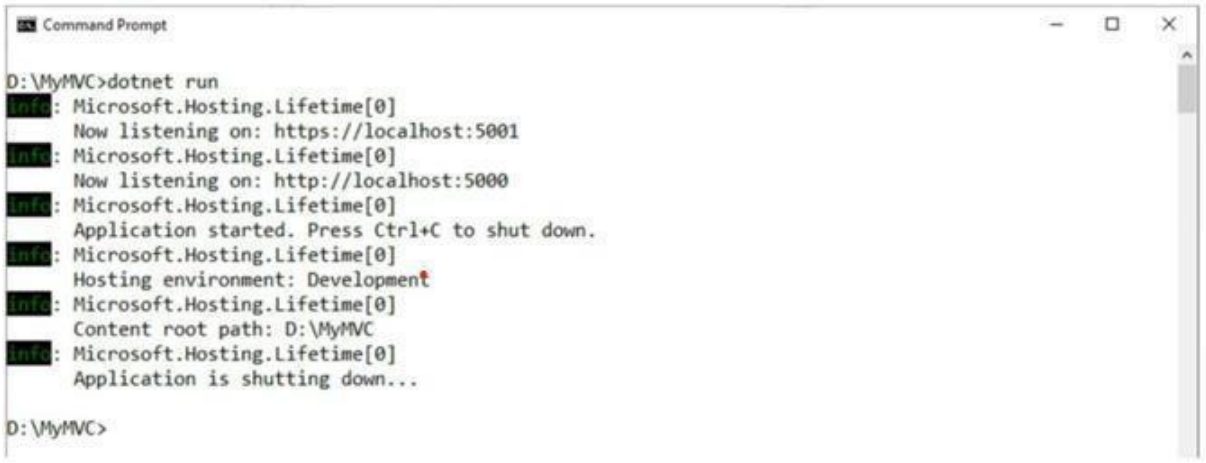
4) Go to controllers folder and modify HomeController.cs file to match following:

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using  System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Logging;
using MyMVC.Models;
namespace MyMVC.Controllers
{
 public class HomeController : Controller
{
 public String Index()
{
return "Hello World";     }
}
}
```

5) Run the project

Now open browser and and type URL: localhost:5000

6) Now go back to command prompt and stop running project using CTRL+C



7) Go to models folder and add new file StockQuote.cs to it with following content

```
using System;
namespace MyMVC.Models
{
public class StockQuote
{
public string Symbol {get;set;}
public int Price{get;set;}
}
}
```

8) Now Add View to folder then home folder in it and modify index.cshtml file to match following

```
@{
ViewData["Title"] = "Home Page";
}
<div>
Symbol: @Model.Symbol <br/>
Price: $@Model.Price <br/>
</div>
```

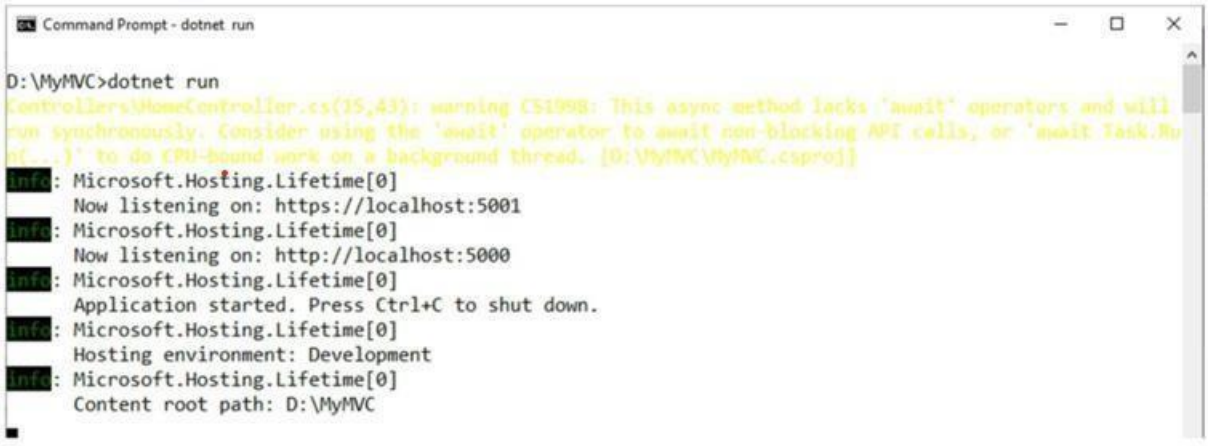
9) Now modify HomeController.cs file to match following:

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using  System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Logging;
using MyMVC.Models;
namespace MyMVC.Controllers
{
public class HomeController : Controller
{
public async Task <IActionResult> Index()
{
var model= new StockQuote{ Symbol='HLLO', Price=3200};
```

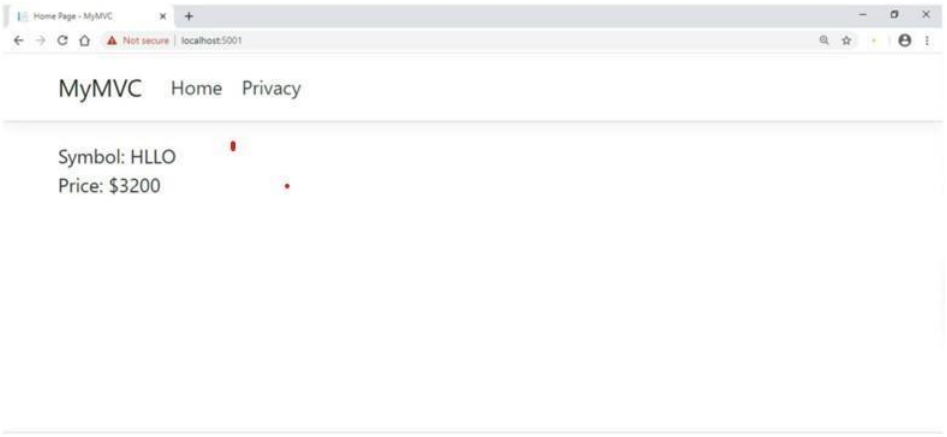
```
return View(model);
}
}
}
```

10) Now run the project using

dotnet run



11) Now go back to browser and refresh to get modified view response



PRACTICAL 2

Aim: Building ASP.Net core REST API

Software requirement:

1. Download and install

To start building .NET apps you just need to download and install the .NET SDK (Software Development Kit version3.0 above).

Link: https://dotnet.microsoft.com/learn/dotnet/hello-world-tutorial/install

2. Check everything installed correctly

Once you've installed, open a new command prompt and run the following command:

Command prompt

> dotnet

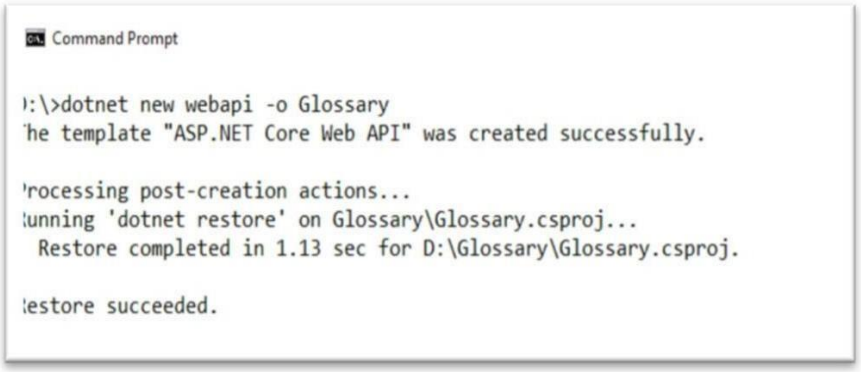
Create your web API

1. Open two command prompts

Command prompt 1:

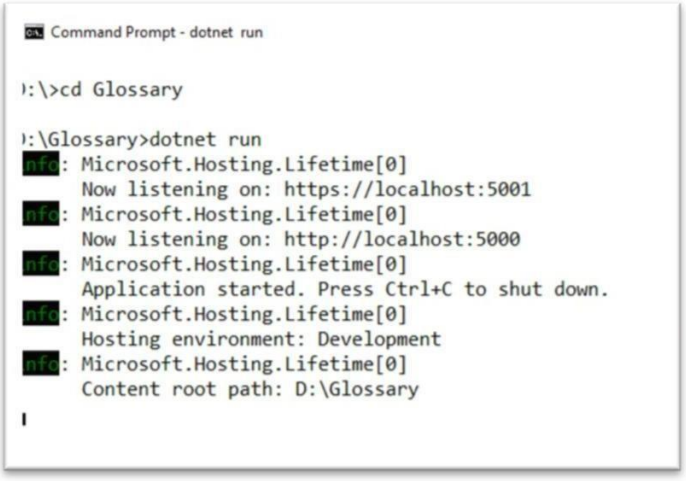
Command: dotnet new webapi -o Glossary

OUTPUT:



Command: cd Glossary & dotnet run

Output:



2. Command Prompt 2: (try running ready made weather forecast class for testing)

Command: curl --insecure https://localhost:5001/weatherforecast

OUTPUT:

3. Now Change the content:

To get started, remove the WeatherForecast.cs file from the root of the project and the WeatherForecastController.cs file from the Controllers folder.

Add Following two files

1) D:\Glossary\GlossaryItem.cs (type it in notepad and save as all files)

```
//GlossaryItem.cs
namespace Glossary
{
  public class GlossaryItem
  {
    public string Term { get; set; }
    public string Definition { get; set; }
  }
}
```

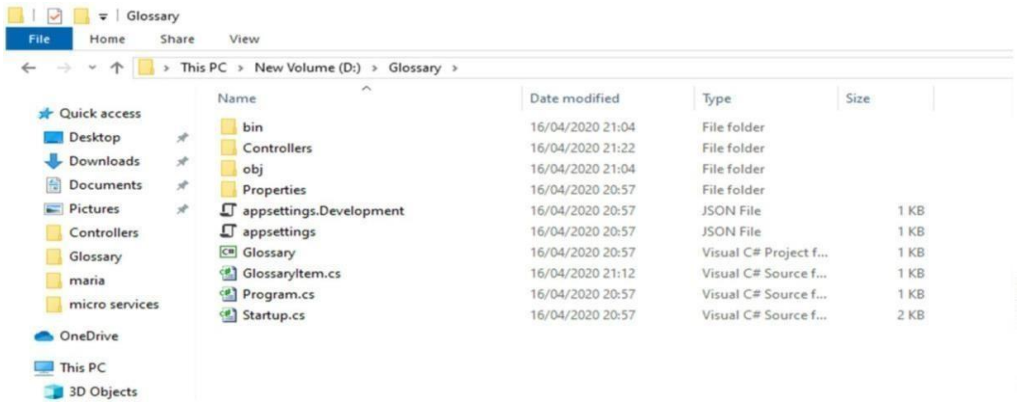
2) D:\Glossary\Controllers\ GlossaryController.cs (type it in notepad and save as all files)

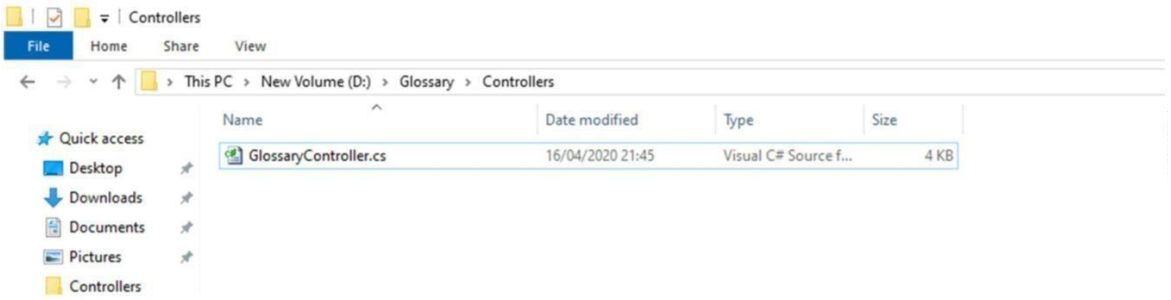
```
//Controllers/GlossaryController.cs
using System;
using  System.Collections.Generic;
using Microsoft.AspNetCore.Mvc;
using System.IO;
namespace Glossary.Controllers
{
[ApiController]
[Route("api/[controller]")]
public class GlossaryController: ControllerBase
{
new GlossaryItem
{
Term= "HTML",
Definition = "Hypertext Markup Language"
}
new GlossaryItem
{
```

```
Term= "MVC",
Definition = "Model View Controller"
}
new GlossaryItem
{
Term= "OpenID",
Definition = "An open standard for authentication"
}
};
[HttpGet]

public ActionResult<List<GlossaryItem>> Get()
{
return Ok(Glossary);
}
[HttpGet]
[Route("{term}")]

public ActionResult<GlossaryItem> Get(string term)
{
var glossaryItem = Glossary.Find(item =>
item.Term.Equals(term, StringComparison.InvariantCultureIgnoreCase));
if (glossaryItem == null)
{
return NotFound();
} else
{
return Ok(glossaryItem);
}
}
[HttpPost]
public ActionResult Post(GlossaryItem glossaryItem)
{
var existingGlossaryItem = Glossary.Find(item =>
item.Term.Equals(glossaryItem.Term, StringComparison.InvariantCultureIgnoreCase));
if (existingGlossaryItem != null)
{
return Conflict("Cannot create the term because it already exists.");
}
else
{
Glossary.Add(glossaryItem);
var resourceUrl = Path.Combine(Request.Path.ToString(),
Uri.EscapeUriString(glossaryItem.Term));
return Created(resourceUrl, glossaryItem);
}
}
[HttpPut]
public ActionResult Put(GlossaryItem glossaryItem)
{
```

```
var existingGlossaryItem = Glossary.Find(item =>
item.Term.Equals(glossaryItem.Term, StringComparison.InvariantCultureIgnoreCase));

if (existingGlossaryItem == null)
{
return BadRequest("Cannot update a nont existing term.");
} else
existingGlossaryItem.Definition = glossaryItem.Definition;
return Ok();
}
}
[HttpDelete]
[Route("{term}")]
public ActionResult Delete(string term)
{
var glossaryItem = Glossary.Find(item =>
item.Term.Equals(term, StringComparison.InvariantCultureIgnoreCase));
if (glossaryItem == null)
{
return NotFound();
}
else
{
Glossary.Remove(glossaryItem);
return NoContent();
}
}
}
}
```

OUTPUT:



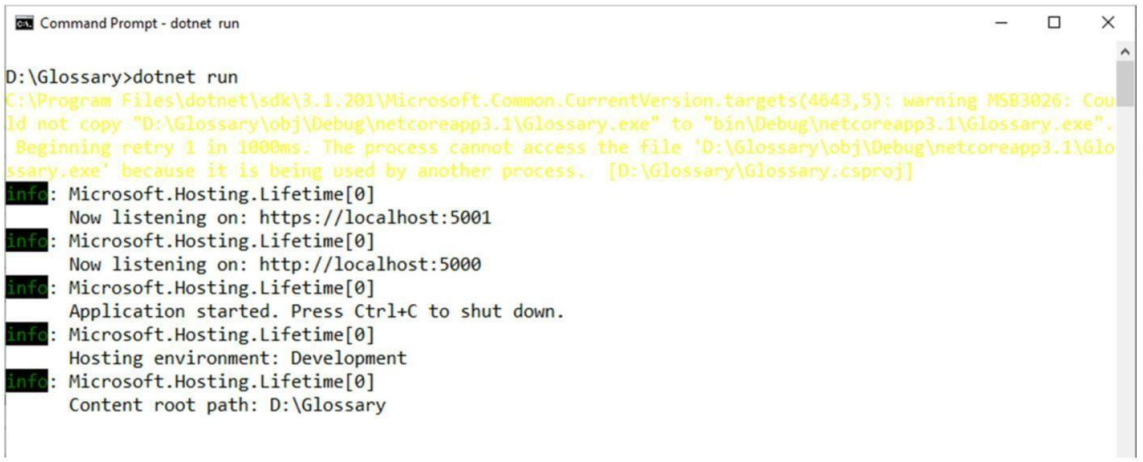


3. Now stop running previous dotnet run on command prompt 1 using Ctrl+C. and Run it again for new code.

On Command prompt1:

Command: **dotnet run**

OUTPUT:



On Command prompt2:

1) Getting a list of items:

Command: curl --insecure https://localhost:5001/api/glossary

OUTPUT:

2) Getting a single item

Command: curl --insecure https://localhost:5001/api/glossary/MVC

OUTPUT:

3) Creating an item

Command: curl --insecure -X POST -d "{\"term\": \"MFA\", \"definition\":\"An authentication process.\"}" -H "Content-Type: application/json" https://localhost:5001/api/glossary

4)Update Item

Command: curl --insecure -X PUT -d "{\"term\": \"MVC\", \"definition\":\"Modified record of Model ViewController.\"}" -H "Content-Type:application/json" https://localhost:5001/api/glossary

OUTPUT:

5) Delete Item

Command:

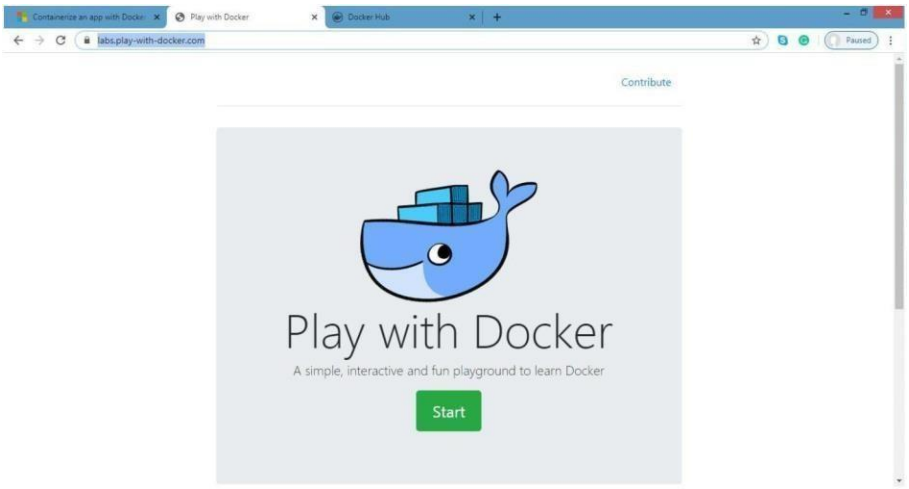
curl --insecure --request DELETE --url https://localhost:5001/api/glossary/openid

OUTPUT:

Practical 3

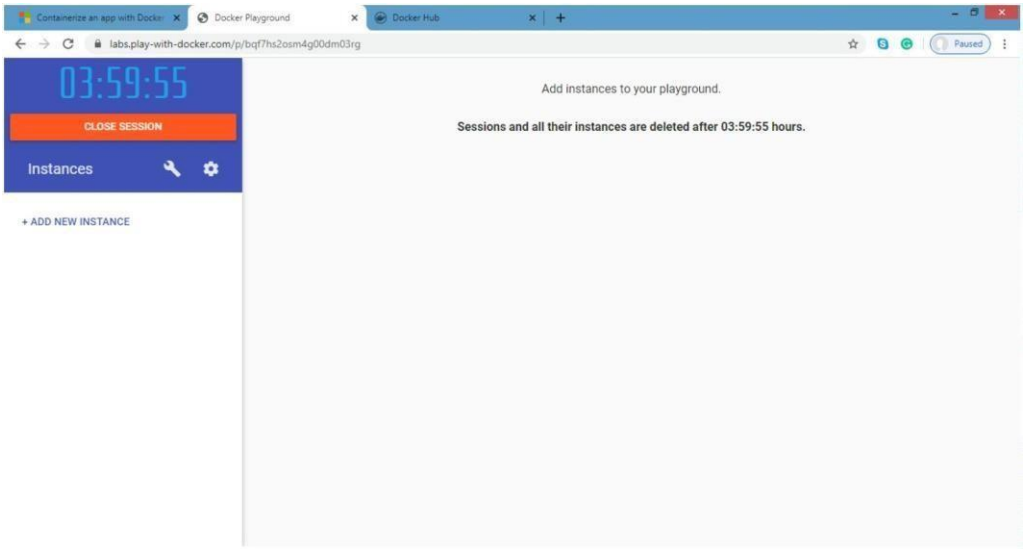
Working with Docker

1) Create Docker Hub account (sign up)
2) Login to https://labs.play-with-docker.com/



Click on start

3) Add new instance



4) Perform following:

Method1: To pull and push images using docker

Command: to check docker version

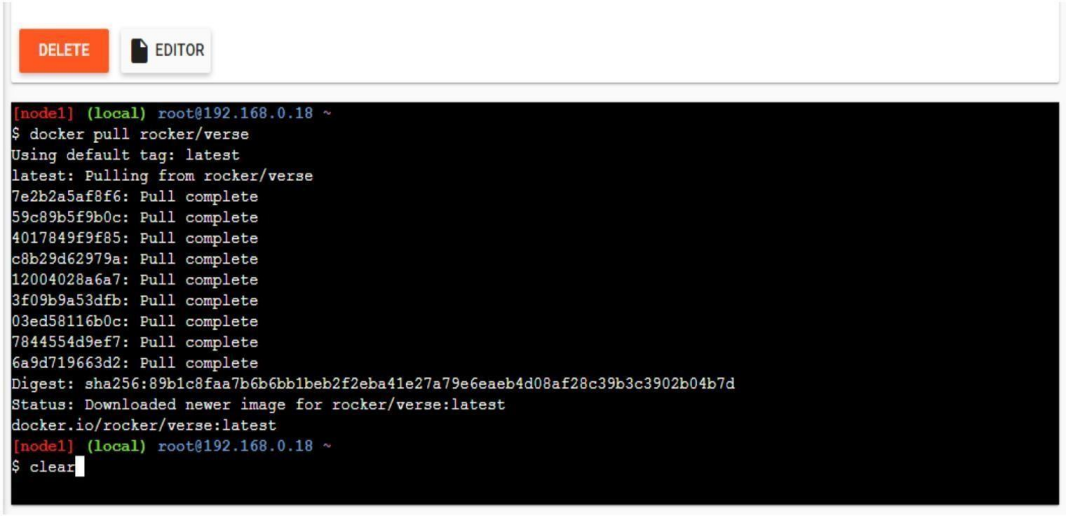
docker –version

OUTPUT:

Command: to pull readymade image

docker pull rocker/verse

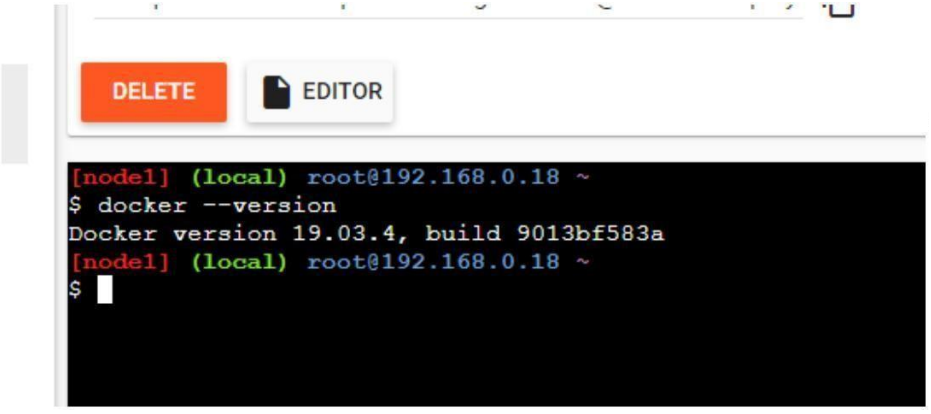
OUTPUT:



Command: to check images in docker

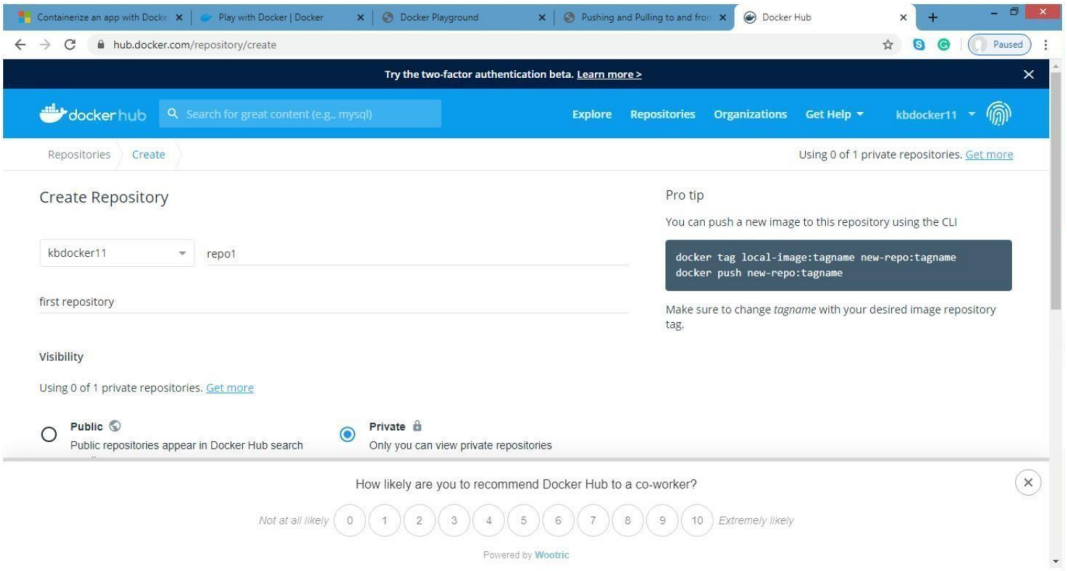
docker images

Output:

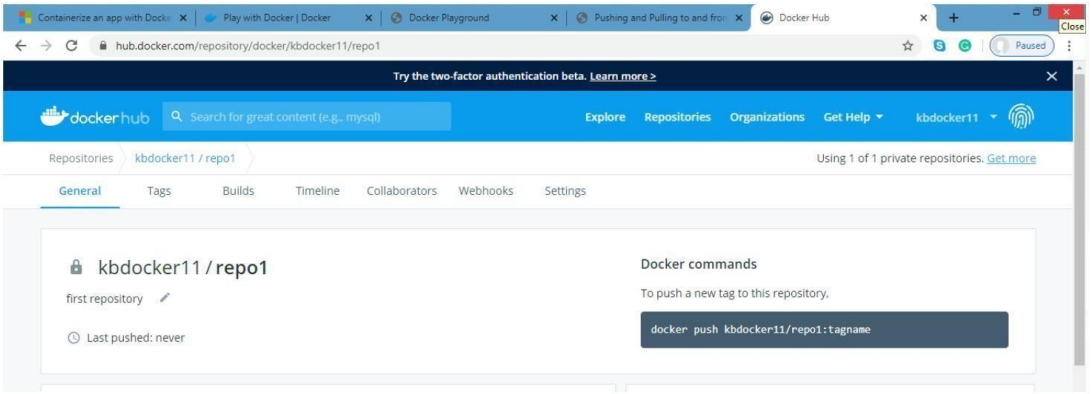


Now Login to docker hub and create repository

Output:



Click on Create button
Now check repository created



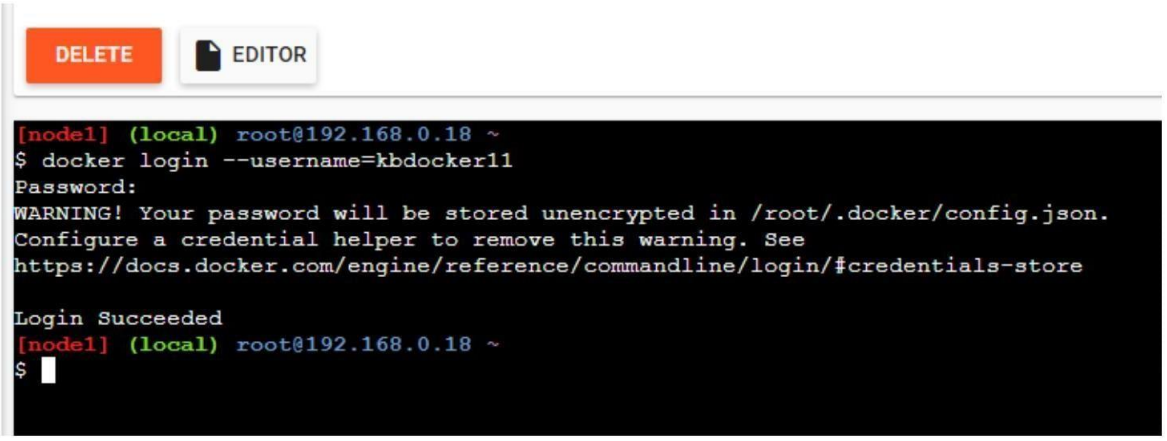
Command: to login to your docker account

docker login –username=kbdocker11

password

Note: kbdocker11 is my docker ID . You will use your docker ID here. And enter your Password.

OUTPUT:

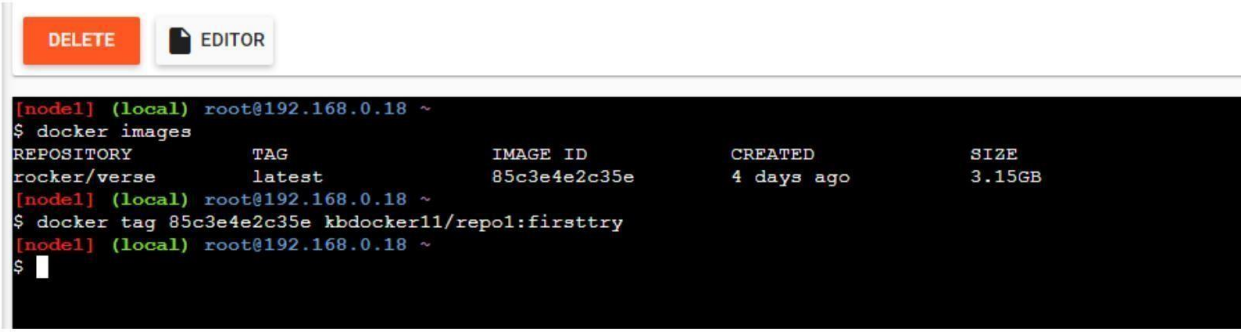


Command to tag image

docker tag 8c3e4e2c3e kbdocker11/repo1:firsttry

Note: here8c3e4e2c3e this is image id which you can get from docker images command.

OUTPUT:



Command: to push image to docker hub account

docker push kbdocker11/repo1:firsttry

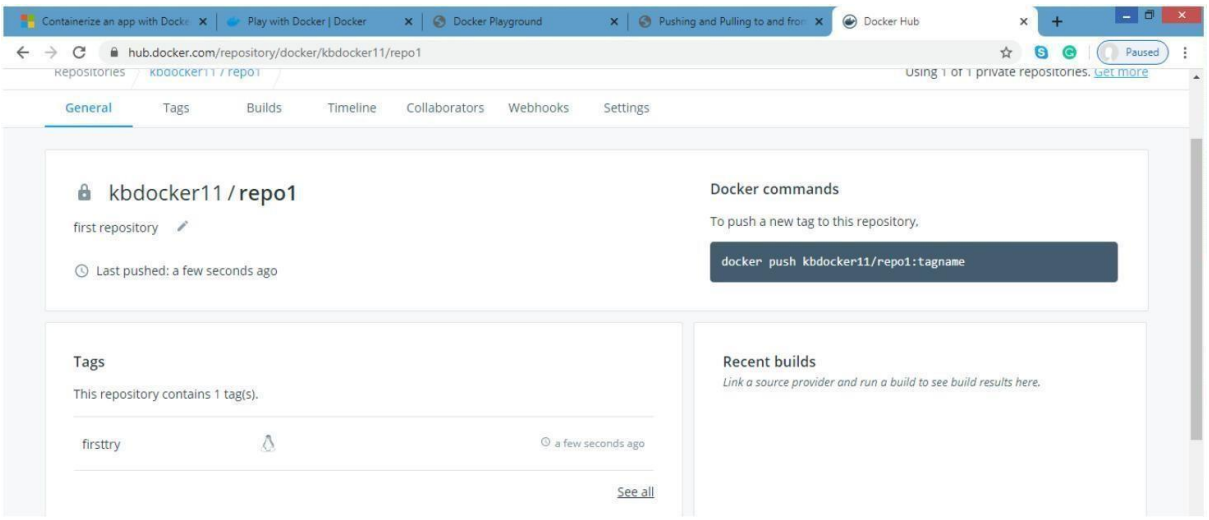
Note: firsttry is tag name created above.

Output:

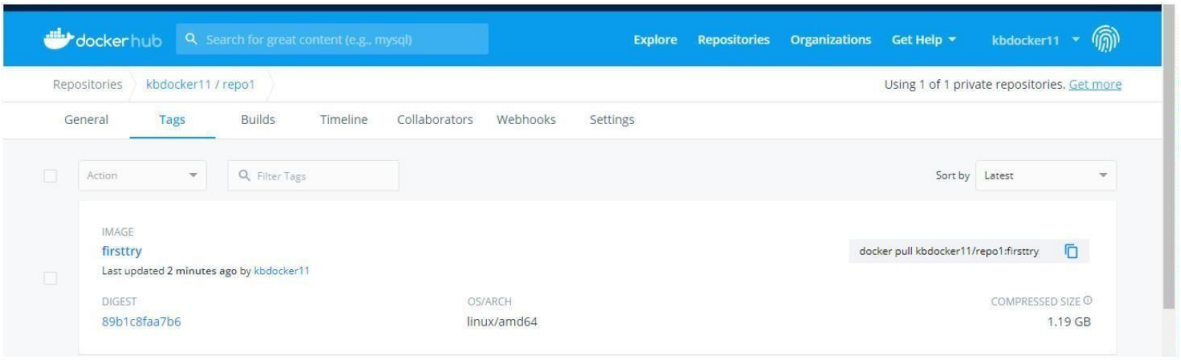
DELETE EDITOR

```
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
rocker/verse        latest              85c3e4e2c35e        4 days ago          3.15GB
[node1] (local) root@192.168.0.18 ~
$ docker tag 85c3e4e2c35e kbdocker11/repo1:firsttry
[node1] (local) root@192.168.0.18 ~
$ docker push kbdocker11/repo1:firsttry
The push refers to repository [docker.io/kbdocker11/repo1]
3e43a21d810a: Mounted from rocker/verse
8fdb254334fd: Mounted from rocker/verse
6611ef73af7c: Mounted from rocker/verse
7ec16b3cc818: Mounted from rocker/verse
a2f3120be52c: Mounted from rocker/verse
beb6bc4429d0: Mounted from rocker/verse
828281284548: Mounted from rocker/verse
61fb5e16e303: Mounted from rocker/verse
461719022993: Mounted from rocker/verse
firsttry: digest: sha256:89b1c8faa7b6b6bb1beb2f2eba41e27a79e6eaeb4d08af28c39b3c3902b04b7d size: 2211
[node1] (local) root@192.168.0.18 ~
$
```

Check it in docker hub now



Click on tags and check

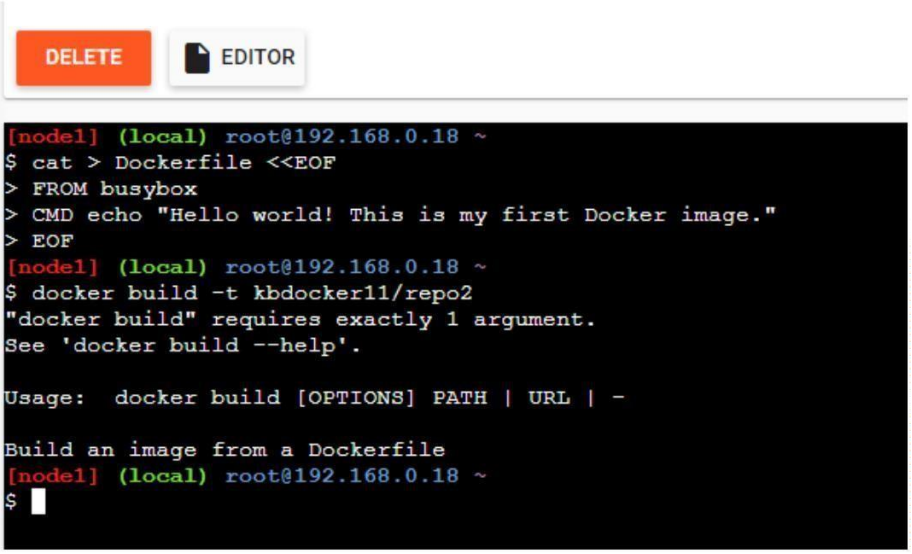


Method 2:

Build an image then push it to docker and run it
Command: to create docker file

1. **cat > Dockerfile <<EOF**
2. **FROM busybox**
3. **CMD echo "Hello world! This is my first Docker image."**
4. **EOF**

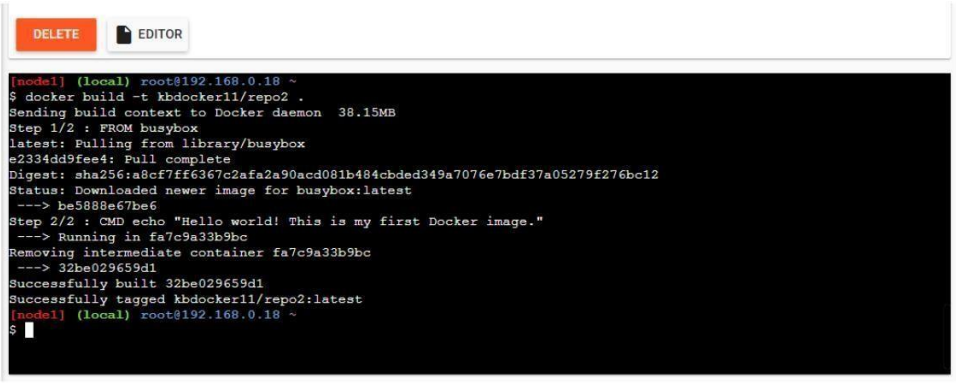
OUTPUT:



Command: to build image from docker file

docker build –t kbdocker11/repo2

OUTPUT:



Command: to check docker images

docker images

Command: to create docker file

Output:

```
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
kbdocker11/repo2    latest              32be029659d1        About a minute ago  1.22MB
kbdocker11/repo1    firsttry            85c3e4e2c35e        4 days ago          3.15GB
rocker/verse        latest              85c3e4e2c35e        4 days ago          3.15GB
busybox             latest              be5888e67be6        6 days ago          1.22MB
```

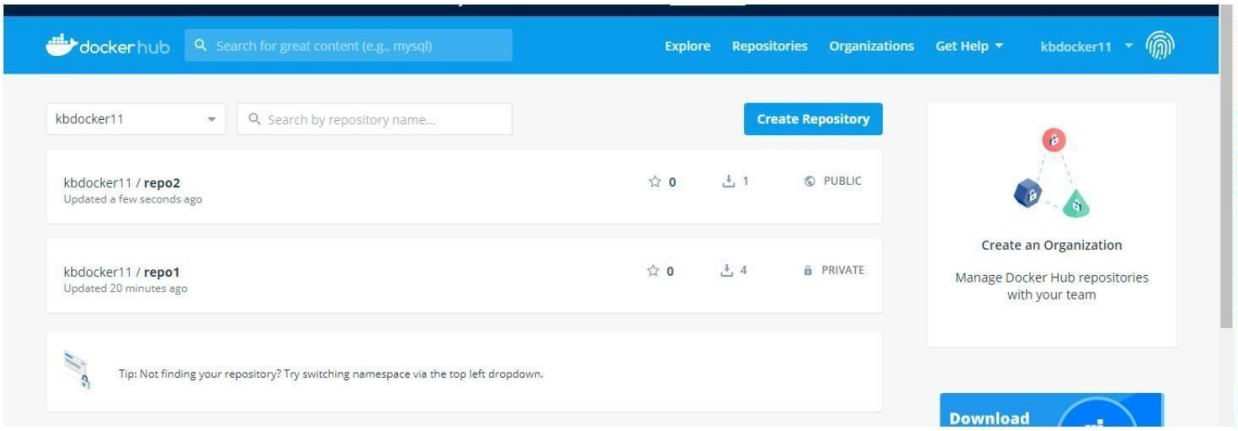
Command: to push image to docker hub

docker push kbdocker11/repo2

Output:

```
[node1] (local) root@192.168.0.18 ~
$ docker push kbdocker11/repo2
The push refers to repository [docker.io/kbdocker11/repo2]
5b0d2d635df8: Mounted from library/busybox
latest: digest: sha256:afa7a4103608d128764a15889501141a10eb9e733f19e4f57645a5ac01c85407 size: 527
[node1] (local) root@192.168.0.18 ~
$
```

Now check it on docker hub



Command: to run docker image:

docker run kbdocker11/repo2

Output:

```
[node1] (local) root@192.168.0.18 ~
$ docker run kbdocker11/repo2
Hello world! This is my first Docker image.
[node1] (local) root@192.168.0.18 ~
$
```

Now close session.

PRACTICAL 4

Installing software packages on dockers, working with docker working with docker volume & networks.

Steps for Installing Docker:

1. Open the terminal on Ubuntu.

2. Remove any Docker files that are running in the system, using the following command:

$ sudo apt-get remove docker docker-engine docker.io

```
ubuntu@ip-172-31-83-130:~$ sudo apt-get remove docker docker-engine docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
E: Unable to locate package docker-engine
ubuntu@ip-172-31-83-130:~$ sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [108 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Fetched 337 kB in 1s (461 kB/s)
Reading package lists... Done
```

3. Check if the system is up-to-date using the following command:

$ sudo apt-get update

```
ubuntu@ip-172-31-83-130:~$ sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [108 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Fetched 337 kB in 1s (461 kB/s)
Reading package lists... Done
```

4. Install Docker using the following command:

$ sudo apt install docker.io

You will then get a prompt asking you to choose between y/n - choose y

```
ubuntu@ip-172-31-83-130:~$ sudo apt install docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
docker.io is already the newest version (20.10.21-0ubuntu1~22.04.3).
0 upgraded, 0 newly installed, 0 to remove and 66 not upgraded.
```

5. Install all the dependency packages using the following command:

$ sudo snap install docker

```
ubuntu@ip-172-31-83-130:~$ sudo snap install docker
snap "docker" is already installed, see 'snap help refresh'
```

6. Before testing Docker, check the version installed using the following command:

$ docker –version

```
ubuntu@ip-172-31-83-130:~$ docker --version
Docker version 20.10.21, build 20.10.21-0ubuntu1~22.04.3
```

7. Pull an image from the Docker hub using the following command:

$ sudo docker run hello-world

```
ubuntu@ip-172-31-83-130:~$ sudo docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

8. Check if the docker image has been pulled and is present in your system using the following command:

$ sudo docker images

```
ubuntu@ip-172-31-83-130:~$ sudo docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
hello-world   latest    9c7a54a9a43c   6 weeks ago    13.3kB
```

9. To display all the containers pulled, use the following command:

$ sudo docker ps –a

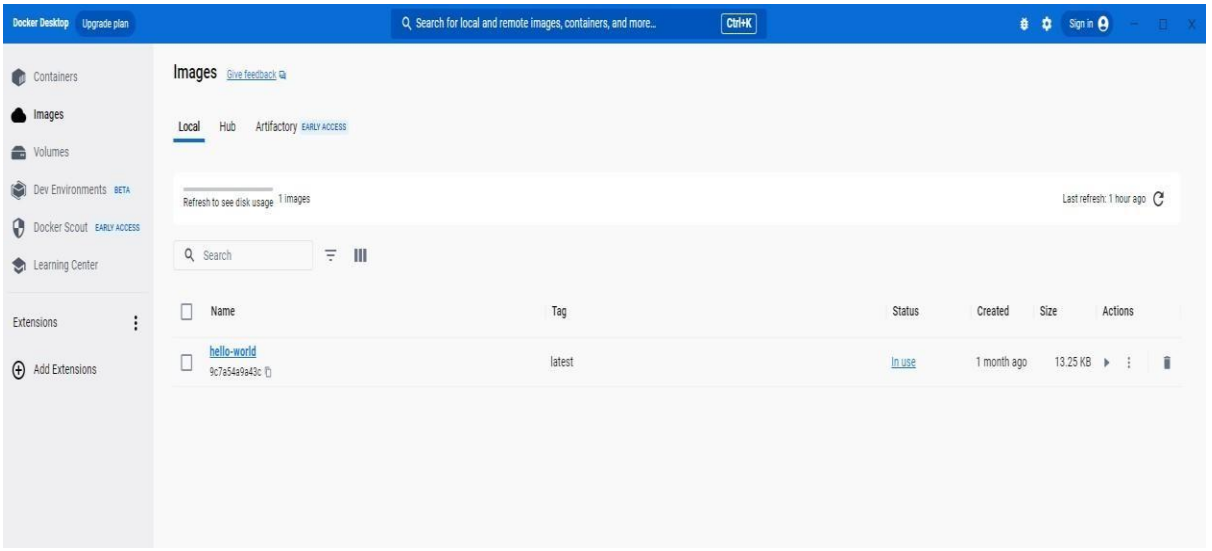
```
ubuntu@ip-172-31-83-130:~$ sudo docker ps -a
CONTAINER ID   IMAGE         COMMAND    CREATED          STATUS                      PORTS     NAMES
b30807cbd51f   hello-world   "/hello"   25 seconds ago   Exited (0) 23 seconds ago             pedantic_pare
c60279a2120d   hello-world   "/hello"   51 minutes ago   Exited (0) 51 minutes ago             gracious_mendel
```

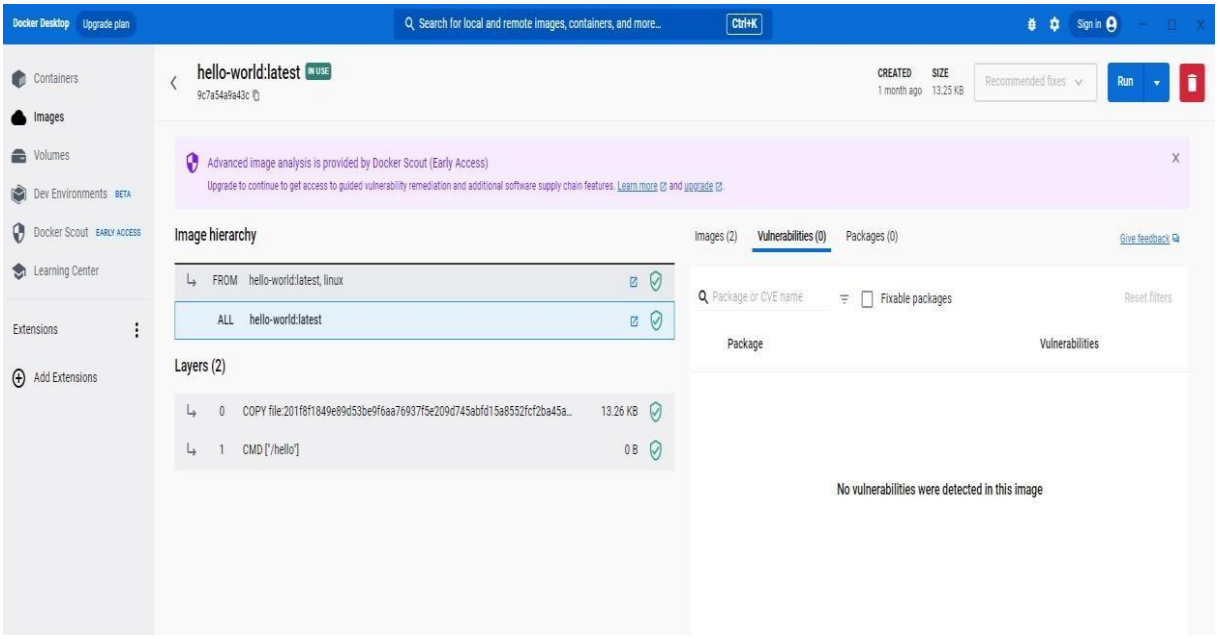
10. To check for containers in a running state, use the following command:

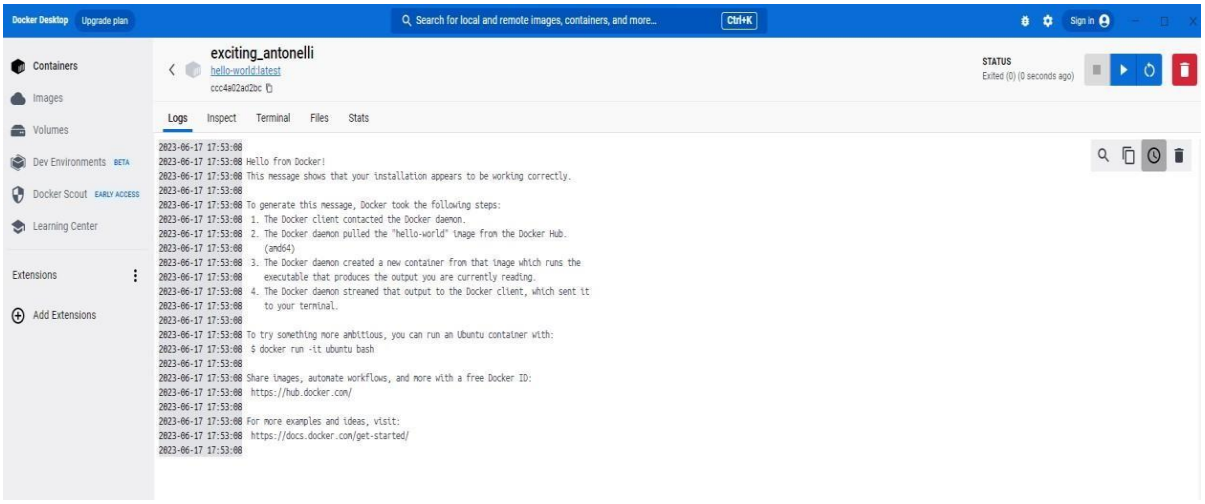
$ sudo docker ps

```
ubuntu@ip-172-31-83-130:~$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

You have just successfully installed Docker on Ubuntu!





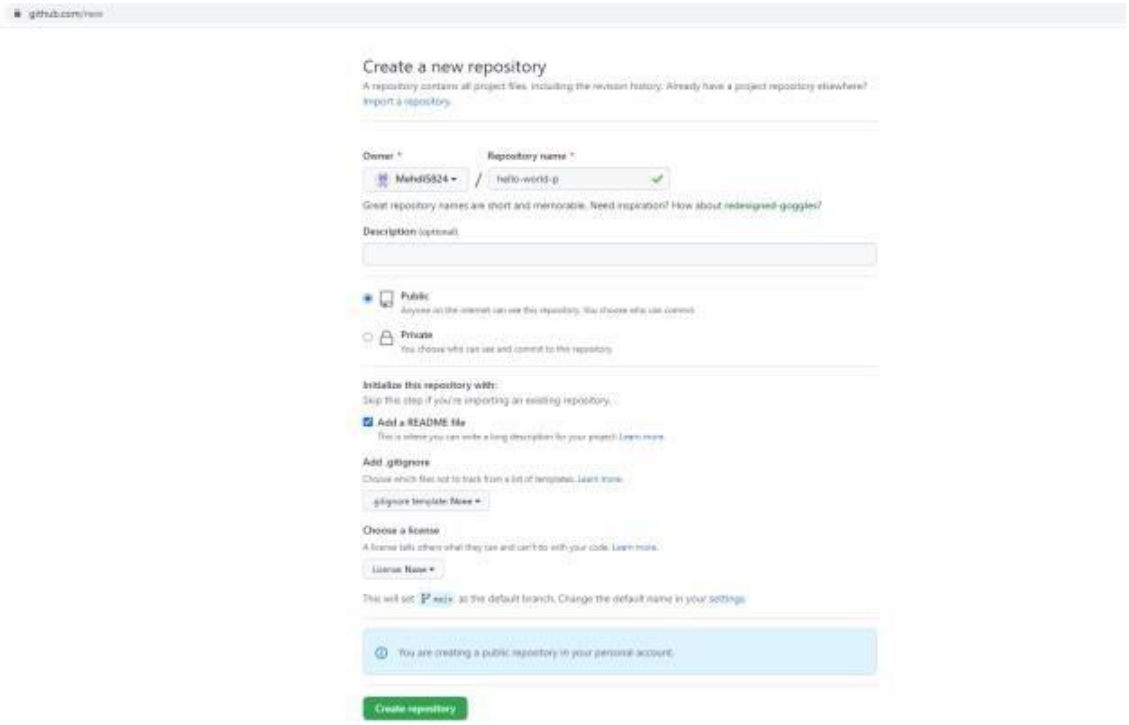


PRACTICAL 5

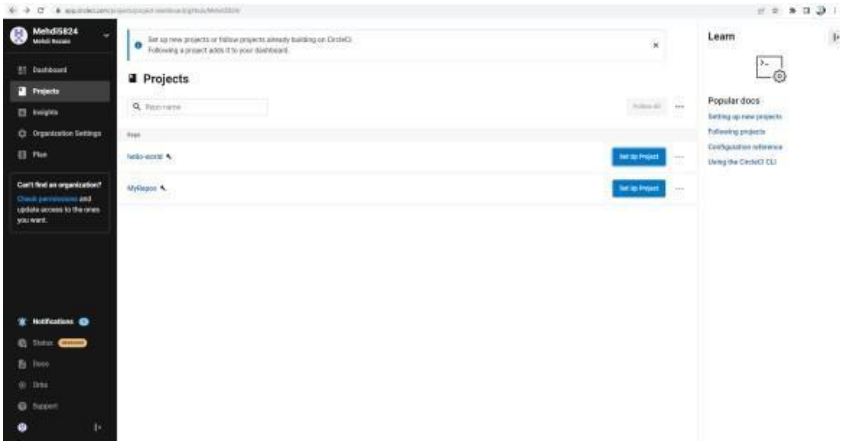
(Working with Circle CI for continuous integration)

Step 1 - Create a repository

1. Log in to GitHub and begin the process to create a new repository.
2. Enter a name for your repository (for example, hello-world).
3. Select the option to initialize the repository with a README file.
4. Finally, click Create repository.
5. There is no need to add any source code for now.

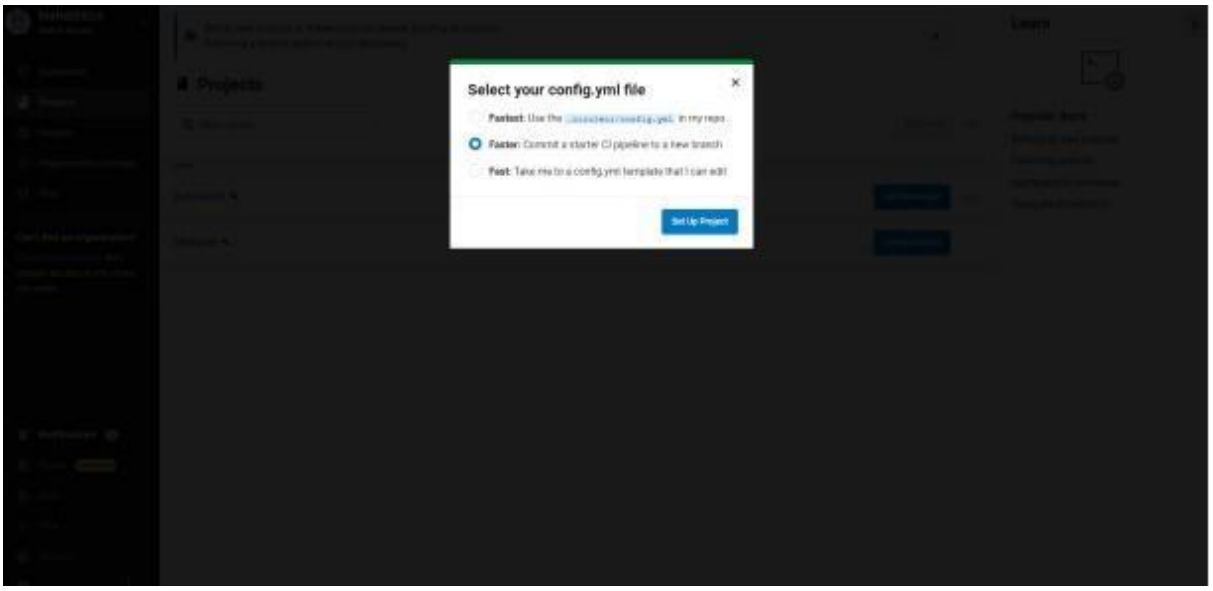


Login to Circle CI https://app.circleci.com/ Using GitHub Login, Once logged in navigate to Projects.



Step 2 - Set up CircleCI

1. Navigate to the CircleCI Projects page. If you created your new repository under an organization,you will need to select the organization name.
2. You will be taken to the Projects dashboard. On the dashboard, select the project you want to set up (hello-world).
3. Select the option to commit a starter CI pipeline to a new branch, and click Set Up Project. This will create a file .circleci/config.yml at the root of your repository on a new branch called circleci-project-setup.



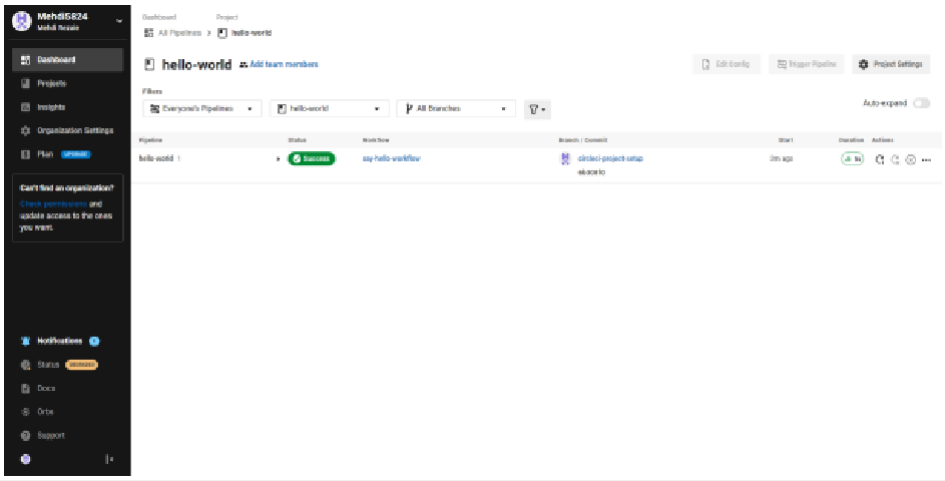
Step 3 - Your first pipeline

On your project's pipeline page, click the green Success button, which brings you to the workflow that ran (say-helloworkflow).

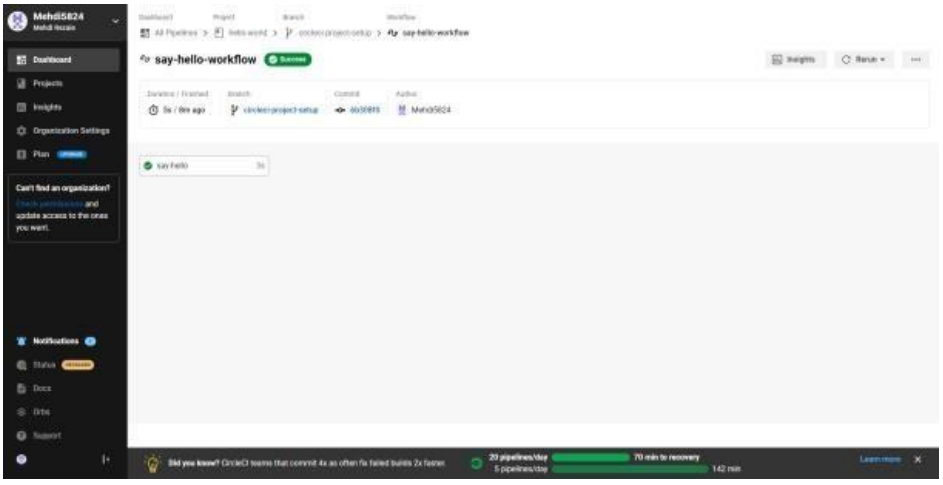
Within this workflow, the pipeline ran one job, called say-hello. Click say-hello to see the steps in this job:

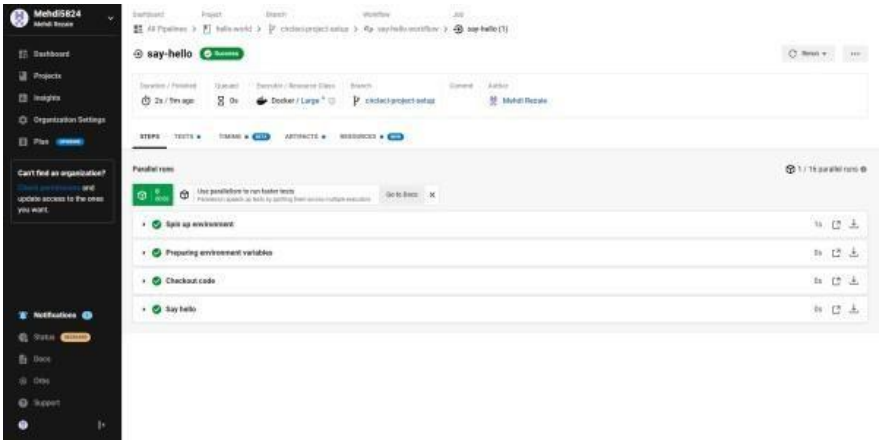
a. Spin up environment
b. Preparing environment variables
c. Checkout code
d. Say hello

Now select the "say-hello-workflow" to the right of Success status column

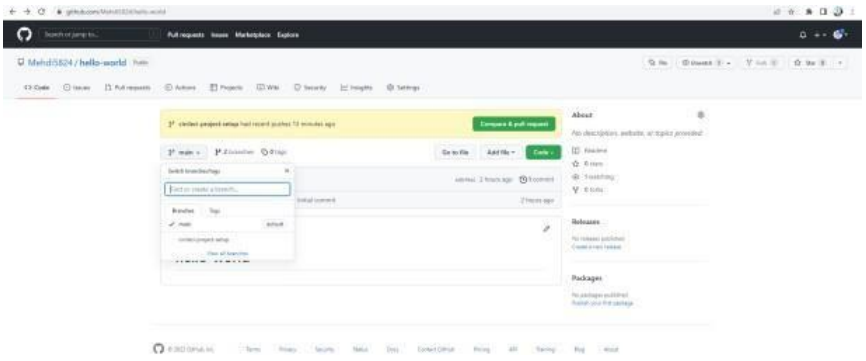


Select “say-hello” Job with a green tick





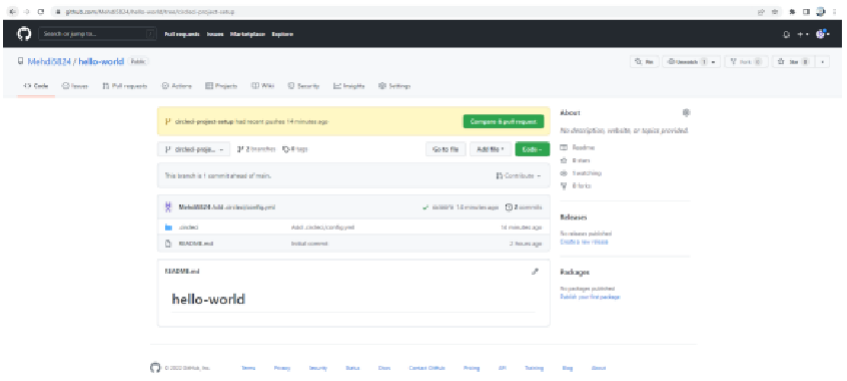
Select Branch and option circleci-project-setup

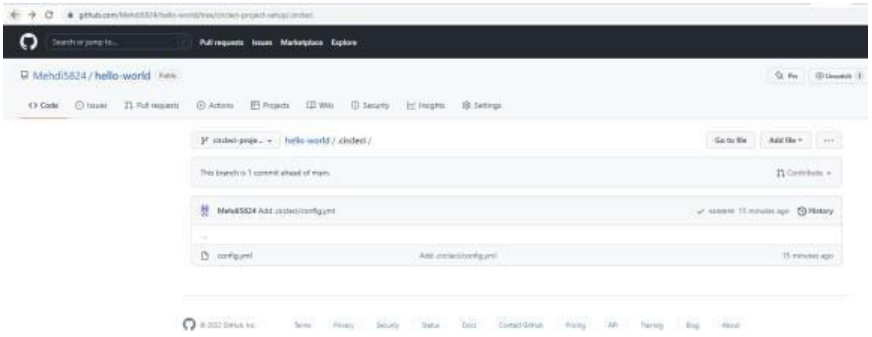


Step 4 - Break your build

In this section, you will edit the .circleci/config.yml file and see what happens if a build does not complete successfully.

It is possible to edit files directly on GitHub

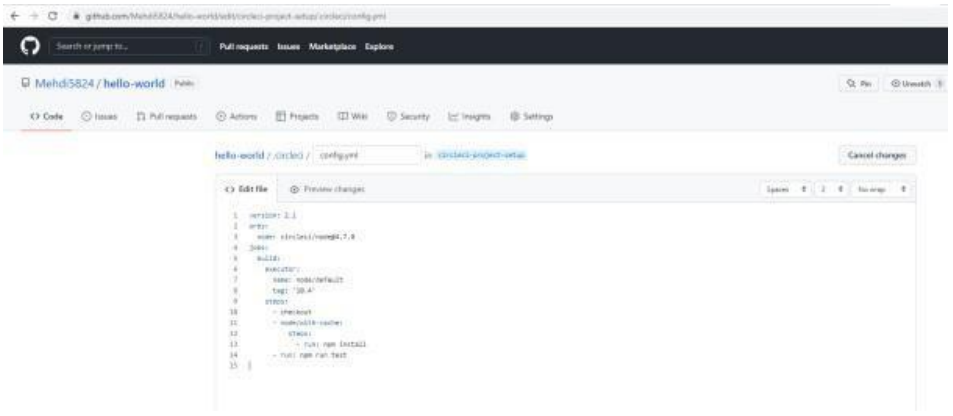




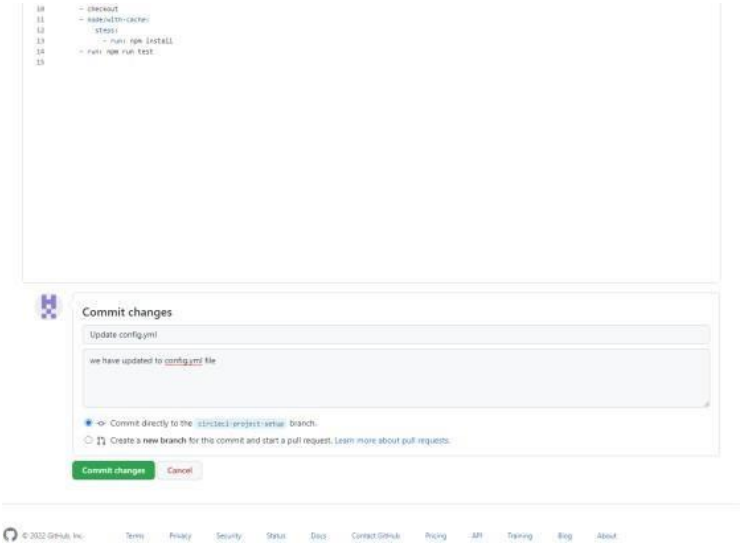
Let's use the Node orb. Replace the existing config by pasting the following code:

```
version: 2.1
orbs:
  node: circleci/node@4.7.0
jobs:
  build:
    executor:
      name: node/default
      tag: '10.4'
    steps:
      - checkout
      - node/with-cache:
          steps:
            - run: npm install
      - run: npm run test
```

The GitHub file editor should look like this

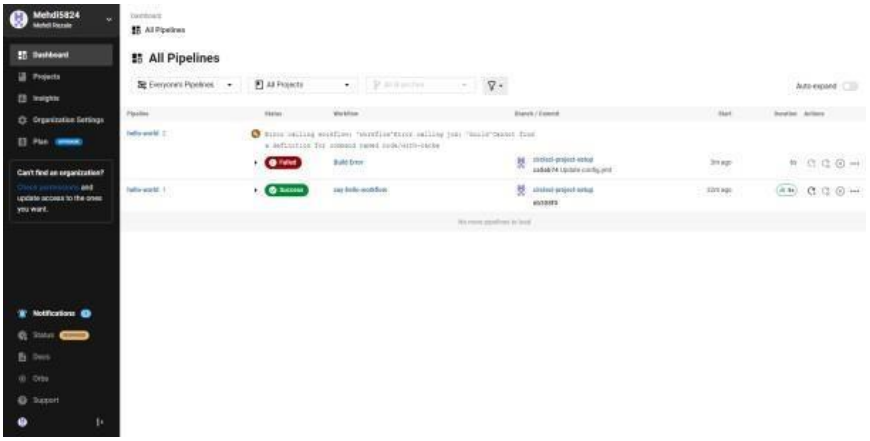


Scroll down and Commit your changes on GitHub



After committing your changes, then return to the Projects page in CircleCI. You should see a new pipeline running… and it will fail! What's going on? The Node orb runs some common Node tasks. Because you are working with an empty repository,

running npm run test, a Node script, causes the configuration to fail. To fix this, you need to set up a Node project in your repository.



Step 5 – Use Workflows

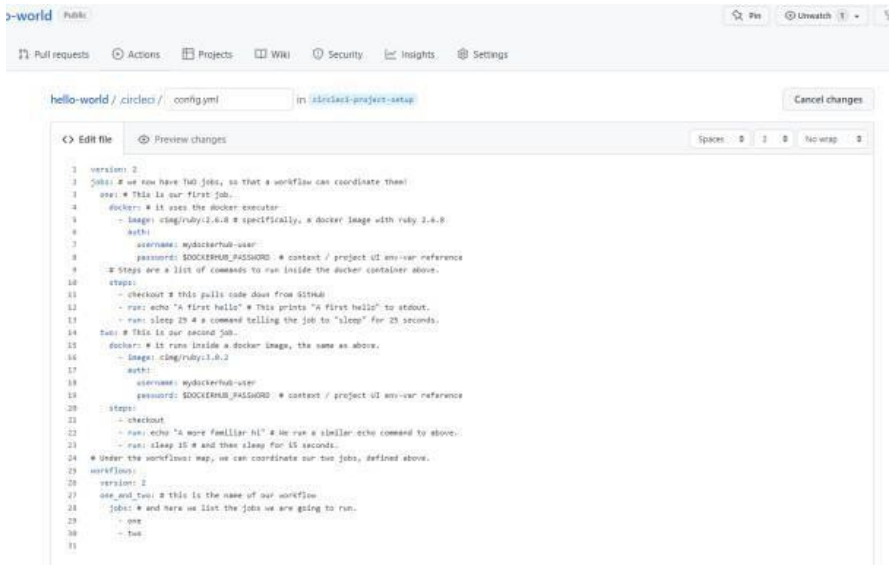
You do not have to use orbs to use CircleCI. The following example details how to create a custom configuration that also uses the workflow feature of CircleCI.

1) Take a moment and read the comments in the code block below. Then, to see workflows in action, edit your .circleci/config.yml file and copy and paste the following text into it.

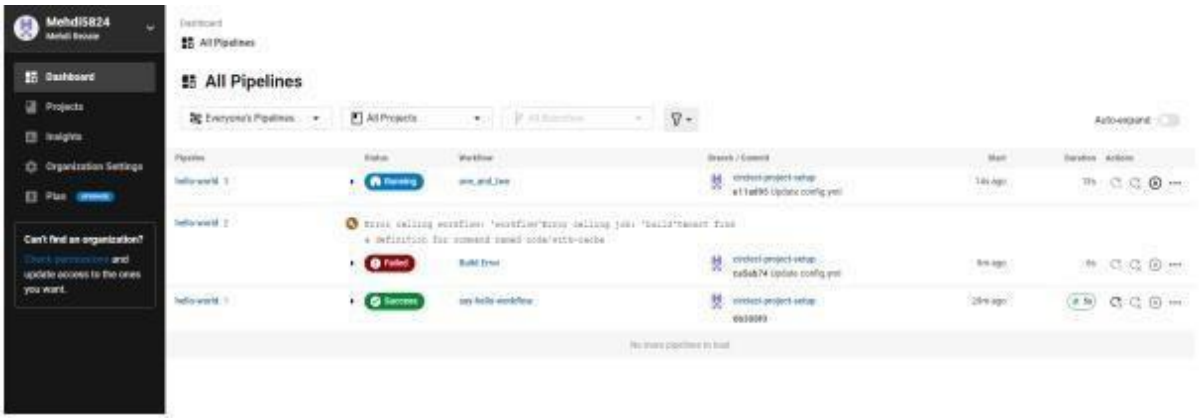
```
version: 2
jobs: # we now have TWO jobs, so that a workflow can coordinate them!
  one: # This is our first job.
    docker: # it uses the docker executor
      - image: cimg/ruby:2.6.8 # specifically, a docker image with ruby 2.6.8
        auth:
          username: mydockerhub-user
          password: $DOCKERHUB_PASSWORD  # context / project UI env-var reference
    # Steps are a list of commands to run inside the docker container above.
    steps:
      - checkout # this pulls code down from GitHub
      - run: echo "A first hello" # This prints "A first hello" to stdout.
      - run: sleep 25 # a command telling the job to "sleep" for 25 seconds.
  two: # This is our second job.
    docker: # it runs inside a docker image, the same as above.
      - image: cimg/ruby:3.0.2
        auth:
          username: mydockerhub-user
          password: $DOCKERHUB_PASSWORD  # context / project UI env-var reference
    steps:
      - checkout
      - run: echo "A more familiar hi" # We run a similar echo command to above.
      - run: sleep 15 # and then sleep for 15 seconds.
# Under the workflows: map, we can coordinate our two jobs, defined above.
workflows:
  version: 2
  one_and_two: # this is the name of our workflow
    jobs: # and here we list the jobs we are going to run.
      - one
      - two
```

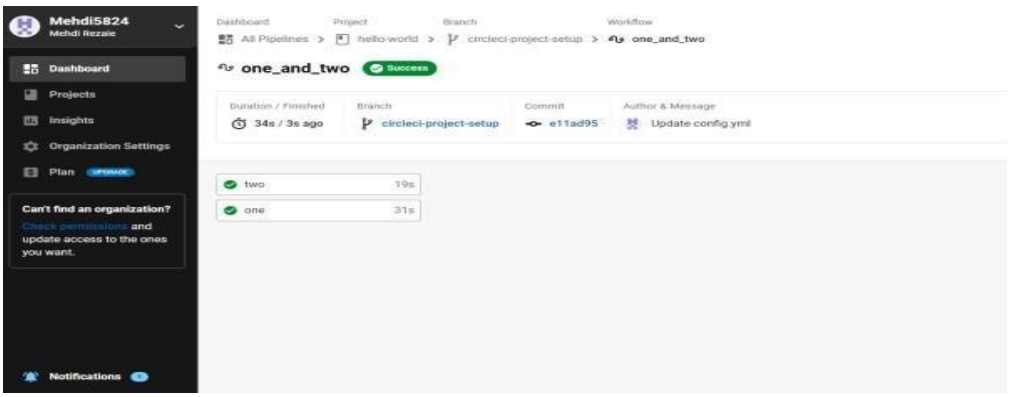
You don’t need to write the comments which are the text after #

2) Commit these changes to your repository and navigate back to the CircleCI Pipelines page. You should see your pipeline running.



3) Click on the running pipeline to view the workflow you have created. You should see that two jobs ran (or are currently running!) concurrently.



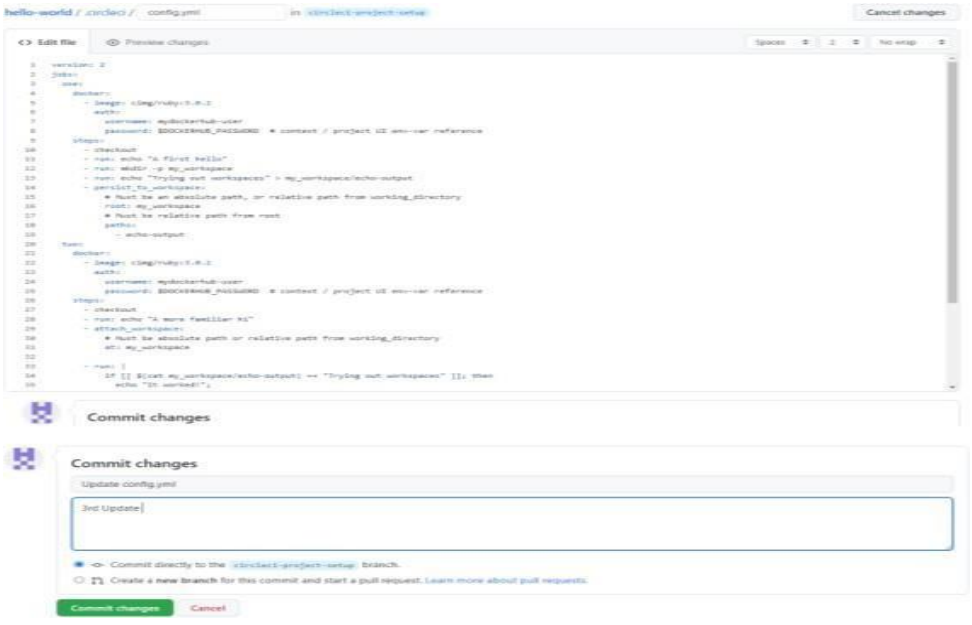


Step 5 – Add some changes to use workspaces

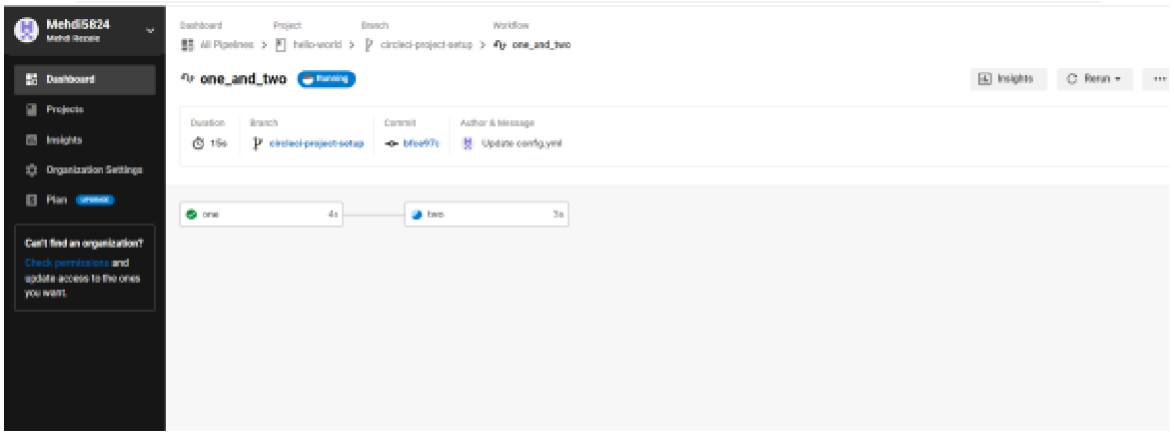
Each workflow has an associated workspace which can be used to transfer files to downstream jobs as the workflow progresses. You can use workspaces to pass along data that is unique to this run and which is needed for downstream jobs. Try updating config.yml to the following:

```yaml
version: 2
jobs:
  one:
    docker:
      - image: cimg/ruby:3.0.2
        auth:
          username: mydockerhub-user
          password: $DOCKERHUB_PASSWORD  # context / project UI env-var reference
    steps:
      - checkout
      - run: echo "A first hello"
      - run: mkdir -p my_workspace
      - run: echo "Trying out workspaces" > my_workspace/echo-output
      - persist_to_workspace:
          # Must be an absolute path, or relative path from working_directory
          root: my_workspace
          # Must be relative path from root
          paths:
            - echo-output
  two:
    docker:
      - image: cimg/ruby:3.0.2
        auth:
          username: mydockerhub-user
          password: $DOCKERHUB_PASSWORD  # context / project UI env-var reference
    steps:
      - checkout
      - run: echo "A more familiar hi"
      - attach_workspace:
          # Must be absolute path or relative path from working_directory
          at: my_workspace
```

Updated config.yml in GitHub file editor should be updated like this



Finally your workflow with the jobs running should look like this



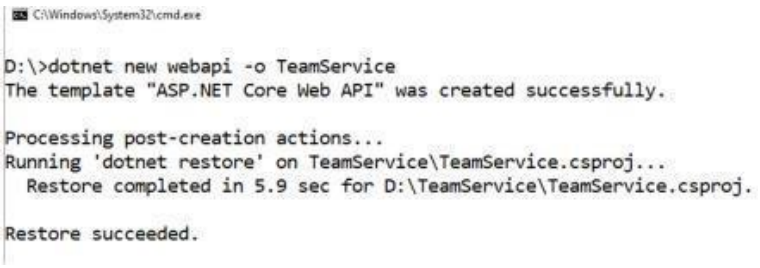
PRACTICAL 6

(Install .Net core sdk first)

Link: https://dotnet.microsoft.com/learn/dotnet/hello-world-tutorial/install

1) **Create new project:**

Command : dotnet new webapi -o TeamService



2) Remove existing weatherforecast files both model and controller files.
3) Add new files as follows:
4) Add Member.cs to "D:\TeamService\Models" folder

```
using System;
namespace TeamService.Models
{ public class Member {
public Guid ID { get; set; }
public string FirstName { get; set;}
public string LastName { get; set; }
public Member() { }
public Member(Guid id) : this()
{
this.ID = id;
}
public Member(string firstName, string lastName, Guid id) : this(id)
{
this.FirstName = firstName;
 this.LastName = lastName;
} public override string ToString() { return this.LastName;
}
}
}
```

5) Add Team.cs to "D:\TeamService\Models" folder

```
using System;
using System.Collections.Generic;
namespace TeamService.Models
{
 public class Team
```

```
{
public string Name { get; set; }
public Guid ID { get; set; }
public ICollection Members { get; set; }
public Team()
{
this.Members = new List();
}
public Team(string name) : this()
{
this.Name = name;
}
public Team(string name, Guid id) : this(name)
{
this.ID = id;
}

public override string ToString() {

return this.Name;

}

 }

 }
```

3)add TeamsController.cs file to "D:\TeamService\Controllers" folder

```
using System; using
System.Collections.Generic;
namespace TeamService.Models { public class Team
{
public string Name { get; set; }
public Guid ID { get; set; }
public ICollection Members { get; set; }
public Team()
{
this.Members = new List();
}
public Team(string name) : this()
{
this.Name = name;
}
public Team(string name, Guid id) : this(name)
{
this.ID = id;
}
public override string ToString() {
```

```
return this.Name;
}
}
}
```

3) add TeamsController.cs file to "D:\TeamService\Controllers" folder

```
using System;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Mvc;
using System.Collections.Generic;
using System.Linq;
using TeamService.Models;
using System.Threading.Tasks;
using TeamService.Persistence;
namespace TeamService
{ [Route("[controller]")]
public class TeamsController : Controller
{ ITeamRepository repository;
public TeamsController(ITeamRepository repo)
{
repository = repo;
}
 [HttpGet]
public virtual IActionResult GetAllTeams()
{
 return this.Ok(repository.List());
}
[HttpGet("{id}")]
public IActionResult GetTeam(Guid id)
{ Team team = repository.Get(id);
if (team != null) // I HATE NULLS, MUST FIXERATE THIS.
{ return this.Ok(team);
}
else {
return this.NotFound();
}
}
[HttpPost]
public virtual IActionResult CreateTeam([FromBody]Team newTeam)
{
repository.Add(newTeam);
return this.Created($"/teams/{newTeam.ID}", newTeam);
}
[HttpPut("{id}")]
public virtual IActionResult UpdateTeam([FromBody]Team team, Guid id)
```

```
 { team.ID = id;
if(repository.Update(team) == null)
{
return this.NotFound();
}
Else
{ return this.Ok(team);
}
}
[HttpDelete("{id}")]
public virtual IActionResult DeleteTeam(Guid id)
{ Team team = repository.Delete(id);
if (team == null)
{
return this.NotFound();
}
else {
return this.Ok(team.ID);
}
}
}
}
```

4) add MembersController.cs file to "D:\TeamService\Controllers" folder

```
using System;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Mvc;
using System.Collections.Generic;
using System.Linq;
using TeamService.Models;
using System.Threading.Tasks;
using TeamService.Persistence;
namespace TeamService
{ [Route("/teams/{teamId}/[controller]")]
public class MembersController : Controller
{ ITeamRepository repository;
public MembersController(ITeamRepository repo)
{
repository = repo;
}
[HttpGet]
public virtual IActionResult GetMembers(Guid teamID)
{
Team team = repository.Get(teamID);
if(team == null)
{
```

```
return this.NotFound();
}
else {
return this.Ok(team.Members);
}
}
[HttpGet] [
Route("/teams/{teamId}/[controller]/{memberId}")]
public virtual IActionResult GetMember(Guid teamID, Guid memberId)
{ Team team = repository.Get(teamID);
if(team == null)
{
return this.NotFound();
}
else
{
var q = team.Members.Where(m => m.ID == memberId);
if(q.Count() < 1)
{
return this.NotFound();
}
 Else
 {
return this.Ok(q.First());
}
}
}
[HttpPut]
[Route("/teams/{teamId}/[controller]/{memberId}")]
public virtual IActionResult UpdateMember([FromBody]Member updatedMemb
{ Team team = repository.Get(teamID);
if(team == null)
{ return this.NotFound();
}
else {
var q = team.Members.Where(m => m.ID == memberId);
if(q.Count() < 1)
{
 return this.NotFound();
}
else {
team.Members.Remove(q.First());
team.Members.Add(updatedMember);
return this.Ok();
}
}
```

```
}
[HttpPost]
public virtual IActionResult CreateMember([FromBody]Member newMember, Guid teamID)
{
Team team = repository.Get(teamID);
if(team == null)
{
return this.NotFound();
}
else {
team.Members.Add(newMember);
var teamMember = new {TeamID = team.ID, MemberID = newMember.ID};
return
this.Created($"/teams/{teamMember.TeamID}/[controller]/{teamMember.MemberID}",
teamMember);
}
}
[HttpGet]
[Route("/members/{memberId}/team")]
public IActionResult GetTeamForMember(Guid memberId)
{
var teamId = GetTeamIdForMember(memberId);
if (teamId != Guid.Empty)
{
return this.Ok(new {TeamID = teamId });
}
else {
return this.NotFound();
}
}
private Guid GetTeamIdForMember(Guid memberId)
{ foreach (var team in repository.List())
{ var member = team.Members.FirstOrDefault( m => m.ID == memberId);
if (member != null)
{ return team.ID;
}
}
return Guid.Empty;
}
}
}
```

5) create folder "D:\TeamService\Persistence":

6) add file ITeamReposiroty.cs in "D:\TeamService\Persistence" folder

```
using System;
using System.Collections.Generic;
```

```
using TeamService.Models;
namespace TeamService.Persistence
{
public interface ITeamRepository
{
IEnumerable List();
Team Get(Guid id);
Team Add(Team team);
Team Update(Team team);
Team Delete(Guid id);
}
}
```

7) Add MemoryTeamRepository.cs in "D:\TeamService\Persistence" folder

```
using System;
using System.Collections.Generic;
using System.Linq;
using TeamService;
using TeamService.Models;
namespace TeamService.Persistence
{
public class MemoryTeamRepository : ITeamRepository
{
protected static ICollection teams;
public MemoryTeamRepository() {
if(teams == null) {
teams = new List();
}
}
public MemoryTeamRepository(ICollection teams)
{
MemoryTeamRepository.teams = teams;
}
public IEnumerable List()
{
return teams;
}
public Team Get(Guid id)
{
 return teams.FirstOrDefault(t => t.ID == id);
}
public Team Update(Team t)
{
Team team = this.Delete(t.ID);
if(team != null)
{
team = this.Add(t);
```

```
}
return team;
}
public Team Add(Team team)
{
teams.Add(team);
return team;
}
public Team Delete(Guid id)
{
var q = teams.Where(t => t.ID == id);
Team team = null;
if (q.Count() > 0)
{
team = q.First();
teams.Remove(team);
}
return team;
}
}
}
```

8) add following line to Startup.cs in public void ConfigureServices(IServiceCollection services) method

services.AddScoped();

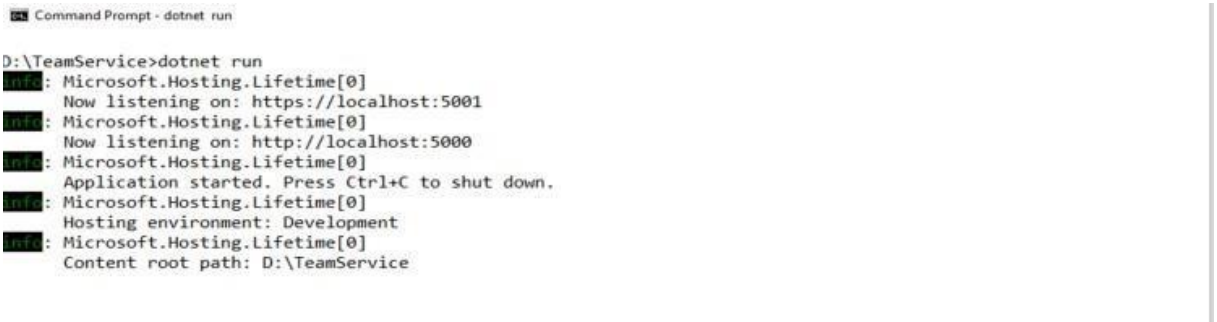
9) Now open two command prompts to run this project

10) On Command prompt 1: (go inside folder teamservice first)

Commands:

dotnet run

OUTPUT:

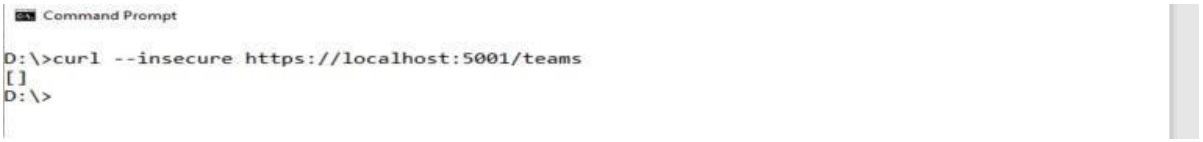


11) On command prompt 2

Command: To get all teams

curl --insecure https://localhost:5001/teams

OUTPUT:



Command : To create new team

curl --insecure -H “Content-Type:application/json” –X POST –d “{\“id\”:\”e52baa63-d511-417e-9e54- 7aab04286281\”, \”name\”:\”KC\”}” https://localhost:5001/teams

OUTPUT

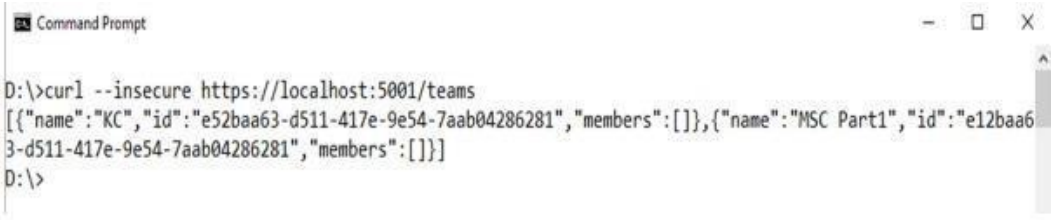
Command : To create one more new team

curl --insecure -H “Content-Type:application/json” –X POST –d “{\“id\”:\”e12baa63-d511-417e-9e54- 7aab04286281\”, \”name\”:\”MSC Part1\”}” https://localhost:5001/teams

OUTPUT:

Command : To get all teams curl --insecure https://localhost:5001/teams

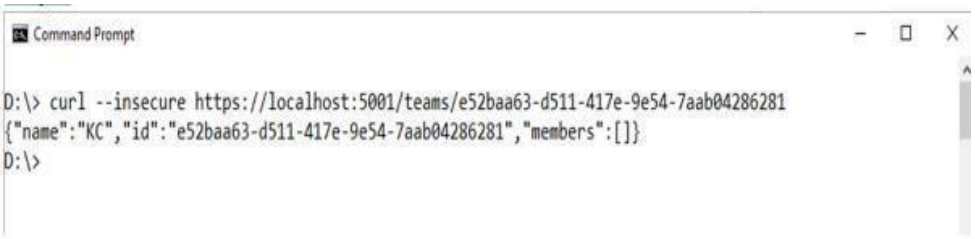
OUTPUT:



Command : to get single team with team-id as parameter

curl --insecure https://localhost:5001/teams/e52baa63-d511-417e-9e54-7aab04286281

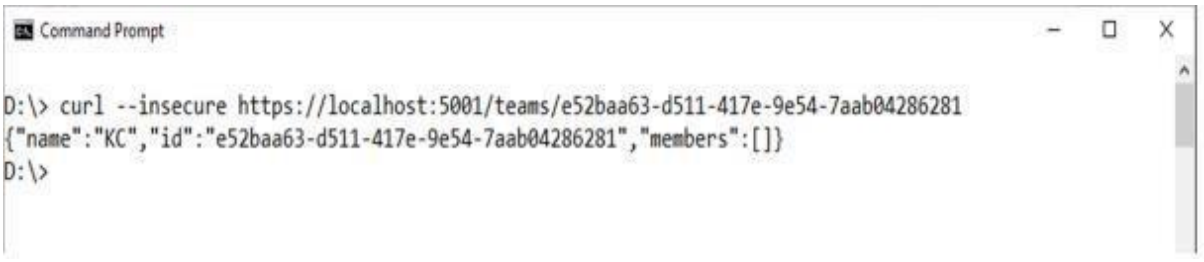
OUTPUT



Command : to update team details (change name of first team from "KC" to "KC IT DEPT")

curl --insecure -H "Content-Type:application/json" –X PUT –d "{\"id\":\"e52baa63-d511-417e-9e54- 7aab04286281\", \"name\":\"KC IT DEPT\"}" https://localhost:5001/teams/e52baa63-d511-417e-9e54- 7aab04286281

OUTPUT:



Command: to delete team

curl --insecure -H "Content-Type:application/json" –X DELETE https://localhost:5001/teams/e52baa63-d511-417e-

9e54-7aab04286281

OUTPUT:

Confirm: with get all teams now it shows only one team (first one is deleted)

Command: curl –insecure https://localhost:5001/teams

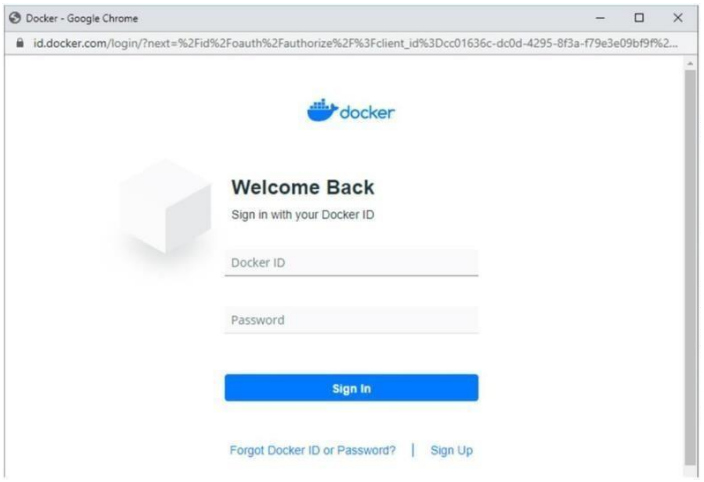
OUTPUT:


Command Prompt
D:\>curl --insecure https://localhost:5001/teams
[{"name":"MSC Part1","id":"e12baa63-d511-417e-9e54-7aab04286281","members":[]}]
D:\>

PRACTICAL 7

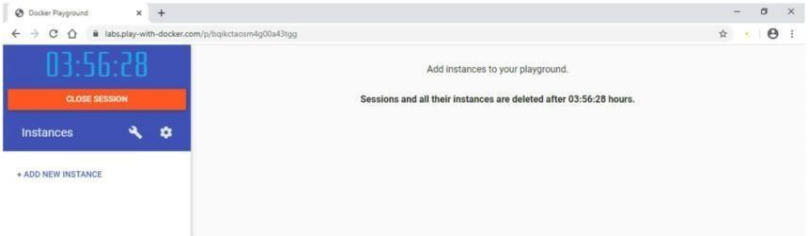
Running Location Service in Docker

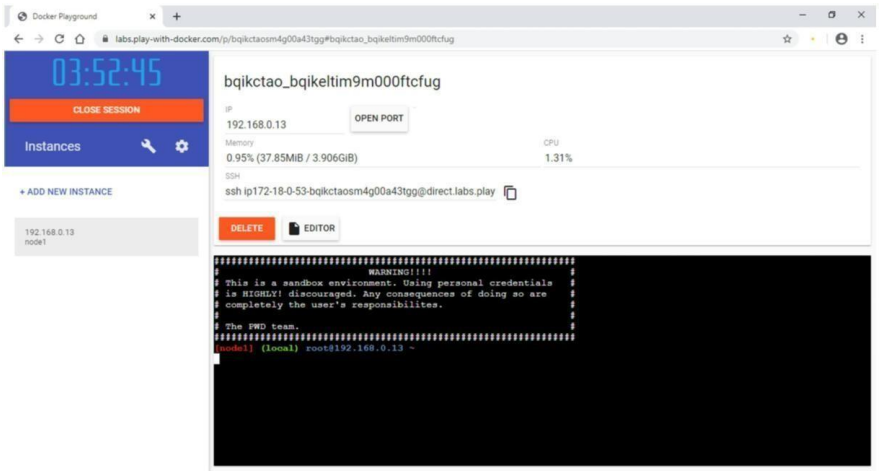
Create docker hub login first to use it in play with docker
Now login in to Play-With-Docker



Click on Start

Click on Add New Instance





Start typing following commands

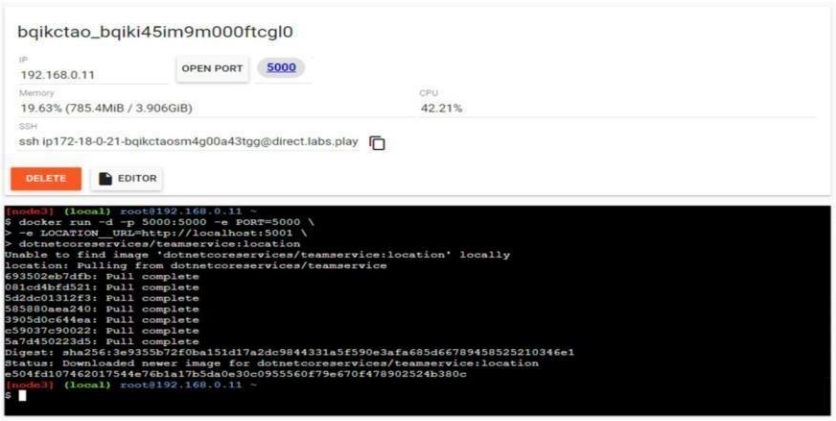
Command to run team service

**docker run -d -p 5000:5000 -e PORT=5000 **

**-e LOCATION_URL=http://localhost:5001 **

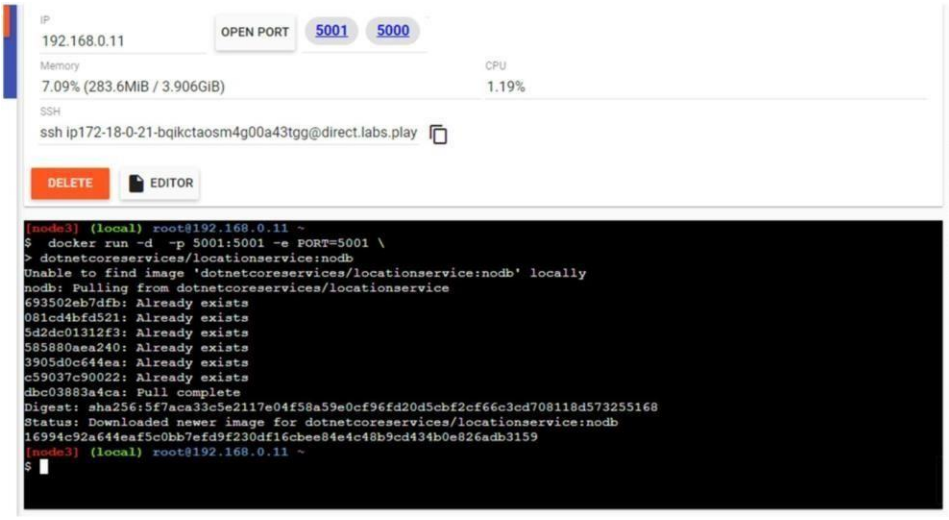
dotnetcoreservices/teamservice:location

OUTPUT: (you can observe that it has started port 5000 on top)



Command: to run location service
**docker run -d -p 5001:5001 -e PORT=5001 **
dotnetcoreservices/locationservice:nodb

OUTPUT:



Command to check running images in docker

docker images

OUTPUT:

Command: to create new team

curl -H "Content-Type:application/json" -X POST -d \'{"id":"e52baa63-d511-417e-9e54-7aab04286281", "name":"KC"}' http://localhost:5000/teams

OUTPUT:

Command to confirm that team is added

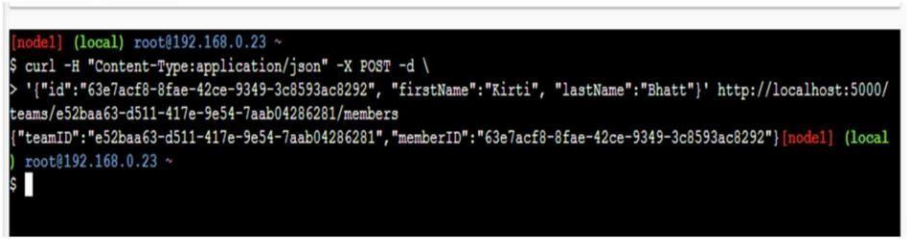
curl http://localhost:5000/teams/e52baa63-d511-417e-9e54-7aab04286281

OUTPUT:

Command to add new member to team

curl -H "Content-Type:application/json" -X POST -d \ '{"id":"63e7acf8-8fae-42ce-9349-3c8593ac8292", "firstName":"Kirti", "lastName":"Bhatt"}' http://localhost:5000/teams/e52baa63-d511-417e-9e54-7aab04286281/members

OUTPUT:



Command to confirm member added

curl http://localhost:5000/teams/e52baa63-d511-417e-9e54-7aab04286281

OUTPUT:

Command to add location for member

curl -H "Content-Type:application/json" -X POST -d \ '{"id":"64c3e69f-1580-4b2f-a9ff-2c5f3b8f0e1f", "latitude":12.0,"longitude":12.0,"altitude":10.0, "timestamp":0,"memberId":"63e7acf8-8fae-42ce-9349-3c8593ac8292"}' http://localhost:5001/locations/63e7acf8-8fae-42ce-9349-3c8593ac8292

OUTPUT:

Command: To confirm location is added in member

curl http://localhost:5001/locations/63e7acf8-8fae-42ce-9349-3c8593ac8292

OUTPUT:

IMAGE PROCESSING

NURTURING POTENTIAL

SAKET GYANPEETH'S

SAKET COLLEGE OF ARTS, SCIENCE, AND COMMERCE

(Permanently Affiliated to University of Mumbai)

NAAC Accredited B Grade

Saket Vidyanagari Marg, Chinchpada Road, Katemanivali, Kalyan (East) – 421306, Dist. Thane (MAH)

Department of Information Technology

CERTIFICATE

This is to certify that
NA of **MSc Information Technology Part-I**
Class has satisfactorily carried out the required practical in the subject.

IMAGE PROCESSING

For the Academic year 2022-2023

Practical in Charge **Head of Department** **External Examiner**

INDEX

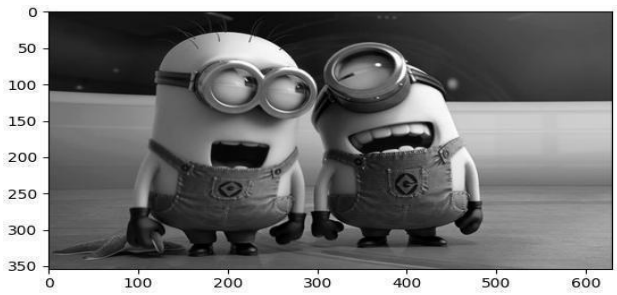
PRACT NO.	PRACTICAL	SIGN
1.	A. Display Of An Image And Conversion Of Gray Scale B. Negative Transform Of An Image	
2.	A. To Perform Image Negation B. Log Trans C. Bit Plane Slicing D. Histogram	
3.	A. High Pass B. Low Pass C. Discrete Fourier Transform	
4.	A. Mean Filter B. Median Filter C. Histogram	
5.	A. Gray To False Color B. Gray Level With Bg C. Gray To Color	
6.	To implement Morphological operations	
7.	To implement Boundary	
8.	To perform Addition and subtraction on two image.	

PRACTICAL 1

A. DISPLAY OF AN IMAGE AND CONVERSION OF GRAY SCALE

```
import PIL
import matplotlib.pyplot as plt
img = PIL.Image.open("image.jpg")
gray_img = img.convert("L")
plt.imshow(gray_img, cmap='gray')
plt.savefig("gray_pic.jpg")
```

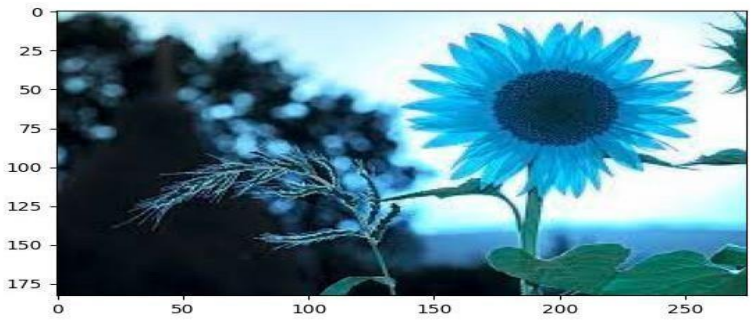
OUTPUT:

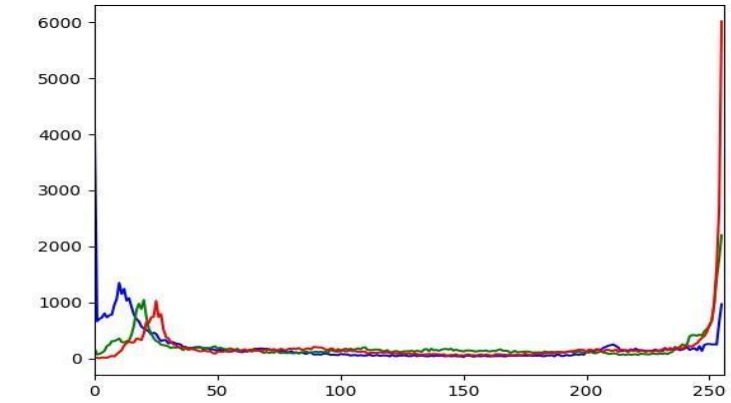


B. NEGATIVE TRANSFORM OF AN IMAGE

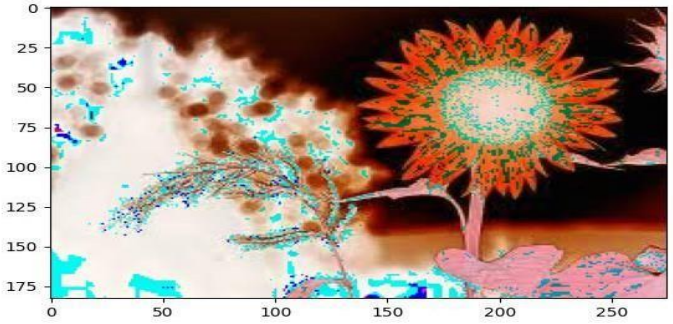
```
import cv2
import matplotlib.pyplot as plt
img_bgr = cv2.imread('image.jpg', 1)
plt.imshow(img_bgr)
plt.show()
color = ('b', 'g', 'r')
for i, col in enumerate(color):
    histr = cv2.calcHist([img_bgr],[i], None,[256],[0, 256])
    plt.plot(histr, color = col)
    plt.xlim([0, 256])
plt.show()
img_neg = 1 - img_bgr
plt.imshow(img_neg)
plt.show()
color = ('b', 'g', 'r')
for i, col in enumerate(color):
    histr = cv2.calcHist([img_neg],[i], None, [256],[0, 256])
    plt.plot(histr, color = col)
    plt.xlim([0, 256])
plt.show()
```

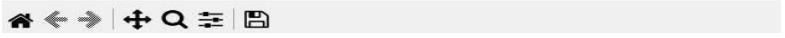
OUTPUT:

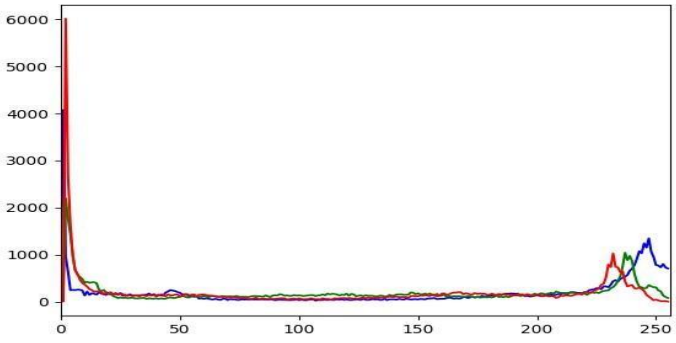

0
25
50
75
100
125
150
175
0
50
100
150
200
250

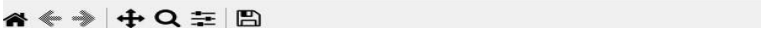

6000
5000
4000
3000
2000
1000
0
0
50
100
150
200
250


x=142.5 y=4.57e+03


0
25
50
75
100
125
150
175
0
50
100
150
200
250




6000
5000
4000
3000
2000
1000
0
0
50
100
150
200
250

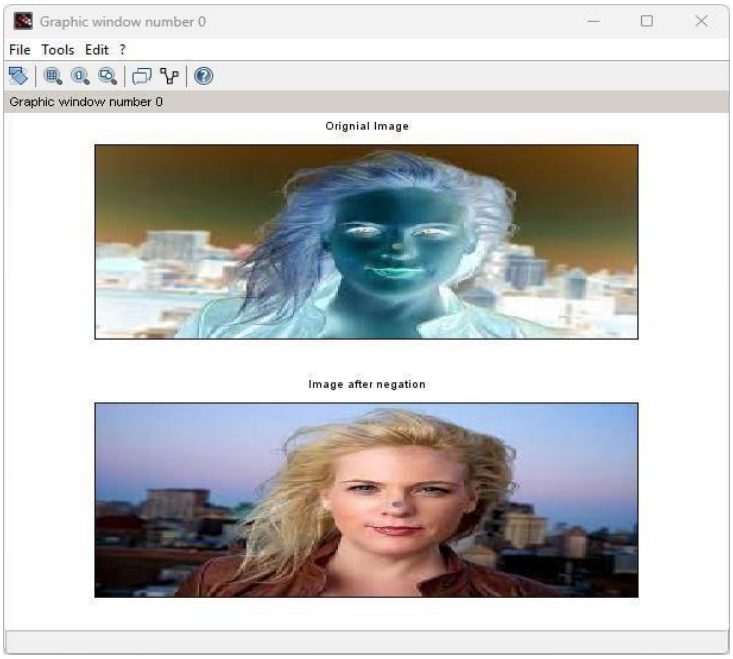


PRACTICAL 2

A) TO PERFORM IMAGE NEGATION

```
clc;
clear all;
A
=imread("C:\Users\Student\Desktop\NA\negimg.jpg");
subplot(2,1,1);
imshow(A);
title('Orignial Image');
R = A(:,:,1);
G = A(:,:,2);
B = A(:,:,3);
[row col]=size(A);
for x=1:row
   for y=1:col
      R(x,y)=255-R(x,y);
      G(x,y)=255-G(x,y);
      B(x,y)=255-B(x,y);
   end
end
A(:,:,1)=R;
A(:,:,2)=G;
A(:,:,3)=B;
subplot(2,1,2);
imshow(A);
title('Image after negation');
```

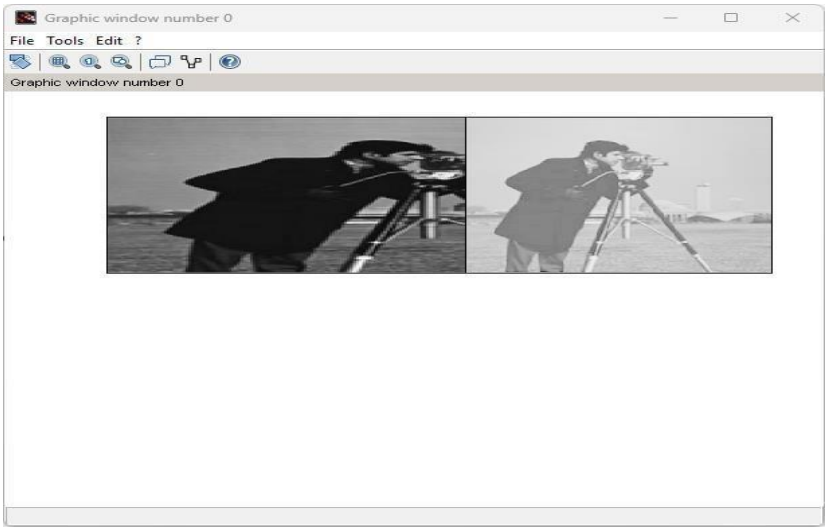
OUTPUT:



B) LOG TRANS

```
clc;
clear all;
a=imread("C:\Users\Student\Downloads\camera.png");
a=rgb2gray(a);
subplot(2,1,1);
imshow(a);
c=1;
[r1,c1]=size(a);
for i=1:r1
   for j=1:c1
      b=double(a(i,j));
      s(i,j)=c*log10(1+b);
   end
end
new1=uint8(s*100);
//imshow(new1);
subplot(2,2,2);
imshow(new1);
```

OUTPUT:

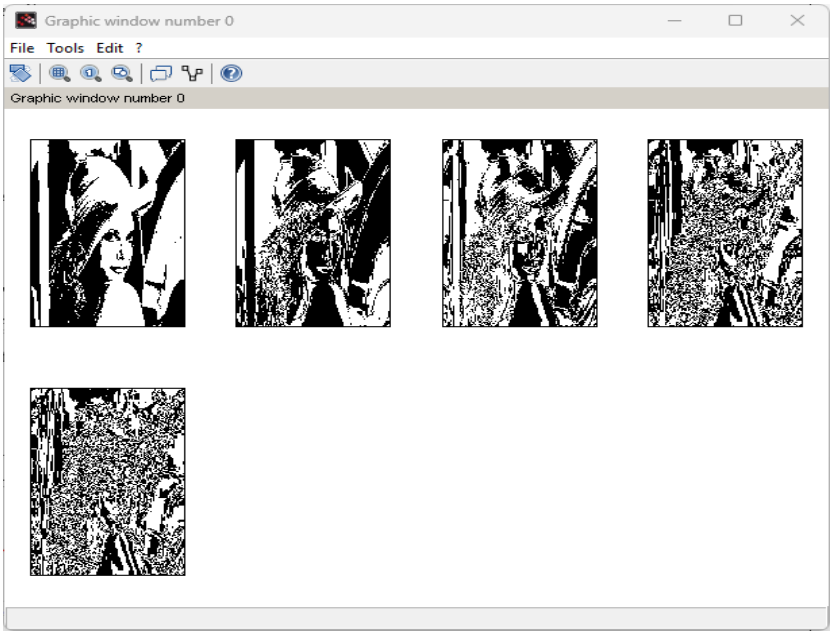


C) BIT PLANE SLICING

```
clc;
clear all;
f=imread("C:\Users\Student\Downloads\lenag.jpeg");
f=double(f);
[r,c]=size(f);
com=[128 64 32 16 8 4 2 1];
for k=1:1:length(com);
   for i=1:r
      for j=1:c
      new(i,j)=bitand(f(i,j),com(k));
   end
```

```
    subplot(2,4,k);
    imshow(new);
    end
end
```

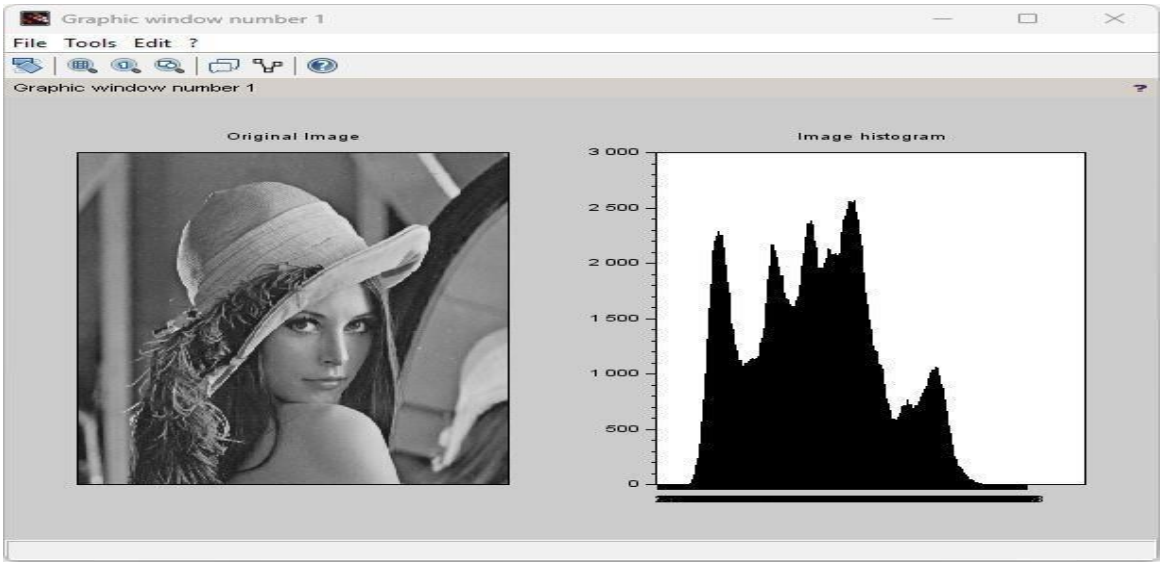
OUTPUT:



D) HISTOGRAM

```
clc;
clear all;
a=imread('C:\Users\Student\Downloads\lena.jpeg');
a=rgb2gray(a);
h=zeros(1,258);
[r,c]=size(a);
for i=1:r
    for j=1:c
        if (a(i,j)==0)
            h(0)=h(0)+1;
        end
        k=a(i,j);
        h(k)=h(k)+1;
    end
end
figure(1);
subplot(1,2,1);
imshow(uint8(a));
title('Original Image')
subplot(1,2,2);
bar(h);
title('Image histogram');
```

OUTPUT:

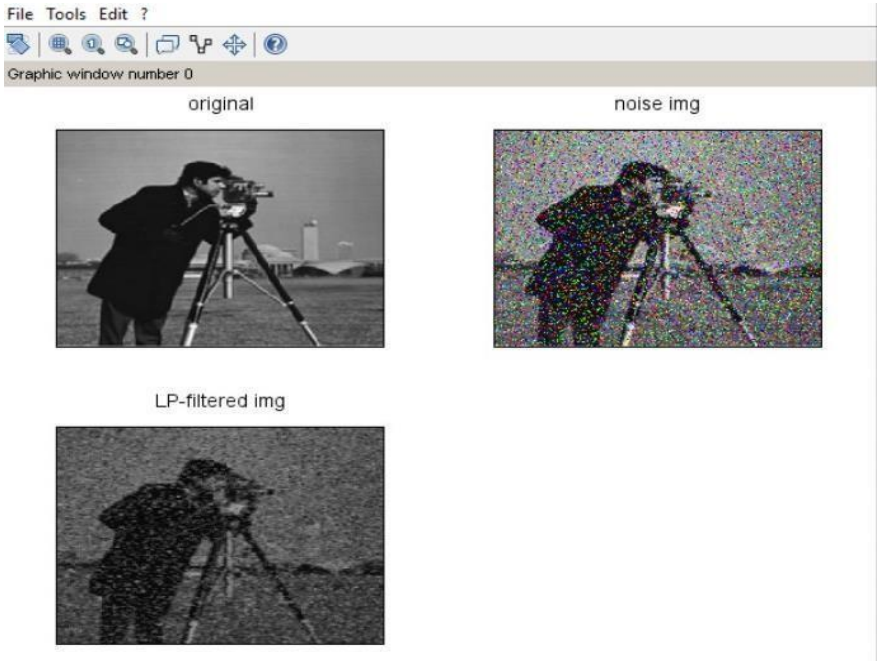

Graphic window number 1
File Tools Edit ?
Graphic window number 1
Original Image
Image histogram
3 000
2 500
2 000
1 500
1 000
500
0

PRACTICAL 3

A. HIGH PASS

```
a=imread('C:\Users\Student\Downloads\NA\OIP.jpg'
);b=double(a);
c=imnoise(a,'salt & pepper',0.2);
d=double(c);
subplot(2,2,1);
imshow(a);
title('original');
subplot(2,2,2);
imshow(c);
title('noise img');
m=[-1 -1 -1;-1 8 -1;-1 -1 -1];
[r1,c1]=size(a);
for i=2:1:r1-1
    for j=2:1:c1-1
   new(i,j)=(m(1)*d(i-1,j-1))+(m(2)*d(i-1,j))+(m(3)*d(i-1,j+1))    +(m(4)*d(i,j
1))+(m(5)*d(i,j))+(m(6)*d(i,j+1)) +(m(7)*d(i+1,j-1))+(m(8)*d(i+1,j))+(m(9)*d(i+1,j+1));
    end
end
subplot(2,2,3);
imshow(uint8(new));
title('LP-filtered img');
```

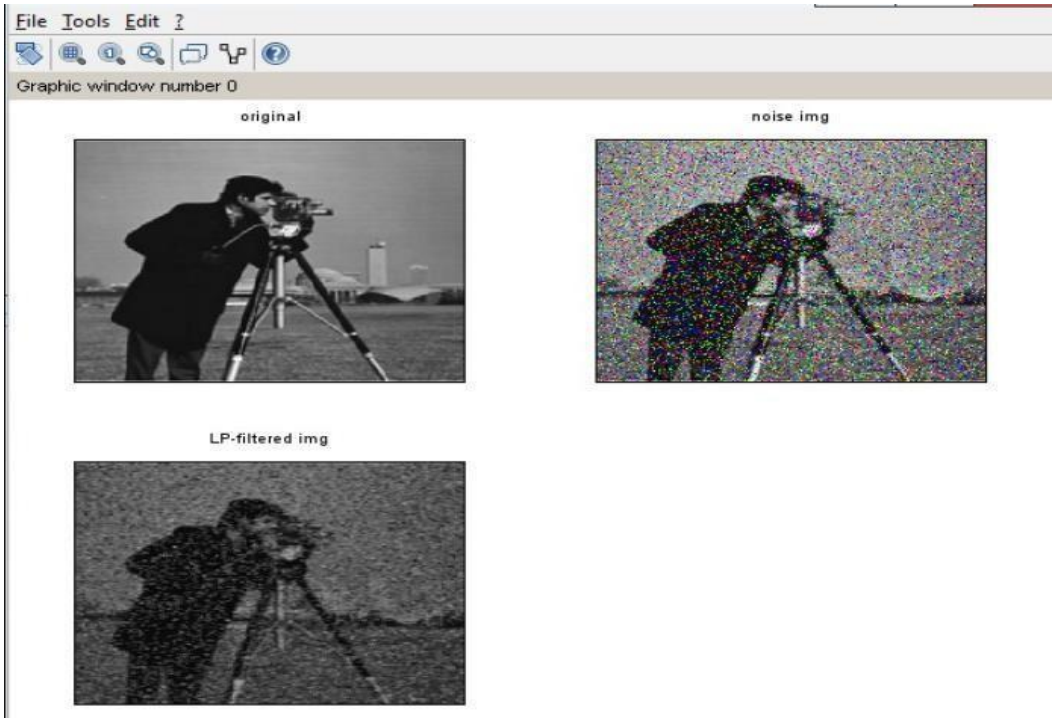
OUTPUT:



B. LOW PASS

```
clc;
clear all;
a=imread('C:\Users\Student\Downloads\NA\camera (1)
(1).png');b=double(a);
c=imnoise(a,'salt & pepper',0.2);
d=double(c);
subplot(2,2,1);
imshow(a);
title('original');
subplot(2,2,2);
imshow(c);
title('noise img');
m=(1/9)*(ones(3,3));
[r1,c1]=size(a);
for i=2:1:r1-1
    for j=2:1:c1-1
  new(i,j)=(m(1)*d(i-1,j-1))+(m(2)*d(i-1,j))+(m(3)*d(i-1,j+1))    +(m(4)*d(i,j
1))+(m(5)*d(i,j))+(m(6)*d(i,j+1))  +(m(7)*d(i+1,j-
1))+(m(8)*d(i+1,j))+(m(9)*d(i+1,j+1));
    end
end
subplot(2,2,3);
imshow(uint8(new));
title('LP-filtered img');
```

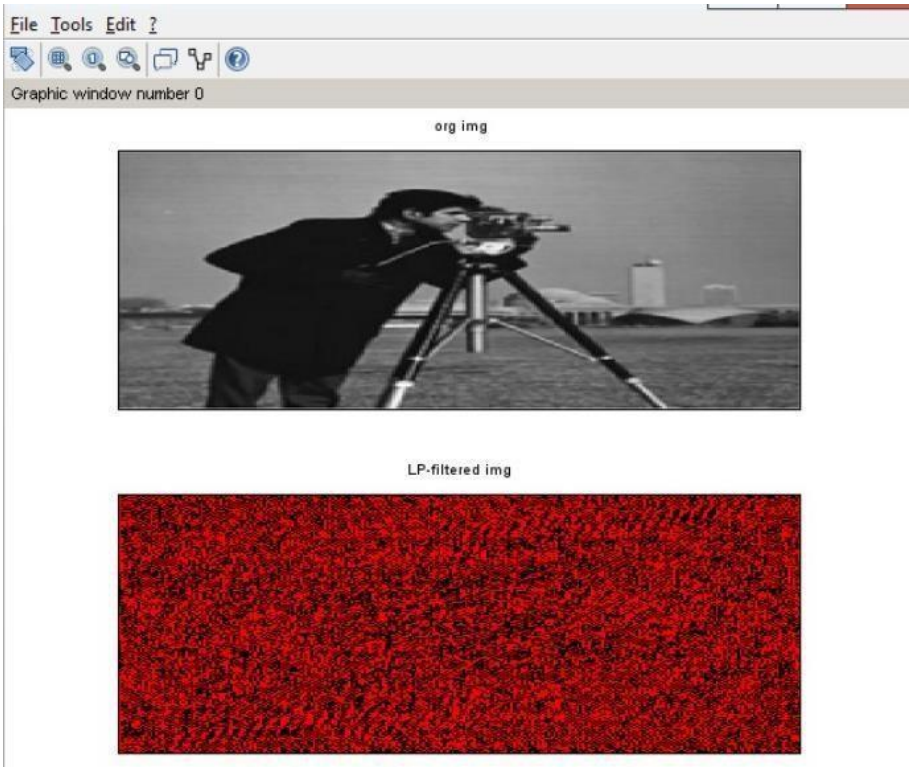
OUTPUT:



C. DISCRETE FOURIER TRANSFORM

```
a=imread('C:\Users\Student\Downloads\camera (1).png');
subplot(2,1,1);
imshow(a);
title('org img');
b=double(a);
c=fft(b);
subplot(2,1,2);
imshow(c);
title('LP-filtered img');
```

OUTPUT:

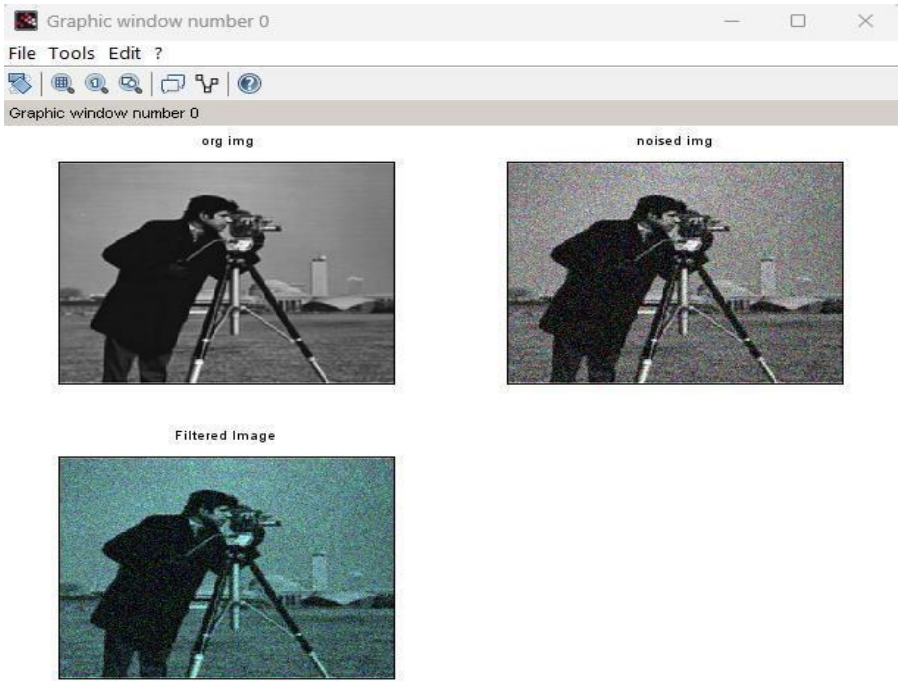


PRACTICAL 4

A) MEAN FILTER

```
a=imread('C:\Users\Student\Downloads\NA\camera (1)
(1).png');b1=double(a);
c=imnoise(a,'gaussian');
d=double(c);
b=d;
m=(1/9)*(ones(3,3));
[r1,c1]=size(a);
subplot(2,2,1);
imshow(a);
title('org img');
subplot(2,2,2);
imshow(c);
title('noised img');
for i=2:r1-1
for j=2:c1-1
a1=d(i-1,j-1)+d(i-1,j)+d(i-1,j+1)+d(i,j-1)+d(i,j)+d(i,j+1)+d(i+1,j-1)+d(i+1,j)+d(i+1,j+1);
b(i,j)=a1*(1/9);
end
end
subplot(2,2,3);
imshow(uint8(b));
title('Filtered Image');
clc;
clear all;
```

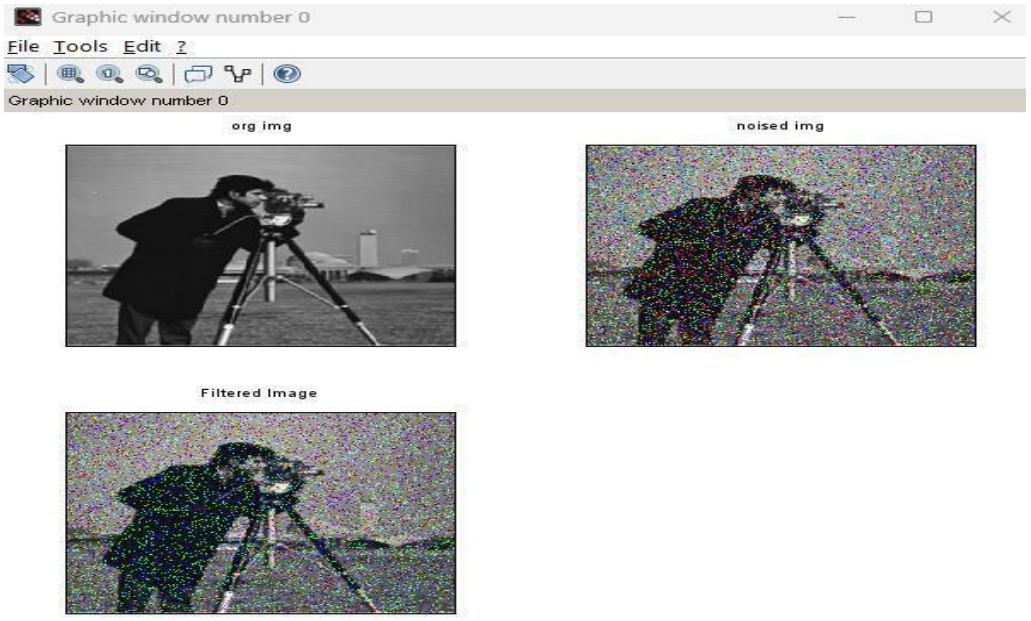
OUTPUT:



B) MEDIAN FILTER

```
a=imread("C:\Users\Student\Downloads\NA\camera (1)
(1).png');b1 = double(mtlb_double(a));
c = imnoise(a,"salt & pepper",0.2);
d = double(mtlb_double(c));
b = d;
m = (1/9)*ones(3,3);
subplot(2,2,1);
imshow(a);
title('org img');
subplot(2,2,2);
imshow(c);
title('noised img');
[r1,c1] = size(mtlb_double(a));
for i = 2:r1-1
      for j = 2:c1-1
      a1 = [d(i-1,j-1),d(i-1,j),d(i-1,j+1),d(i,j-1),d(i,j), d(i,j+1),d(i+1,j1),d(i+1,j),d(i+1,j+1)];
      a2 = gsort(a1,"g","i");//gsort(A,'g','i') sort the elements of the array A in the
increasing order.
      med = a2(5);
      b(i,j) = med;
      end;
end;
subplot(2,2,3);
imshow(uint8(b));
title('Filtered Image');
```

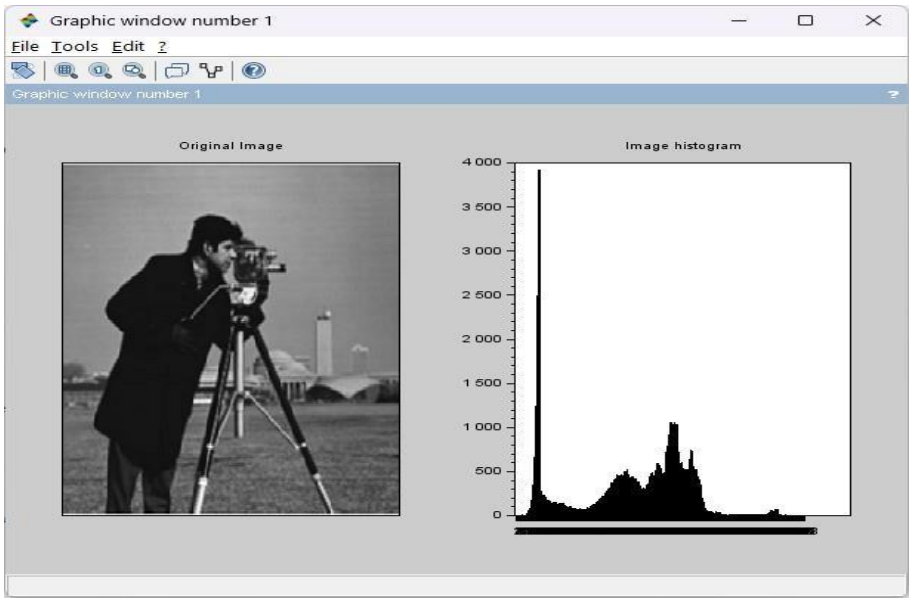
OUTPUT:



C) HISTOGRAM

```
clc;
clear all;
a=imread('C:\Users\Student\Downloads\NA\camera (1)
(1).png');a=rgb2gray(a);
h=zeros(1,258);
[r,c]=size(a);
for i=1:r
   for j=1:c
      if (a(i,j)==0)
         h(0)=h(0)+1;
      end
      k=a(i,j);
      h(k)=h(k)+1;
   end
end
figure(1);
subplot(1,2,1);
imshow(uint8(a));
title('Original Image')
subplot(1,2,2);
bar(h);
title('Image histogram');
```

OUTPUT:

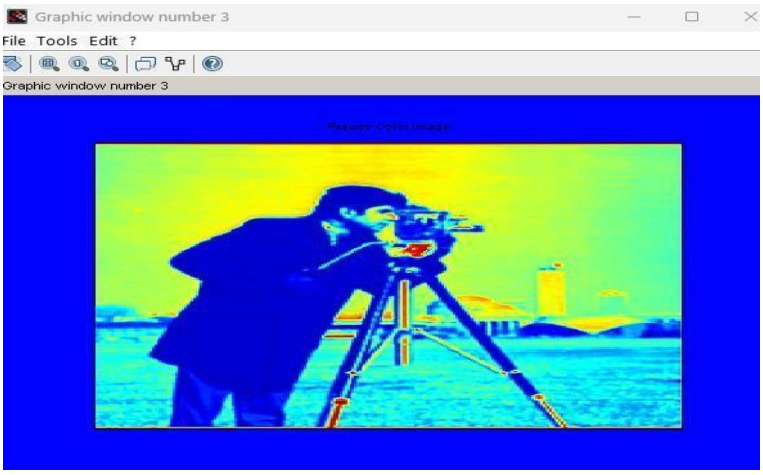


PRACTICAL 5

A) GRAY TO FALSE COLOR

```
clc;


close;
a = imread('C:\Users\Student\Downloads\NA\camera (1) (1).png');
//Displaying Original RGB image
figure(1);
imshow(a);
title("Original Image")
//Displaying Gray level image
b = rgb2gray(a);
figure(2);
imshow(b);
title("Gray Level Image")
//Displaying False coloring(Pseudo) image
figure(3)
imshow(b,jetcolormap(256));
title("Pseudo Color Image");
```

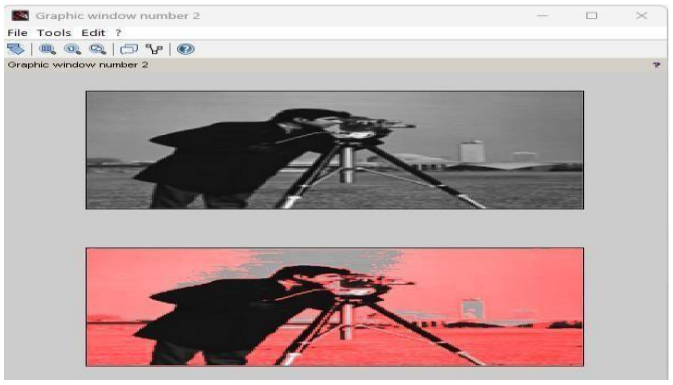
OUTPUT:



B) GRAY LEVEL WITH BG

```
clc;
clear all;
a=imread('C:\Users\Student\Downloads\NA\camera (1)
(1).png');a1=58; // This value is user defined
b1=158; // This value is user defined
[r,c]=size(a);
figure(2);
subplot(2,1,1);
imshow(a);
for i=1:r
   for j=1:c
      if (a(i,j)>a1 & a(i,j)<b1)
         x(i,j)=255;
 else
         x(i,j)=a(i,j);
      end
   end
end
x=uint8(x);
subplot(2,1,2);
imshow(x);
```

OUTPUT:

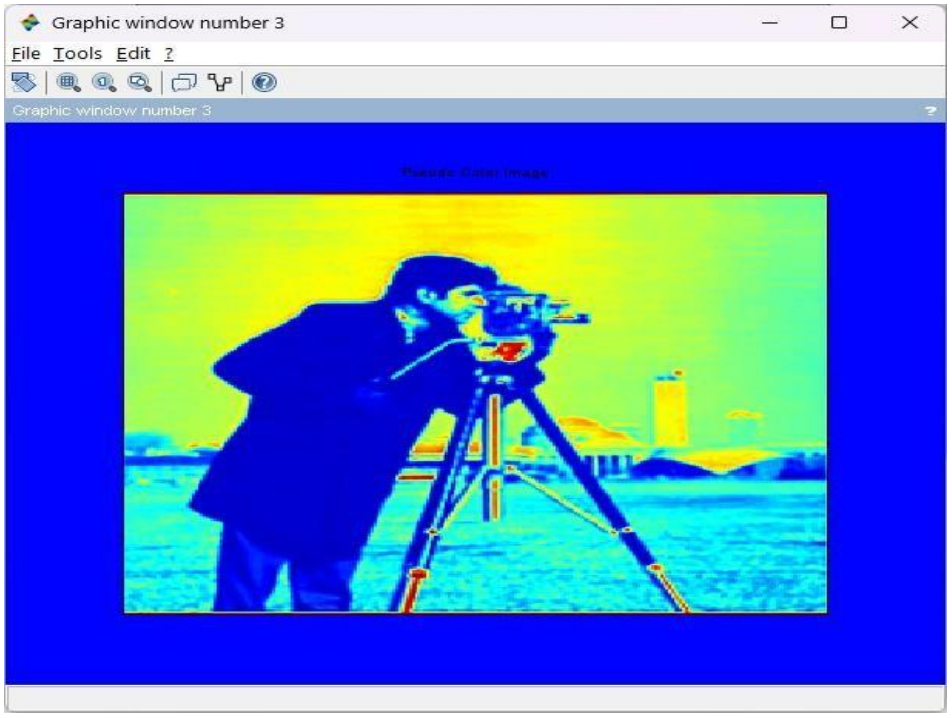


C) GRAY TO COLOR

```
clc;
close;
a = imread('C:\Users\Student\Downloads\NA\camera (1) (1).png');
//Displaying Original RGB image
figure(1);
imshow(a);
title("Original Image")
//Displaying Gray level image
b = rgb2gray(a);
figure(2);
imshow(b);
title("Gray Level Image")
//Displaying False coloring(Pseudo) image
figure(3)
imshow(b,jetcolormap(256));
title("Pseudo Color Image");
```

OUTPUT:


Graphic window number 2
File Tools Edit ?
Graphic window number 2
Gray Level Image


Graphic window number 3
File Tools Edit ?
Graphic window number 3

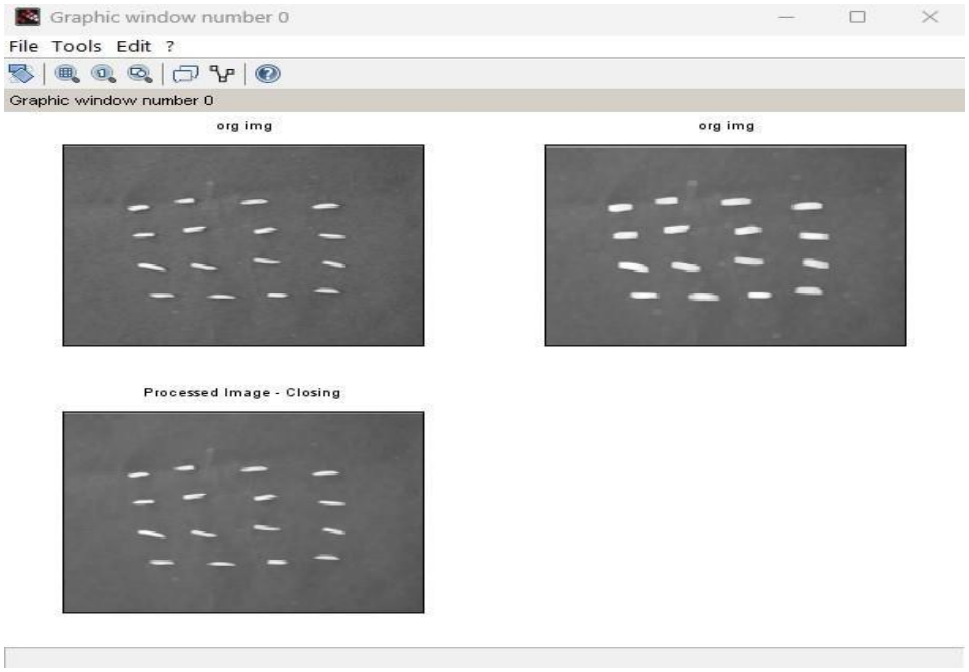
PRATICAL 6

Aim: To implement Morphological operations

A) Closing

```
clc;
clear all;
a=imread('C:\NA\rice.jpeg'
);a=rgb2gray(a);
d=a;
A2=d;
A1=d;
subplot(2,2,1);
imshow(a);
title('org img');
[r,c]=size(d);
m=[1 1 1;1 1 1;1 1 1];
for i=2:1:r-1
for j=2:1:c-1
new=[(m(1)*d(i-1,j-1)) (m(2)*d(i-1,j)) (m(3)*d(i-1,j+1)) (m(4)*d(i,j-1)) (m(5)*d(i,j))
(m(6)*d(i,j+1)) (m(7)*d(i+1,j-1)) (m(8)*d(i+1,j)) (m(9)*d(i+1,j+1))];
A2(i,j)=max(new);
end
subplot(2,2,2);
imshow(A2);
title('org img');
end
d = A2;
A1=A2;
[r,c]=size(d);
for  i=2:1:r-1
for j=2:1:c-1
new=[(m(1)*d(i-1,j-1)) (m(2)*d(i-1,j)) (m(3)*d(i-1,j+1)) (m(4)*d(i,j-1)) (m(5)*d(i,j))
(m(6)*d(i,j+1))(m(7)*d(i+1,j-1)) (m(8)*d(i+1,j)) (m(9)*d(i+1,j+1))];
A1(i,j)=min(new);
end
subplot(2,2,3);
imshow(A1);title('Processed Image - Closing');
end
```

OUTPUT:

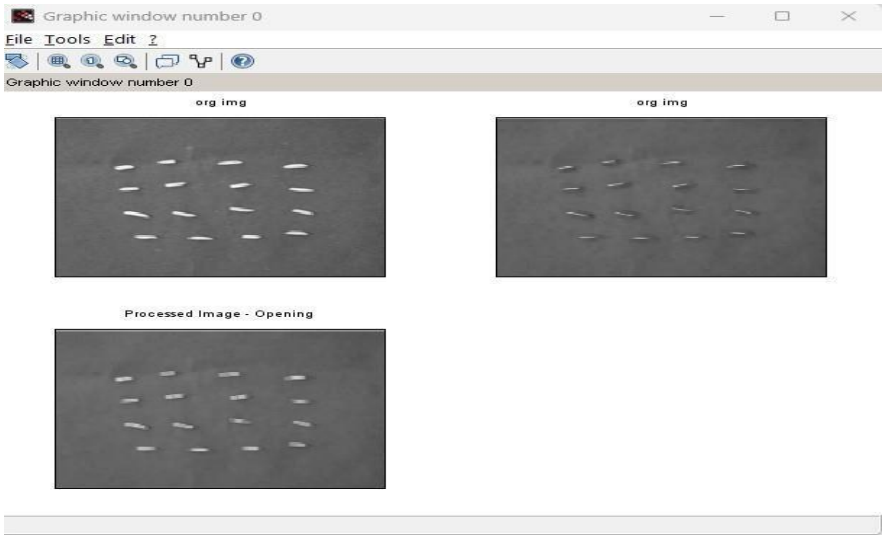


B) Opening

```
clc;
clear all;
a=imread("C:\NA\rice.jpeg"
);a=rgb2gray(a);
d=a;
A2=d;
A1=d;
subplot(2,2,1);
imshow(a);
title('org img');
[r,c]=size(d);
m=[1 1 1;1 1 1;1 1 1];
for i=2:1:r-1
for j=2:1:c-1
new=[(m(1)*d(i-1,j-1)) (m(2)*d(i-1,j)) (m(3)*d(i-1,j+1))
(m(4)*d(i,j-1)) (m(5)*d(i,j)) (m(6)*d(i,j+1))
(m(7)*d(i+1,j-1)) (m(8)*d(i+1,j)) (m(9)*d(i+1,j+1))];
A2(i,j)=min(new);
end
subplot(2,2,2);
imshow(A2);
title('org img');
end
d = A2;
A1=A2;
```

```
[r,c]=size(d);
for  i=2:1:r-1
for j=2:1:c-1
new=[(m(1)*d(i-1,j-1)) (m(2)*d(i-1,j)) (m(3)*d(i-1,j+1))
(m(4)*d(i,j-1)) (m(5)*d(i,j)) (m(6)*d(i,j+1))
(m(7)*d(i+1,j-1)) (m(8)*d(i+1,j)) (m(9)*d(i+1,j+1))];
A1(i,j)=max(new);
end
subplot(2,2,3);
imshow(A1);title('Processed Image - Opening');
end
```

OUTPUT:

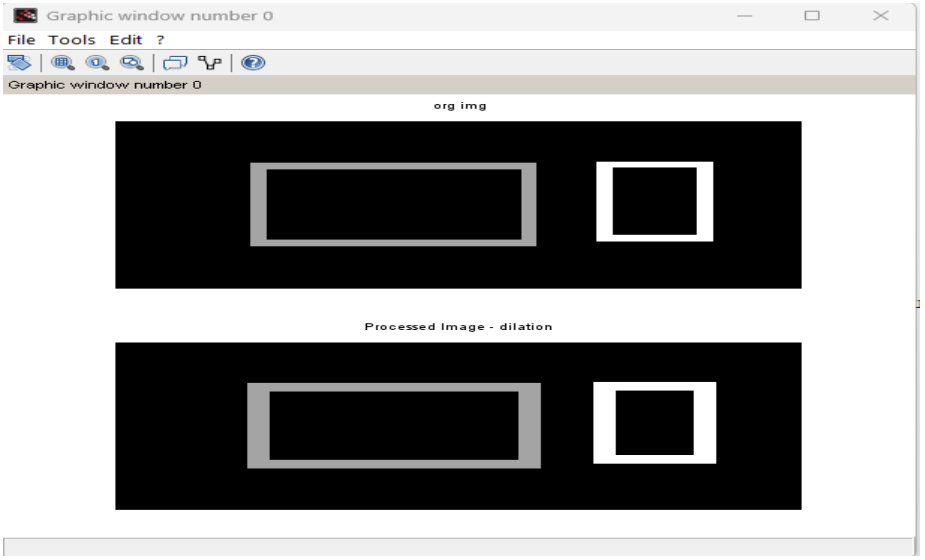


C)Dialation

```
clc;
clear all;
a=imread('C:\NA\rectb.png'
);a=rgb2gray(a);
d=a;
A1=a;
[r,c]=size(d);
subplot(2,1,1);
imshow(a);
title('org img');
m=[1 1 1;1 1 1;1 1 1];
// m=ones(5,5);
for i=2:1:r-1
for j=2:1:c-1
```

```
new=[(m(1)*d(i-1,j-1)) (m(2)*d(i-1,j)) (m(3)*d(i-1,j+1)) (m(4)*d(i,j-1)) (m(5)*d(i,j))
(m(6)*d(i,j+1)) (m(7)*d(i+1,j-1)) (m(8)*d(i+1,j)) (m(9)*d(i+1,j+1))];
A1(i,j)=max(new);
end
subplot(2,1,2);
imshow(A1);title('Processed Image - dilation');
end
```

OUTPUT:

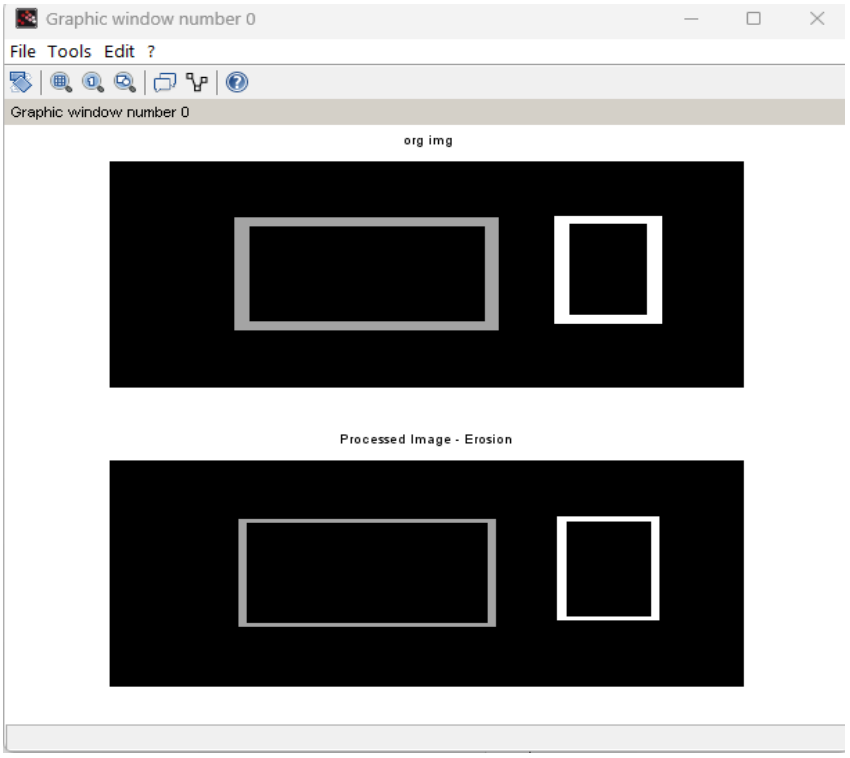


D) Erosion

```
clc;
clear all;
a=imread('C:\NA\rectb.png'
);a=rgb2gray(a);
subplot(2,1,1);
imshow(a);
title('org img');
A1=a;
d=a;
[r,c]=size(d);
m=[1 1 1;1 1 1;1 1 1];
// m=ones(5,5);
for i=2:1:r-1
for j=2:1:c-1
   new=[(m(1)*d(i-1,j-1)) (m(2)*d(i-1,j)) (m(3)*d(i-1,j+1)) (m(4)*d(i,j-1))
(m(5)*d(i,j)) (m(6)*d(i,j+1)) (m(7)*d(i+1,j-1)) (m(8)*d(i+1,j)) (m(9)*d(i+1,j+1))];
A1(i,j)=min(new);
end
subplot(2,1,2);
```

```
title('org img');imshow(A1);title('Processed Image - Erosion');
end
```

OUTPUT:

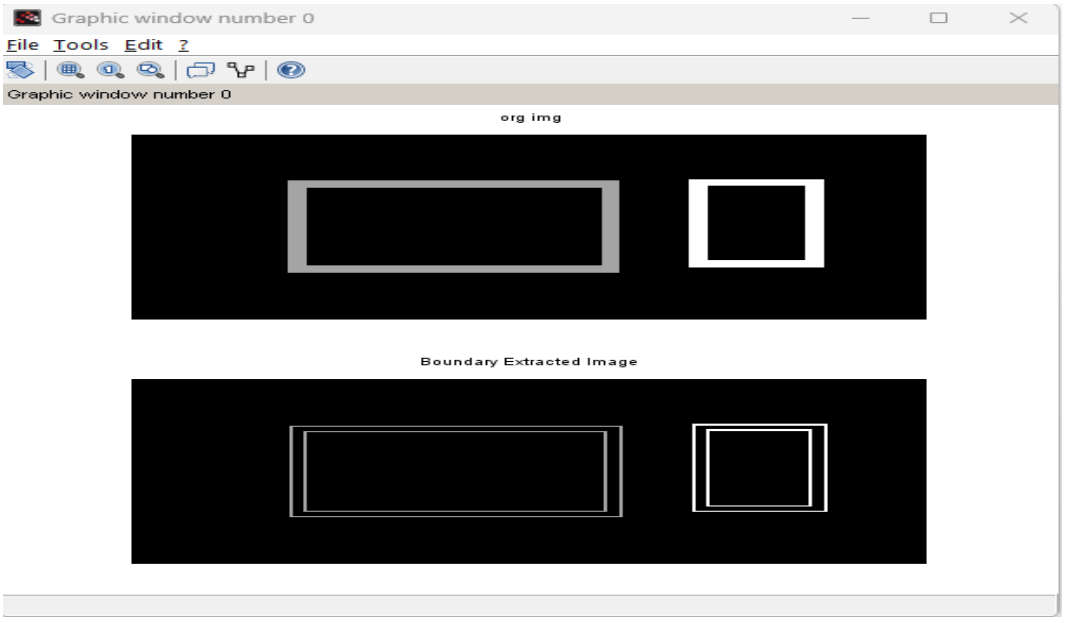


PRACTICAL 7

Aim : To implement Boundary

```
clc;
clear all;
a=imread('C:\NA\rectb.png'
);a=rgb2gray(a);
subplot(2,1,1);
imshow(a);
title('org img');
d=a;
[r,c]=size(d);
m=[1 1 1;1 1 1;1 1 1];
for i=2:1:r-1
for j=2:1:c-1
new=[(m(1)*d(i-1,j-1)) (m(2)*d(i-1,j)) (m(3)*d(i-1,j+1))
(m(4)*d(i,j-1)) (m(5)*d(i,j)) (m(6)*d(i,j+1))
(m(7)*d(i+1,j-1)) (m(8)*d(i+1,j)) (m(9)*d(i+1,j+1))];
A2(i,j)=min(new);
aa(i,j)=d(i,j)-A2(i,j);
end
end
subplot(2,1,2);
imshow(aa);title('Boundary Extracted Image');
```

OUTPUT:



PRATICAL 8

Aim: To perform Addition and subtraction on two image

```
clc;
clear all;
A=imread('C:\NA\circle.png');
B=imread('C:\NA\camera
(2).png');A=rgb2gray(A);
B=rgb2gray(B);
C=imadd(B, A);
D=imsubtract(B, A);
figure(1);
subplot(2,2,1);
imshow(A);
subplot(2,2,2);
imshow(B);
subplot(2,2,3);
imshow(C);
subplot(2,2,4);
imshow(D);
```

OUTPUT: