UNIVERSITY OF MUMBAI

PRACTICAL REPORT
M.Sc. I.T (PART 1) 2022-2023

PAPER 1: BIG DATA ANALYTICS

PAPER 2: MODERN NETWORKING

PAPER 3: MICROSERVICES ARCHITECTURE

PAPER 4: IMAGE PROCESSING

SUBMITTED BY:
FAROOQUI ANAM FATMA RAIS AHMAD

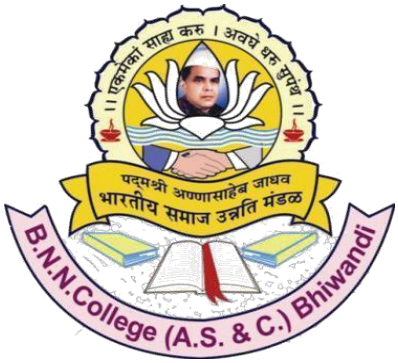
PROF. SIMRAN SHAIKH
{TEACHER INCHARGE}

PROF. SUMIYA MADOO
{TEACHER INCHARGE}

PROF. SRAVANTI GAJJELLI
{TEACHER INCHARGE}

PROF. BHAVNA PATIL
{TEACHER INCHARGE}

Padmashri Annasaheb Jadhav Bhartiya Samaj Unnati Mandal's
B.N.N COLLEGE
(Arts, Science, Commerce & Self-Funded Course)
(Affiliated to University of Mumbai)
DEPARTMENT OF INFORMATION TECHNOLOGY
BHIWANDI, MAHARASHTRA-421302



Padmashri Annasaheb Jadhav Bhartiya Samaj Unnati Mandal's

B.N.N College of Arts, Science and Commerce, Bhiwandi.

(Self-Funded Course)

(Department of Information Technology)

CERTIFICATE

This is to certify that Mr./Miss. **Farooqui Anam Fatma Rais Ahmad** .

Roll No._____ Class: **MSc-IT** Exam Seat No. **4132807** .

has satisfactorily completed practical in **Big Data Analytics** .

As laid down in the regulation of University of Mumbai for the purpose of **Semester- II Practical** Examination 2022-2023.

Date:

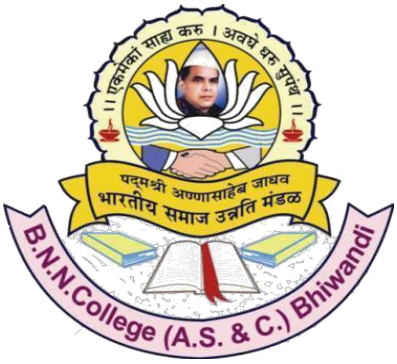
Place: **Bhiwandi**

_____ _____

In-Charge Professor **Signature of External Examiner**

_____

Signature of HOD



Padmashri Annasaheb Jadhav Bhartiya Samaj Unnati Mandal's

B.N.N College of Arts, Science and Commerce, Bhiwandi.

(Self-Funded Course)

(Department of Information Technology)

CERTIFICATE

This is to certify that Mr./Miss. **Farooqui Anam Fatma Rais Ahmad** .

Roll No._____ Class: **MSc-IT** Exam Seat No. **4132807** .

has satisfactorily completed practical in **Mordern Networking** .

As laid down in the regulation of University of Mumbai for the purpose of **Semester- II Practical** Examination 2022-2023.

Date:

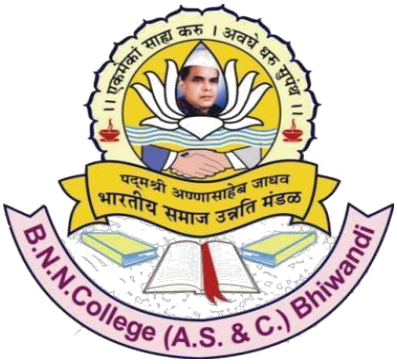
Place: **Bhiwandi**

_____ _____

In-Charge Professor **Signature of External Examiner**

_____

Signature of HOD



Padmashri Annasaheb Jadhav Bhartiya Samaj Unnati Mandal's

B.N.N College of Arts, Science and Commerce, Bhiwandi.

(Self-Funded Course)

(Department of Information Technology)

CERTIFICATE

This is to certify that Mr./Miss. **Farooqui Anam Fatma Rais Ahmad** .

Roll No._____ Class: **MSc-IT** Exam Seat No. **4132807**

has satisfactorily completed practical in **Microservices Architecture** .

As laid down in the regulation of University of Mumbai for the purpose of **Semester- II Practical** Examination 2022-2023.

Date:

Place: **Bhiwandi**

_____ _____

In-Charge Professor **Signature of External Examiner**

_____

Signature of HOD

Padmashri Annasaheb Jadhav Bhartiya Samaj Unnati Mandal's

B.N.N College of Arts, Science and Commerce, Bhiwandi.

(Self-Funded Course)

(Department of Information Technology)

CERTIFICATE

This is to certify that Mr./Miss. **Farooqui Anam Fatma Rais Ahmad** --

Roll No._____ Class: **MSc-IT** Exam Seat No. **4132807** .

has satisfactorily completed practical in **Image Processing** .

As laid down in the regulation of University of Mumbai for the purpose of **Semester- II Practical** Examination 2022-2023.

Date:

Place: **Bhiwandi**

_____ _____

In-Charge Professor **Signature of External Examiner**

_____

Signature of HOD

BIG DATA ANALYTICS

INDEX

PRACTICAL 1: Install, Configure and Run Hadoop and HDFS.

Step 1: First Downlaod java jdk.

1)Download Link: https://www.oracle.com/in/java/technologies/javase/javase8-archive-downloads.html

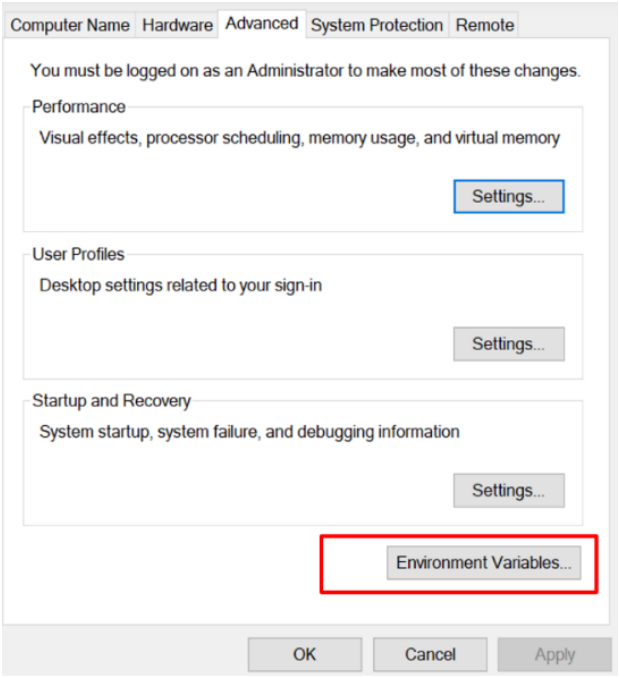
2)Open cmd and type -> javac -version

Step 2: download Hadoop **hadoop-3.3.0.tar.gz** https://hadoop.apache.org/release/3.3.0.html

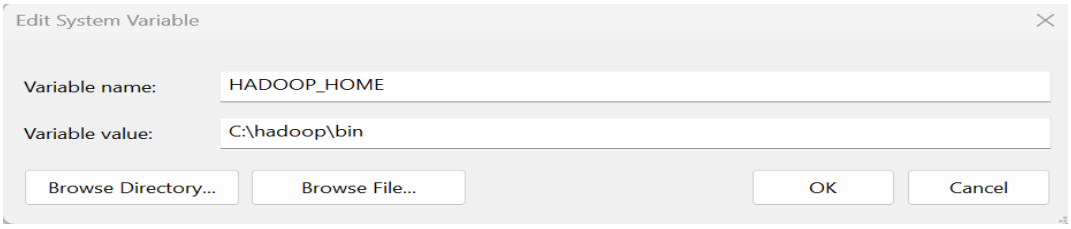
- right click .rar.gz file and extract to C:\Hadoop-3.3.0\

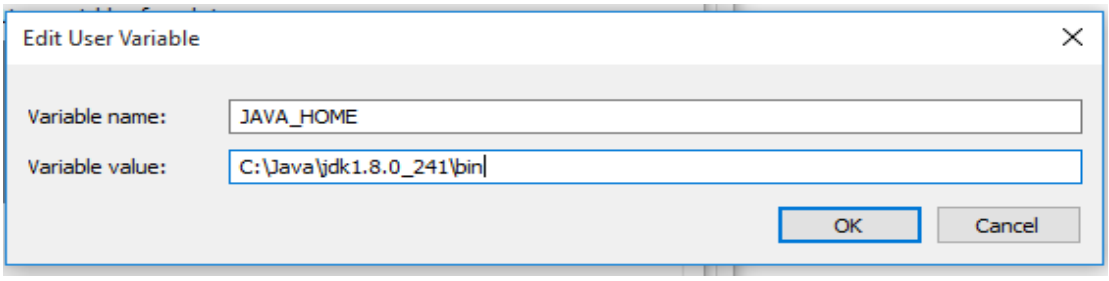
Step 3: Set Environment variables

- Set the path JAVA_HOME Environment variable
- Set the path HADOOP_HOME Environment variable

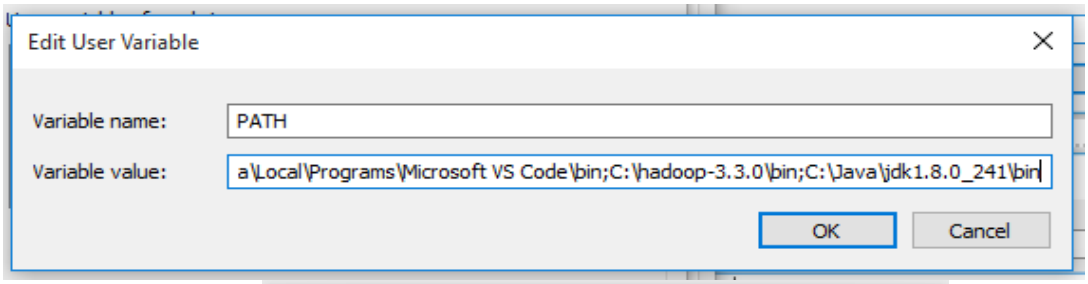


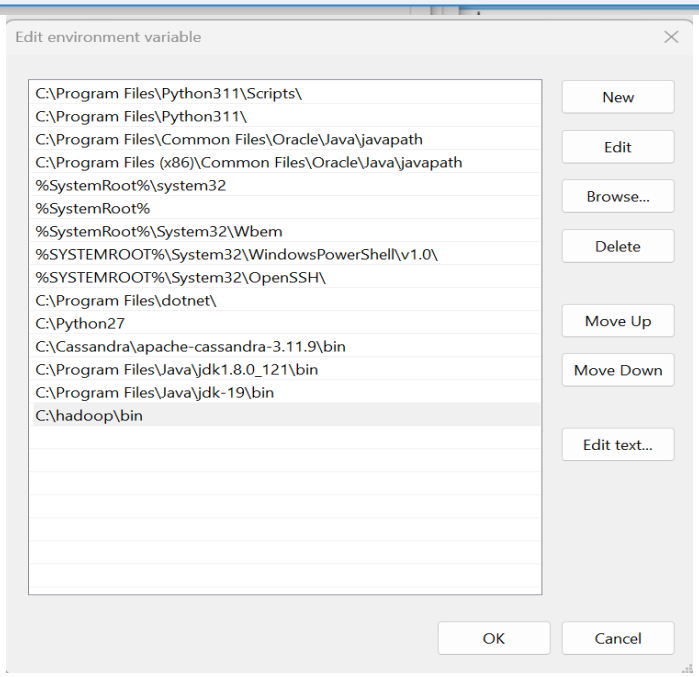
- Click on New to both user variables and system variables.





- Click on user variable -> path -> edit-> add path for Hadoop and java upto ‘bin





Step 4: Configurations

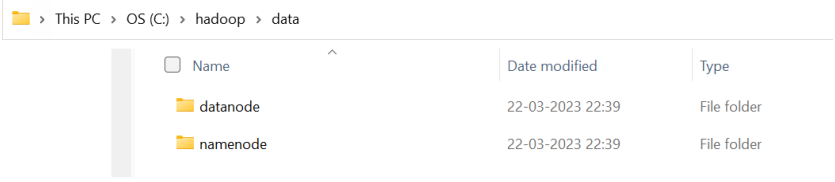
- Edit file C:\hadoop\etc\hadoop\core-site.xml, paste the xml code in folder and save:

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

- Edit file C:\hadoop-3.3.0\etc\hadoop\mapred-site.xml, paste the xml code and save:

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

- Create folder “data” under “C:\Hadoop-3.3.0”
- Create folder “datanode” under “C:\Hadoop-3.3.0\data”
- Create folder “namenode” under “C:\Hadoop-3.3.0\data”



- Edit file C:\Hadoop-3.3.0/etc/hadoop/hdfs-site.xml, paste xml code and save this file.

```
<configuration>
<property>
     <name>dfs.replication</name>
     <value>1</value>
  </property>
<property>
     <name>dfs.namenode.name.dir</name>
     <value>/hadoop-3.3.0/data/namenode</value>
  </property>
  <property>
     <name>dfs.datanode.data.dir</name>
     <value>/hadoop-3.3.0/data/datanode</value>
  </property>
 </configuration>
```

- Edit file C:/Hadoop-3.3.0/etc/hadoop/yarn-site.xml, paste xml code and save this file.

```
<configuration>
  <property>
          <name>yarn.nodemanager.aux-services</name>
          <value>mapreduce_shuffle</value>
  </property>

  <property>
         <name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
          <value>org.apache.hadoop.mapred.ShuffleHandler</value>
  </property>
  <property>
             <name>yarn.resourcemanager.address</name>
             <value>127.0.0.1:8032</value>
  </property>
  <property>
             <name>yarn.resourcemanager.scheduler.address</name>
             <value>127.0.0.1:8030</value>
   </property>
   <property>
             <name>yarn.resourcemanager.resource-tracker.address</name>
             <value>127.0.0.1:8031</value>
    </property>
</configuration>
```

Step 5: Edit file C:/Hadoop-3.3.0/etc/hadoop/hadoop-env.cmd
Find "JAVA_HOME=%JAVA_HOME%" and replace it as
set JAVA_HOME="C:\java\jdk1.8.0_121"

Step 6: Download "redistributable" package. Download and run VC_redist.x64.exe

Step 7: Hadoop Configurations

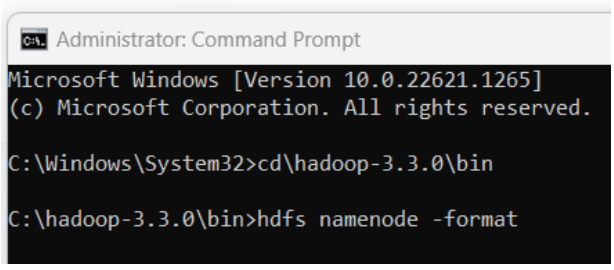
- Download bin folder from

https://github.com/s911415/apache-hadoop-3.1.0-winutils

- Copy the bin folder to c:\hadoop-3.3.0. Replace the existing bin folder.
- copy "hadoop-yarn-server-timelineservice-3.0.3.jar" from ~\hadoop-3.0.3\share\hadoop\yarn\timelineservice to ~\hadoop-3.0.3\share\hadoop\yarn folder.

Step 8: Format the NameNode

- Open cmd 'Run as Administrator' and type command "hdfs namenode –format"




```
2023-03-07 21:31:34,685 INFO namenode.FSImageFormatProtobuf: Saving image file C:\hadoop-3.3.0\data\namenode\current\fsi
mage.ckpt_0000000000000000000 using no compression
2023-03-07 21:31:34,844 INFO namenode.FSImageFormatProtobuf: Image file C:\hadoop-3.3.0\data\namenode\current\fsimage.ck
pt_0000000000000000000 of size 400 bytes saved in 0 seconds .
2023-03-07 21:31:34,860 INFO namenode.NNStorageRetentionManager: Going to retain 1 images with txid >= 0
2023-03-07 21:31:34,869 INFO namenode.FSImage: FSImageSaver clean checkpoint: txid=0 when meet shutdown.
2023-03-07 21:31:34,870 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at DESKTOP-OL8EULH/192.168.1.19
************************************************************/
```

Step 9: Testing

– Open cmd 'Run as Administrator' and change directory to C:\Hadoop-3.3.0\sbin

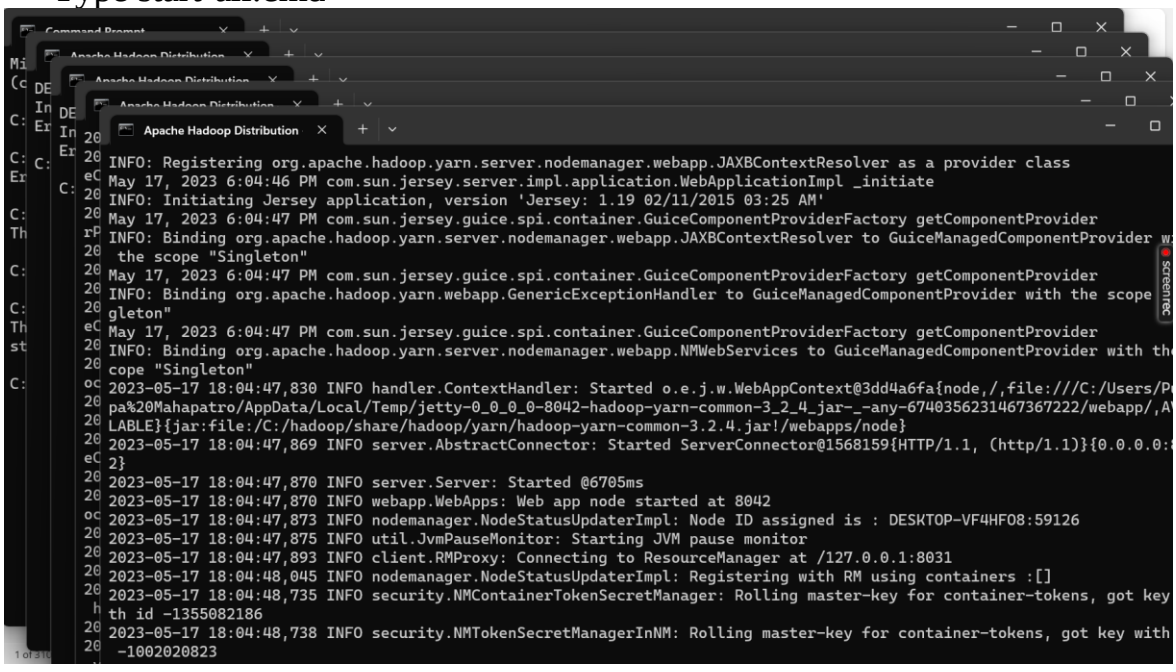
– type start-all.cmd

OR

- type start-dfs.cmd

– type start-yarn.cmd

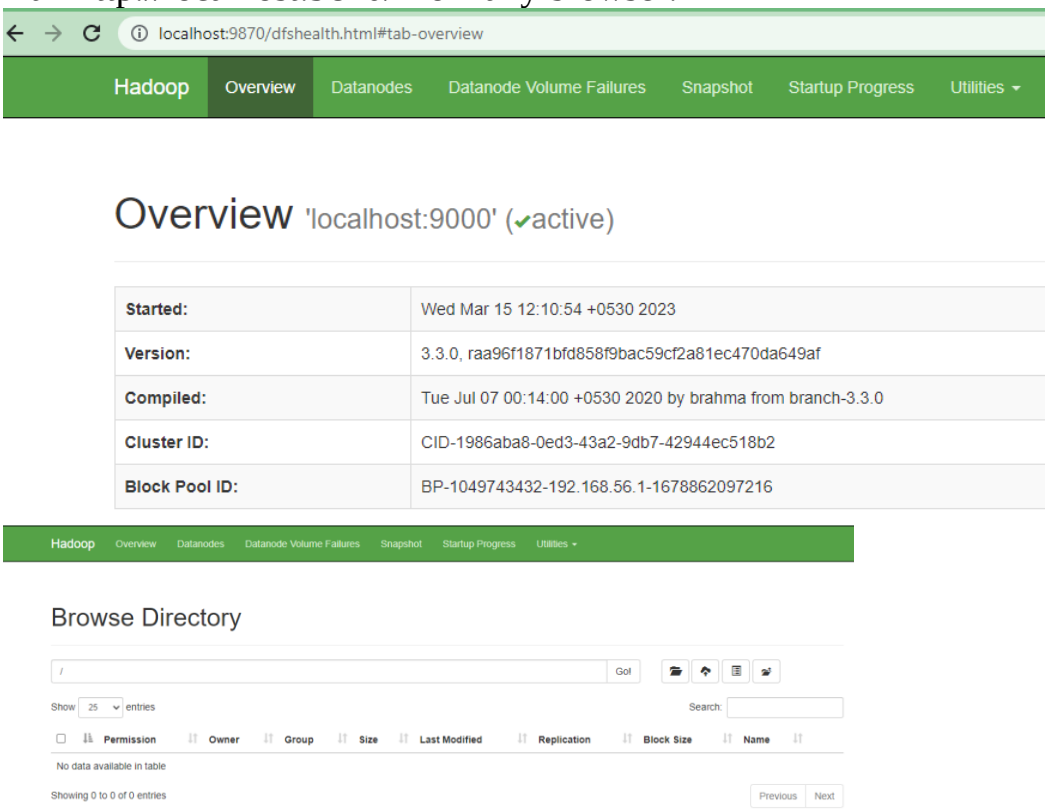
- Type start-all.cmd



- You will get 4 more running threads for Datanode, namenode, resouce manager and node manager.
- Type JPS command to start-all.cmd command prompt, you will get following output.

```
C:\hadoop-3.3.0\sbin>jps
5632 Jps
7572 DataNode
3752 ResourceManager
7992 NameNode
8028 NodeManager
```

- Run http://localhost:9870/ from any browser.

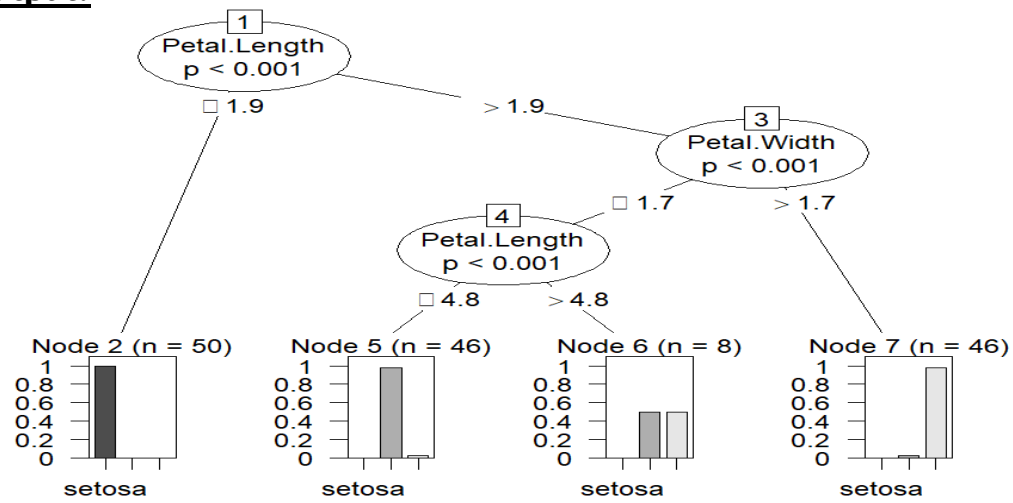


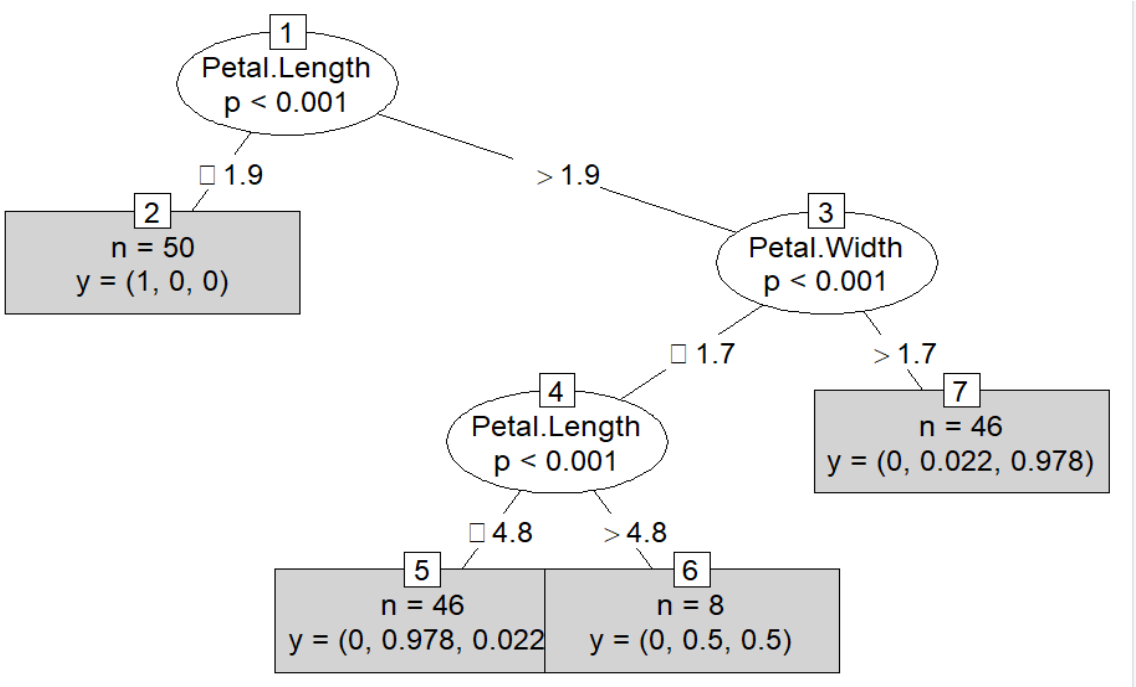
PRACTICAL 2: Implement Decision tree classification techniques

Code:

```
install.packages("party")
library("party")
print(head(readingSkills))
library("party")
str(iris)
iris_ctree<- ctree(Species ~ Sepal.Length + Sepal.Width +Petal.Length + Petal.Width, data=iris)
print(iris_ctree)
plot(iris_ctree)
plot(iris_ctree, type="simple")
```

Output:



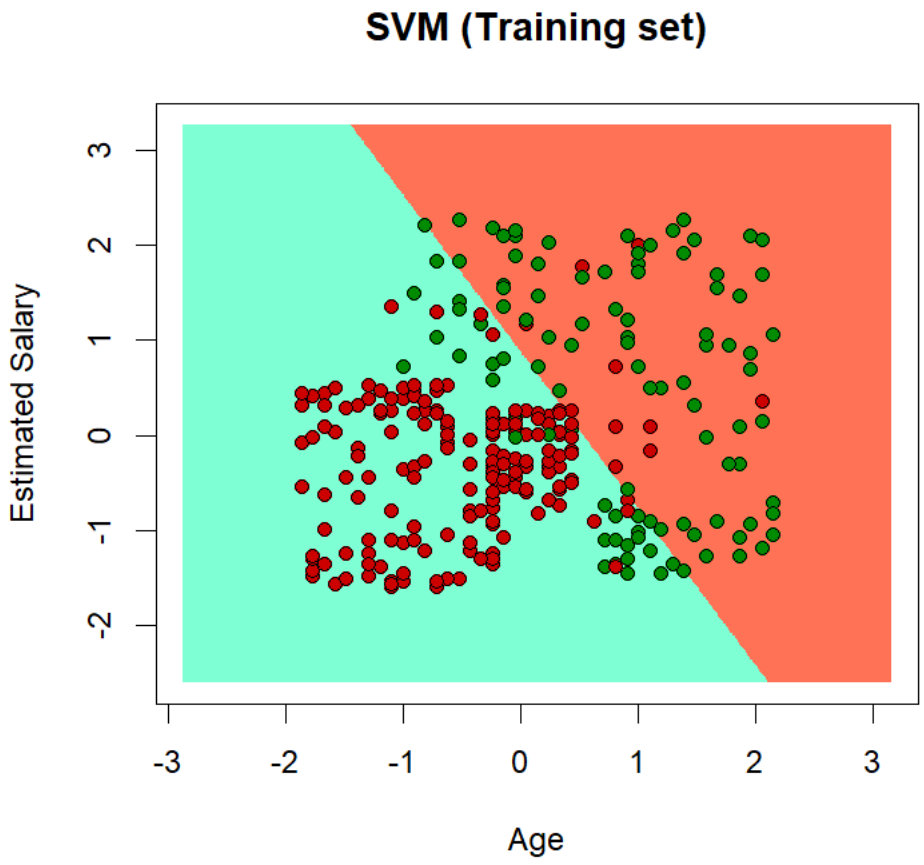


PRACTICAL 3: Classification using SVM

Code:

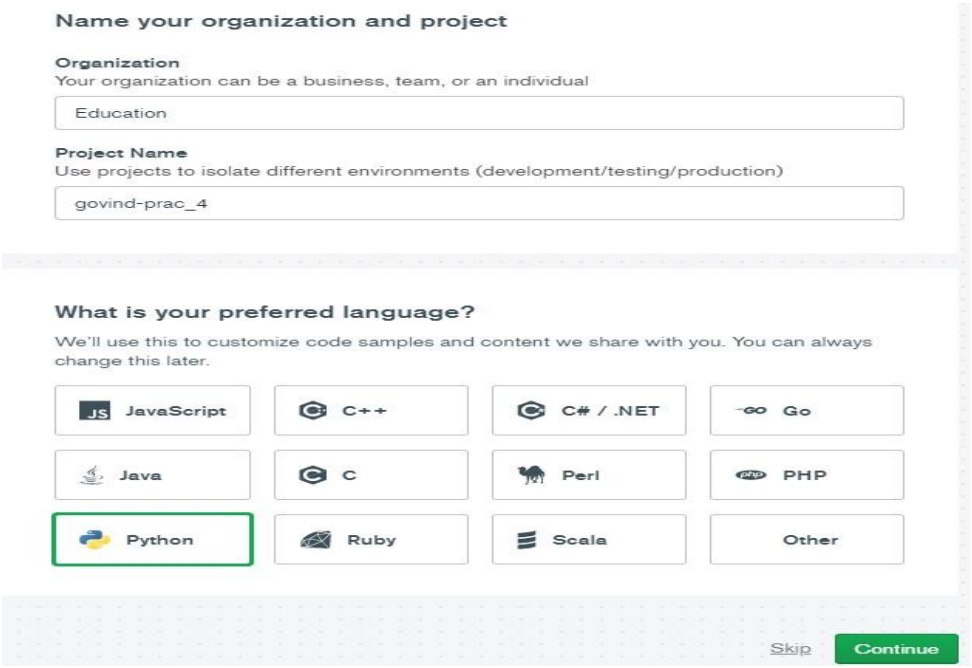
```
dataset = read.csv('Social_Network_Ads.csv')
dataset = dataset[3:5]

install.packages('caTools')
library(caTools)

set.seed(123)
split = sample.split(dataset$Purchased, SplitRatio = 0.75)
training_set = subset(dataset, split == TRUE)
test_set = subset(dataset, split == FALSE)
training_set[-3] = scale(training_set[-3])
test_set[-3] = scale(test_set[-3])

install.packages('e1071')
library(e1071)

classifier = svm(formula = Purchased ~ .,
                 data = training_set,
                 type = 'C-classification',
                 kernel = 'linear')
classifier

y_pred = predict(classifier, newdata = test_set[-3])
y_pred
cm = table(test_set[, 3], y_pred)
cm

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'))
```

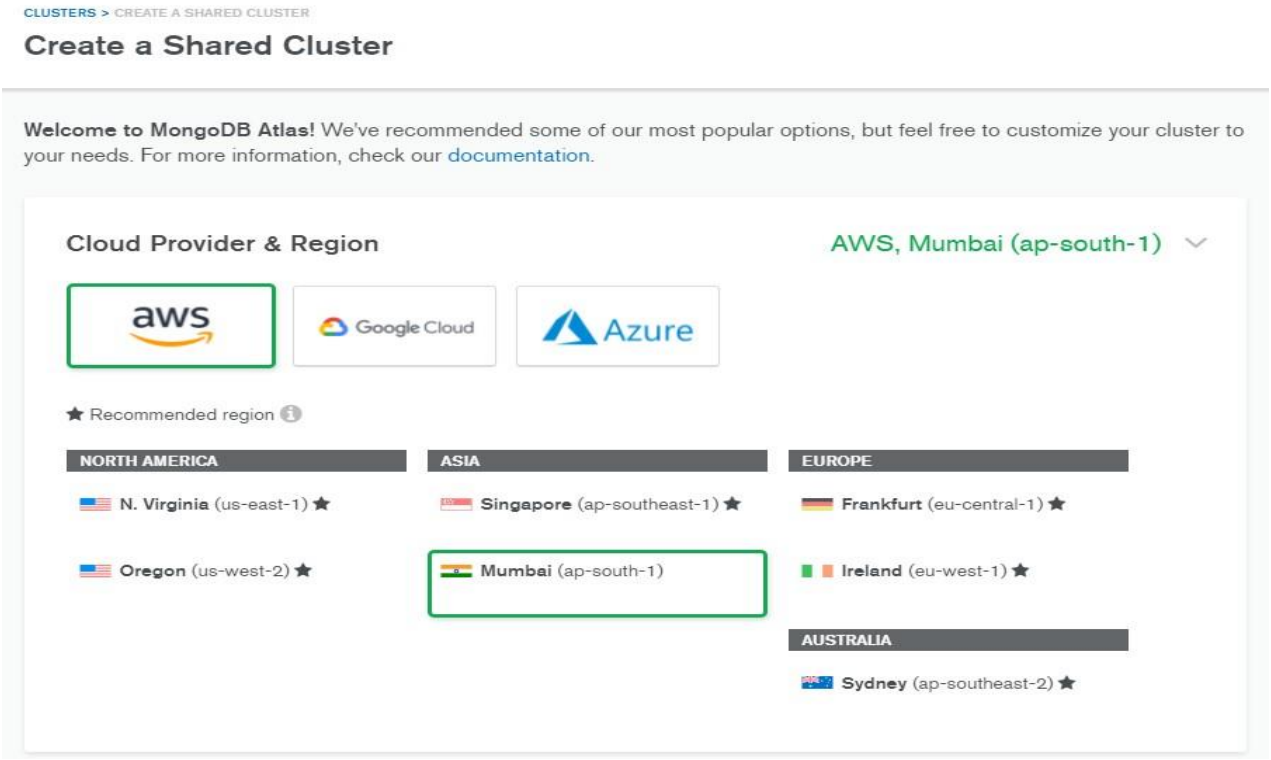
Output:



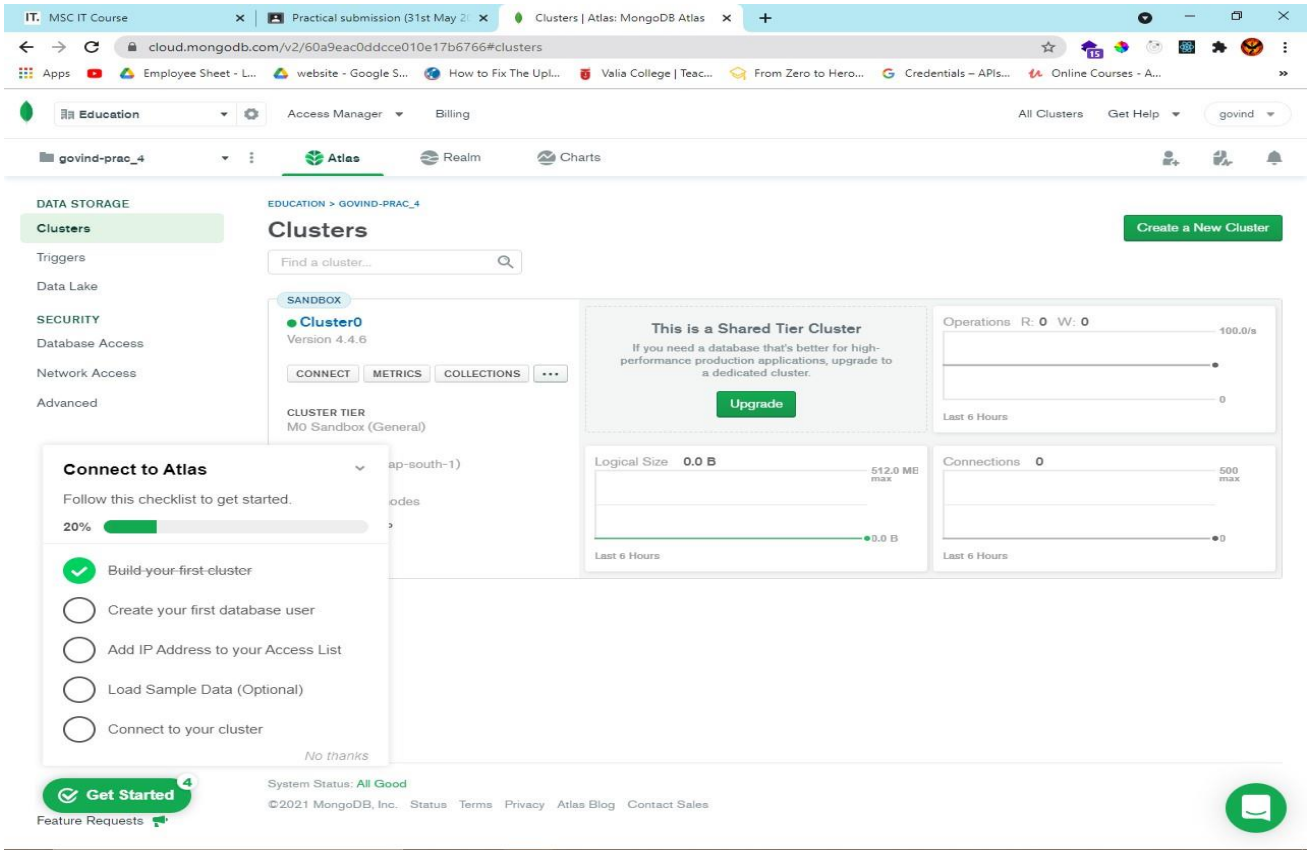
PRACTICAL 4: Implement an application that stores big data in base / MongoDB and manipulate it using R / Python H

Step 1 : Sign up and create a cluster.





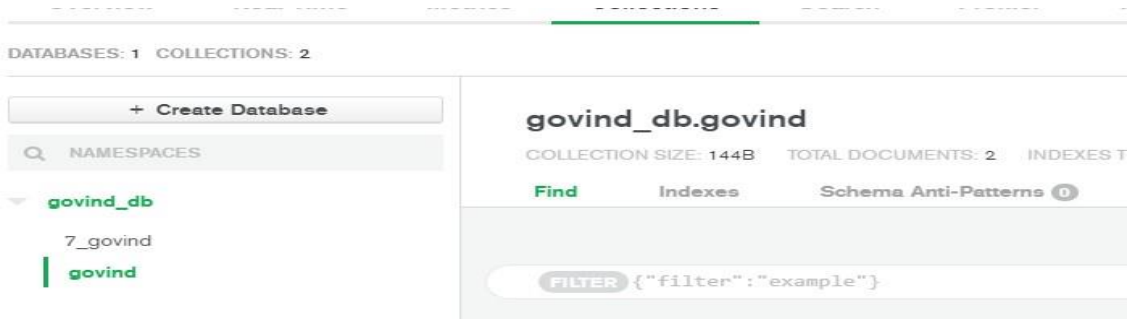
This is the home page of mongoDB Atlas.



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

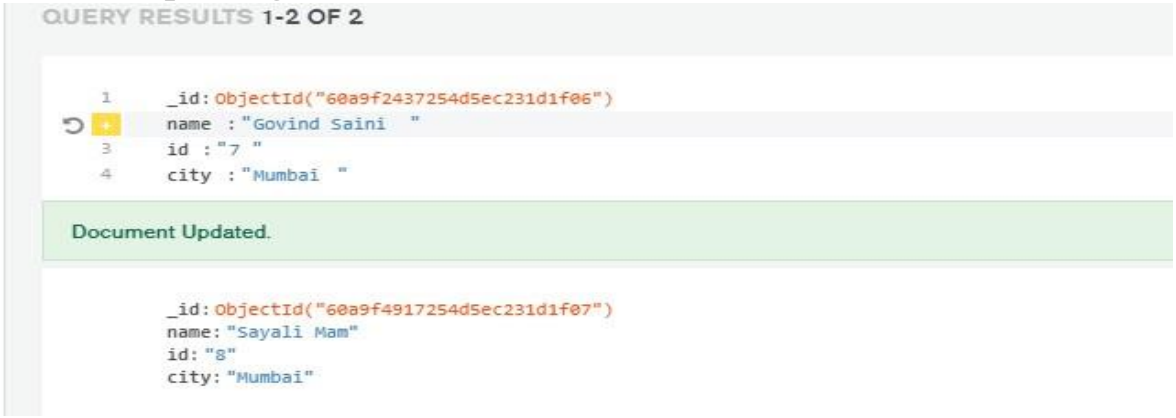
Step 3 : Click on 'Add My Own Data' to create a database.

Step 4 : Click on insert document to add records.



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

Perform updating data



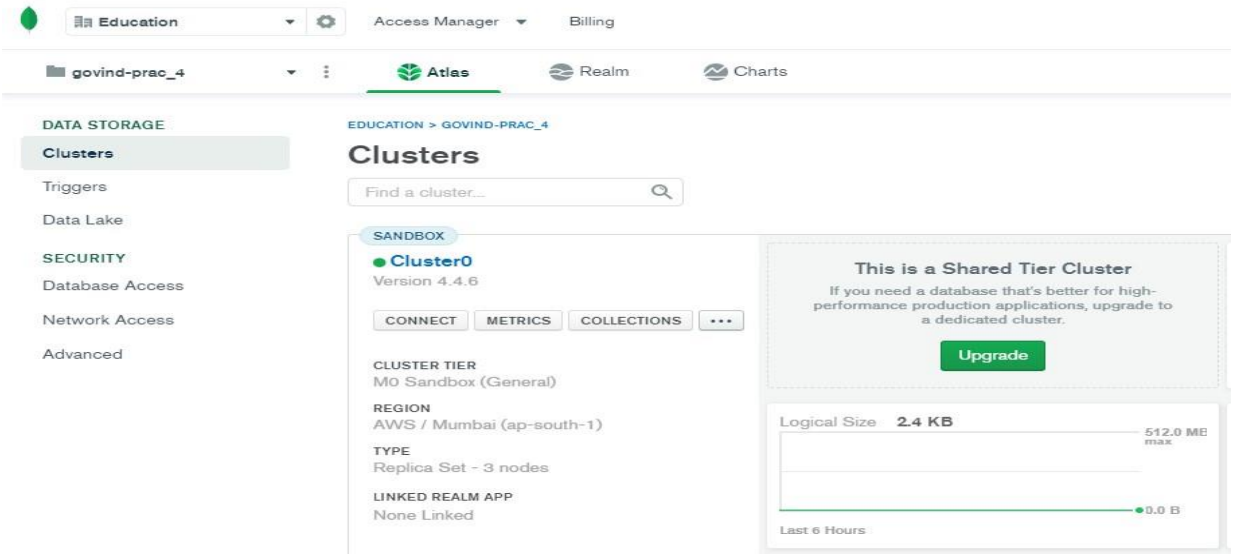
Performing deleting data

Performing Insert data

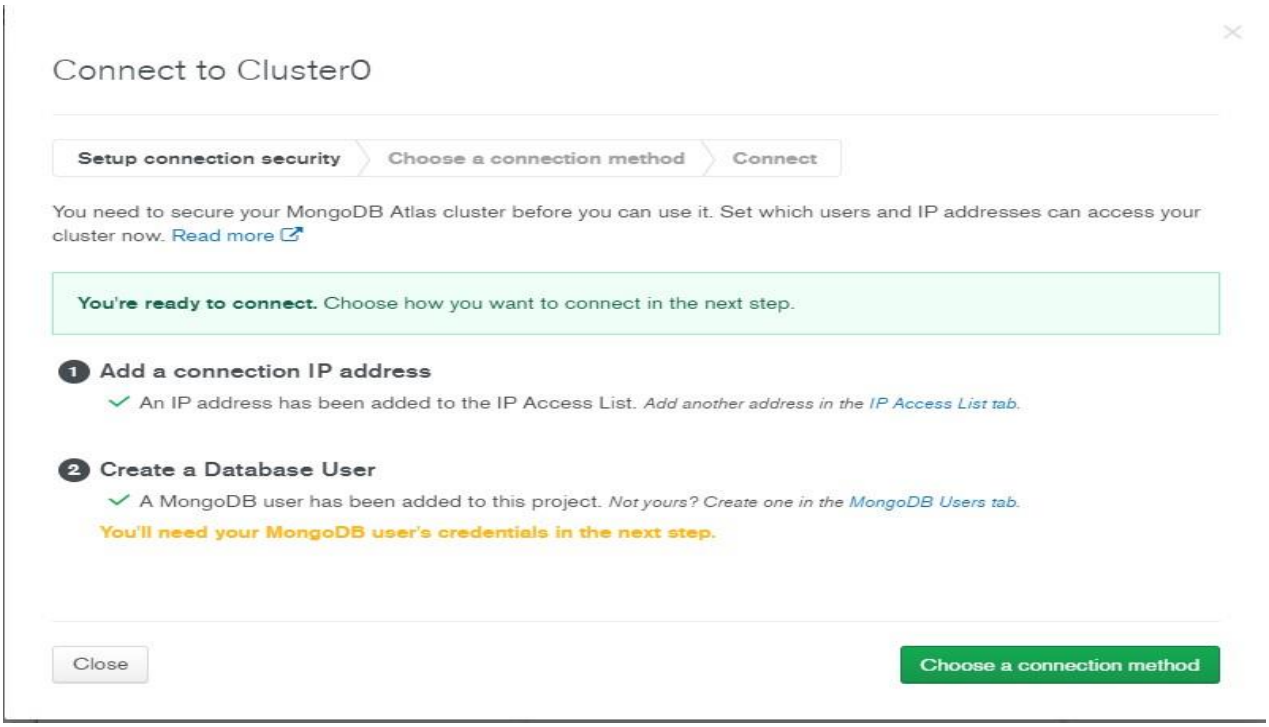
QUERY RESULTS 1-2 OF 2

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

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

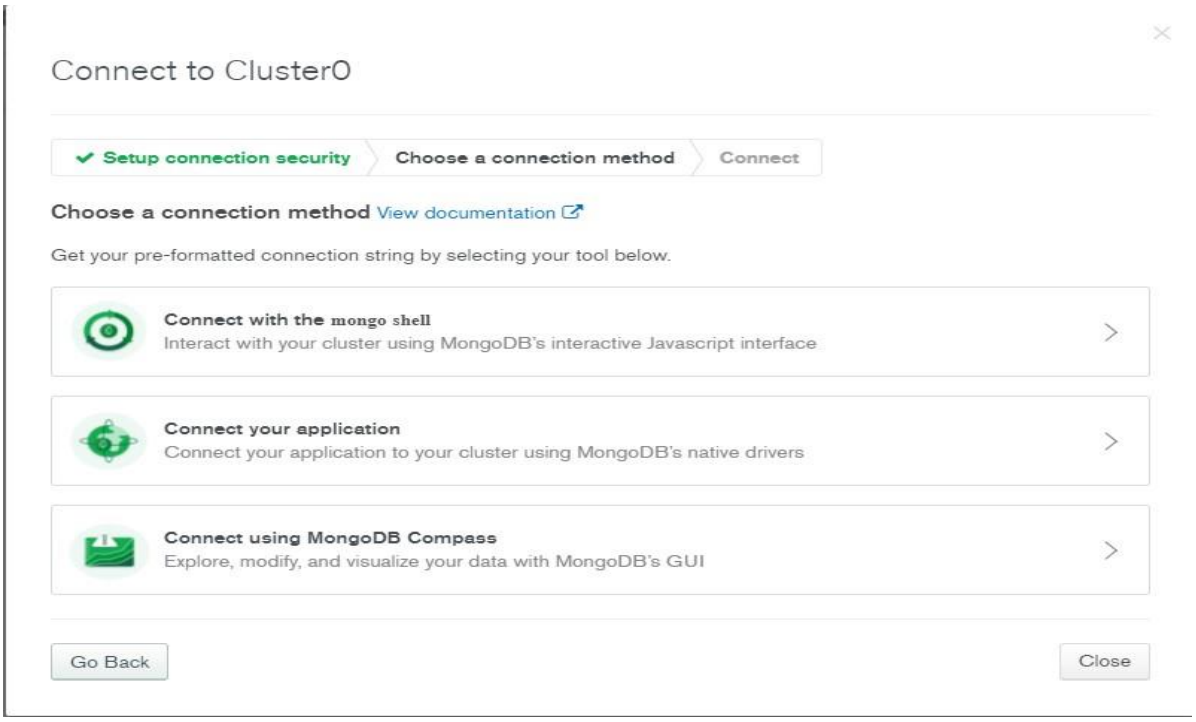
Step 5 : To start with the connection click on Overview, and then click on Connect.



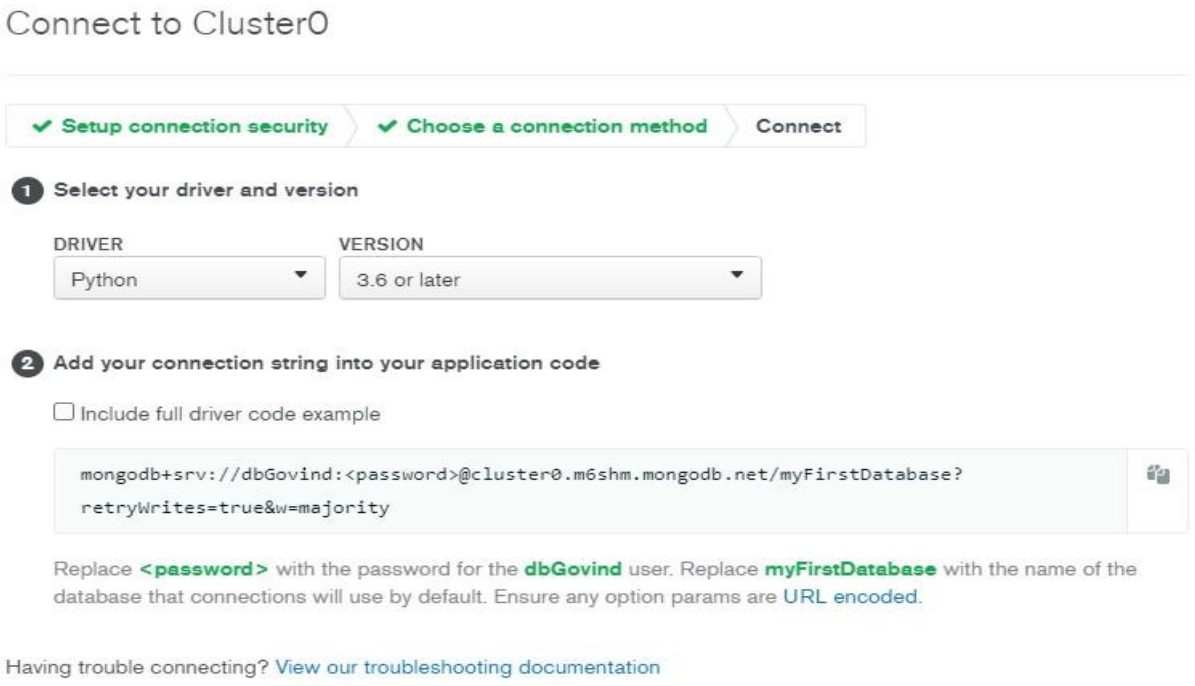
Step 6 : Select on add your current IP and create a MongoDB user.



Step 7 : Click on 'Connect your application'.



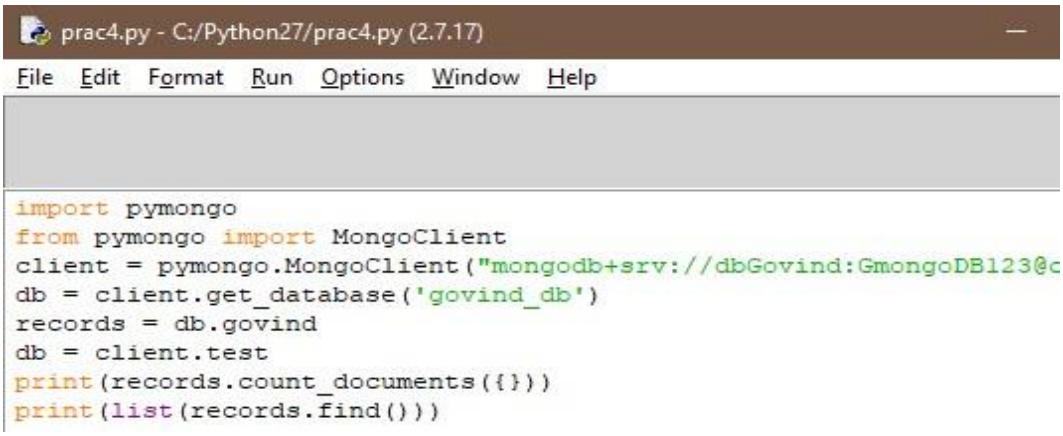
Step 8 : Select the driver as 'Python' and version as '3.6 or later'. (Select the version as 3.6 or later only if your Python's version is 3.6 or later.)



Step 9 : on command prompt, install pymongo by following commands :

- **pip install pymongo**
- **pip install pymongo [srv]**

Step 10 : Write the code given below in a Python file.



Output:

```
======================== RESTART: C:/Python27/prac.py ========================
2[{"_id":{"$oid":"60a9ff027254d5ec231d1f0b"},"name":"Govind Saini","id":" 7","city"
:"Mumbai"}{"_id":{"$oid":"60a9ff3a7254d5ec231d1f0c"},"name":"Sohrab Sir","id":" 5",
"city":"Mumbai"}]
>>>
```

PRACTICAL 5: Write program in R of Naive bayes theorem

Code:

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

split <- sample.split(iris, SplitRatio = 0.7)
train_cl <- subset(iris, split == "TRUE")
test_cl <- subset(iris, split == "FALSE")

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

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

y_pred <- predict(classifier_cl, newdata = test_cl)

cm <- table(test_cl$Species, y_pred)
cm
confusionMatrix(cm)
```

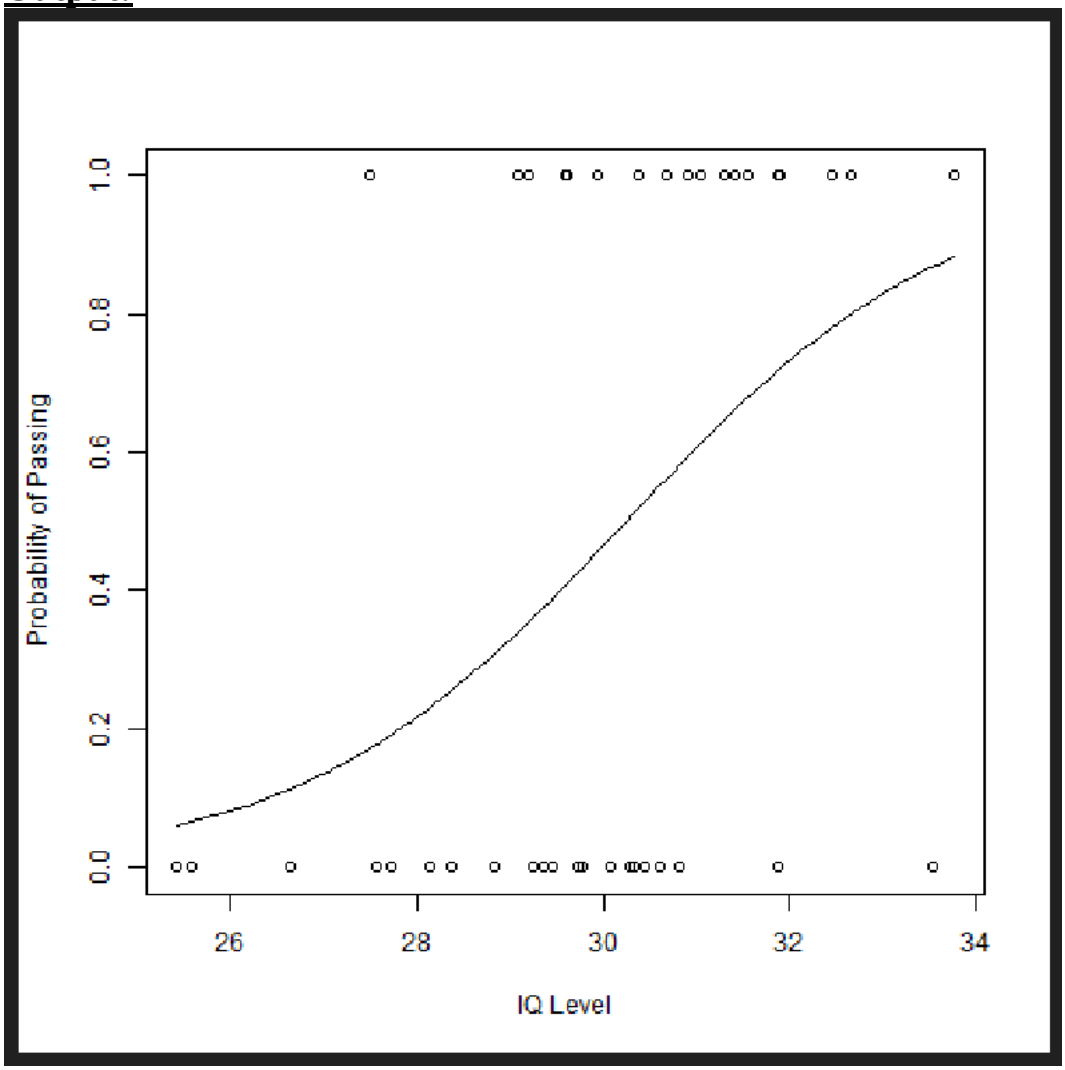
Output:

```
Confusion Matrix and Statistics

            y_pred
             setosa versicolor virginica
  setosa         20          0         0
  versicolor      0         17         3
  virginica       0          0        20

Overall Statistics

               Accuracy : 0.95
                 95% CI : (0.8608, 0.9896)
    No Information Rate : 0.3833
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.925

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: setosa Class: versicolor Class: virginica
Sensitivity                 1.0000            1.0000           0.8696
Specificity                 1.0000            0.9302           1.0000
Pos Pred Value              1.0000            0.8500           1.0000
Neg Pred Value              1.0000            1.0000           0.9250
Prevalence                  0.3333            0.2833           0.3833
Detection Rate              0.3333            0.2833           0.3333
Detection Prevalence        0.3333            0.3333           0.3333
Balanced Accuracy           1.0000            0.9651           0.9348
```

PRACTICAL 6: Write a program showing implementation of R egression model.

```
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)
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)
dev.off()
```

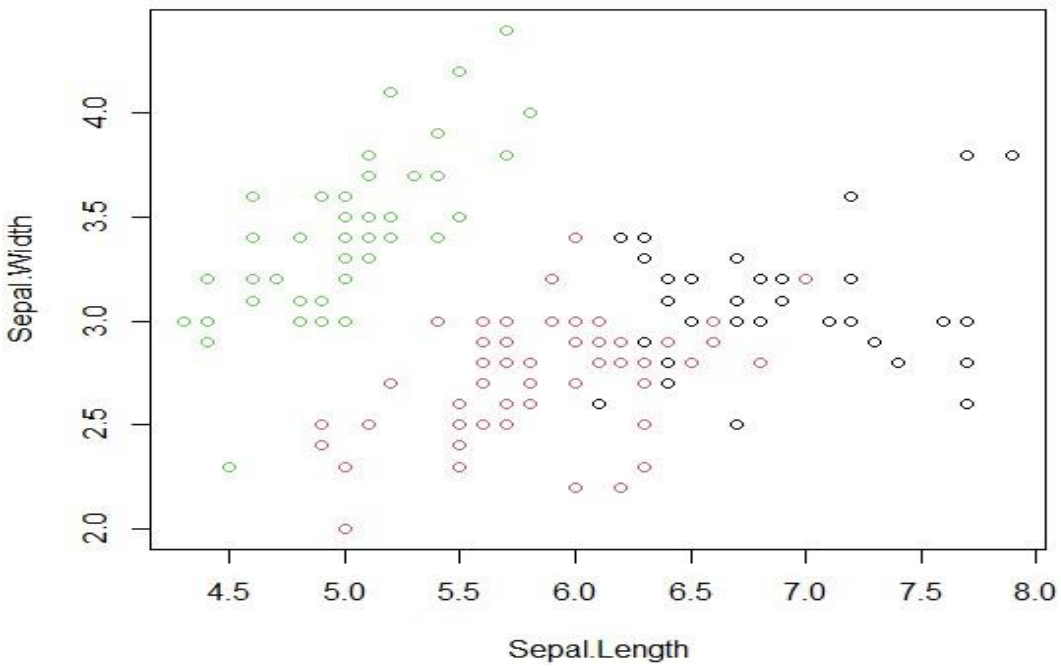
Output:



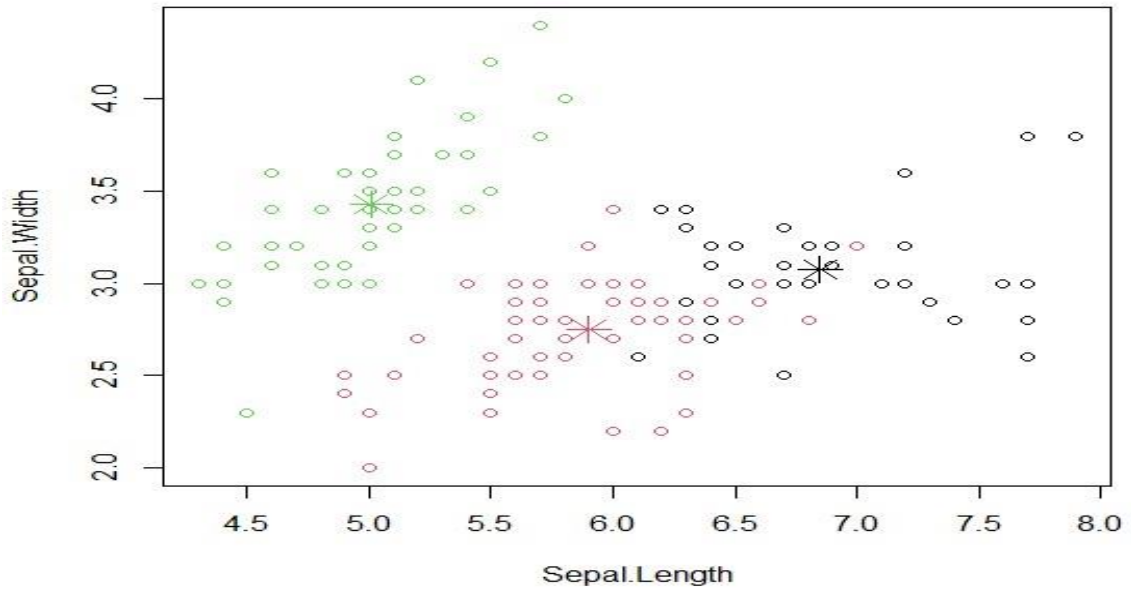
PRACTICAL 7: Write a program showing clustering.

Code:

```
newiris <- iris
newiris$Species <- NULL
(kc <- kmeans(newiris,3))
table(iris$Species, kc$cluster)
plot(newiris[c("Sepal.Length","Sepal.Width")], col=kc$cluster)
```




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

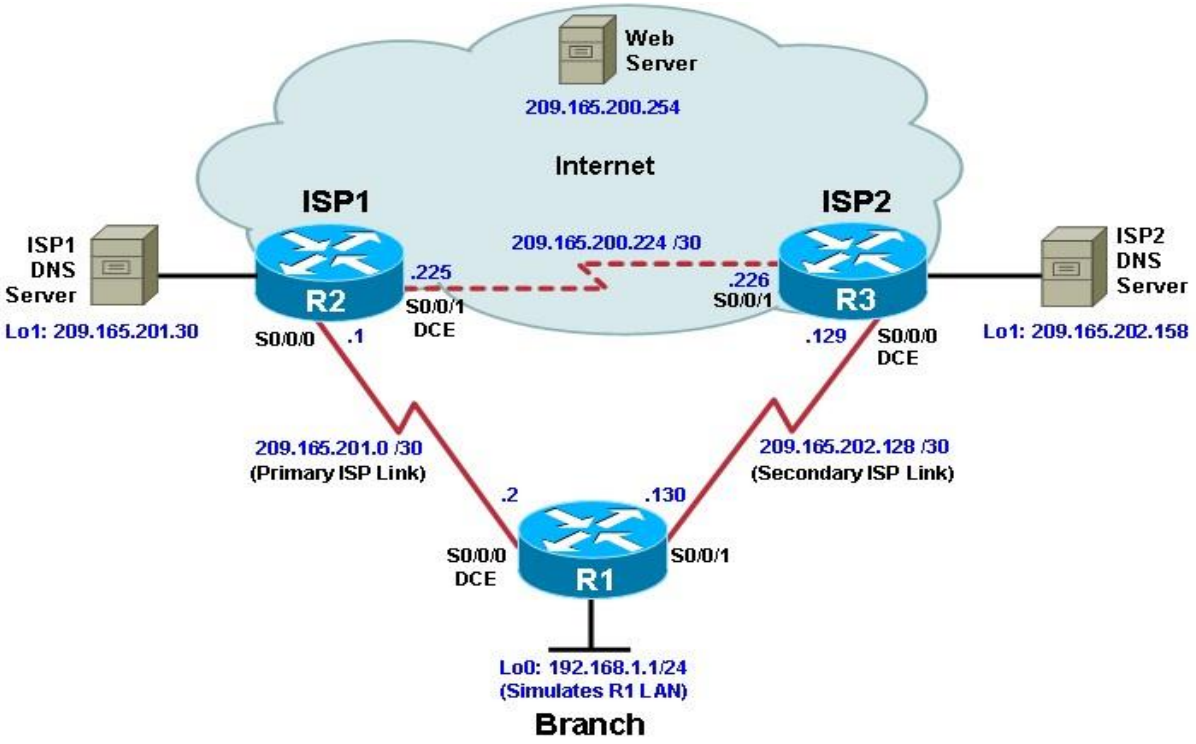


MODERN NETWORKING

INDEX

PRACTICAL 1: Configure IP SLA Tracking and Path Control Topology

Topology



Objectives

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

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

Router(R1)

```
conf t
hostname R1
interface Loopback 0
description R1 LAN
ip address 192.168.1.1 255.255.255.0
exit

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

conf t
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

```
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.

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

Step 2: Verify server reachability.

R1# tclsh
foreach address { 209.165.200.254 209.165.201.30 209.165.202.158} {ping $address source 192.168.1.1 }

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

foreach address { 209.165.200.254 209.165.201.30 209.165.202.158} { trace $address source 192.168.1.1 }

```
Type escape sequence to abort.
Tracing the route to 209.165.200.254

  1 209.165.201.1 24 msec 24 msec 16 msec
Type escape sequence to abort.
Tracing the route to 209.165.201.30

  1 209.165.201.1 16 msec 24 msec 24 msec
Type escape sequence to abort.
Tracing the route to 209.165.202.158

  1 209.165.201.1 24 msec 24 msec 32 msec
  2 209.165.200.226 28 msec 44 msec 72 msec
```

Step 3: Configure IP SLA probes.

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

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

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

```
R1(tcl)#show ip sla configuration 11
IP SLAs, Infrastructure Engine-II.
Entry number: 11
Owner:
Tag:
Type of operation to perform: icmp-echo
Target address/Source address: 209.165.201.30/0.0.0.0
Type Of Service parameter: 0x0
Request size (ARR data portion): 28
Operation timeout (milliseconds): 5000
Verify data: No
Vrf Name:
Schedule:
   Operation frequency (seconds): 10  (not considered if
   Next Scheduled Start Time: Start Time already passed
   Group Scheduled : FALSE
   Randomly Scheduled : FALSE
   Life (seconds): Forever
   Entry Ageout (seconds): never
   Recurring (Starting Everyday): FALSE
   Status of entry (SNMP RowStatus): Active
```

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

R1# show ip sla statistics

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

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

d. Although not actually required because IP SLA session 11 alone could provide the desired fault tolerance, create a second probe, 22, to test connectivity to the second DNS server located on router ISP2. You can copy and paste the following commands on R1.

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

```
R1#show ip sla configuration 22
IP SLAs, Infrastructure Engine-II.
Entry number: 22
Owner:
Tag:
Type of operation to perform: icmp-echo
Target address/Source address: 209.165.202.158/0.0.0.0
Type Of Service parameter: 0x0
Request size (ARR data portion): 28
Operation timeout (milliseconds): 5000
Verify data: No
```

R1# show ip sla statistics 22

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

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

Step 4: Configure tracking options.

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

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

Gateway of last resort is 209.165.201.1 to network 0.0.0.0
```

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

R1(config)# track 1 ip sla 11 reachability

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.

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

R1# debug ip routing
R1# conf t
R1(config)# ip route 0.0.0.0 0.0.0.0 209.165.201.1 2 track 1

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

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

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

g. Verify the routing table again.

R1# show ip route

Step 5: Verify IP SLA operation.

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

b. Verify the routing table.
R1# show ip route

c. Verify the IP SLA statistics.
R1# show ip sla statistics

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

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

d. Initiate a trace to the web server from the internal LAN IP address.
R1# trace 209.165.200.254 source 192.168.1.1

e. To examine the routing behavior when connectivity to the ISP1 DNS is restored, re-enable the DNSaddress on ISP1 (R2) by issuing the no shutdown command on the loopback 1 interface on ISP2.
ISP1(config-if)# no shutdown

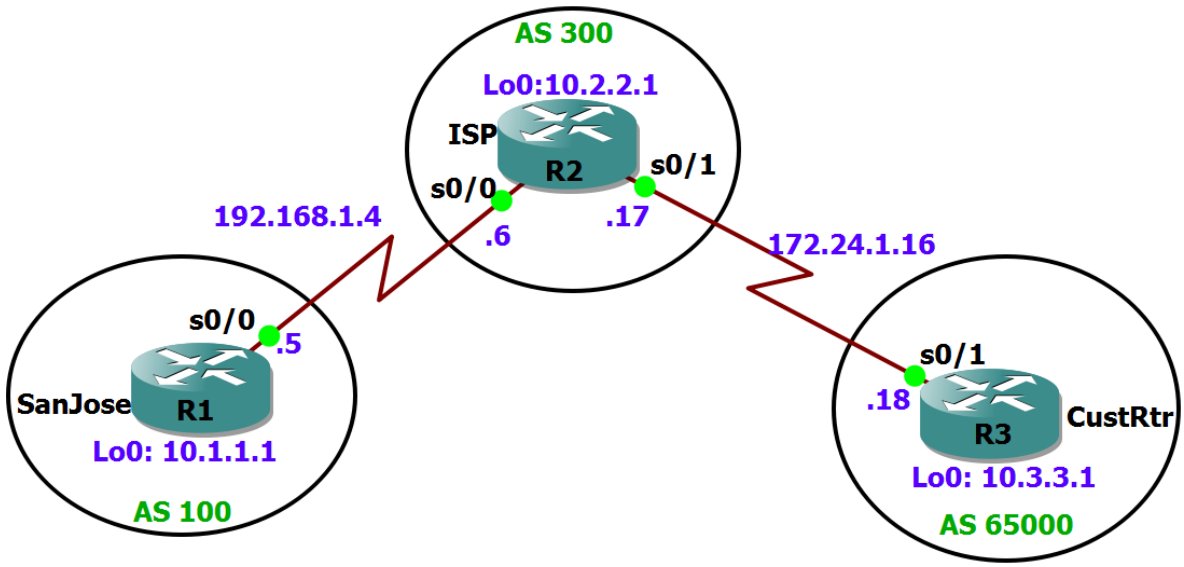
f. Again examine the IP SLA statistics.
R1# show ip sla statistics

g. Verify the routing table.
R1# show ip route

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

Gateway of last resort is 209.165.201.1 to network 0.0.0.0

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

PRATICAL 2: Using the AS_PATH Attribute

Topology



Objectives

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

Step 0: Suggested starting configurations.

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

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

Step 1: Configure interface addresses.

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

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

```
ISP(config)# interface Loopback0
```

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

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

c. Use ping **to test the connectivity between the directly connected routers.**

Step 2: Configure BGP.

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

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

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

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

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

```
ISP# show ip bgp neighbors
```

Step 3: Remove the private AS.

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

```
SanJose#show ip route
```

b. Ping the 10.3.3.1 address from SanJose.

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

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

SanJose# ping 10.3.3.1 source 10.1.1.1

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,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

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

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

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

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

Note: The **clear ip bgp * soft** command can also be used to force each router to resend its BGP table.

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

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

```
SanJose# ping 10.3.3.1 source lo0
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.3.3.1, timeout is 2 seconds:
Packet sent with a source address of 10.1.1.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/28/32 ms
```

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

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

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

Step 4: Use the AS_PATH attribute to filter routes.

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 .*
```

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
```

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.

```
ISP# clear ip bgp *
ISP# show ip route
```

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

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

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

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

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

ISP#

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

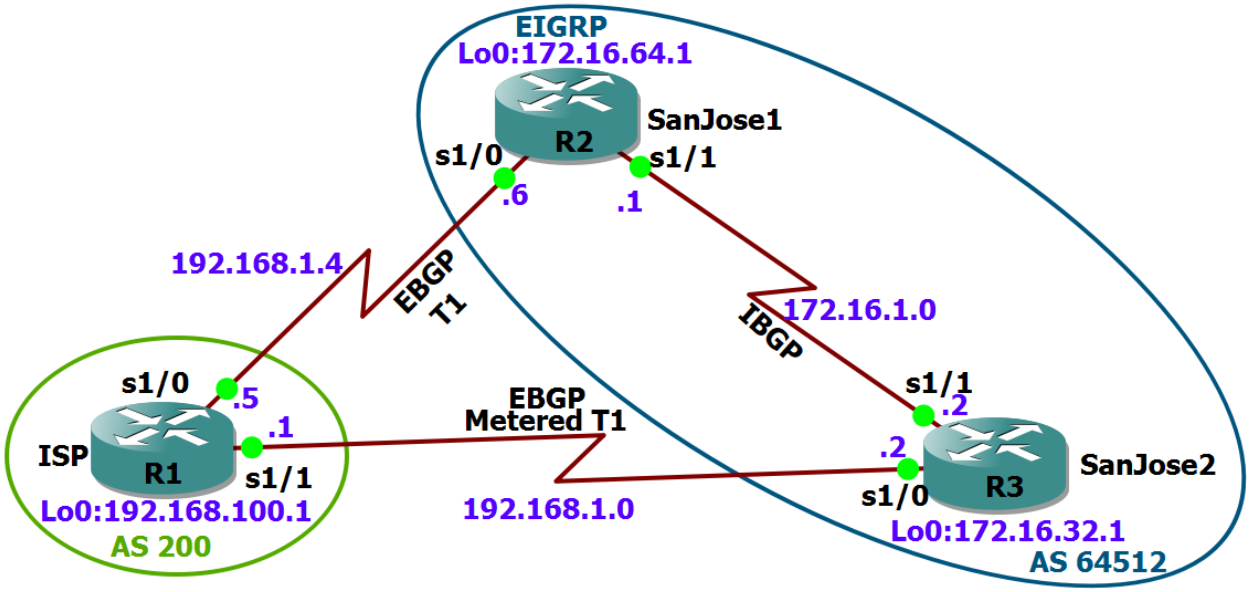
f. Run the following Tcl script on all routers to verify whether there is connectivity. All pings from ISP should be successful. SanJose should not be able to ping the CustRtr loopback 10.3.3.1 or the WAN link 172.24.1.16/30. CustRtr should not be able to ping the SanJose loopback 10.1.1.1 or the WAN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

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

```
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/40/64 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.2.2.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/4/4 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.3.3.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 16/32/48 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.1.5, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 16/28/40 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.1.6, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 48/59/64 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.24.1.17, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 48/56/68 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.24.1.18, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 16/31/48 ms
```

PRATICAL 3:Configuring IBGP and EBGP Sessions, Local Preference, and MED

Topology



Objectives

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

Step 0: Suggested starting configurations.

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

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

Step 1: Configure interface addresses.

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

Router R1 (hostname ISP)

```
ISP(config)# interface Loopback0
ISP(config-if)# ip address 192.168.100.1 255.255.255.0
ISP(config-if)# exit
ISP(config)# interface Serial1/0
ISP(config-if)# ip address 192.168.1.5 255.255.255.252
ISP(config-if)# clock rate 128000
ISP(config-if)# no shutdown
ISP(config-if)# exit
ISP(config)# interface Serial1/1
ISP(config-if)# ip address 192.168.1.1 255.255.255.252
ISP(config-if)# no shutdown
```

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

Router R2 (hostname SanJose1)

```
SanJose1(config)# interface Loopback0
SanJose1(config-if)# ip address 172.16.64.1 255.255.255.0
SanJose1(config-if)# exit
SanJose1(config)# interface Serial1/0
SanJose1(config-if)# ip address 192.168.1.6 255.255.255.252
SanJose1(config-if)# no shutdown
SanJose1(config-if)# exit
SanJose1(config)# interface Serial1/1
SanJose1(config-if)# ip address 172.16.1.1 255.255.255.0
SanJose1(config-if)# clock rate 128000
SanJose1(config-if)# no shutdown
SanJose1(config-if)# end
SanJose1#
```

Router R3 (hostname SanJose2)

```
SanJose2(config)# interface Loopback0
SanJose2(config-if)# ip address 172.16.32.1 255.255.255.0
SanJose2(config-if)# exit
SanJose2(config)# interface Serial1/0
SanJose2(config-if)# ip address 192.168.1.2 255.255.255.252
SanJose2(config-if)# clock rate 128000
SanJose2(config-if)# no shutdown
SanJose2(config-if)# exit
SanJose2(config)# interface Serial1/1
SanJose2(config-if)# ip address 172.16.1.2 255.255.255.0
SanJose2(config-if)# no shutdown
SanJose2(config-if)# end
SanJose2#
```

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

Step 2: Configure EIGRP.

Configure EIGRP between the SanJose1 and SanJose2 routers.

```
SanJose1(config)# router eigrp 1
SanJose1(config-router)# network 172.16.0.0
SanJose2(config)# router eigrp 1
SanJose2(config-router)# network 172.16.0.0
```

Step 3: Configure IBGP and verify BGP neighbors.

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

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

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

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

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

SanJose2# show ip bgp neighbors

BGP neighbor is 172.16.64.1, remote AS 64512, internal link
BGP version 4, remote router ID 172.16.64.1
BGP state = Established, up for 00:00:22
Last read 00:00:22, last write 00:00:22, hold time is 180, keepalive interval is
60 seconds

Step 4: Configure EBGP and verify BGP neighbors.

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

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

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

SanJose1(config)# ip route 172.16.0.0 255.255.0.0 null0

c. Configure SanJose1 as an EBGP peer to ISP.

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

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

SanJose1# show ip bgp neighbors

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

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

Step 5: View BGP summary output.

In Step 4, the show ip bgp neighbors command was used to verify that SanJose1 and ISP had reached the established state.

A useful alternative command is show ip bgp summary. The output should be similar to the following.

SanJose2# show ip bgp summary

BGP router identifier 172.16.32.1, local AS number 64512
BGP table version is 6, main routing table version 6
2 network entries using 288 bytes of memory

```
4 path entries using 320 bytes of memory
4/2 BGP path/bestpath attribute entries using 640 bytes of memory
1 BGP AS-PATH entries using 24 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 1272 total bytes of memory
BGP activity 2/0 prefixes, 4/0 paths, scan interval 60 secs
Neighbor V AS MsgRcvd MsgSent TblVer InQ OutQ Up/Down
State/PfxRcd
172.16.64.1 4 64512 27 26 6 0 0 00:18:15 2
192.168.1.1 4 200 10 7 6 0 0 00:01:42 1
```

Step 6: Verify which path the traffic takes.

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

ISP# clear ip bgp *

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

ISP# ping 172.16.64.1

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

ISP# ping 172.16.1.1

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

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

ISP# ping 172.16.32.1

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

ISP# ping 172.16.1.2

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

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

ISP# show ip bgp

```
ISP#show ip bgp
BGP table version is 3, local router ID is 192.168.100.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

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

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

ISP# ping 172.16.1.1 source 192.168.100.1

ISP# ping 172.16.32.1 source 192.168.100.1

ISP# ping 172.16.1.2 source 192.168.100.1

ISP# ping 172.16.64.1 source 192.168.100.1

ISP# ping

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

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

a. Issue the following commands on the ISP router.

ISP(config)# router bgp 200

ISP(config-router)# network 192.168.1.0 mask 255.255.255.252

ISP(config-router)# network 192.168.1.4 mask 255.255.255.252

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

ISP# show ip bgp

```
ISP#show ip bgp
BGP table version is 5, local router ID is 192.168.100.1
Status codes: s suppressed, d damped, h history, * valid, > best, i -
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

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

c. Verify on SanJose1 and SanJose2 that the opposite WAN link is included in the routing table.

SanJose2# show ip route

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

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

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

SanJose2# show ip bgp

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

SanJose2# show ip route

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

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

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

SanJose1# clear ip bgp *
SanJose2# clear ip bgp *

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

SanJose2# show ip bgp

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

SanJose2# show ip route

D 172.16.64.0/24 [90/2297856] via 172.16.1.1, 04:27:19, Serial0/0/1
B 192.168.100.0/24 [200/0] via 172.16.64.1, 00:00:46

j. Before configuring the next BGP attribute, restore the WAN link between ISP and SanJose3. This will change the BGP table and routing table on both routers.

For example, SanJose2's routing table shows 192.168.100.0/24 will now have a better path through ISP.

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

SanJose2# show ip route
B 192.168.100.0/24 [20/0] via 192.168.1.1, 00:01:35

Step 8: Set BGP local preference.

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

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

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

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

Step 9: Set BGP MED.

a. Examine what the return path ISP takes to reach AS 64512. Notice that the return path is different from the original path.

ISP# show ip bgp
ISP# show ip route\
B 172.16.0.0/16 [20/0] via 192.168.1.2, 00:12:45

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

SanJose2# ping
Protocol [ip]:
Target IP address: 192.168.100.1
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface: 172.16.32.1

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

```
Reply to request 4 (20 ms). Received packet has options
Total option bytes= 40, padded length=40
Record route:
(172.16.1.2)
(192.168.1.6)
(192.168.100.1)
(192.168.1.1)
(172.16.32.1) &lt;*&gt;
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
End of list
Success rate is 100 percent (5/5), round-trip min/avg/max = 20/20/24 ms
SanJose2#
```

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

```
SanJose1(config)#route-map PRIMARY_T1_MED_OUT permit 10
SanJose1(config-route-map)#set Metric 50
SanJose1(config-route-map)#exit
SanJose1(config)#router bgp 64512
SanJose1(config-router)#neighbor 192.168.1.5 route-map PRIMARY_T1_MED_OUT out
SanJose2(config)#route-map SECONDARY_T1_MED_OUT permit 10
SanJose2(config-route-map)#set Metric 75
SanJose2(config-route-map)#exit
SanJose2(config)#router bgp 64512
SanJose2(config-router)#neighbor 192.168.1.1 route-map SECONDARY_T1_MED_OUT
out
```

d. Use the clear ip bgp * soft command after issuing this new policy. Issuing the show ip bgp command as follows onSanJose1 or SanJose2 does not indicate anything about this newly defined policy.

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

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

SanJose2# ping
Protocol [ip]:
Target IP address: 192.168.100.1
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface: 172.16.32.1
Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]: record
Number of hops [9]:
Loose, Strict, Record, Timestamp, Verbose[RV]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.100.1, timeout is 2 seconds:
Packet sent with a source address of 172.16.32.1
Packet has IP options: Total option bytes= 39, padded length=40
Record route: <*>
(0.0.0.0)
(0.0.0.0) (0.0.0.0)
(0.0.0.0)
Reply to request 0 (28 ms). Received packet has options
Total option bytes= 40, padded length=40
Record route:
(172.16.1.2)
(192.168.1.6)
(192.168.100.1)
(192.168.1.5)
(172.16.1.1)
(172.16.32.1) <*>
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
End of list
Reply to request 1 (28 ms). Received packet has options
Total option bytes= 40, padded length=40
Record route:
(172.16.1.2)
(192.168.1.6)
(192.168.100.1)
(192.168.1.5)
(172.16.1.1)
(172.16.32.1) <*>
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
End of list
Reply to request 2 (28 ms). Received packet has options
Total option bytes= 40, padded length=40
Record route:

```
(172.16.1.2)
(192.168.1.6)
(192.168.100.1)
(192.168.1.5)
(172.16.1.1)
(172.16.32.1) &lt;*&gt;
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
End of list
Reply to request 3 (28 ms). Received packet has options
Total option bytes= 40, padded length=40
Record route:
(172.16.1.2)
(192.168.1.6)
(192.168.100.1)
(192.168.1.5)
(172.16.1.1)
(172.16.32.1) &lt;*&gt;
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
End of list
Reply to request 4 (28 ms). Received packet has options
Total option bytes= 40, padded length=40
Record route:
(172.16.1.2)
(192.168.1.6)
(192.168.100.1)
(192.168.1.5)
(172.16.1.1)
(172.16.32.1) &lt;*&gt;
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
End of list
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/28/28 ms
SanJose2#
```

ISP# show ip bgp

Step 10: Establish a default route.

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

ISP(config)# router bgp 200
ISP(config-router)# neighbor 192.168.1.6 default-originate
ISP(config-router)# neighbor 192.168.1.2 default-originate
ISP(config-router)# exit
ISP(config)# interface loopback 10
ISP(config-if)# ip address 10.0.0.1 255.255.255.0
ISP(config-if)#

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

SanJose2# show ip bgp

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

SanJose2# traceroute 10.0.0.1

Type escape sequence to abort.

Tracing the route to 10.0.0.1
VRF info: (vrf in name/id, vrf out name/id)
1 172.16.1.1 8 msec 4 msec 8 msec
2 192.168.1.5 [AS 200] 12 msec * 12 msec

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

ISP(config)# interface serial 1/0
ISP(config-if)# shutdown
ISP(config-if)#

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

SanJose1# show ip route

B* 0.0.0.0/0 [20/0] via 192.168.1.5, 00:00:36

SanJose2# show ip route

B* 0.0.0.0/0 [200/0] via 172.16.64.1, 00:00:45

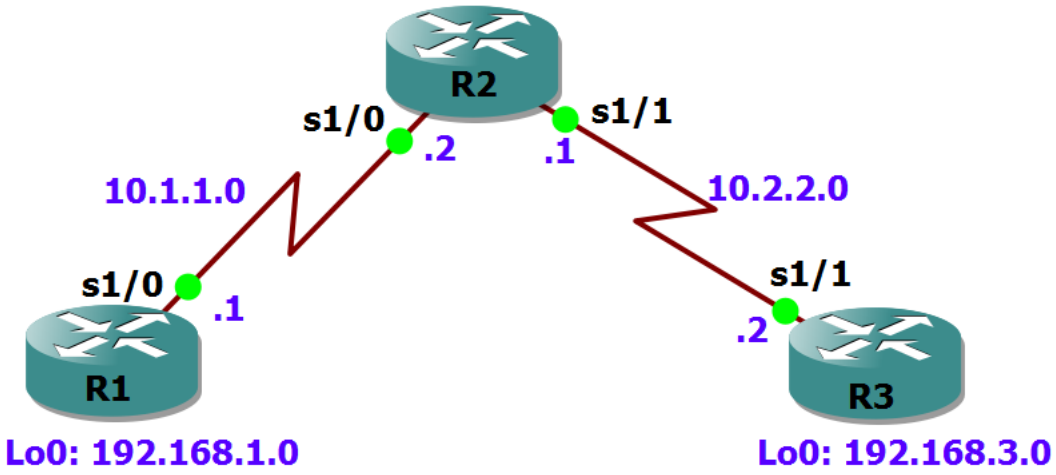
g. Verify the new path using the traceroute command to 10.0.0.1 from SanJose1. Notice the default route is now through SanJose2.

SanJose1# trace 10.0.0.1

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

PRATICAL 4: Secure the Management Plane

Topology



Objectives

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

Step 1: Configure loopbacks and assign addresses.

R1

```
hostname R1
interface Loopback 0
description R1 LAN
ip address 192.168.1.1 255.255.255.0
exit

interface Serial1/0
description R1 --> R2
ip address 10.1.1.1 255.255.255.252
clock rate 128000
no shutdown
exit
end
```

R2

```
hostname R2
interface Serial1/0
description R2 --> R1
ip address 10.1.1.2 255.255.255.252
no shutdown
exit
interface Serial1/1
```

```
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 Serial1/1
description R3 --> R2
ip address 10.2.2.2 255.255.255.252
no shutdown
exit
end
```

Step 2: Configure static routes.

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

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

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

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

c. On R2, configure two static routes.

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

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

R1# **tclsh**
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 }

```
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)#
```

Step 3: Secure management access.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

h. 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.

i. Repeat the configuration portion of steps 3a through 3h on router R3.

Step 4: Configure enhanced username password security.

a. To create local database entry encrypted to level 4 (SHA256), use the **username** *name* **secret** *password* global configuration command. In global configuration mode, enter the following command:
R1(config)# **username JR-ADMIN secret class12345**
R1(config)# **username ADMIN secret class54321**

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

c. Set the vty lines to use the locally defined login accounts.
R1(config)# **line vty 0 4**
R1(config-line)# **login local**
R1(config-line)# **end**
R1(config)#

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

e. To verify the configuration, telnet to R3 from R1 and login using the ADMIN local database account.
R1# **telnet 10.2.2.2**
Trying 10.2.2.2 ... Open
Unauthorized access strictly prohibited!
User Access Verification
Username: **ADMIN**
Password:
R3>

Note:- Password is class54321 and it will be invisible while writing. Just write the password and press enter.

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

a. Always have local database accounts created before enabling AAA. Since we created two local database accounts in the previous step, then we can proceed and enable AAA on R1.
R1(config)# **aaa new-model**

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

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

d. Assign both RADIUS servers to a server group.
R1(config)# **aaa group server radius RADIUS-GROUP**
R1(config-sg-radius)# **server name RADIUS-1**
R1(config-sg-radius)# **server name RADIUS-2**
R1(config-sg-radius)# **exit**
R1(config)#

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

f. Enable the default AAA authentication Telnet login to attempt to validate against the server group. If they are not available, then authentication should be validated against a case sensitive local database.
R1(config)# **aaa authentication login TELNET-LOGIN group RADIUS-GROUP local-case**
R1(config)#

g. Alter the VTY lines to use the TELNET-LOGIN AAA authentiaito0n method.
R1(config)# **line vty 0 4**
R1(config-line)# **login authentication TELNET-LOGIN**
R1(config-line)# **exit**

R1(config)#

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

i. To verify the configuration, telnet to R3 from R1 and login using the ADMIN local database account.
R1# **telnet 10.2.2.2**
Trying 10.2.2.2 ... Open
Unauthorized access strictly prohibited!

User Access Verification
Username: **ADMIN**
Password:

R3>

Note:- Password is class54321 and it will be invisible while writing. Just write the password and press enter.

Step 6: Enabling secure remote management using SSH.

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

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

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

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

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

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

d. Configure SSH version 2 on R1.
R1(config)# **ip ssh version 2**

e. Configure the vty lines to use only SSH connections.
R1(config)# **line vty 0 4**
R1(config-line)# **transport input ssh**
R1(config-line)# **end**

f. Verify the SSH configuration using the **show ip ssh** command.
R1# **show ip ssh**

SSH Enabled - version 2.0
Authentication timeout: 120 secs; Authentication retries: 3
Minimum expected Diffie Hellman key size : 1024 bits
IOS Keys in SECSH format(ssh-rsa, base64 encoded):

```
ssh-rsa
        AAAAB3NzaC1yc2EAAAADAQABAAAAgQC3Lehh7ReYlgyDzls6wq+mFzxqzoa
        ZFr9XGx+Q/yio
R1#
```

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

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

R1# **ssh -l ADMIN 10.2.2.2**

Password:

Note:- Password is class54321 and it will be invisible while writing. Just write the password and press enter.

Unauthorized access strictly prohibited!

R3>

R3> **en**

Password:

Note:- Password is class12345 and it will be invisible while writing. Just write the password and press enter.

R3#

For device configuration use command **show run** on every router.

```
R1#show run
Building configuration...

Current configuration : 2768 bytes
!
! Last configuration change at 16:00:02 UTC Sat Jun 24 2023
!
version 15.2
service timestamps debug datetime msec
service timestamps log datetime msec
service password-encryption
!
hostname R1
!
boot-start-marker
boot-end-marker
!
!
security passwords min-length 10
enable secret 5 $1$AhFY$WxXLG5dZKtVTToUFLkhbw0
!
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
!
!
!
```

R1 R2

```
!
!
interface Loopback0
 description R1 LAN
 ip address 192.168.1.1 255.255.255.0
!
interface FastEthernet0/0
 no ip address
 shutdown
 duplex full
!
interface Serial1/0
 description R1 --> R2
 ip address 10.1.1.1 255.255.255.252
 serial restart-delay 0
 clock rate 128000
!
interface Serial1/1
 no ip address
 shutdown
 serial restart-delay 0
!
interface Serial1/2
 no ip address
 shutdown
 serial restart-delay 0
!
interface Serial1/3
 no ip address
 shutdown
 serial restart-delay 0
!
interface Serial2/0
 no ip address
 shutdown
 serial restart-delay 0
```

R1 R2 R3

```
!
aaa session-id common
no ip icmp rate-limit unreachable
ip cef
!
!
!
!
!
!
no ip domain lookup
ip domain name ccnasecurity.com
no ipv6 cef
!
!
multilink bundle-name authenticated
!
!
!
!
!
!
!
username JR-ADMIN secret 5 $1$JQyj$SGT1rWeevvos2oD85BDd10
username ADMIN secret 5 $1$a1sj$1tYJooDpZd/LFES3npSHk/
!
!
ip tcp synwait-time 5
ip ssh version 2
!
!
!
!
!
!
!
```

R1 R2 R3

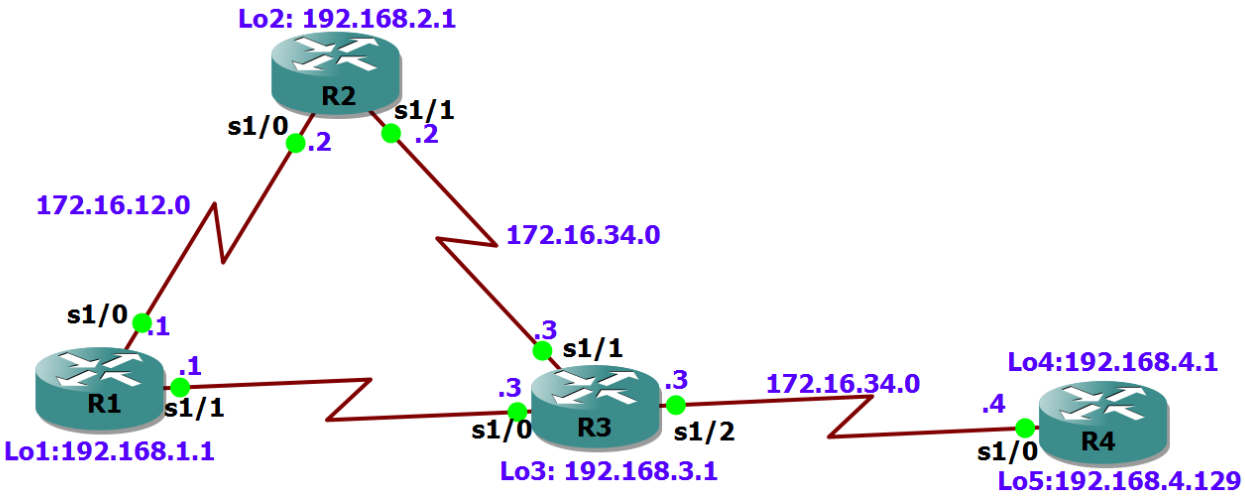
```
!
ip forward-protocol nd
!
!
no ip http server
no ip http secure-server
ip route 0.0.0.0 0.0.0.0 10.1.1.2
!
!
!
!
radius server RADIUS-1
 address ipv4 192.168.1.101 auth-port 1645 acct-port 1646
 key 7 113B38213E27384155673B257D6622720103
!
radius server RADIUS-2
 address ipv4 192.168.1.102 auth-port 1645 acct-port 1646
 key 7 15202A282D1F18697A7E2523465201531352
!
!
control-plane
!
banner motd ^CUnauthorized access strictly prohibited!^C
!
line con 0
 exec-timeout 5 0
 privilege level 15
 password 7 104D000A061811040214 2B3837
 logging synchronous
 stopbits 1
line aux 0
 exec-timeout 0 0
 privilege level 15
 logging synchronous
 no exec
 stopbits 1
```

```
 no exec
 stopbits 1
line vty 0 4
 exec-timeout 5 0
 password 7 13061E010803123E3234292026
 login authentication TELNET-LOGIN
 transport input ssh
!
!
end

R1#
R1#
```

Use **show run** command on **R2#** and **R3#**

PRATICAL 5: Configure and Verify Path Control Using PBR

Topology



Objectives

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

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. On the serial interfaces connecting R1 to R2 and R2 to R3, specify the bandwidth as 128 Kb/s and set a clock rate on the DCE using the **clock rate 128000** command.

Router R1

```
hostname R1
interface Lo1
description R1 LAN
ip address 192.168.1.1 255.255.255.0

interface Serial1/0
description R1 --> R2
ip address 172.16.12.1 255.255.255.248
clock rate 128000
bandwidth 128
 no shutdown

interface Serial1/1
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 Serial1/0
description R2 --> R1
ip address 172.16.12.2 255.255.255.248
bandwidth 128
no shutdown

interface Serial1/1
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 Serial1/0
description R3 --> R1
ip address 172.16.13.3 255.255.255.248
clock rate 64000
bandwidth 64
no shutdown

interface Serial1/1
description R3 --> R2
ip address 172.16.23.3 255.255.255.248
bandwidth 128
no shutdown
interface Serial1/2
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 Serial1/0
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**

```
Serial1/0          172.16.13.3     YES manual up                    up
Serial1/1          172.16.23.3     YES manual up                    up
Serial1/2          172.16.34.3     YES manual up                    up
Loopback3          192.168.3.1     YES manual up                    up
```

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
Serial1/0 is up, line protocol is up
  Internet address is 172.16.13.3/29
Serial1/1 is up, line protocol is up
  Internet address is 172.16.23.3/29
Serial1/2 is up, line protocol is up
  Internet address is 172.16.34.3/29
Loopback3 is up, line protocol is up
  Internet address is 192.168.3.1/24
```

R3# **show interfaces description | include up**

```
Se1/0              up              up      R3 --> R1
Se1/1              up              up      R3 --> R2
Se1/2              up              up      R3 --> R4
Lo3                up              up      R3 LAN
```

Step 3: Configure basic EIGRP.

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

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

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**

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

R2# **show ip eigrp neighbors**

```
R2#show ip eigrp neighbors
*May 19 23:13:42.783: %SYS-5-CONFIG_I: Configured from console by console
R2#show ip eigrp neighbors
IP-EIGRP neighbors for process 1
H   Address                 Interface       Hold Uptime   SRTT   RTO  Q  Seq
                                            (sec)         (ms)       Cnt Num
1   172.16.23.3             Se1/1            11 00:00:50   41   246  0  20
0   172.16.12.1             Se1/0            11 00:01:04   30   200  0  18
```

R3# **show ip eigrp neighbors**

```
R3#show ip eigrp neighbors
IP-EIGRP neighbors for process 1
H   Address                 Interface       Hold Uptime   SRTT   RTO  Q  Seq
                                            (sec)         (ms)       Cnt Num
2   172.16.34.4             Se1/2            12 00:00:44   48   288  0  6
1   172.16.23.2             Se1/1            11 00:00:58   26   200  0  19
0   172.16.13.1             Se1/0            12 00:00:58  281  1686  0  20
```

R4# **show ip eigrp neighbors**

```
R4#show ip eigrp neighbors
IP-EIGRP neighbors for process 1
H   Address                 Interface       Hold Uptime   SRTT   RTO  Q  Seq
                                            (sec)         (ms)       Cnt Num
0   172.16.34.3             Se1/0            10 00:00:55   23   200  0  26
```

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 }

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**

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

R4# **traceroute 192.168.1.1 source 192.168.4.1**
R4# **traceroute 192.168.1.1 source 192.168.4.129**

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**

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

R3# **show interfaces serial1/0**
R3# **show interfaces serial1/0 | include BW**
R3# **show interfaces serial1/1 | include BW**

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**

Step 6: Configure PBR to provide path control.

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

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

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

Step 7: Test the policy.

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 ?**
R3# **debug ip policy 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**

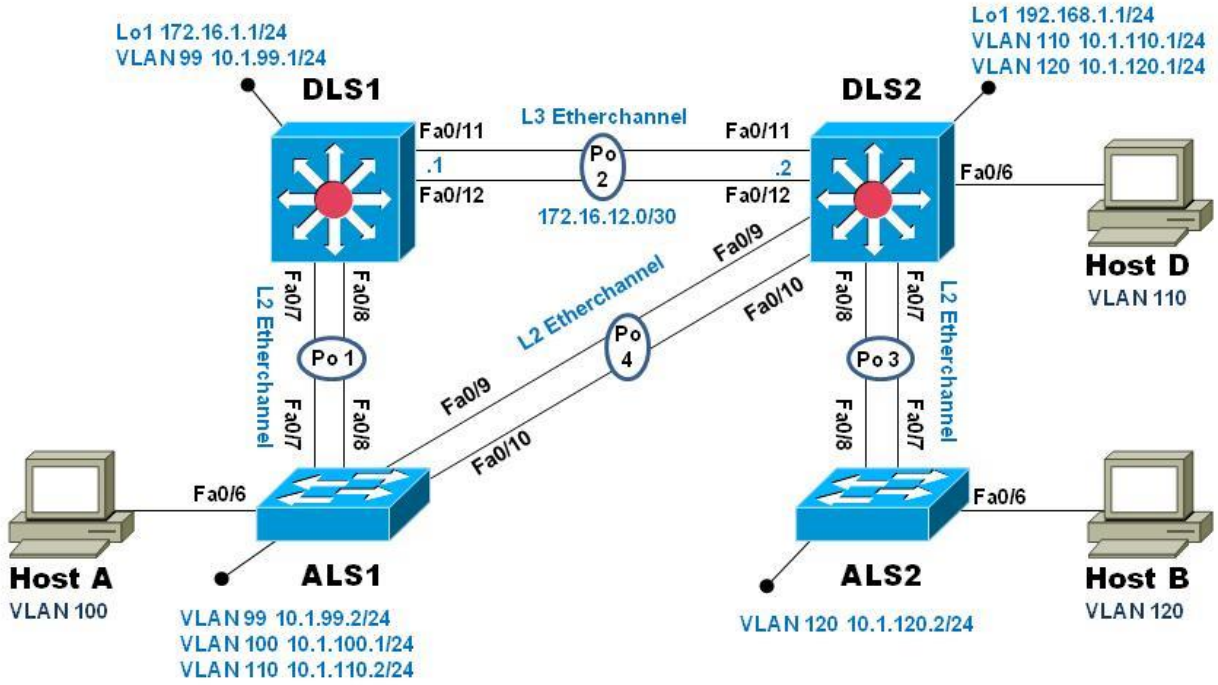
d. Test the policy from R4 with the **traceroute** command, using R4 LAN B as the source network.
R4# **traceroute 192.168.1.1 source 192.168.4.129**

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
```

PRATICAL 6: Inter-VLAN Routing

Topology



Objectives

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

Part 1: Configure Multilayer Switching using Distribution Layer Switches

Step 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. Thenload 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 to administratively down
Switch# **copy BASE.CFG running-config**

Destination filename [running-config]?
184 bytes copied in 0.310 secs (594 bytes/sec)
DLS1#

Step 2: 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]

Step 3: Configure layer 3 interfaces on the DLS switches

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

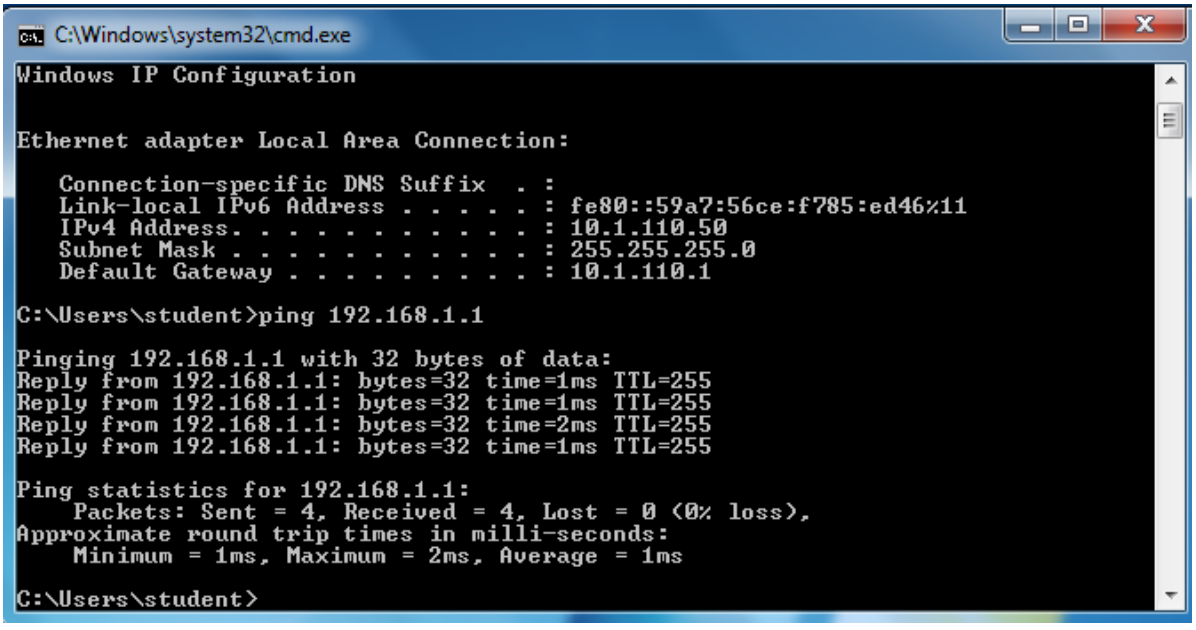
An example from DLS2:

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 intervlan 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). This should work just like a hardware router; the switch will provide connectivity between two directly connected interfaces. 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)#
```



Step 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. This will provide the benefit of increased available bandwidth between the two multilayer 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
```

Step 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#
```

Step 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         1-109,111-4094
Po4         110
ALS1#
```

Step 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 0 msec 8 msec
  3 10.1.99.2 0 msec 0 msec *
ALS2#
```

Part 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.

Step 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
```

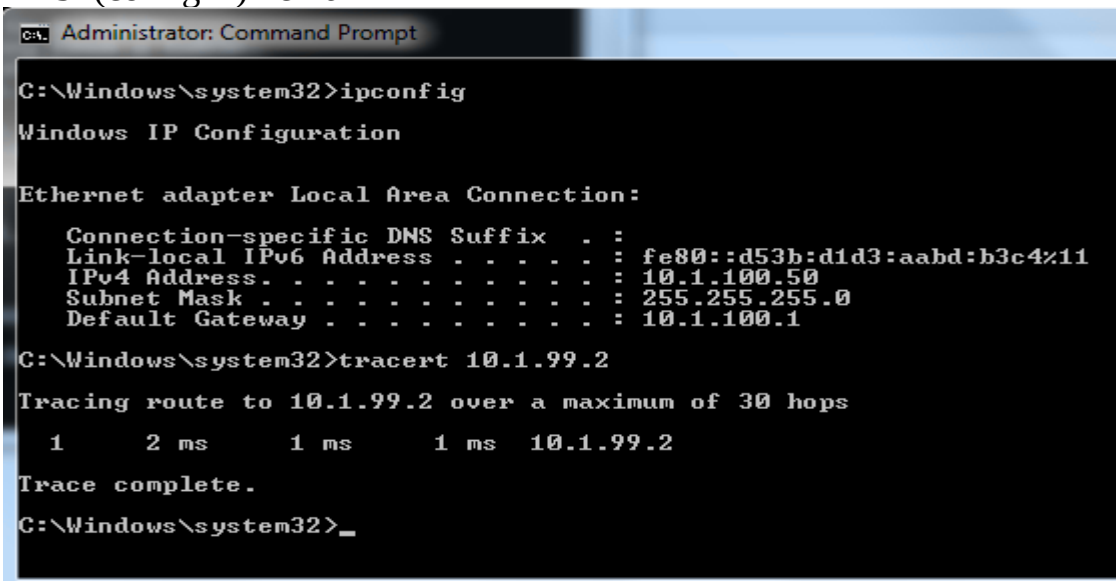
```
ALS1(config-vlan)# exit
ALS1(config)# vlan 110
ALS1(config-vlan)# name InterNode
ALS1(config-vlan)# exit
ALS1(config)# int vlan 100
ALS1(config-if)# ip address 10.1.100.1 255.255.255.0
ALS1(config-if)# no shut
ALS1(config-if)# exit
ALS1(config)# int vlan 110
ALS1(config-if)# ip address 10.1.110.2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

Step 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.

Step 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, Vlan100
L        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, Vlan110
S     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:

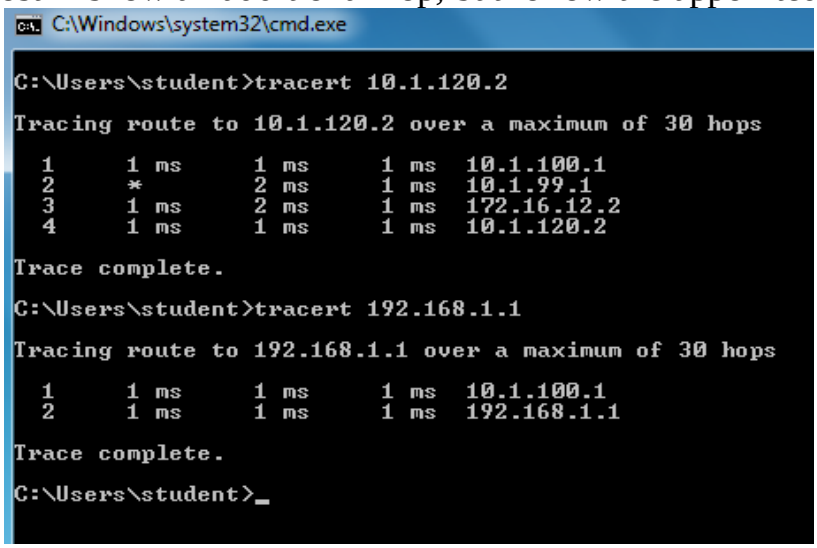
```
ALS1# traceroute 10.1.120.2
Type escape sequence to abort.
Tracing the route to 10.1.120.2
VRF info: (vrf in name/id, vrf out name/id)
  1 10.1.99.1 0 msec 0 msec 0 msec
  2 172.16.12.2 9 msec 0 msec 0 msec
  3 10.1.120.2 0 msec 8 msec *
```

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

```
ALS1# traceroute 192.168.1.1
Type escape sequence to abort.
```

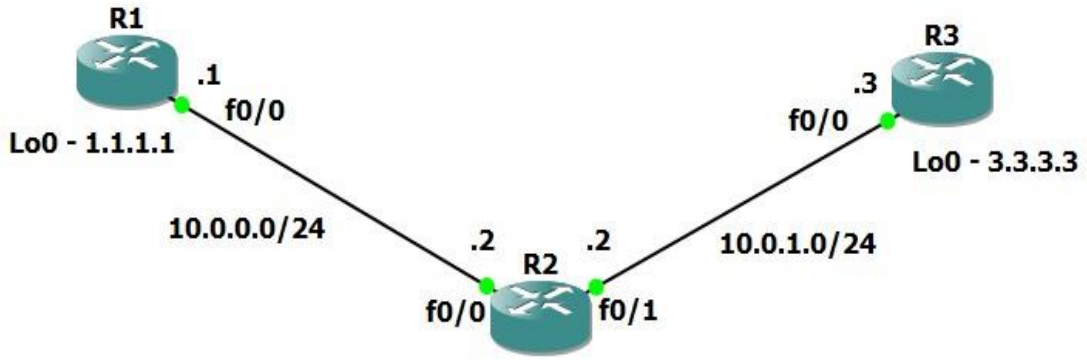
Tracing the route to 192.168.1.1
VRF info: (vrf in name/id, vrf out name/id)
1 10.1.110.1 0 msec 0 msec *

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



PRATICAL 7: Simulating MPLS Environment

Topology



Step 1 – IP addressing of MPLS Core and OSPF
First bring 3 routers into your topology R1, R2, R3 position them as below. We are going to address the routers and configure ospf to ensure loopback to loopback connectivity between R1 and R3

R1

```
hostname R1
int lo0
ip add 1.1.1.1 255.255.255.255
ip ospf 1 area 0

int f0/0
ip add 10.0.0.1 255.255.255.0
no shut
ip ospf 1 area 0
```

R2

```
hostname R2
int lo0
ip add 2.2.2.2 255.255.255.255
ip ospf 1 area 0

int f0/0
ip add 10.0.0.2 255.255.255.0
no shut
ip ospf 1 area 0

int f0/1
ip add 10.0.1.2 255.255.255.0
no shut
ip ospf 1 area 0
```

R3

```
hostname R3
int lo0
ip add 3.3.3.3 255.255.255.255
ip ospf 1 area 0

int f0/0
ip add 10.0.1.3 255.255.255.0
no shut
ip ospf 1 area 0
```

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

```
R1#ping 3.3.3.3 source lo0
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 3.3.3.3, timeout is 2
seconds:
Packet sent with a source address of 1.1.1.1
!!!!!
Success rate is 100 percent (5/5), round-trip
min/avg/max = 40/52/64 ms
R1#
```

You could show the routing table here, but the fact that you can ping between the loopbacks is verification enough and it is safe to move on.

Step 2 – Configure LDP on all the interfaces in the MPLS Core

In order to run MPLS you need to enable it, there are two ways to do this.

- At each interface enter the mpls ip command
- Under the ospf process use the mpls ldp autoconfig command

For this tutorial we will be using the second option, so go int the ospf process and enter mpls ldp autoconfig – this will enable mpls label distribution protocol on every interface running ospf under that specific process.

R1

```
router ospf 1
mpls ldp autoconfig
```

R2

```
router ospf 1
mpls ldp autoconfig
```

R3

```
router ospf 1
mpls ldp autoconfig
```

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

up.

```
R2#
*Mar 1 00:31:53.643: %SYS-5-CONFIG_I: Configured from
console
*Mar 1 00:31:54.423: %LDP-5-NBRCHG: LDP Neighbor
1.1.1.1:0 (1) is UP
R2#
*Mar 1 00:36:09.951: %LDP-5-NBRCHG: LDP Neighbor
3.3.3.3:0 (2) is UP
```

To verify the mpls interfaces the command is very simple – **sh mpls Interface**

This is done on R2 and you can see that both interfaces are running mpls and using LDP

R2#sh mpls interface

```
Interface              IP            Tunnel
Operational
FastEthernet0/0        Yes  (ldp)    No Yes
FastEthernet0/1        Yes  (ldp)    No Yes
```

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

R2#sh mpls ldp neigh

```
Peer LDP Ident: 1.1.1.1:0; Local LDP Ident 2.2.2.2:0
TCP connection: 1.1.1.1.646 - 2.2.2.2.37909

State: Oper; Msgs sent/rcvd: 16/17; Downstream
Up time: 00:07:46
LDP discovery sources:
FastEthernet0/0, Src IP addr: 10.0.0.1
Addresses bound to peer LDP Ident:
10.0.0.1 1.1.1.1
Peer LDP Ident: 3.3.3.3:0; Local LDP Ident 2.2.2.2:0
TCP connection: 3.3.3.3.22155 - 2.2.2.2.646
State: Oper; Msgs sent/rcvd: 12/11; Downstream
Up time: 00:03:30
LDP discovery sources:
FastEthernet0/1, Src IP addr: 10.0.1.3
Addresses bound to peer LDP Ident:
10.0.1.3 3.3.3.3
```

One more verification to confirm LDP is running ok is to do a trace between R1 and R3 and verify if you get MPLS Labels show up in the trace.

R1#trace 3.3.3.3

```
Type escape sequence to abort.
Tracing the route to 3.3.3.3

1 10.0.0.2 [MPLS: Label 17 Exp 0] 84 msec 72 msec 44
msec
2 10.0.1.3 68 msec 60 msec *
```

As you can see the trace to R2 used an MPLS Label in the path, as this is a very small MPLS core only one label was used as R3 was the final hop.

Step 3 – MPLS BGP Configuration between R1 and R3

We need to establish a Multi Protocol BGP session between R1 and R3 this is done by configuring the vpnv4 address family as below

R1#

router bgp 1
neighbor 3.3.3.3 remote-as 1
neighbor 3.3.3.3 update-source Loopback0
no auto-summary
address-family vpnv4
neighbor 3.3.3.3 activate

R3#

router bgp 1
neighbor 1.1.1.1 remote-as 1
neighbor 1.1.1.1 update-source Loopback0
no auto-summary
address-family vpnv4
neighbor 1.1.1.1 activate

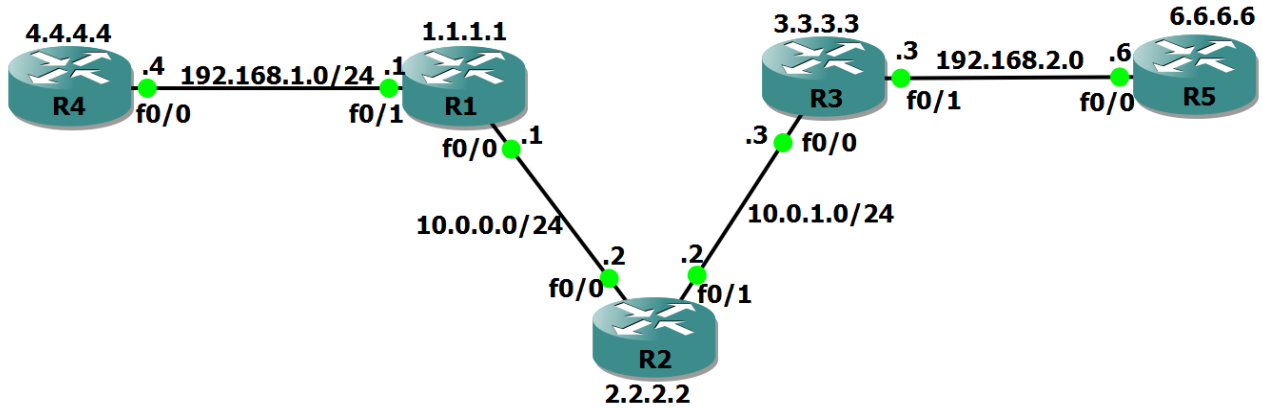
To verify the BGP session between R1 and R3 issue the command sh bgp vpnv4 unicast all summary

R1#sh bgp vpnv4 unicast all summary

```
BGP router identifier 1.1.1.1, local AS number 1
BGP table version is 1, main routing table version 1

Neighbor  V    AS    MsgRcvd MsgSent TblVer InQ  OutQ     Up/Down     State/PfxRcd
3.3.3.3   4    1     218             218     1    0     0     03:17:48            0
```

Step 4 – Add two more routers, create VRFs

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



Router 4 will peer OSPF using process number 2 to a VRF configured on R1. It will use the local site addressing of 192.168.1.0/24.

R4

int lo0
ip add 4.4.4.4 255.255.255.255
ip ospf 2 area 2

int f0/0

ip add 192.168.1.4 255.255.255.0
ip ospf 2 area 2
no shut

R1
int f0/1
no shut
ip add 192.168.1.1 255.255.255.0

Now at this point we have R4 peering to R1 but in the global routing table of R1 which is not what we want.
We are now going to start using VRF's, we now need to create a VRF on R1
R1
ip vrf RED
rd 4:4
route-target both 4:4

R1
R1(config)#int f0/1
R1(config-if)#ip vrf forwarding RED
Now notice what happens when you do that – the IP address is removed
R1(config-if)#ip vrf fo

```
% Interface FastEthernet0/1 IP address 192.168.1.1
removed due to enabling VRF RED
```

You just need to re-apply it
R1
int f0/1
ip address 192.168.1.1 255.255.255.0

Now if we view the config on R1 int f0/1 you can see the VRF configured.
R1#sh run int f0/1

```
Building configuration...

Current configuration : 119 bytes
!

interface FastEthernet0/1
ip vrf forwarding RED
ip address 192.168.1.1 255.255.255.0
duplex auto
speed auto
end

R1#
```

R1#sh ip route
R1#sh ip route vrf RED

```
Routing Table: RED
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA – OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2
i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 -
IS-IS level-2
ia - IS-IS inter area, * - candidate default, U - per- user static route
o - ODR, P - periodic downloaded static route
Gateway of last resort is not set
C 192.168.1.0/24 is directly connected, FastEthernet0/1
R1#
```

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

R1

int f0/1
ip ospf 2 area 2

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

R1#sh ip route vrf RED

```
Routing Table: RED
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA – OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2
i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
ia - IS-IS inter area, * - candidate default, U - per-user static route
o - ODR, P - periodic downloaded static route
Gateway of last resort is not set
4.0.0.0/32 is subnetted, 1 subnets
O 4.4.4.4 [110/11] via 192.168.1.4, 00:00:22,
FastEthernet0/1
C 192.168.1.0/24 is directly connected, FastEthernet0/1
```

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

R6

int lo0
ip add 6.6.6.6 255.255.255.255
ip ospf 2 area 2

int f0/0
ip add 192.168.2.6 255.255.255.0
ip ospf 2 area 2
no shut

R3
int f0/1
no shut
ip add 192.168.2.3 255.255.255.0

We also need to configure a VRF onto R3 as well.
R3
ip vrf RED
rd 4:4
route-target both 4:4

So now we have configured the VRF on R3 we need to move the interface F0/1 into that VRF

```
R3#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#int f0/1
R3(config-if)#no shut
R3(config-if)#ip add
*Jun 26 23:22:59.083: %LINK-3-UPDOWN: Interface FastEthernet0/1, changed stat
e to up
*Jun 26 23:23:00.083: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEth
ernet0/1, changed state to up
R3(config-if)#ip add 192.168.2.3 255.255.255.0
R3(config-if)#ip vrf RED
R3(config-vrf)#rd 4:4
R3(config-vrf)#route-target both 4:4
R3(config-vrf)#int f0/1
R3(config-if)#ip vrf forwarding RED
% Interface FastEthernet0/1 IPv4 disabled and address(es) removed due to enab
ling VRF RED
```

R3
int f0/1
ip vrf forwarding RED
Now notice what happens when you do that – the IP address is removed
R3(config-if)#ip vrf forwarding RED
% Interface FastEthernet0/1 IP address 192.168.2.1
removed due to enabling VRF RED
You just need to re-apply it
R3
R3(config-if)#int f0/1
R3(config-if)#ip address 192.168.2.1 255.255.255.0
R3(config-if)#exit
Now if we view the config on R3 int f0/1 you can see the VRF configured.

R3#sh run int f0/1
Finally we just need to enable OSPF on that interface and verify the routes are in the RED routing table.
R3
int f0/1
ip ospf 2 area 2

Check the routes in vrf RED
R3
R3#sh ip route vrf RED

```
Routing Table: RED
Codes: C - connected, S - static, R - RIP, M - mobile, B
- BGP

Gateway of last resort is not set

6.0.0.0/32 is subnetted, 1 subnets
O 6.6.6.6 [110/11] via 192.168.2.6, 00:02:44,
FastEthernet0/1
C 192.168.2.0/24 is directly connected,
FastEthernet0/1
R3#
```

Check the routes on R4
R4#sh ip route

```
4.0.0.0/32 is subnetted, 1 subnets
C 4.4.4.4 is directly connected, Loopback0
C 192.168.1.0/24 is directly connected, FastEthernet0/0
```

Check the routes on R1
R1#sh ip route

```
1.0.0.0/32 is subnetted, 1 subnets
C 1.1.1.1 is directly connected, Loopback0
2.0.0.0/32 is subnetted, 1 subnets
O 2.2.2.2 [110/11] via 10.0.0.2, 00:01:04,
FastEthernet0/0
3.0.0.0/32 is subnetted, 1 subnets
O 3.3.3.3 [110/21] via 10.0.0.2, 00:00:54,
FastEthernet0/0
10.0.0.0/24 is subnetted, 2 subnets
C 10.0.0.0 is directly connected, FastEthernet0/0
O 10.0.1.0 [110/20] via 10.0.0.2, 00:00:54,
FastEthernet0/0
```

R1#sh ip route vrf RED

```
Routing Table: RED
4.0.0.0/32 is subnetted, 1 subnets
O 4.4.4.4 [110/11] via 192.168.1.4, 00:02:32,
FastEthernet0/1
C 192.168.1.0/24 is directly connected, FastEthernet0/1
```

Here you can see Routing Table: RED is shown and the routes to R4 are now visible with 4.4.4.4 being in OSPF.
So we need to do the following;

- Redistribute OSPF into MP-BGP on R1
- Redistribute MP-BGP into OSPF on R1
- Redistribute OSPF into MP-BGP on R3
- Redistribute MP-BGP into OSPF on R3

Redistribute OSPF into MP-BGP on R1
‘
R1

router bgp 1
address-family ipv4 vrf RED
redistribute ospf 2

Redistribute OSPF into MP-BGP on R3
R3
router bgp 1
address-family ipv4 vrf RED
redistribute ospf 2

This has enabled redistribution of the OSPF routes into BGP. We can check the routes from R4 and R6 are now showing in the BGP table for their VRF with this command
sh ip bgp vpnv4 vrf RED

R1#sh ip bgp vpnv4 vrf RED
BGP table version is 9, local router ID is 1.1.1.1
Status codes: s suppressed, d damped, h history, *
valid, > best,
r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
Network Next Hop Metric LocPrf Weight Path
Route Distinguisher: 4:4 (default for vrf RED)
***> 4.4.4.4/32 192.168.1.4 11 32768 ?**
***>i6.6.6.6/32 3.3.3.3 11 100 0 ?**
*> 192.168.1.0 0.0.0.0 0 32768 ?
*>i192.168.2.0 3.3.3.3 0 100 0 ?

Here we can see that 4.4.4.4 is now in the BGP table in VRF RED on R1 with a next hop of 192.168.1.4 (R4) and also 6.6.6.6 is in there as well with a next hop of 3.3.3.3 (which is the loopback of R3 – showing that it is going over the MPLS and R1 is not in the picture)

The same should be true on R3

R3#sh ip bgp vpnv4 vrf RED
BGP table version is 9, local router ID is 3.3.3.3
Status codes: s suppressed, d damped, h history, *
valid, > best, i - internal,
r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

Network Next Hop Metric LocPrf Weight Path
Route Distinguisher: 4:4 (default for vrf RED)
***>i4.4.4.4/32 1.1.1.1 11 100 0 ?**
***> 6.6.6.6/32 192.168.2.6 11 32768 ?**
***>i192.168.1.0 1.1.1.1 0 100 0 ?**
***> 192.168.2.0 0.0.0.0 0 32768 ?**

Which it is! 6.6.6.6 is now in the BGP table in VRF RED on R3 with a next hop of 192.168.2.6 (R6) and also 4.4.4 is in there as well with a next hop of

1.1.1.1 (which is the loopback of R1 – showing that it is going over the MPLS and R2 is not in the picture)
The final step is to get the routes that have come across the MPLS back into OSPF and then we can get end to end connectivity

R1

router ospf 2
redistribute bgp 1 subnets

R3

router ospf 2
redistribute bgp 1 subnets

If all has worked we should be now able to ping 6.6.6.6 from R4
Before we do let's see what the routing table looks like on R4

R4#sh ip route

4.0.0.0/32 is subnetted, 1 subnets
C 4.4.4.4 is directly connected, Loopback0
6.0.0.0/32 is subnetted, 1 subnets
O IA 6.6.6.6 [110/21] via 192.168.1.1, 00:01:31, FastEthernet0/0
C 192.168.1.0/24 is directly connected, FastEthernet0/0

O E2 192.168.2.0/24 [110/1] via 192.168.1.1, 00:01:31, FastEthernet0/0

Great we have 6.6.6.6 in there
Also check the routing table on R6

R6#sh ip route

4.0.0.0/32 is subnetted, 1 subnets
O IA 4.4.4.4 [110/21] via 192.168.2.1, 00:01:22, FastEthernet0/0
6.0.0.0/32 is subnetted, 1 subnets
C 6.6.6.6 is directly connected, Loopback0
O IA 192.168.1.0/24 [110/11] via 192.168.2.1,00:01:22,FastEthernet0/0
C 192.168.2.0/24 is directly connected, FastEthernet0/0

Brilliant we have 4.4.4.4 in there so we should be able to ping across the MPLS

R4#ping 6.6.6.6

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 6.6.6.6, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max= 40/48/52ms

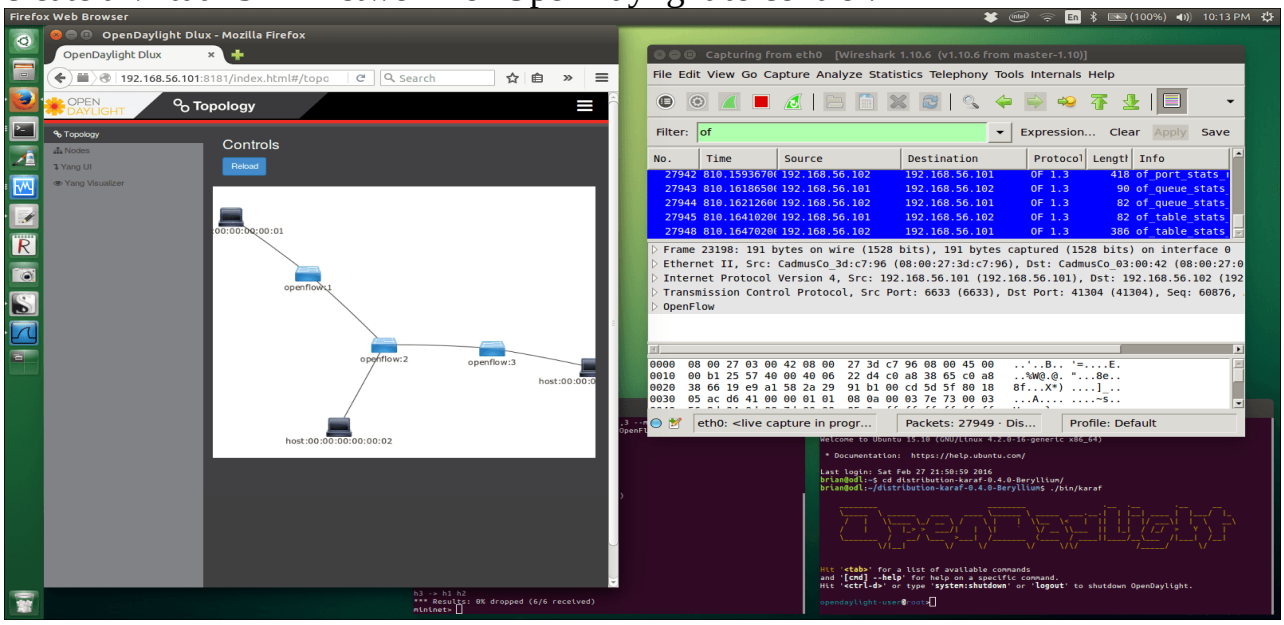
Which we can – to prove this is going over the MPLS and be label switched and not routed, lets do a trace

R4#trace 6.6.6.6

```
Type escape sequence to abort.
Tracing the route to 6.6.6.6

1 192.168.1.1 20 msec 8 msec 8 msec
2 10.0.0.2 [MPLS: Labels 17/20 Exp 0] 36 msec 40 msec
36 msec
3 192.168.2.1 [MPLS: Label 20 Exp 0] 16 msec 40 msec 16
msec
4 192.168.2.6 44 msec 40 msec 56 msec
R4#
```

PRATICAL 8: Simulating OpenDaylight SDN Controller with the Mininet Network Emulator

OpenDaylight (ODL) is a popular open-source SDN controller framework. To learn more about OpenDaylight, it is helpful to use it to manage an emulated network of virtual switches and virtual hosts. Most people use the Mininet network emulator to create a virtual SDN network for OpenDaylight to control.



Using Virtual Machines

Two virtual machines will be used. One will run the Mininet emulated network and the other will run the OpenDaylight controller. I will connect both VMs to a host-only network so they can communicate with each other and with programs running on the host computer, such as **ssh** and the **X11 client**.

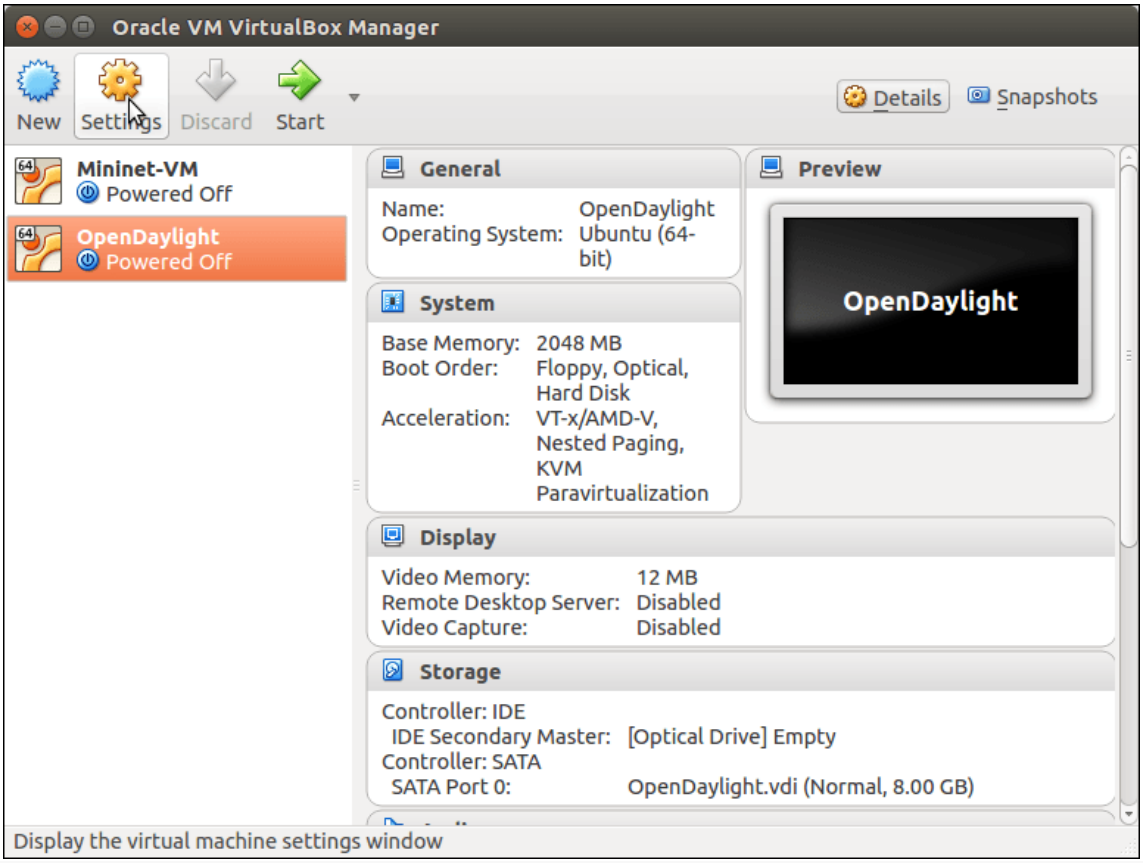
Use VirtualBox to run the Mininet VM that can downloaded from the mininet project web site, which is the easiest way to experiment with Mininet. The Mininet project team provides an Ubuntu 14.04 LTS VM image with Mininet 2.2.1, Wireshark and OpenFlow dissector tools already installed and ready to use.
Install and run the OpenDaylight SDN controller on a new VM I create in VirtualBox.

Setting up the OpenDaylight Virtual Machine

To build the OpenDaylight virtual machine, download the Ubuntu Server ISO image from the ubuntu.com web site. Then installed it in a new VM in VirtualBox. If you need directions on how to install an ISO disk image in a VirtualBox virtual machine, please see post about installing Debian in a VirtualBox VM.

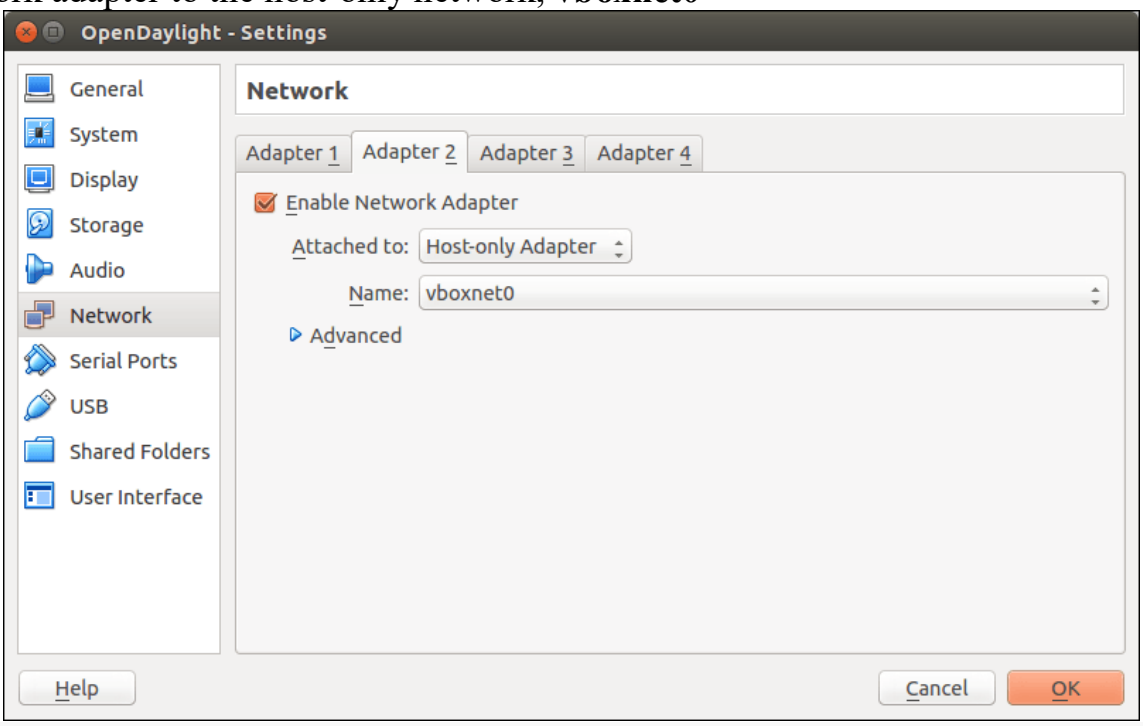
Give the virtual machine a descriptive name. I named the virtual machine **OpenDaylight**. Configure it so it uses two CPUs and 2 GB or RAM. This is the minimum configuration to support OpenDaylight. Then add a host-only network adapter to the VM.

When the VM is powered off, click on the **Settings** button:



The OpenDaylight virtual machine

In the VM's VirtualBox network settings, enable two network interfaces. Connect the first network adapter to the NAT interface (which is the default setting) and the second network adapter to the host-only network, **vboxnet0**



Connecting network adapter 2 to the host-only network

Configure OpenDaylight VM interfaces

By default, the VM's first network adapter is attached to the VirtualBox NAT interface and is already configured when the VM boots up. We need to configure the second network adapter, which is attached to the VirtualBox host-only interface **vboxnet0**.

List all the devices using th **ip** command:

```
brian@odl:~$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_
fast state UP group default qlen 1000
    link/ether 08:00:27:ec:a9:f1 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global enp0s3
valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:feec:a9f1/64 scope link
       valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group
default qlen 1000
    link/ether 08:00:27:b0:f6:70 brd ff:ff:ff:ff:ff:ff
brian@odl:~$
```

Note: starting in 15.10, Ubuntu uses predictable network interface names like **enp0s3** and **enp0s8**, instead of the classic interface names like **eth0** and **eth1**.
We see that interface **enp0s8** has no IP address. This is the second network adapter connected to **vboxnet0**. VirtualBox can assign an IP address on this interface using DHCP if the DCHP client requests it. So, run the following command to set up interface **enp0s8**:

```
brian@odl:~$ sudo dhclient enp0s8
```

Now check the IP address assigned to **enp0s8**:

```
brian@odl:~$ ip addr show enp0s8
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fas
t state UP group default qlen 1000
    link/ether 08:00:27:b0:f6:70 brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.101/24 brd 192.168.56.255 scope global enp0s8
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:feb0:f670/64 scope link
       valid_lft forever preferred_lft forever
brian@odl:~$
```

Now we see the VirtualBox DHCP server connected to the host-only network assigned the IP address **192.168.56.101** to this interface. This is the IP address we should use when connecting to any application running on the VM.

On your system the assigned IP address may be different. You may have set up the VirtualBox preferences to use a different network prefix for the host-only network, or

may have configured the DHCP server to provide a diferent address range. Also, if any other VMs were started and connected to the host-only network before this VM, then the IP address assigned will be different. If the IP address is different, that's OK. Just use the address assigned

.

Now, configure the interface **enp0s8** so it will remain configured after a restart. Edit the **/etc/network/interfaces** file:

brian@odl:~$ **sudo nano /etc/network/interfaces**

Add the following lines to the end of the file **/etc/network/interfaces**:

```
# the host-only network interface
auto enp0s8
iface enp0s8 inet dhcp
```

Connect to the OpenDaylight VM using SSH

Use a terminal application when working on Virtual Machines.

Open a terminal on host computer and login using SSH:

brian@T420:~$ **ssh -X brian@192.168.56.101**

Now you are connected to the OpenDaylight virtual machine and can see that the host name in the prompt is changed to is **odl**, which was configured when installing Ubuntu on the VM.

brian@odl:~$

Also enabled X forwarding when I started SSH so I can run X programs on the OpenDaylight VM.

Install Java

The OpenDaylight SDN controller is a Java program so install the Java run-time environment with the following command:

```
$ sudo apt-get update
$ sudo apt-get install default-jre-headless
```

Set the JAVA_HOME environment variable. Edit the **bashrc** file

brian@odl:~$ **nano ~/.bashrc**

Add the following line to the **bashrc** file:

export JAVA_HOME=/usr/lib/jvm/default-java

Then run the file:

brian@odl:~$ **source ~/.bashrc**

Install OpenDaylight

Download the OpenDaylight software from the OpenDaylight web site. On a Linux or Mac OS host, we can use the **wget** command to download the tar file.

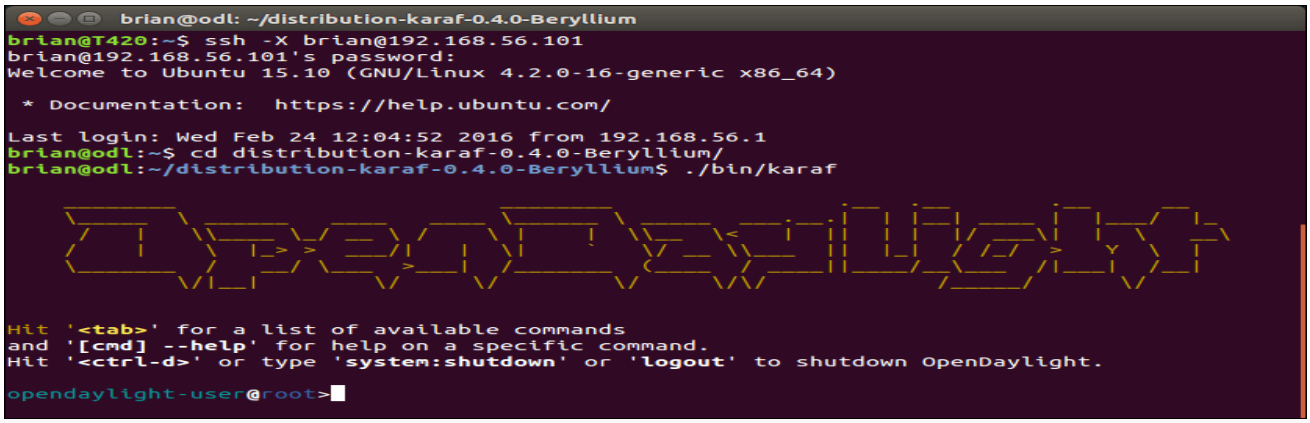
brian@odl:~$ **wget https://nexus.opendaylight.org/content/groups/public/org/opendaylight/integration/distribution-karaf/0.4.0-Beryllium/distribution-karaf-0.4.0-Beryllium.tar.gz**

Install OpenDaylight by extracting the tar file:
brian@odl:~$ **tar -xvf distribution-karaf-0.4.0-Beryllium.tar.gz**

This creates a folder named **distribution-karaf-0.4.0-Beryllium** which contains the OpenDaylight software and plugins.

OpenDaylight is packaged in a **karaf** container. Karaf is a container technology that allows the developers to put all required software in a single distribution folder. This makes it easy to install or re-install OpenDaylight when needed because everything is in one folder. As we will see later, karaf also allows programs to be bundled with optional modules that can be installed when needed.

Start OpenDaylight
To run OpenDaylight, run the **karaf** command inside the package distribution folder.
brian@odl:~$ **cd distribution-karaf-0.4.0-Beryllium**
brian@odl:~$ **./bin/karaf**
Now the OpenDaylight controller is running.



OpenDaylight running in a virtual machine

Install OpenDaylight features
Next, install the minimum set of features required to test OpenDaylight and the OpenDaylight GUI:
opendaylight-user@root> **feature:install odl-restconf odl-l2switch-switch odl-mdsal-apidocs odl-dlux-all**

The above is an example of installing optional modules in a karaf container. You only need to install an optional feature once. Once installed, these features are permanently added to the controller and will run every time it starts.
We installed the following features. Click on each feature to learn more about it:

- odl-restconf: Allows access to RESTCONF API
- odl-l2switch-switch: Provides network functionality similar to an Ethernet switch
- odl-mdsal-apidocs: Allows access to Yang API
- odl-dlux-all: OpenDaylight graphical user interface

To list all available optional features, run the command:

```
opendaylight-user@root> feature:list
```

To list all installed features, run the command:

```
opendaylight-user@root> feature:list --installed
```

Information about OpenDaylight optional features is available on the OpenDaylight wiki.

Stop OpenDaylight

When you want to stop the controller, enter the `<ctrl-d>` key combination or type `system:shutdown` or `logout` at the **opendaylight-user** prompt.

Set up the Mininet Virtual Machine

Start the Mininet VM in the VirtualBox Manager. Now we should have two VMs running: OpenDaylight VM and Mininet VM. If we started the OpenDaylight VM first, it will have IP address 192.168.56.101 and the mininet VM will receive the second available IP address on the host-only network, 192,168.56.102. We can verify this by running the **ip** command on the Mininet VM console:

```
mininet@mininet-vm:~$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN g
roup default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast s
tate UP group default qlen 1000
    link/ether 08:00:27:e2:98:cc brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.102/24 brd 192.168.56.255 scope global eth0
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast s
tate UP group default qlen 1000
    link/ether 08:00:27:1b:c1:07 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.16/24 brd 10.0.2.255 scope global enp0s3
       valid_lft forever preferred_lft forever
mininet@mininet-vm:~$
```

Note: The Mininet VM is based on Ubuntu Server 14.04, which does not yet use the predictable network interface names like **enp0s3** and **enp0s8**, so we see interface names like **eth0** and **eth1**.

We see **eth0** is connected to the host-only interface because it has IP address 192.168.56.102 which is in the address range assigned by the VirtualBox hot-only network DHCP server. So we know we need to use IP address 192.168.56.102 to access applications running on this virtual machine.

Connect to the Mininet VM using SSH

Now open a terminal window on your host computer and SSH into the Mininet VM. Turn X forwarding on. (If you are using Windows, use Xming for an X Window System Server and Putty as an SSH client)

```
brian@T420:~$ ssh -X 198.168.56.102
```

Start Mininet

On the Mininet VM, start a simple network topology. In this case, we will do the following:

- Set up three switches in a linear topology
- Each switch will be connected to one host
- The MAC address on each host will be set to a simple number
- The remote controller, OpenDaylight, is at IP address 192.168.56.101:6633
- We will use OpenFlow version 1.3

The Mininet command to start this is:

```
mininet@mininet-vm:~$ sudo mn --topo linear,3 --mac --controller=remote,ip=192.168.56.101,port=6633 --switch ovs,protocols=OpenFlow13
```

Test the network

Test that the OpenDaylight controller is working by pinging all nodes. Every host should be able to reach every other host:

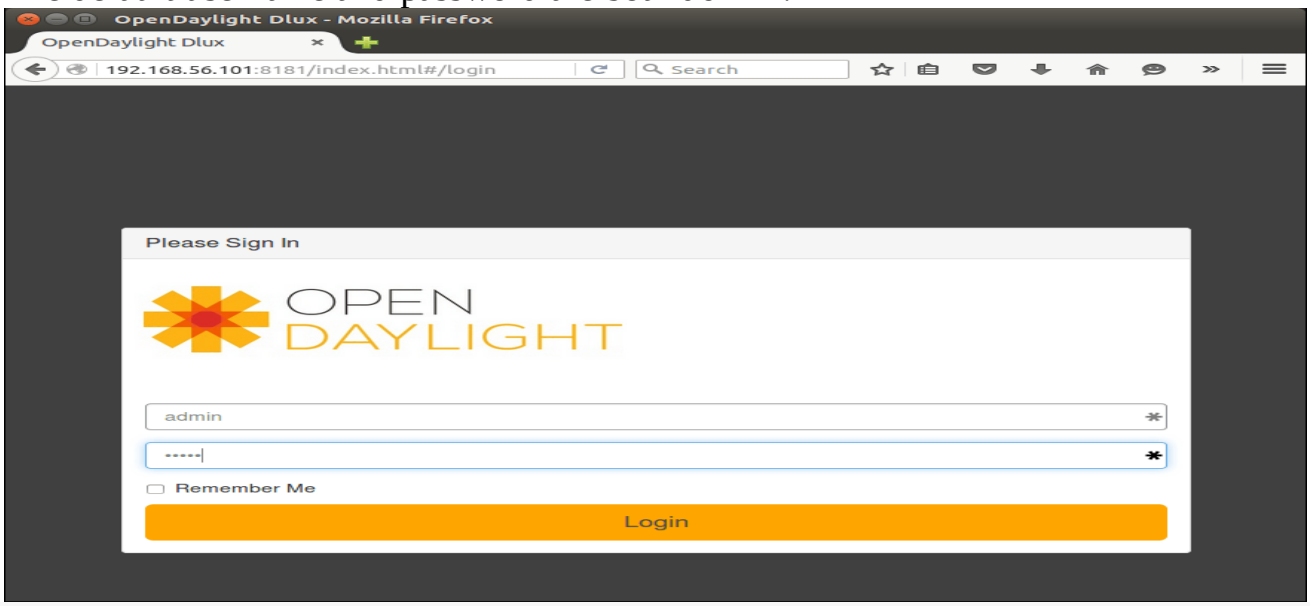
```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
```

The OpenDaylight Graphical User Interface

Open a browser on your host system and enter the URL of the OpenDaylight User Interface (DLUX UI). It is running on the OpenDaylight VM so the IP address is 192.168.56.102 and the port, defined by the application, is 8181:

So the URL is: `http://192.168.56.101:8181/index.html`.

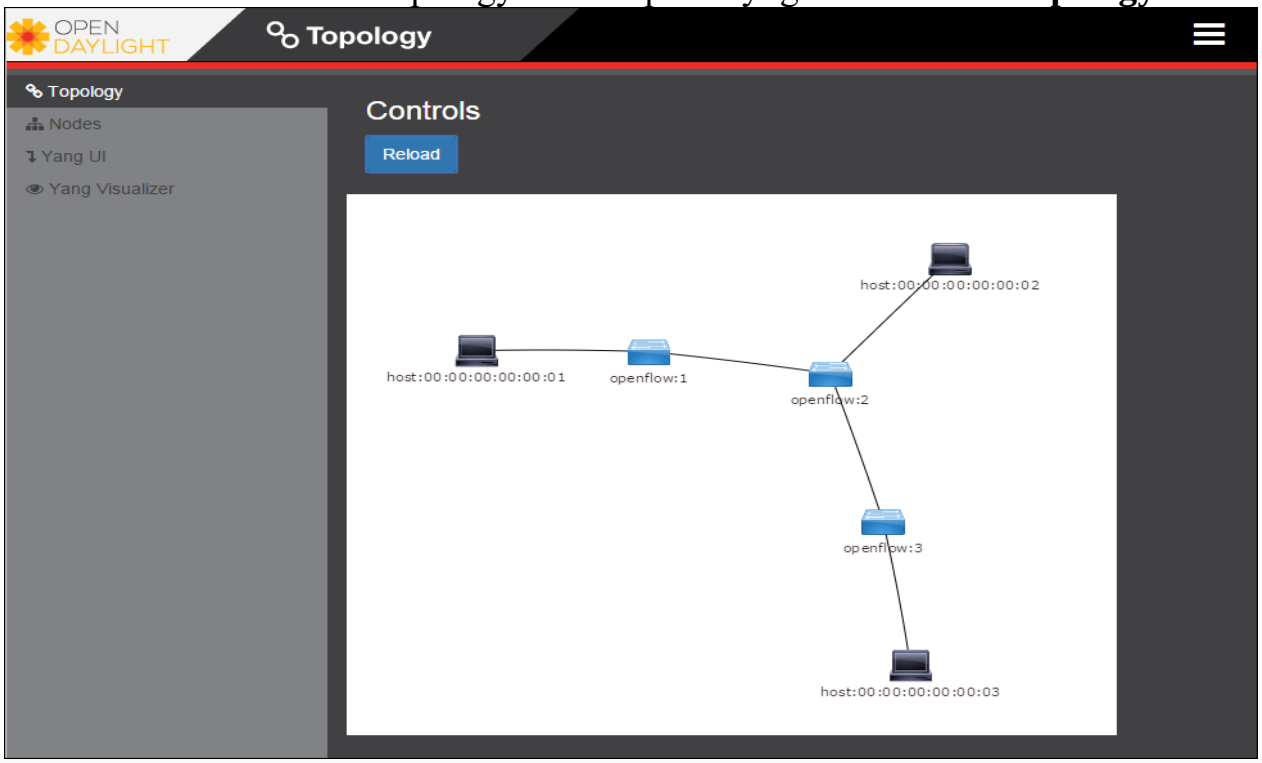
The default username and password are both **admin**.



Log in to OpenDaylight controller

Topology

Now we see the network topology in the OpenDaylight controller's **topology** tab.

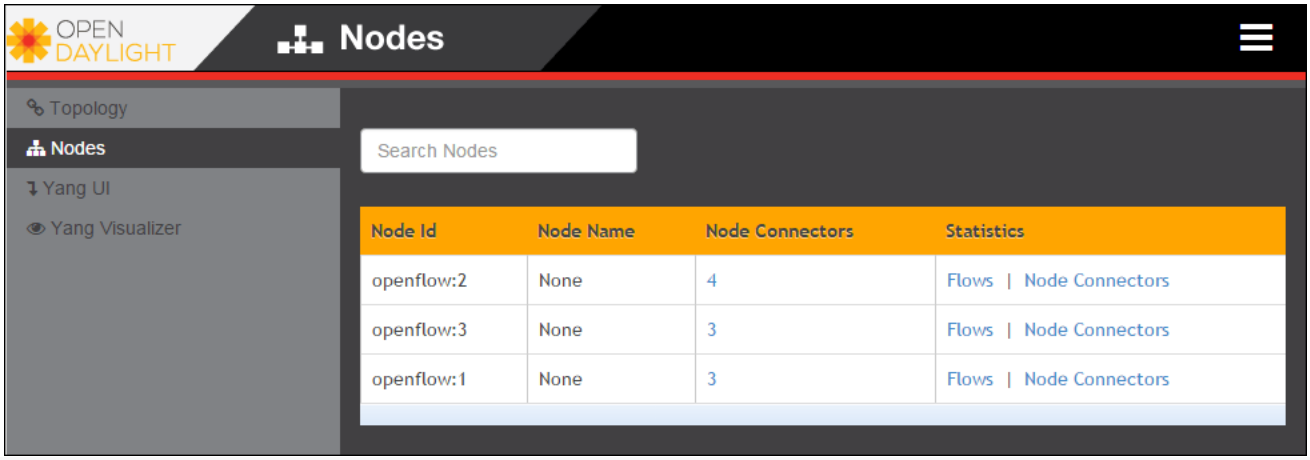


Topology of the Mininet network

You can see the network that is emulated by the Mininet network emulator. You may test OpenDaylight functionality by building different network topologies in Mininet with different attributes, and by using OpenDaylight to run experiments on the emulated network. For example, you may break links between switches in Mininet to test how the network responds to faults.

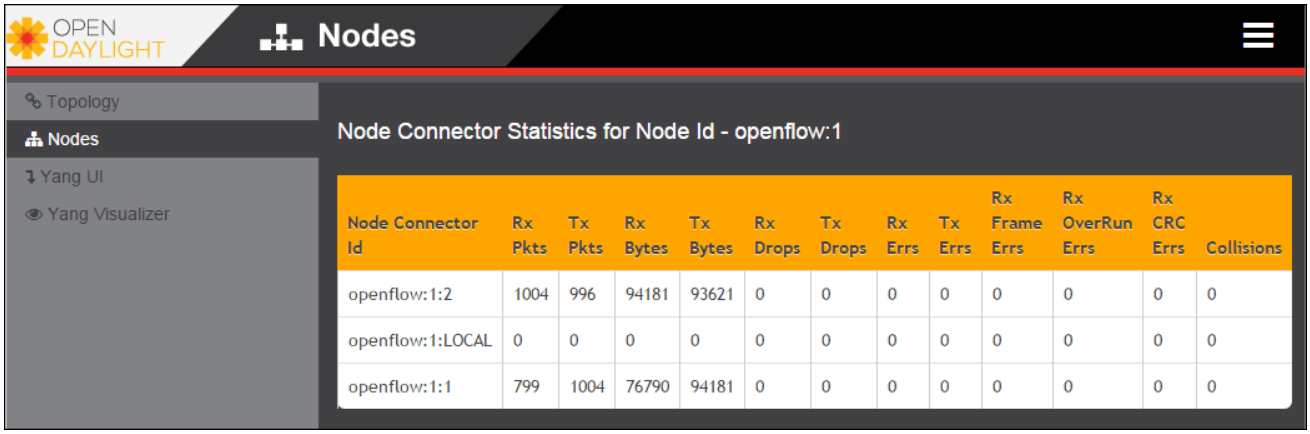
Nodes

Click on the **Nodes** tab to see information about each switch in the network:



List of nodes

Click on the **Node Connectors** link in each row to see information about each port on the switch:



Node Connector Id	Rx Pkts	Tx Pkts	Rx Bytes	Tx Bytes	Rx Drops	Tx Drops	Rx Errs	Tx Errs	Rx Frame Errs	Rx OverRun Errs	Rx CRC Errs	Collisions
openflow:1:2	1004	996	94181	93621	0	0	0	0	0	0	0	0
openflow:1:LOCAL	0	0	0	0	0	0	0	0	0	0	0	0
openflow:1:1	799	1004	76790	94181	0	0	0	0	0	0	0	0

Interfaces

Yang UI

Click on the **Yang UI** tab. Then click on the **Expand all** button to see all available APIs. Not all of them will work because we did not install all features. One API that will work is the **Inventory** API. Click on it, then navigate down to the **nodes** attribute and click on the **Send** button to send the **GET** API method to the controller.

Scroll down to see all the inventory information about the network: nodes, ports, statistics, etc. Click on the switches and interfaces to see the details of each.

Understanding the Yang data model and learning how to read and write to the data store is key to understanding Software Defined Networking with the OpenDaylight controller.

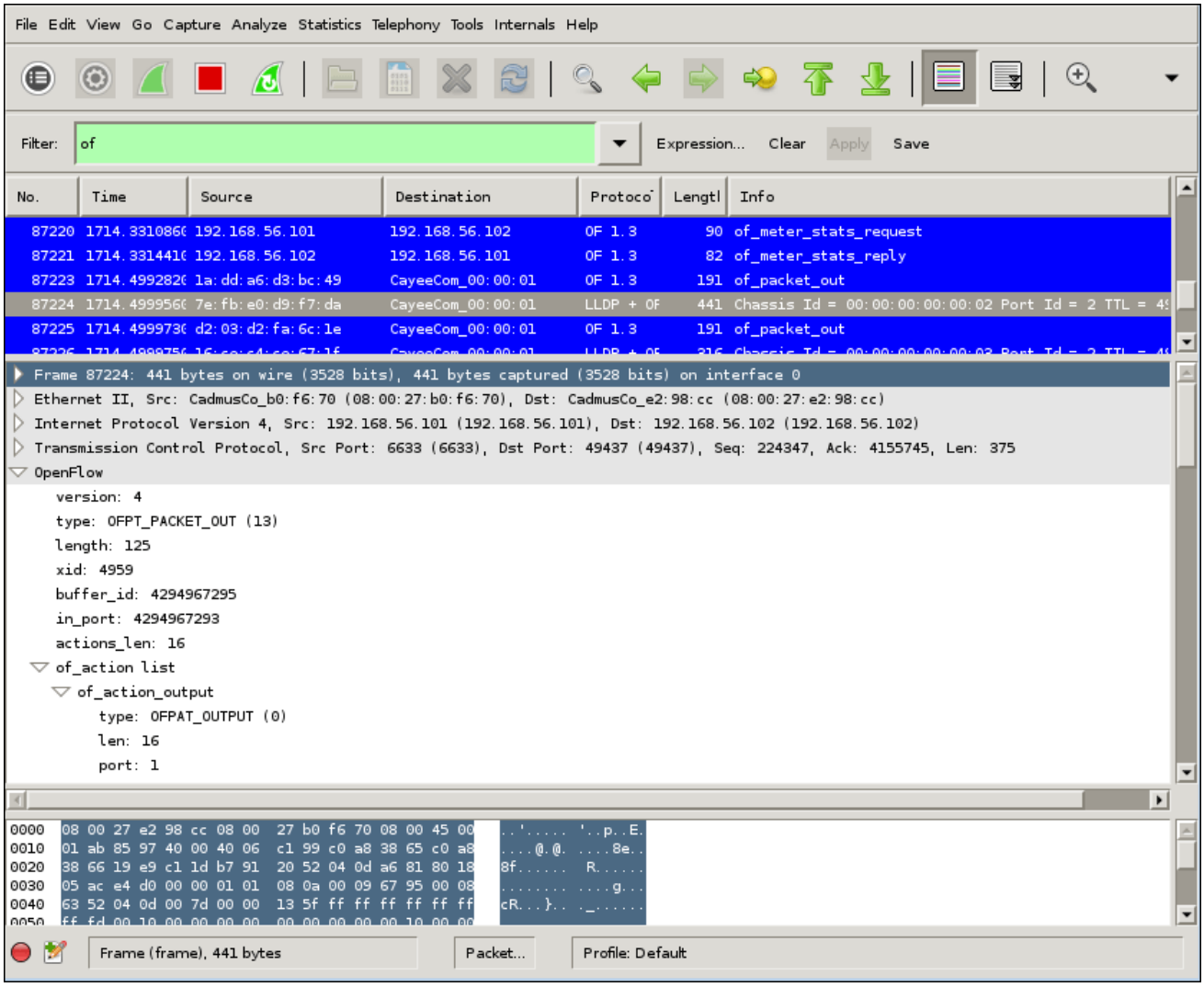
Capturing OpenFlow Messages

To dive deeper into how SDN controllers and switches operate, you may want to view the OpenFlow messages exchanged between the controller and switches in the network. The Mininet VM comes with Wireshark installed, with a custom version of the OpenFlow dissector already set up.
So the easiest way to view OpenFlow messages is to start Wireshark on the Mininet VM and capture data on the interface connected to the host-only network, which is **eth0** in this case.

Open a new terminal window and connect to the Mininet VM using SSH with X Forwarding enabled (or use Putty and Xming if you are using Windows):
brian@T420:~$ **ssh -X 192.168.56.102**

Start Wireshark on the Mininet VM:
mininet@mininet-vm:~$ **sudo wireshark &**
You will see a warning dialog but you can ignore it. Starting Wireshark with root privileges is a security risk but, for our simple testing, we can ignore that — or you can follow the directions in the warning message to set up Wireshark in a more secure way.

Create a display filter for OpenFlow messages. Enter the text, of in the Filter window and click on **Apply**. Now you will see only OpenFlow messages in the Wireshark display, as shown below.



Viewing captured OpenFlow messages in Wireshark

Shut down the project

When it is time to end the project, shut down Mininet and OpenDaylight using the following commands:

On the Mininet VM, stop Mininet and clean up the node, then shut down the VM:

```
mininet> exit
mininet@mininet:~$ sudo mn -c
mininet@mininet:~$ sudo shutdown -h now
```

On the OpenDaylight VM, stop OpenDaylight and shut down the VM:

```
opendaylight-user@root> system:shutdown
brian@odl:~$ sudo shutdown -h now
```

Both VMs should now show that they are stopped in the VirtualBox Manager application.

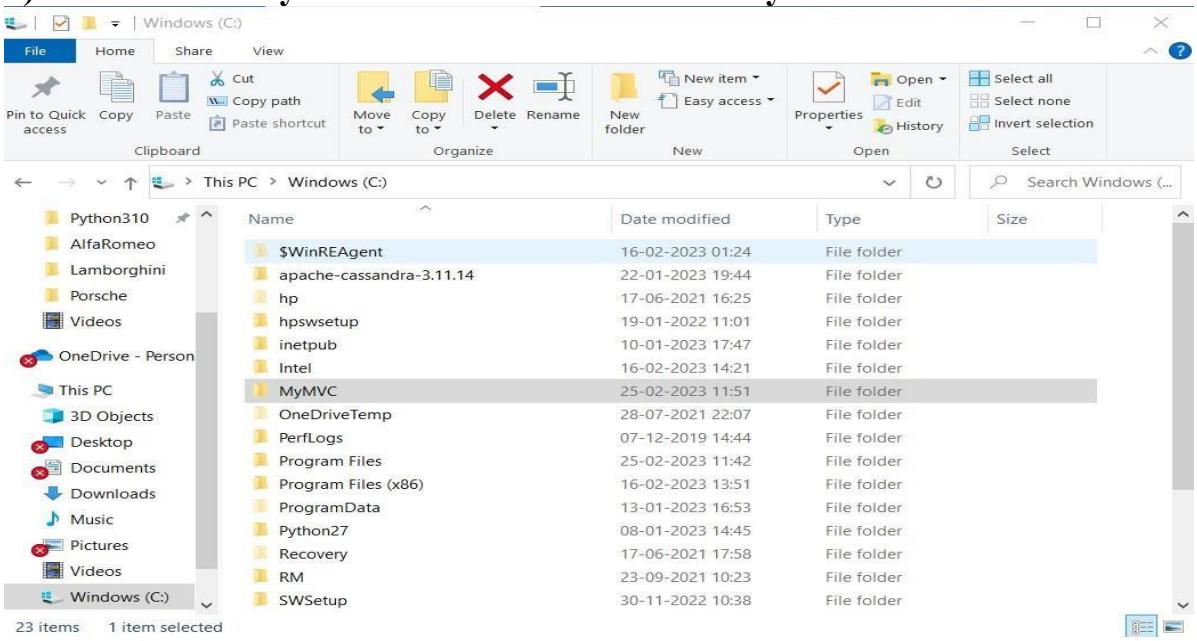
MICROSERVICE ARCHITECTURE

INDEX

PRATICAL 1: Building ASP.NET Core MVC Application

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

2)create folder MyMVC folder in C: drive or any other drive

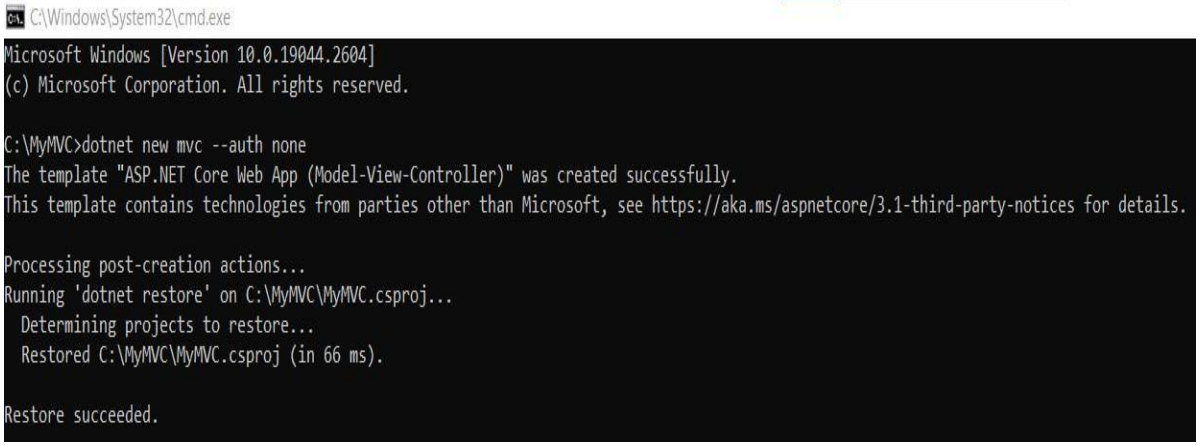


3)open command prompt and perform following operations

Command: to create mvc project

dotnet new mvc --auth none

output:



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

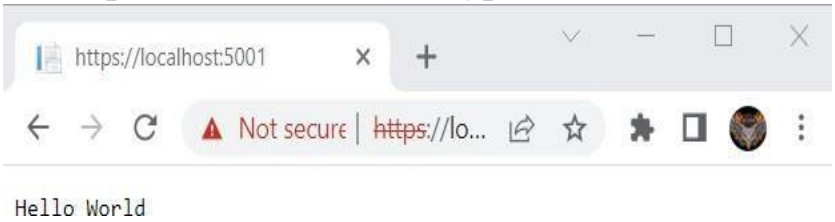
```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Logging;
using MyMVC.Models;
namespace MyMVC.Controllers
{ public class HomeController : Controller
{
```

```
public String Index()
{ return "Hello World"; }
}
}
```

5) Run the project

```
C:\MyMVC>dotnet run
info: Microsoft.Hosting.Lifetime[0]
      Now listening on: https://localhost:5001
info: Microsoft.Hosting.Lifetime[0]
      Now listening on: http://localhost:5000
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: C:\MyMVC
```

Now open browser and and type URL: localhost:5001



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

C:\Windows\System32\cmd.exe

```
C:\MyMVC>dotnet run
info: Microsoft.Hosting.Lifetime[0]
      Now listening on: https://localhost:5001
info: Microsoft.Hosting.Lifetime[0]
      Now listening on: http://localhost:5000
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: C:\MyMVC
info: Microsoft.Hosting.Lifetime[0]
      Application is shutting down...

C:\MyMVC>
```

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

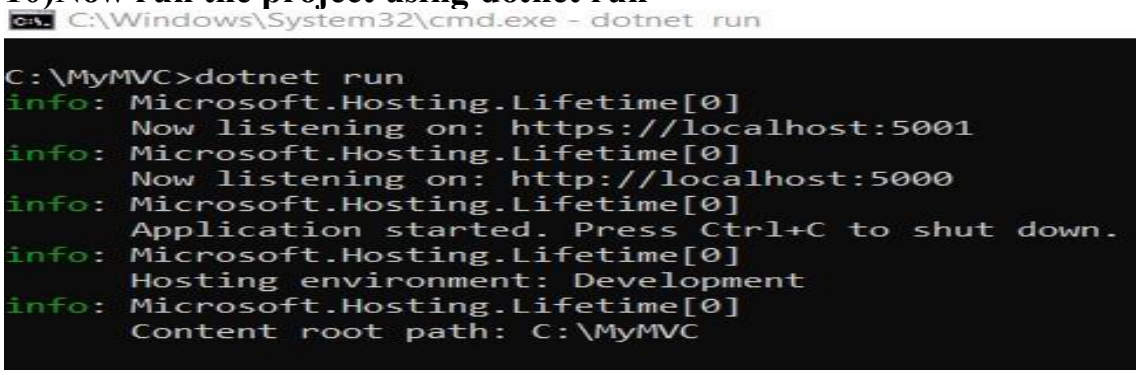
```
@{
ViewData["Title"] = "Home Page";
}
<div>
Symbol: @Model.Symbol <br/>
```

```
Price: $@Model.Price <br/>
</div>
```

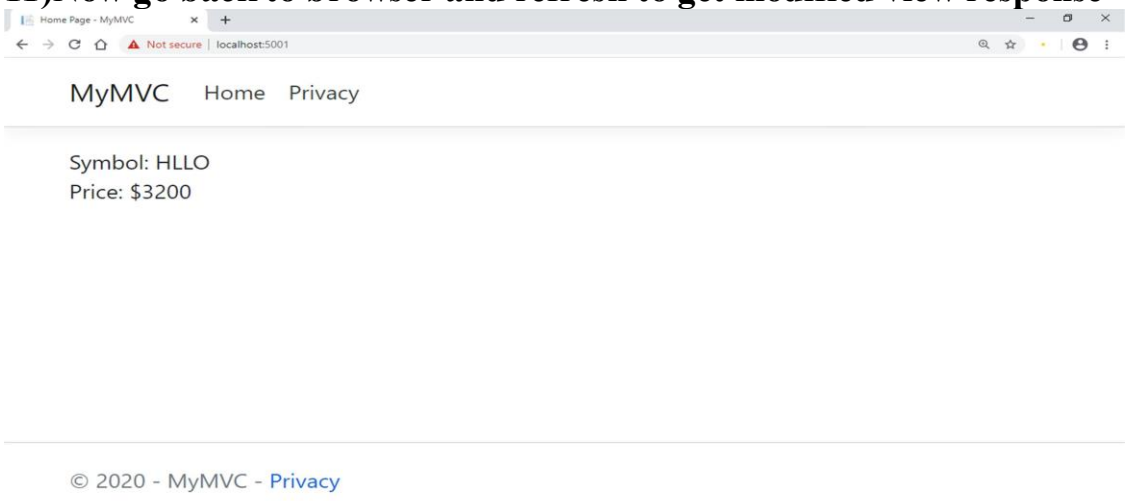
9)Now modify HomeController.cs file to match following:

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
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



PRATICAL 2: 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 version 3.0 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:

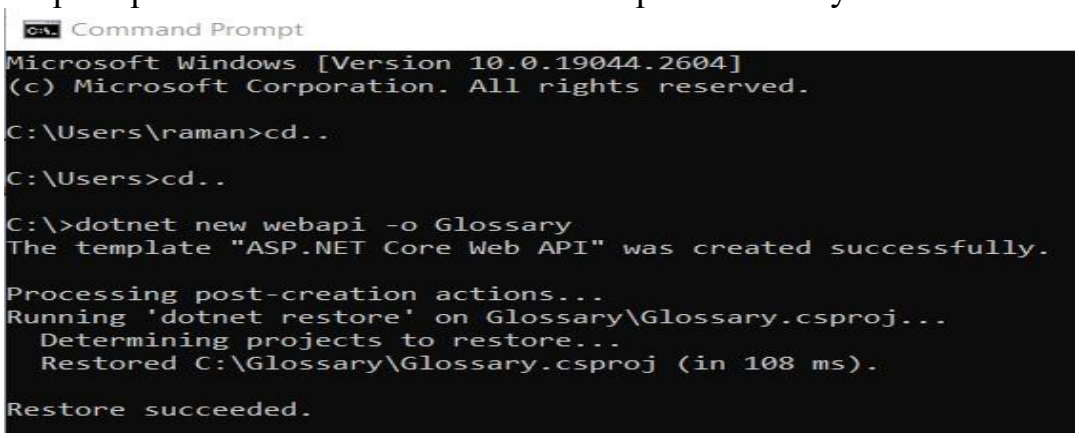
Command prompt

> dotnet

Create your web API

1. Open two command prompts

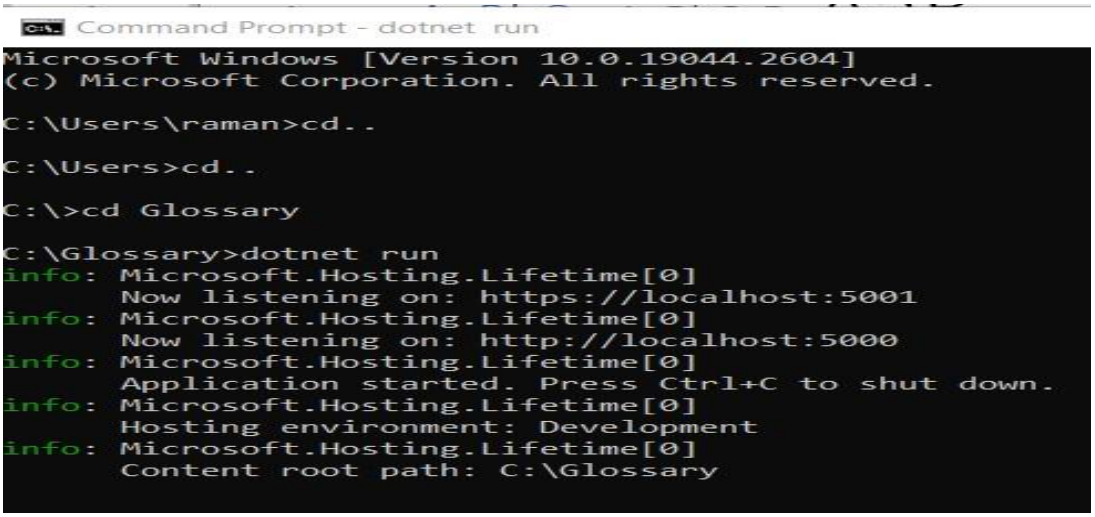
Command prompt 1: Command: dotnet new webapi -o Glossary



CMD-2:

cd Glossary

dotnet run



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

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

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

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

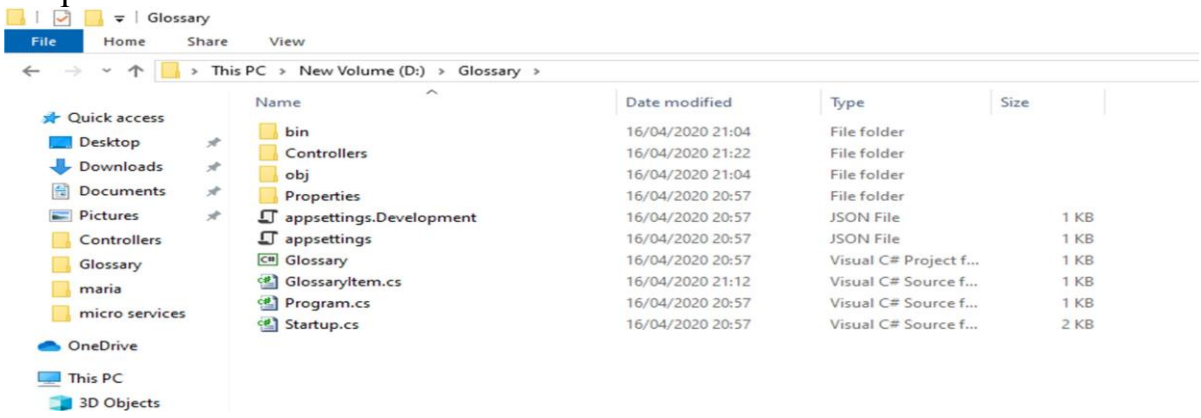
2) D:\Glossary\Controllers\ GlossaryController.cs (type it in notepad and save as all files)

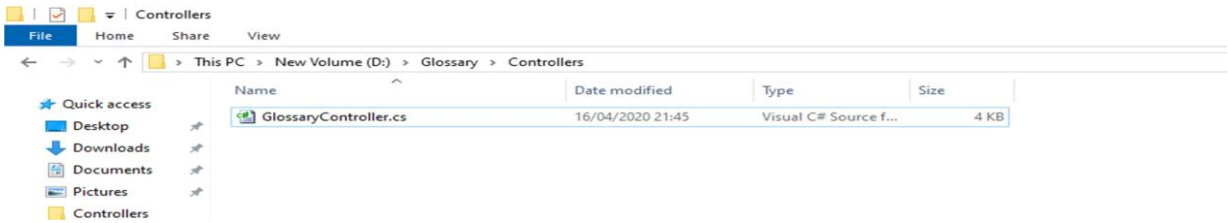
```
using System;
using System.Collections.Generic;
using Microsoft.AspNetCore.Mvc;
using System.IO;
namespace Glossary.Controllers
{
 [ApiController]
 [Route("api/[controller]")]
 public class GlossaryController: ControllerBase
 {
 private static List<GlossaryItem> Glossary = new List<GlossaryItem> {
 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:

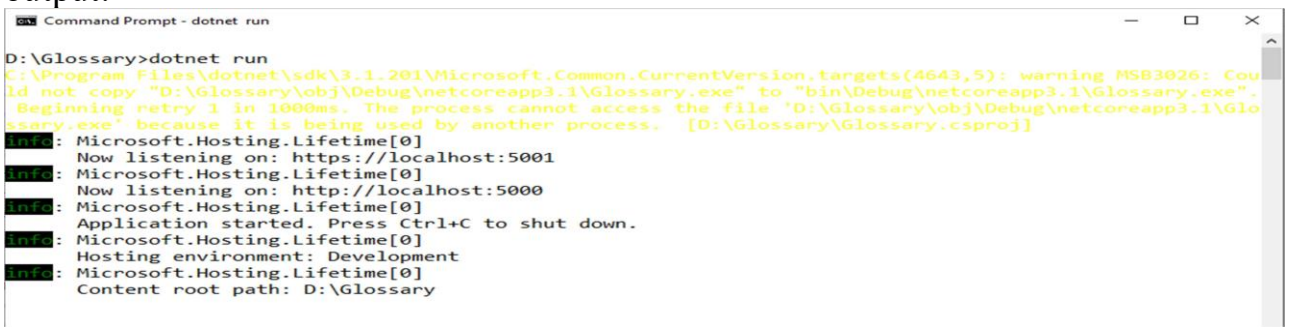




3.Now stop running previous dotnet run on command prompt 1 using Ctrl+C. and Run it again for new code.

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 "ContentType:application/json" https://localhost:5001/api/glossary**

Output:

4)Update Item:

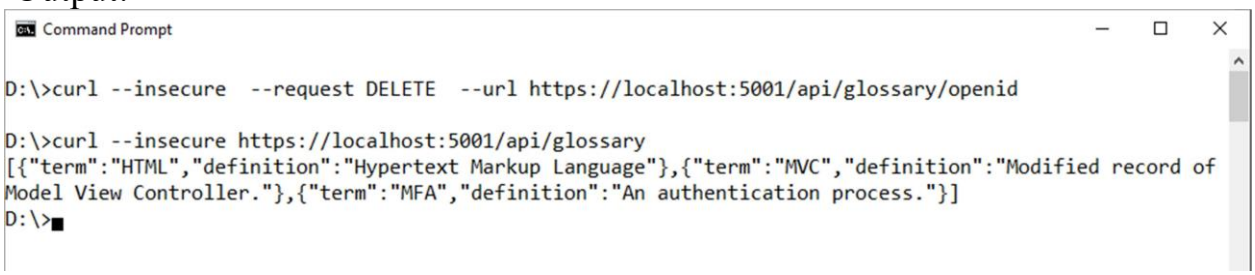
Command: **curl --insecure -X PUT -d "{\"term\": \"MVC\", \"definition\":\"Modified record of Model View Controller.\"}" -H "Content-Type:application/json" https://localhost:5001/api/glossary**

Output:

5) Delete Item:

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

Output:



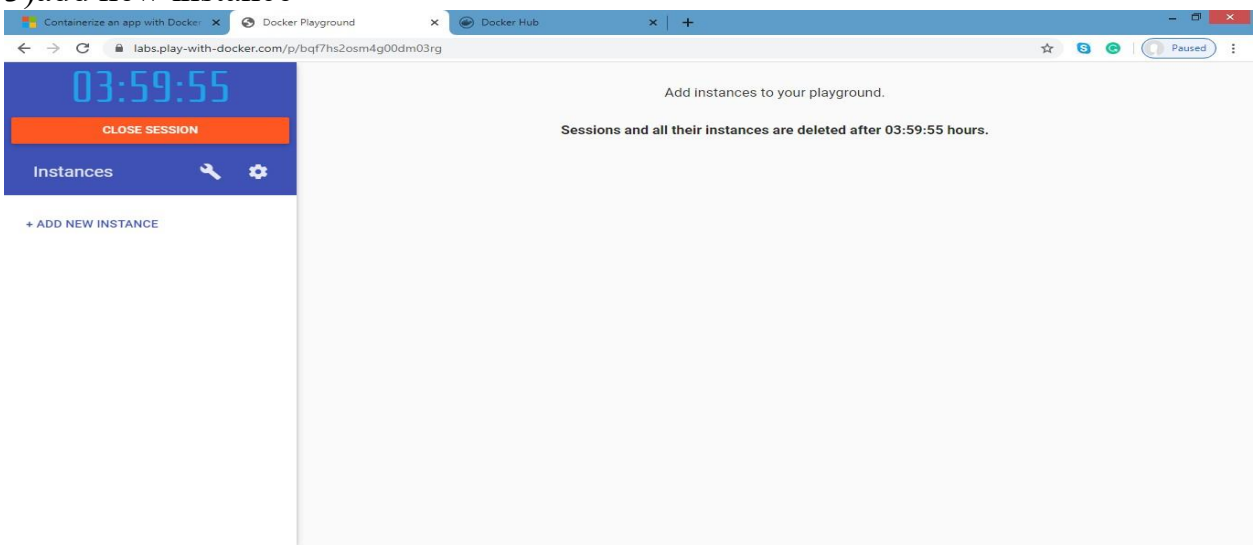
PRATICAL 3: Working with Docker, Docker Commands, Docker Images and Containers

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:

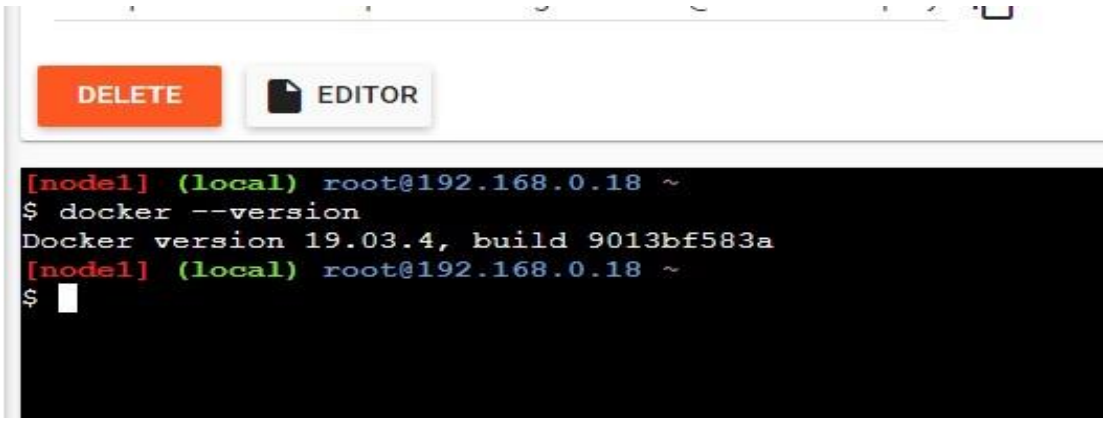
Method1:

To pull and push images using docker

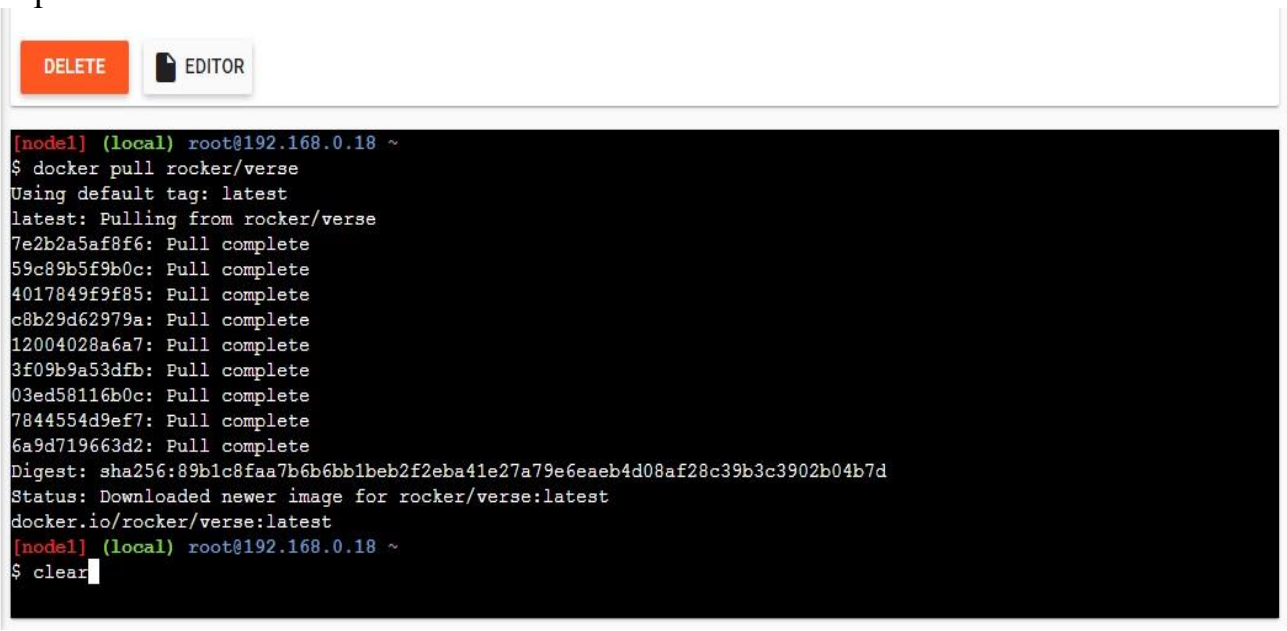
Command: to check docker version

docker –version

output:

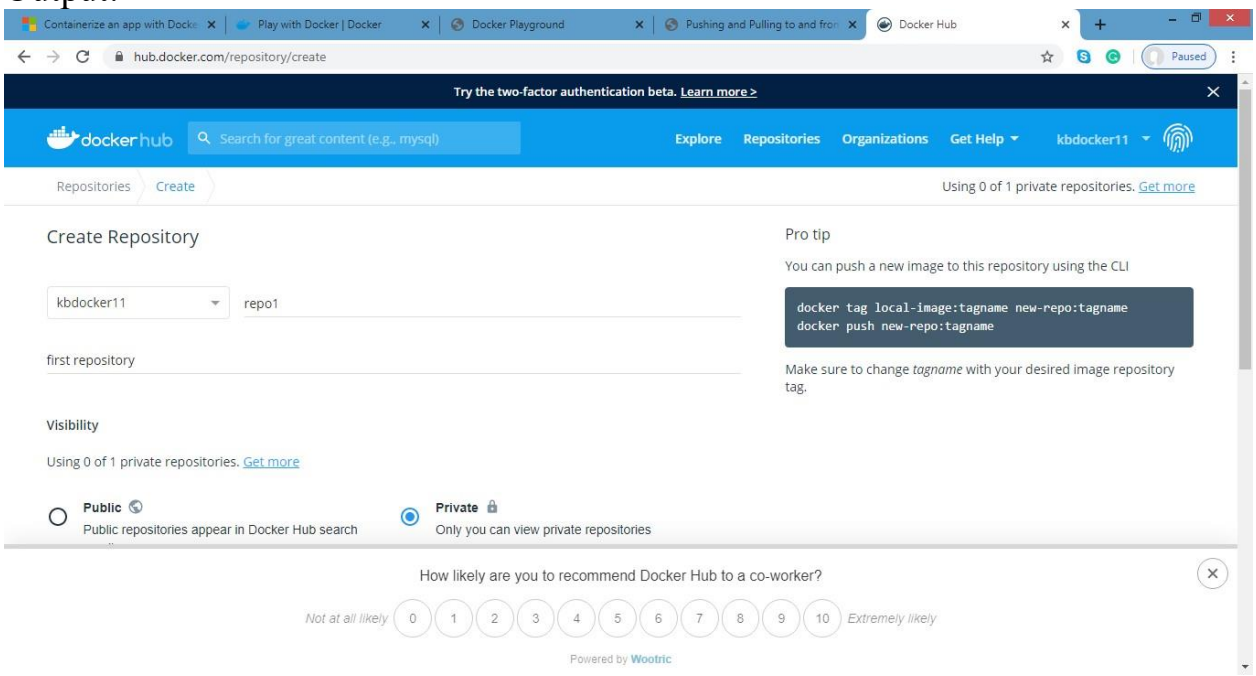


Command: to pull readymade image
docker pull rocker/verse
output:

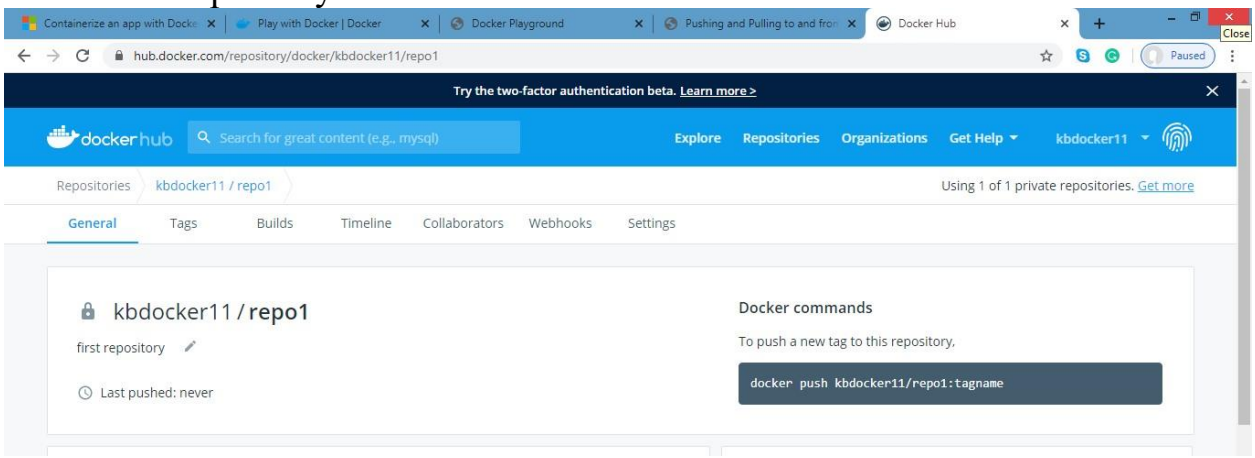


Command: to check images in docker
docker images
output:

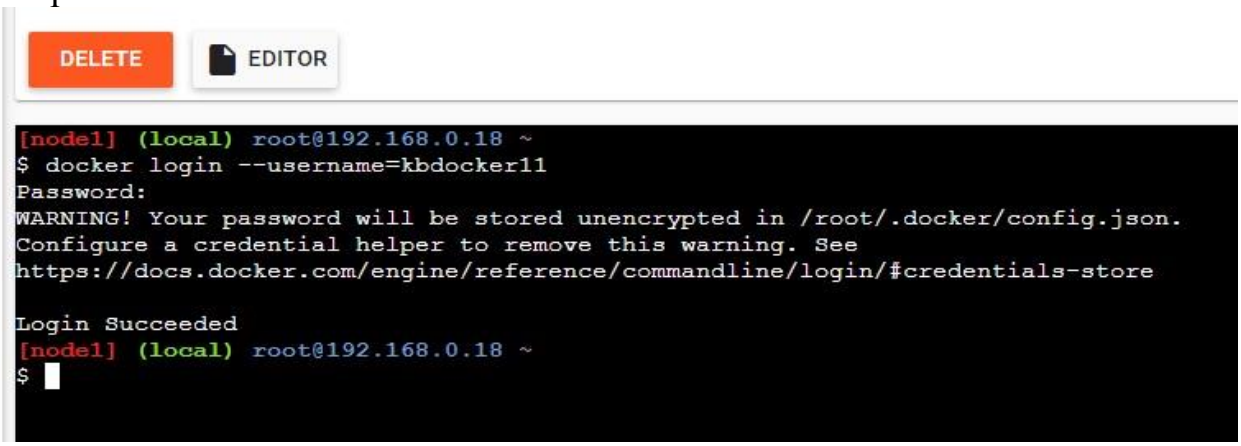
Now Login to docker hub and create repository
Output:



Click on Create button
Now check repository created

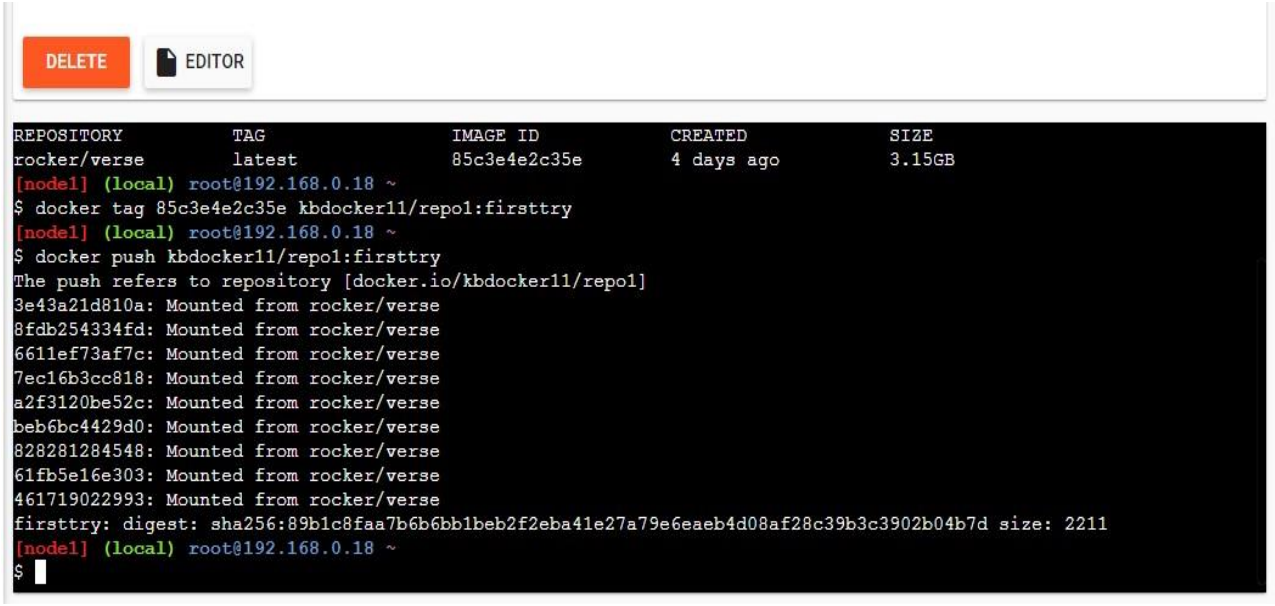


Command: to login to your docker account
docker login –username=kbdocker11
password:
note: kbdocker11 is my docker ID . You will use your docker ID here. And enter your password .
Output:

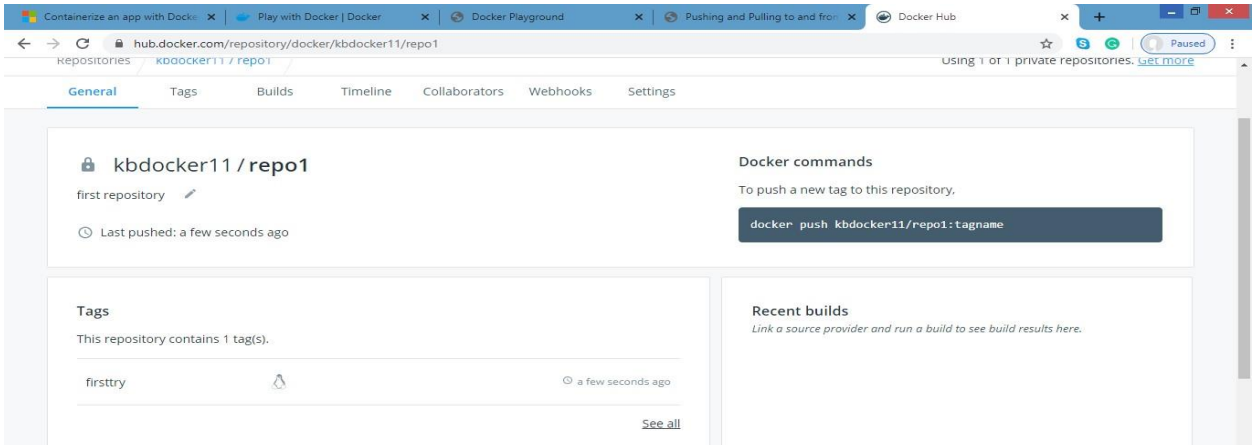


Command : to tag image
docker tag 8c3e4e2c3e kbdocker11/repo1:firsttry
note: here 8c3e4e2c3e this is image id which you can get from docker images command.
Output:

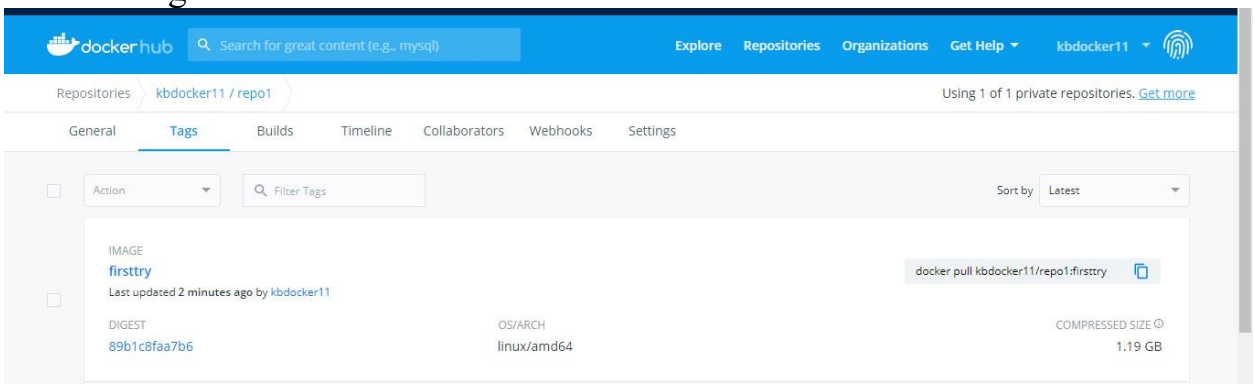
Command: to push image to docker hub account
docker push kbdocker11/repo1:firsttry
note: firsttry is tag name created above.
Output:



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:

DELETE EDITOR

```
[node1] (local) root@192.168.0.18 ~
$ cat > Dockerfile <<EOF
> FROM busybox
> CMD echo "Hello world! This is my first Docker image."
> EOF
[node1] (local) root@192.168.0.18 ~
$ docker build -t kbdocker11/repo2
"docker build" requires exactly 1 argument.
See 'docker build --help'.

Usage:  docker build [OPTIONS] PATH | URL | -

Build an image from a Dockerfile
[node1] (local) root@192.168.0.18 ~
$
```

Command : to build image from docker file

docker build –t kbdocker11/repo2 .

Output:

DELETE EDITOR

```
[node1] (local) root@192.168.0.18 ~
$ docker build -t kbdocker11/repo2 .
Sending build context to Docker daemon  38.15MB
Step 1/2 : FROM busybox
latest: Pulling from library/busybox
e2334dd9fee4: Pull complete
Digest: sha256:a8cf7ff6367c2afa2a90acd081b484cbded349a7076e7bdf37a05279f276bc12
Status: Downloaded newer image for busybox:latest
 ---> be5888e67be6
Step 2/2 : CMD echo "Hello world! This is my first Docker image."
 ---> Running in fa7c9a33b9bc
Removing intermediate container fa7c9a33b9bc
 ---> 32be029659d1
Successfully built 32be029659d1
Successfully tagged kbdocker11/repo2:latest
[node1] (local) root@192.168.0.18 ~
$
```

Command: to check docker images

docker images

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
```

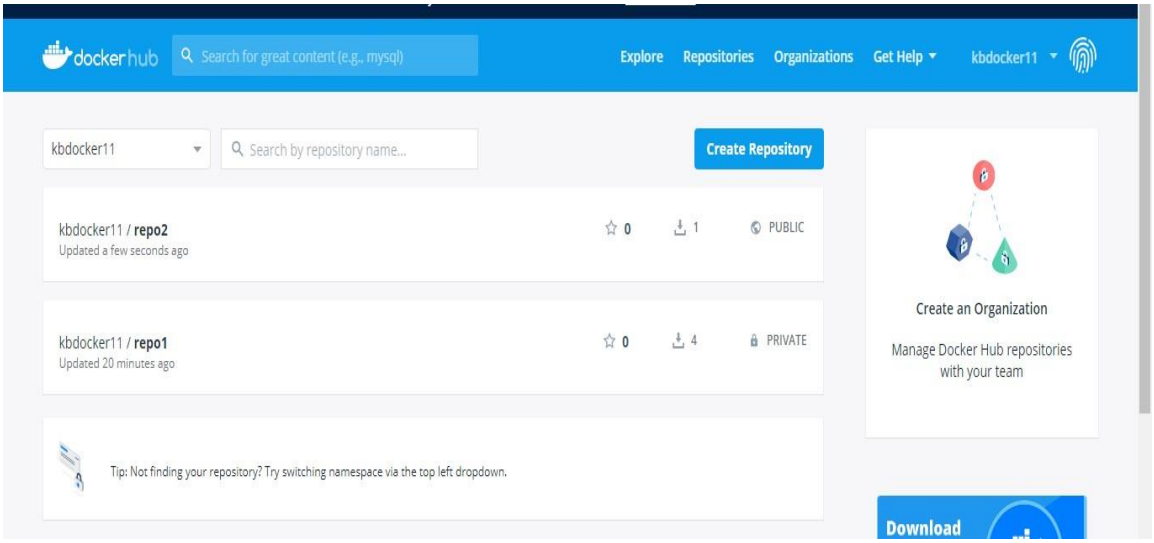
Command: to push image to docker hub

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



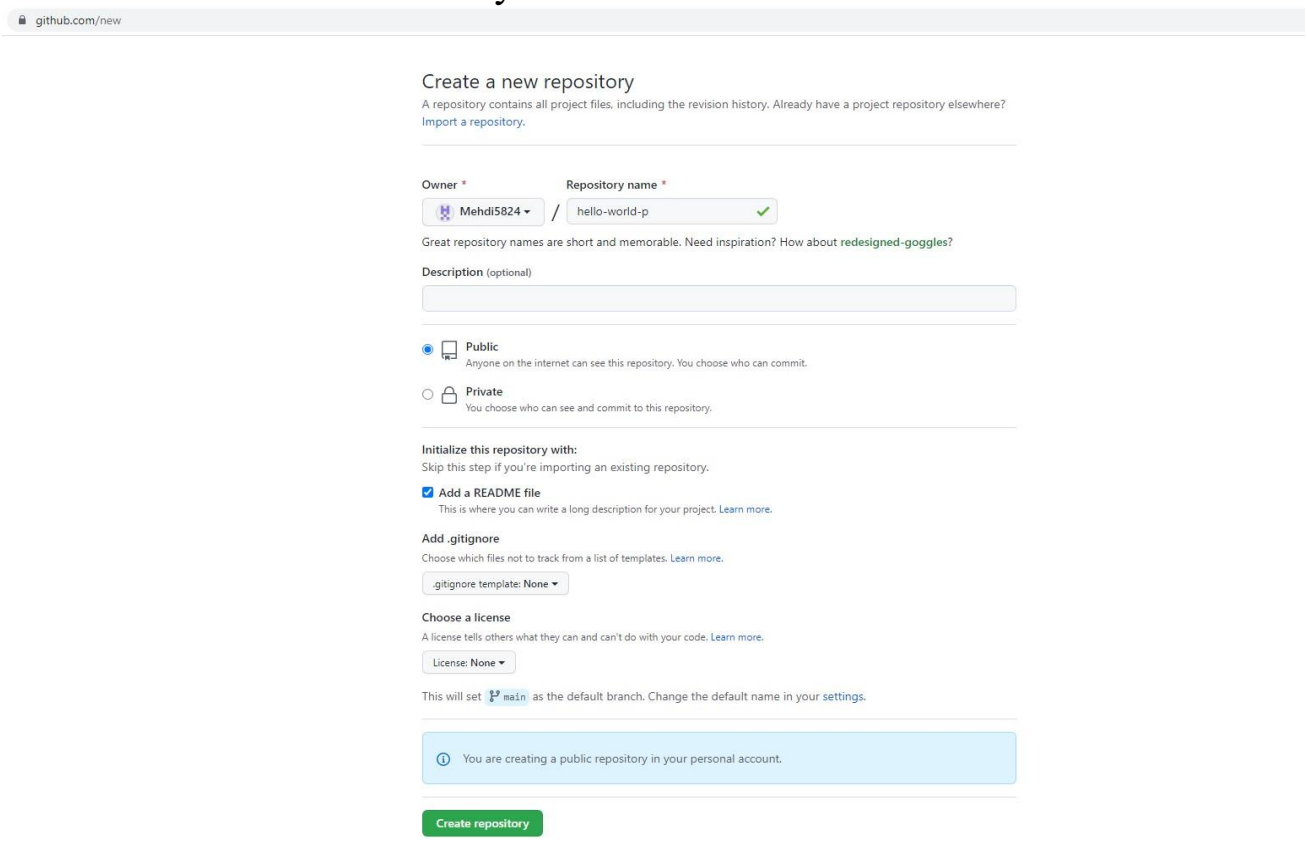
command: to run docker image:
docker run kbdocker11/repo2
output:

Now close session

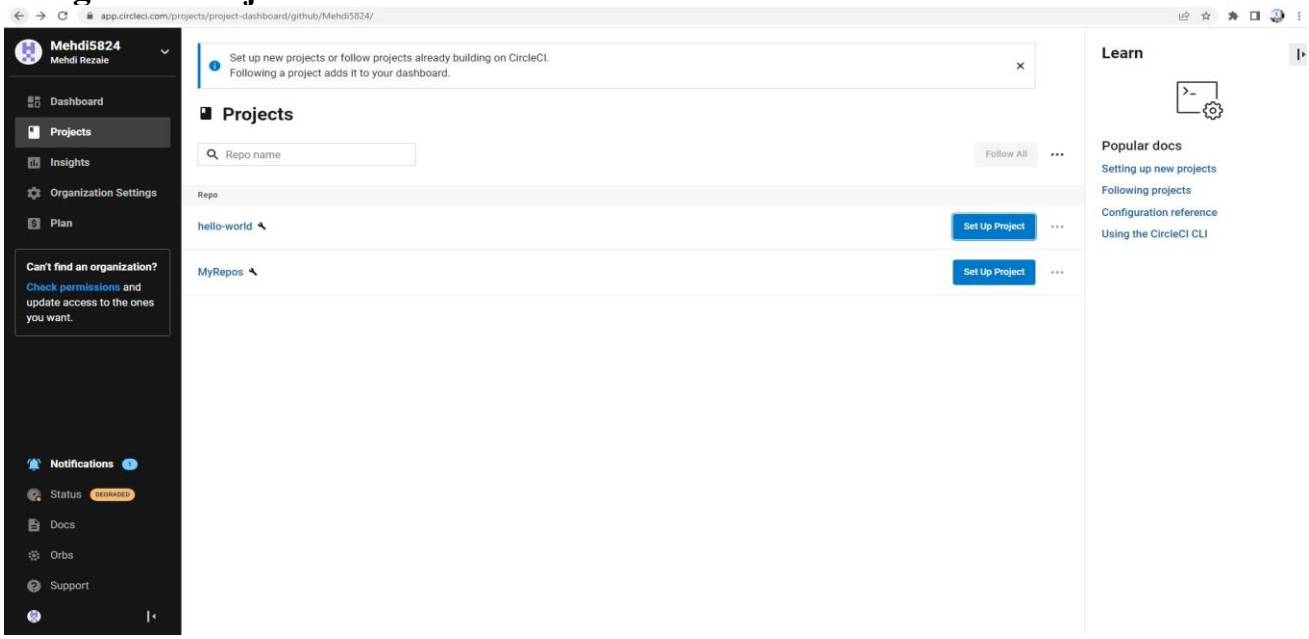
PRATICAL 4: Working with Circle CI for continuous integration.

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.

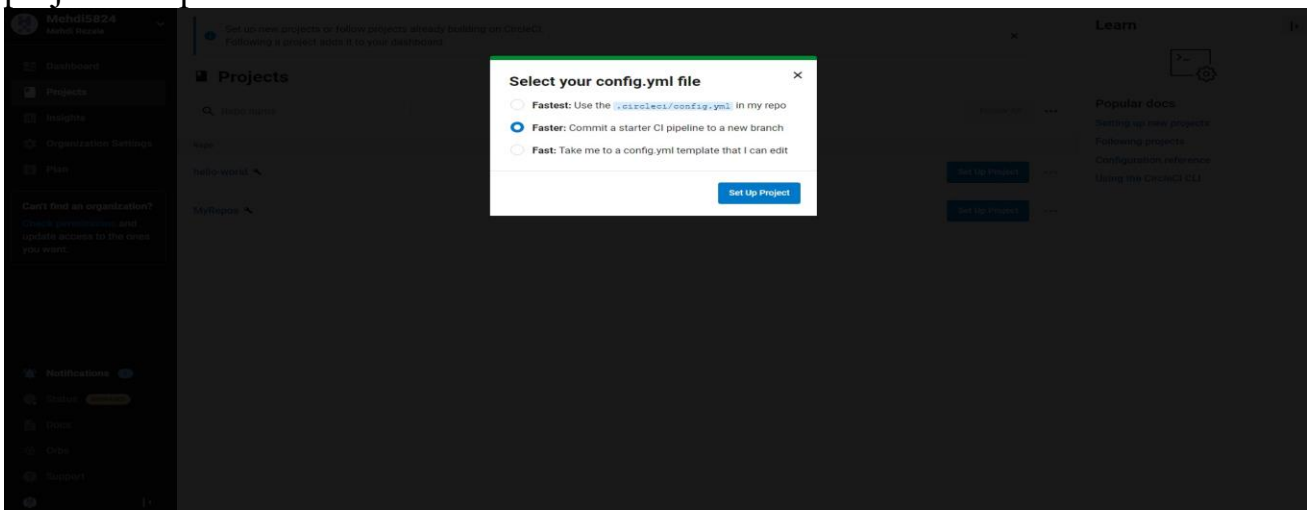


Step 2 - Set up CircleCI

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-hello-
workflow).

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

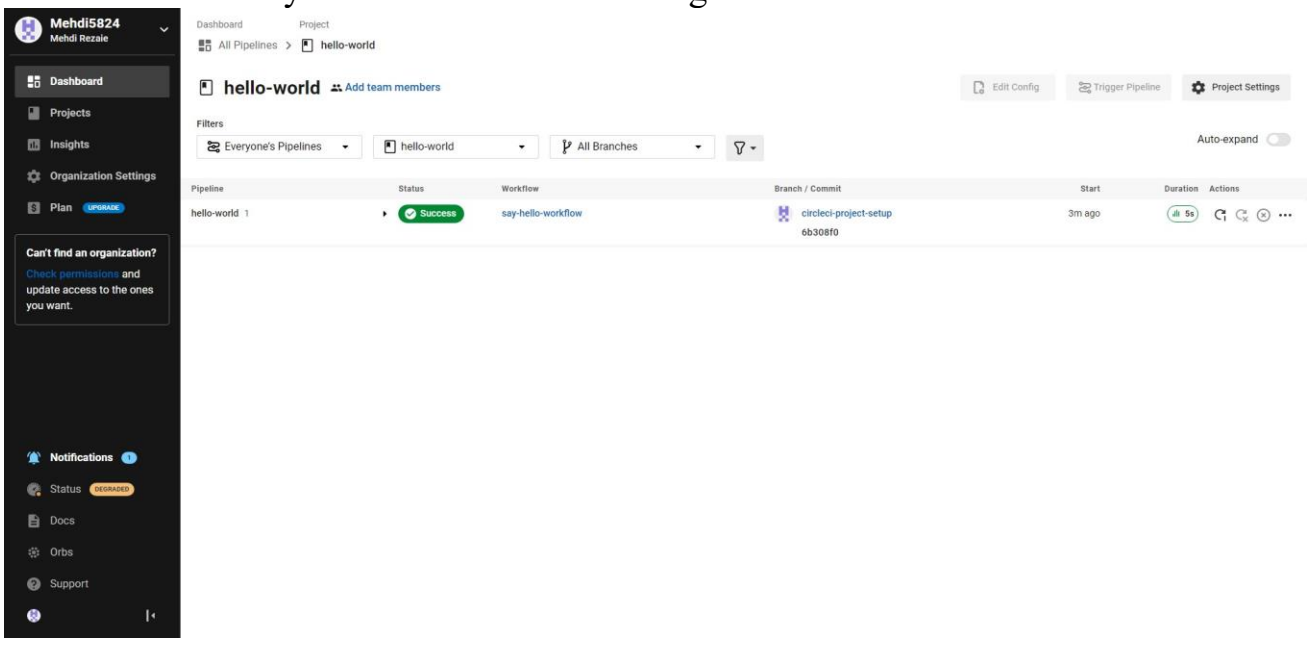
a. Spin up environment

b. Preparing environment variables

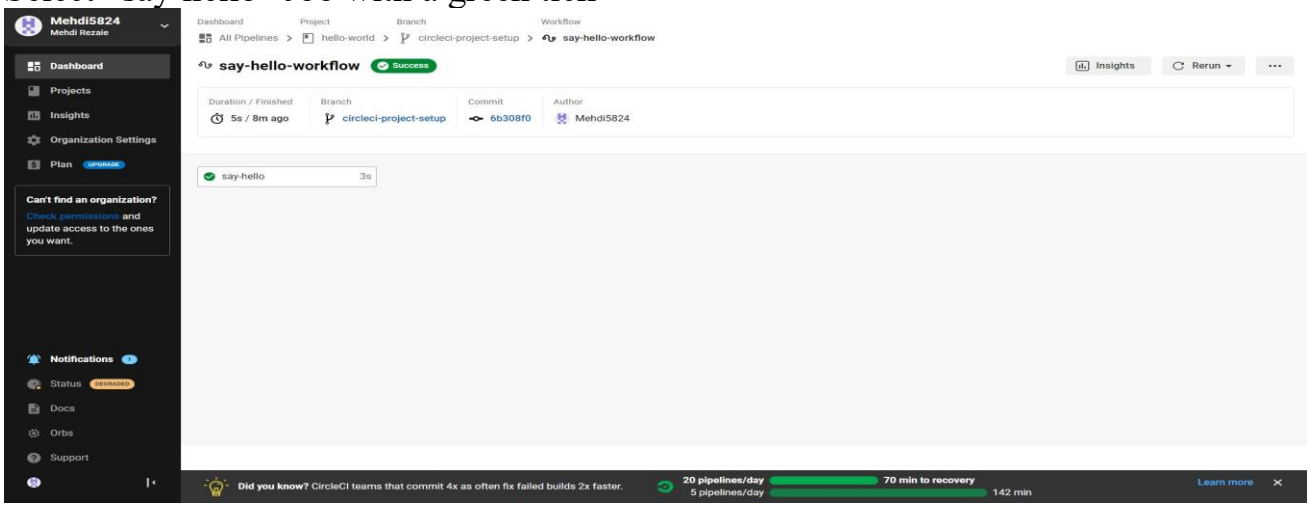
c. Checkout code

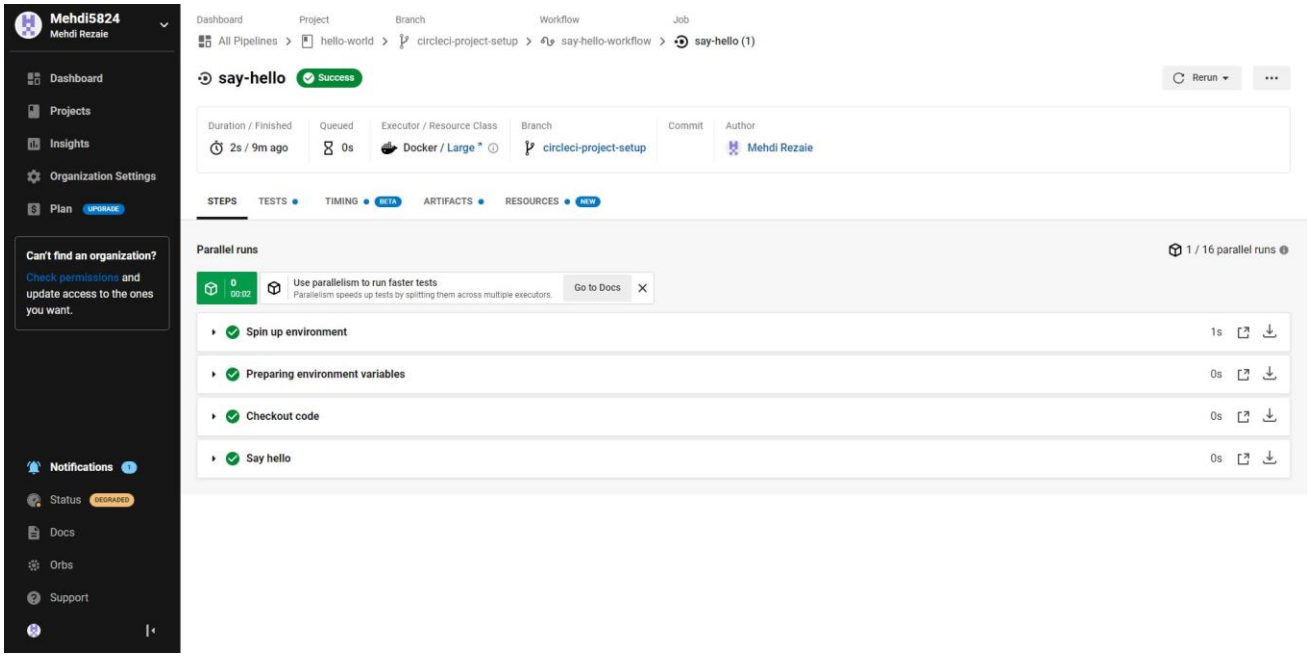
d. Say hello

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

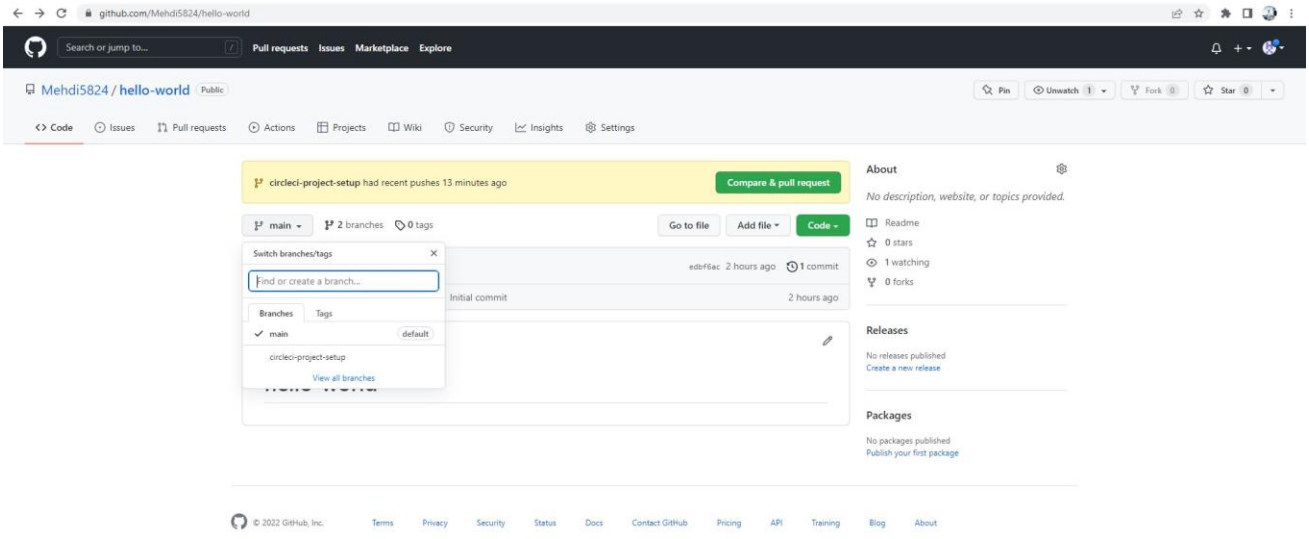


Select “say-hello” Job with a green tick





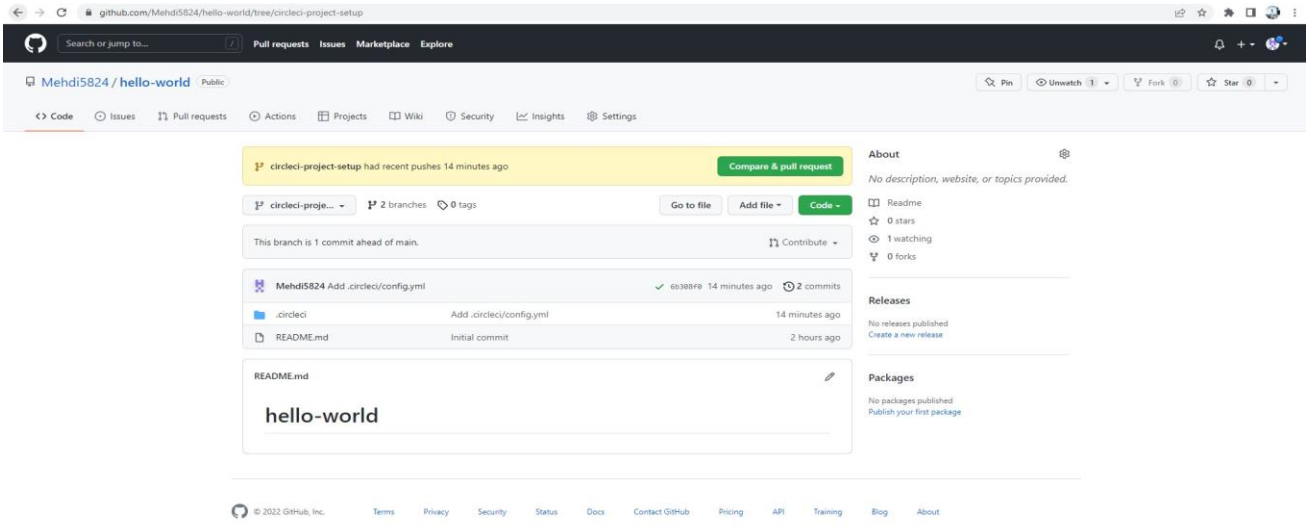
Select Branch and option circleci-project-setup

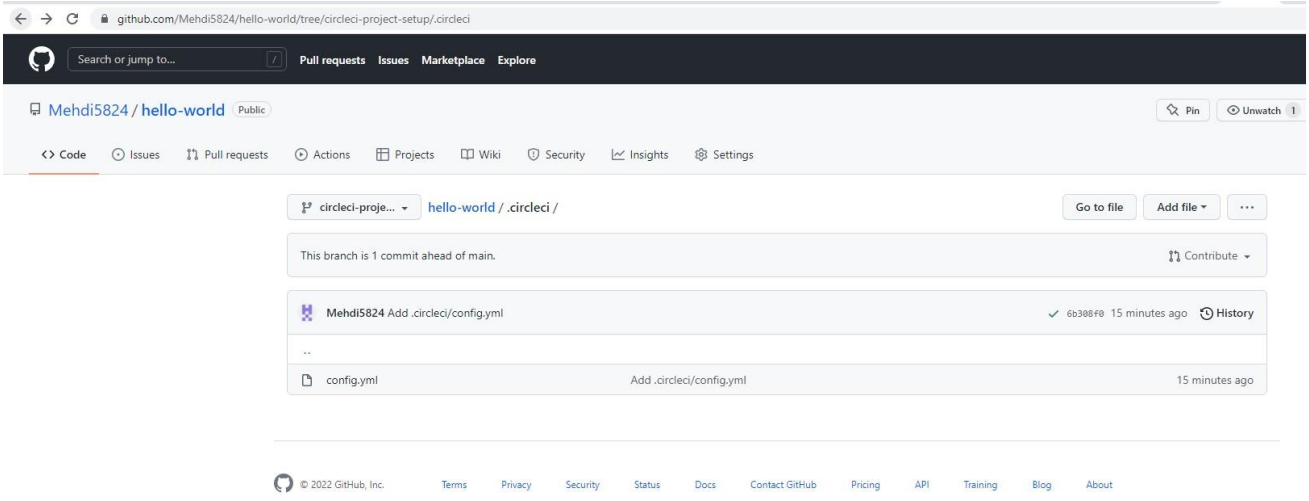


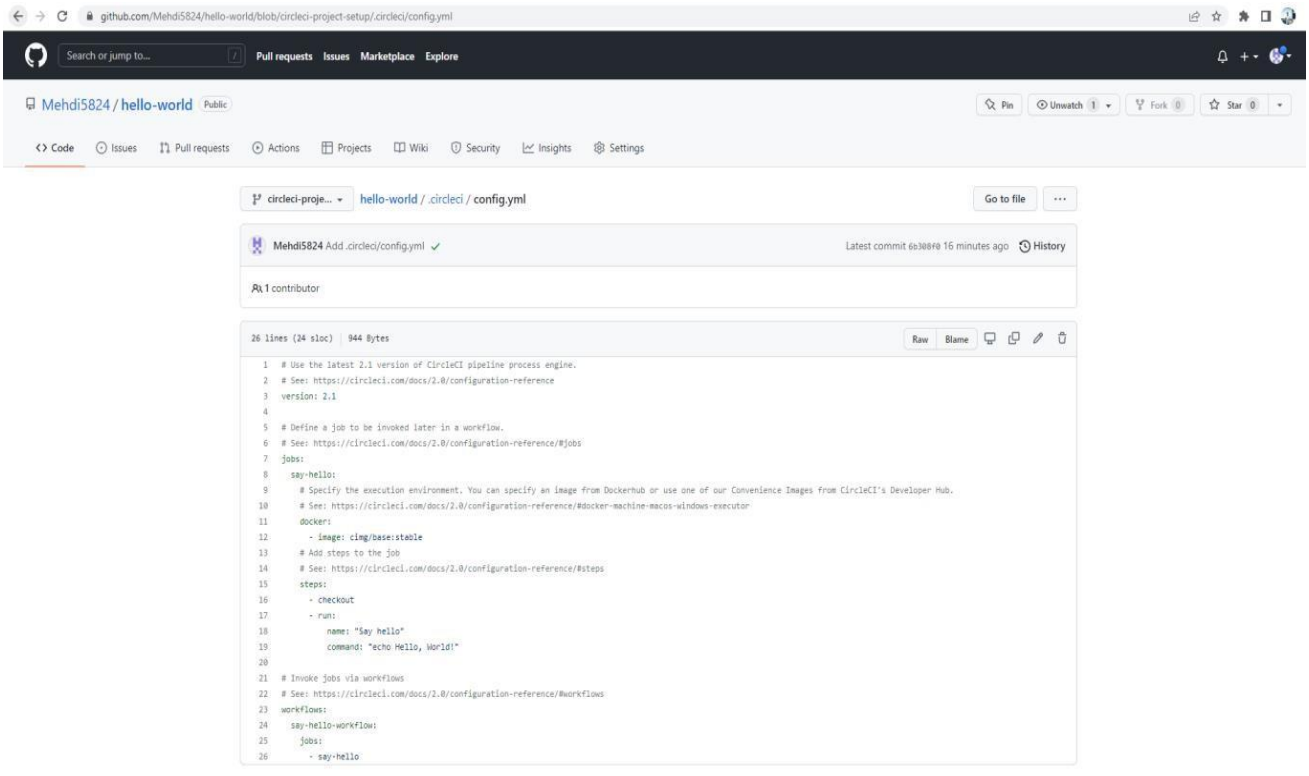
Step 4 - Break your build

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.







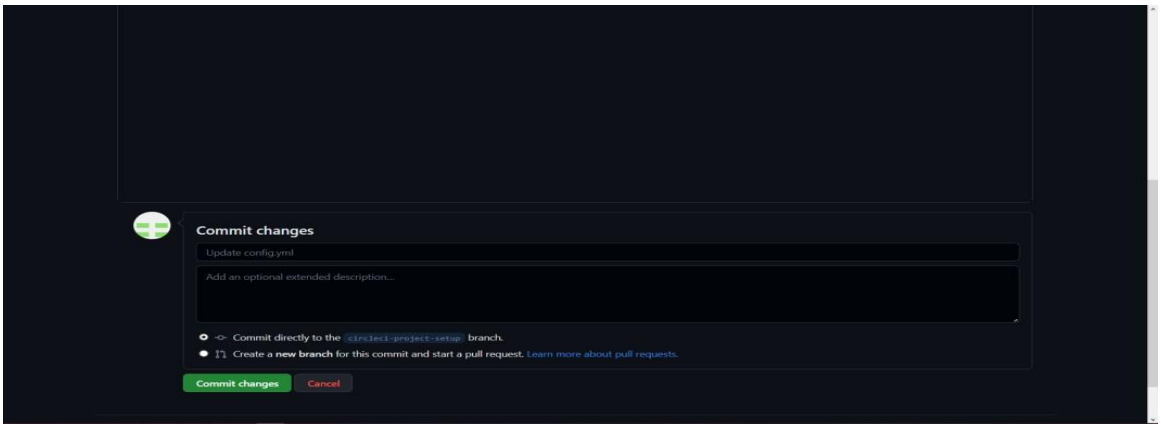
Replace the existing code with new code:

```
version: 2.1
orbs:
        node: cricleci/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
```

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

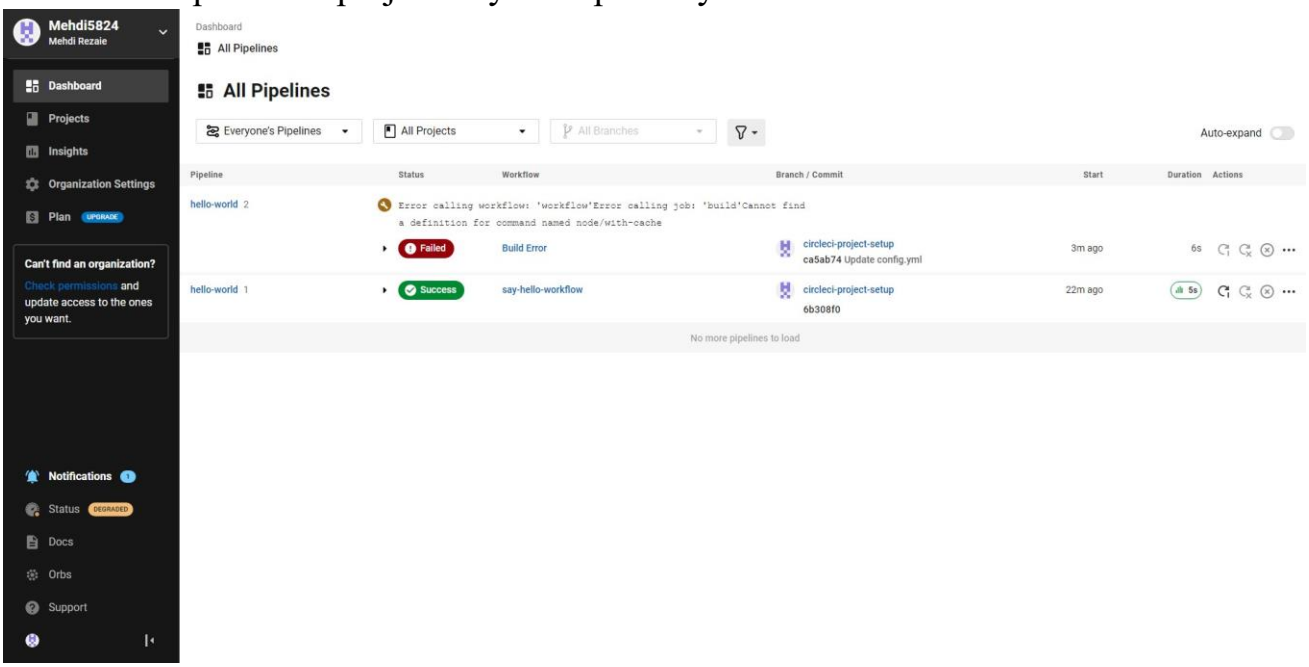
```
version: 2.1
orbs:
  node: circleci/node@4.7.0
jobs:
  build:
    executor:
      name: node/default
      tag: '10.4'
    steps:
      - checkout
      - node/with-cache:
          steps:
            - run: npm install
      - run: npm run test
```

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

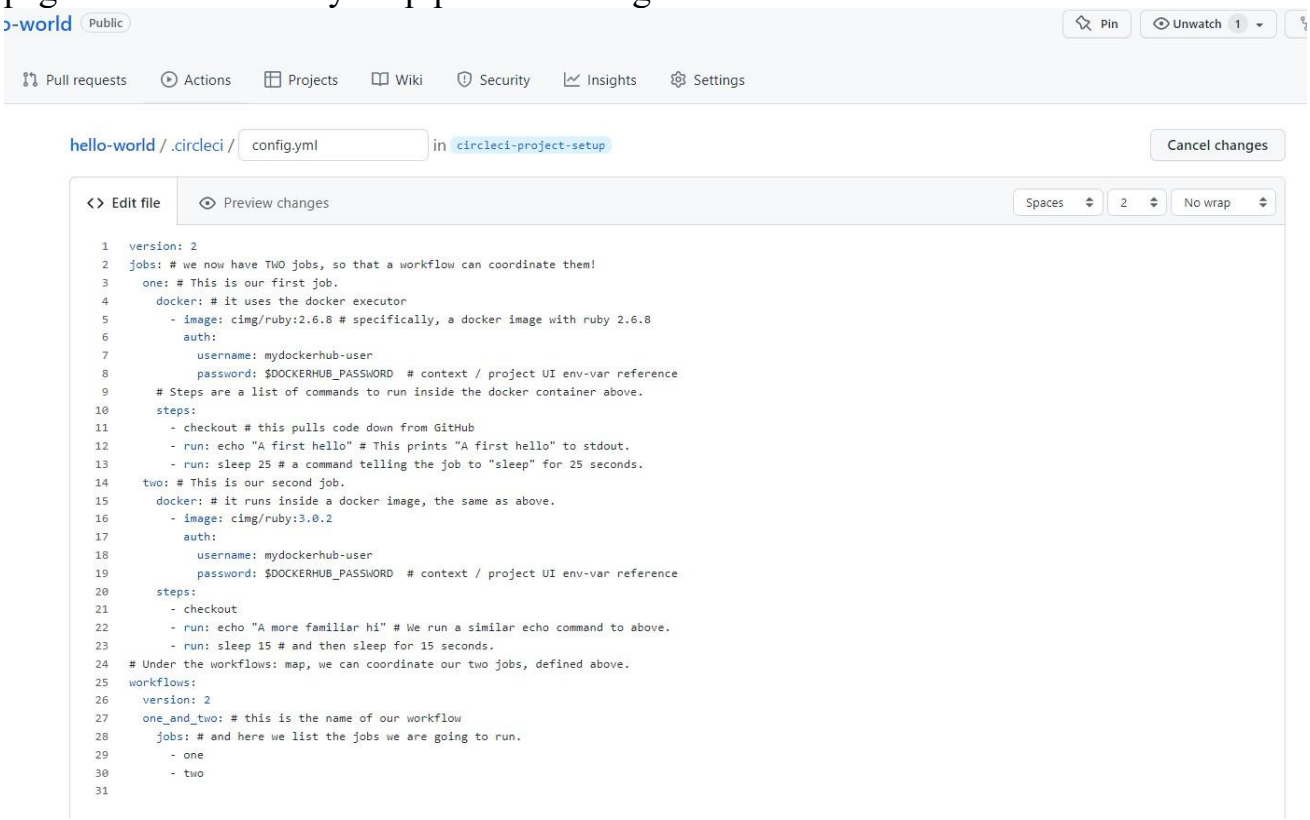
You do not have to use orbs to use CircleCI. The following example details how to create a custom configuration that also uses the workflow feature of CircleCI.

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.

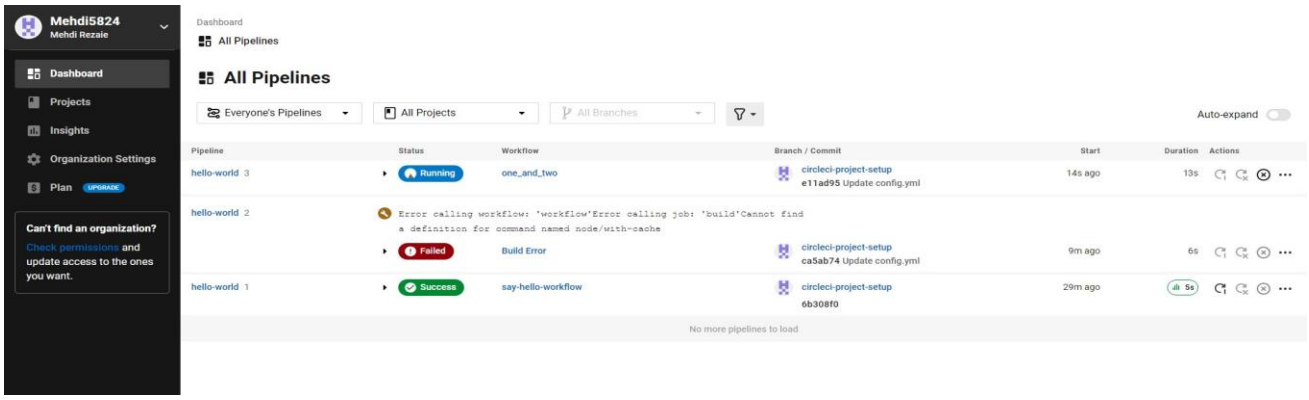
```
version: 2.1
jobs:
  one:
    docker:
     - image: cimg/ruby:2.6.8
       auth:
         username: mydockerhub-user
         password: $DOCKERHUB_PASSWORD
    steps:
     - checkout
     - run: echo "A first hello"
     - run: sleep 25
  two:
    docker:
      - image: cimg/ruby:3.0.2
        auth:
          username: mydockerhub-user
          password: $DOCKERHUB_PASSWORD
    steps:
     - checkout
```

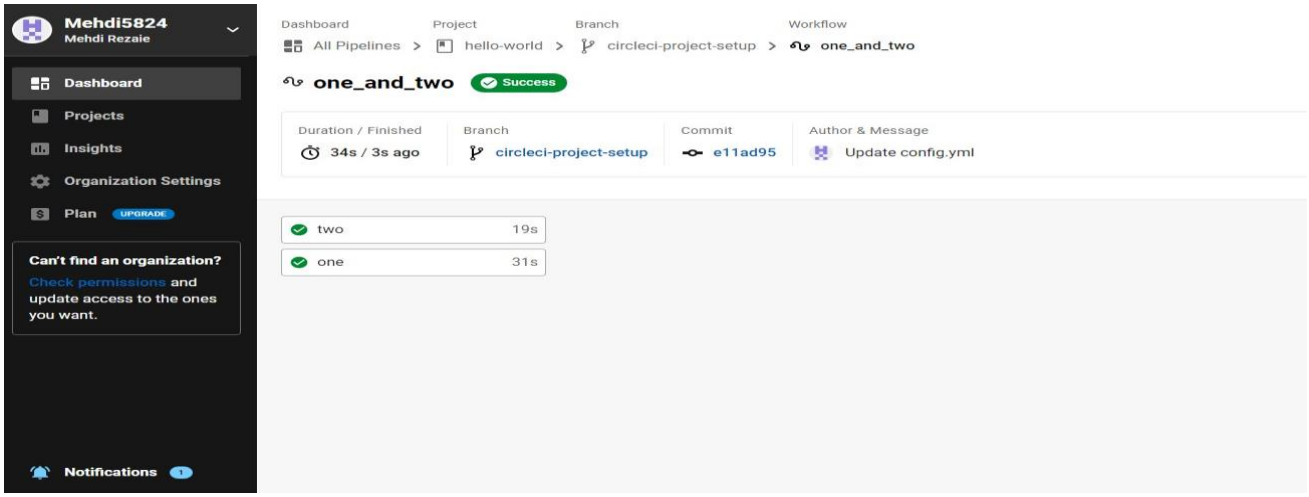
```
      - run: echo "A more familiar hi"
      - run: sleep 15
workflows:
  version: 2.1
  one_and_two:
     jobs:
       - one
       - two
```

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



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

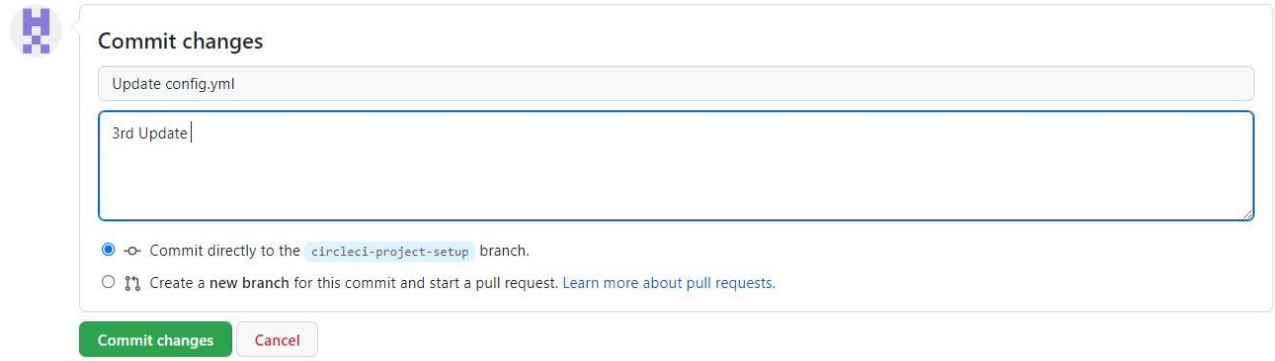
Each workflow has an associated workspace which can be used to transfer files to downstream jobs as the workflow
progresses. You can use workspaces to pass along data that is unique to this run and which is needed for downstream
jobs. Try updating config.yml to the following:

```
version: 2.1
jobs:
  one:
    docker:
      - image: cimg/ruby:3.0.2
        auth:
          username: mydockerhub-user
          password: $DOCKERHUB_PASSWORD
    steps:
      - checkout
      - run: echo "A first hello"
      - run: mkdir -p my_workspace
      - run: echo "Trying out workspaces" > my_workspace/echo-output
      - persist_to_workspace:
          root: my_workspace

          paths:
           - echo-output
  two:
    docker:
      - image: cimg/ruby:3.0.2
        auth:
          username: mydockerhub-user
          password: $DOCKERHUB_PASSWORD
    steps:
      - checkout
      - run: echo "A more familiar hi"
      - attach_workspace:
          at: my_workspace
      - run: |
          if [[ $(cat my_workspace/echo-output) == "Trying out workspaces"]];
            then echo "It worked!";
          else
            echo "Nope!"; exit 1
          fi
workflows:
  version: 2.1
  one_and_two:
```

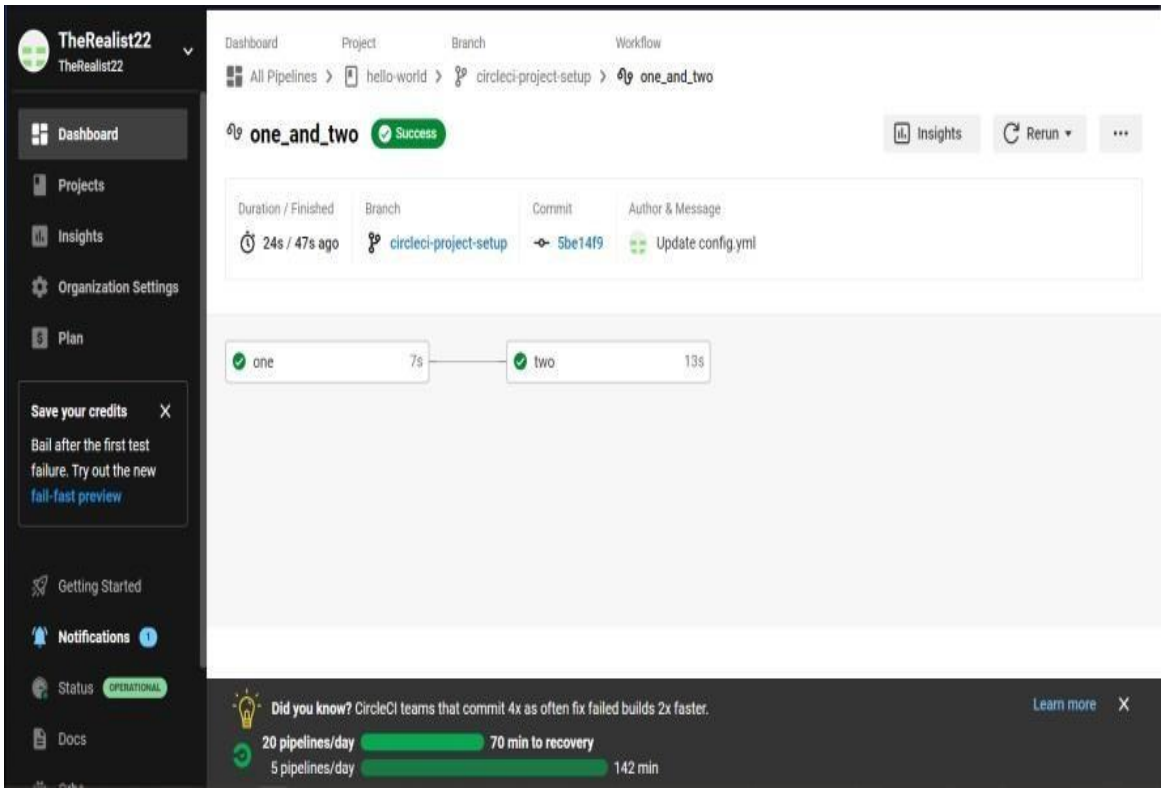
jobs:
- one
- two:
requires:
- one

```
version: 2.1
jobs:
  one:
    docker:
      - image: cimg/ruby:3.0.2
        auth:
          username: mydockerhub-user
          password: $DOCKERHUB_PASSWORD
    steps:
      - checkout
      - run: echo "A first hello"
      - run: mkdir -p my_workspace
      - run: echo "Trying out workspaces" > my_workspace/echo-output
      - persist_to_workspace:
          root: my_workspace

          paths:
           - echo-output
  two:
    docker:
      - image: cimg/ruby:3.0.2
        auth:
          username: mydockerhub-user
          password: $DOCKERHUB_PASSWORD
    steps:
      - checkout
      - run: echo "A more familiar hi"
      - attach_workspace:
            at: my_workspace
      - run: |
          "Trying out workspaces" > my_workspace/echo-output
            then echo "It worked!";
          else
            echo "Nope!";
          fi
workflows:
  version: 2.1
  one_and_two:
```

```
jobs:
  - one
  - two:
      requires:
        - one
```



Finally, your workflow with the jobs running should look like this



PRATICAL 5: Creating Microservice with ASP.NET Core.

(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
output:

2) Remove existing weatherforecast files both model and controller files.
3) Add new files as follows:
a) 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;
}
}
}
```

b) 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<Member> Members { get; set; }

public Team()
{
```

```
this.Members = new List<Member>();
}
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;
}
}
}
```

c)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);
}
}
}
}
```

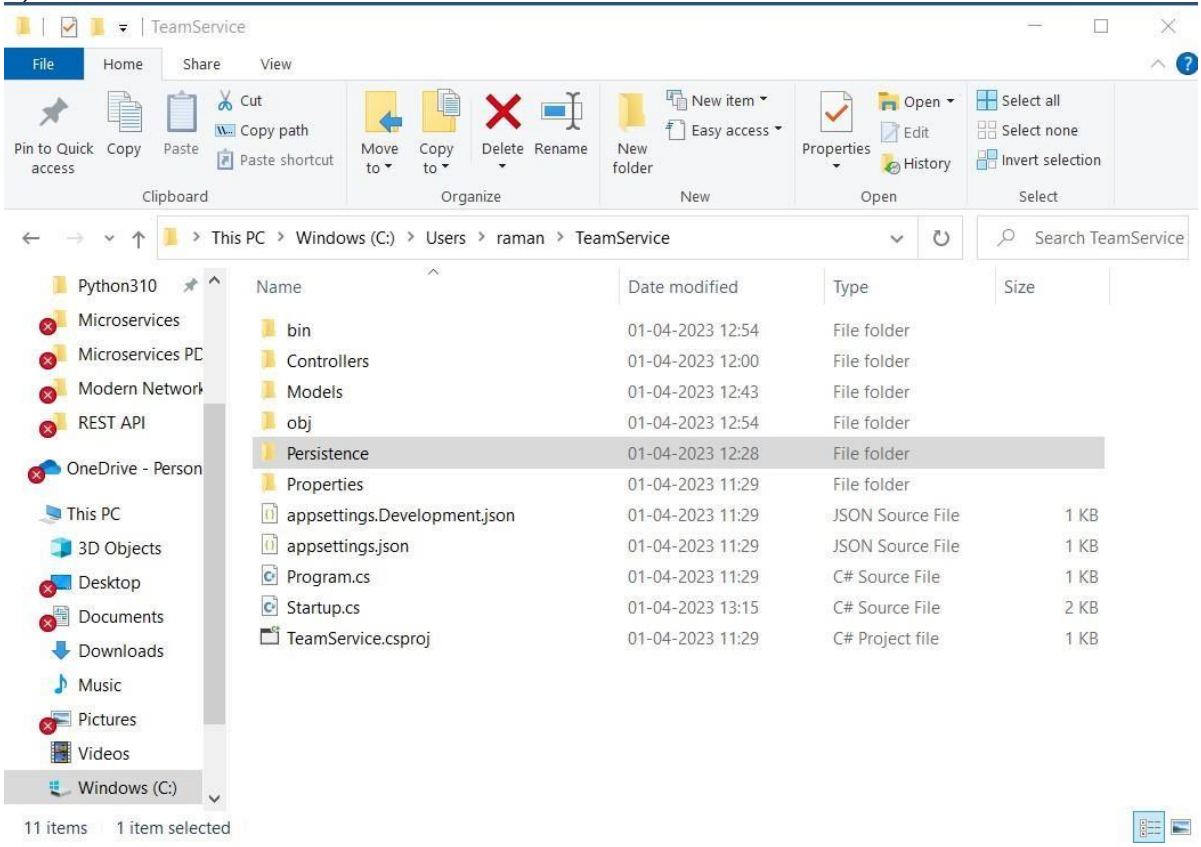
d) add MembersController.cs file to "D:\TeamService\Controllers" folder

```
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 updatedMember, 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 {
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;
}
}
}
```

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



5)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<Team> List();
Team Get(Guid id);
Team Add(Team team);
Team Update(Team team);
Team Delete(Guid id);
}
}
```

6)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<Team> teams;
public MemoryTeamRepository() {
```

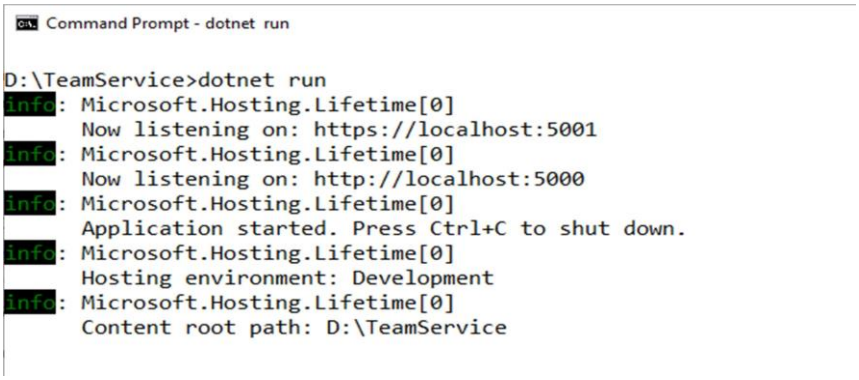
```
if(teams == null) {
teams = new List<Team>();
}
}
public MemoryTeamRepository(ICollection<Team> teams)
{
MemoryTeamRepository.teams = teams;
}
public IEnumerable<Team> List()
{
return teams;
}
public Team Get(Guid id)
{
return teams.FirstOrDefault(t => t.ID == id);
}
public Team Update(Team t)
{
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;
}
}
}
```

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

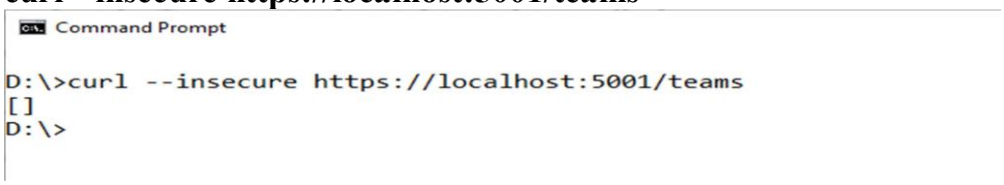
```
Services.AddScoped<IteamRepository, MemoryTeamRepository>();
```

9) Now open two command prompts to run this project

10) On Command prompt 1: (go inside folder teamservice first)
Commands: **dotnet run**



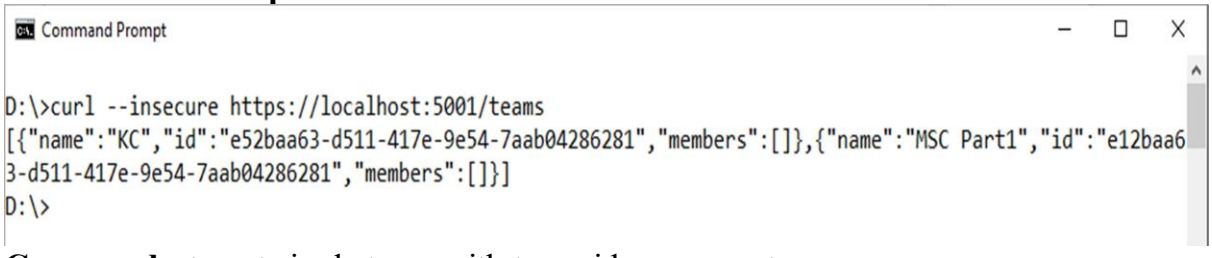
11)On command prompt 2
Command: To get all teams
curl --insecure https://localhost:5001/teams



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

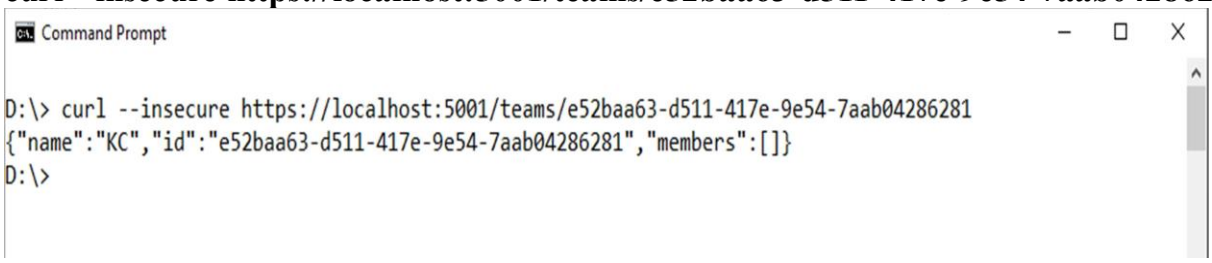
Command : To create one more new team
curl --insecure -H "Content-Type:application/json" –X POST –d "{\"id\":\"e12baa63-d511-417e-9e54-7aab04286281\", \"name\":\"MSC Part1\"}"
https://localhost:5001/teams

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



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

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



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-9e547aab04286281\", \"name\":\"KC IT DEPT\"}" https://localhost:5001/teams/e52baa63-d511-417e-9e547aab04286281

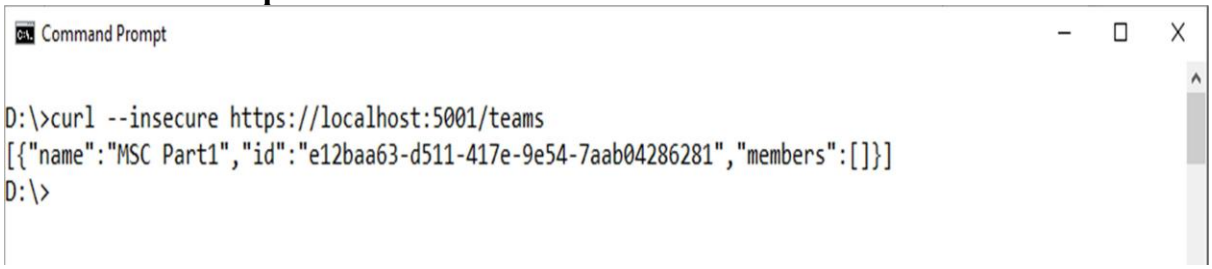
Command: to delete team

curl --insecure -H "Content-Type:application/json" –X DELETE https://localhost:5001/teams/e52baa63-d511-417e-9e54-7aab04286281

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

Command:

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

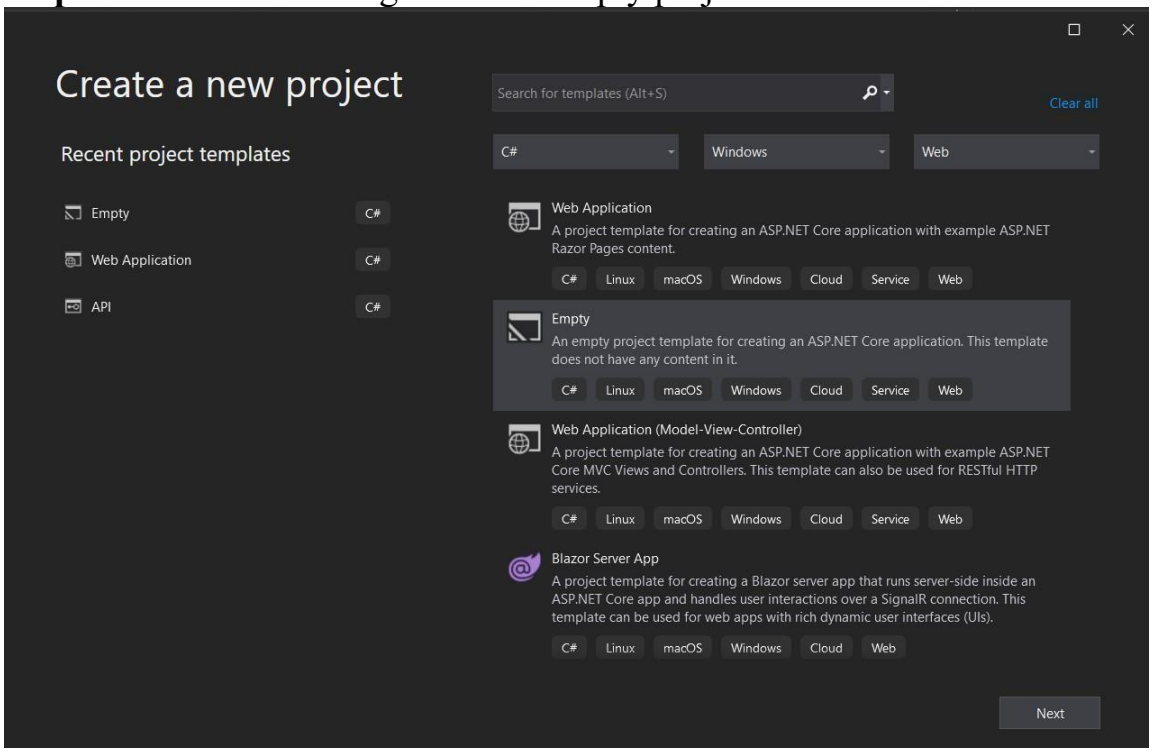


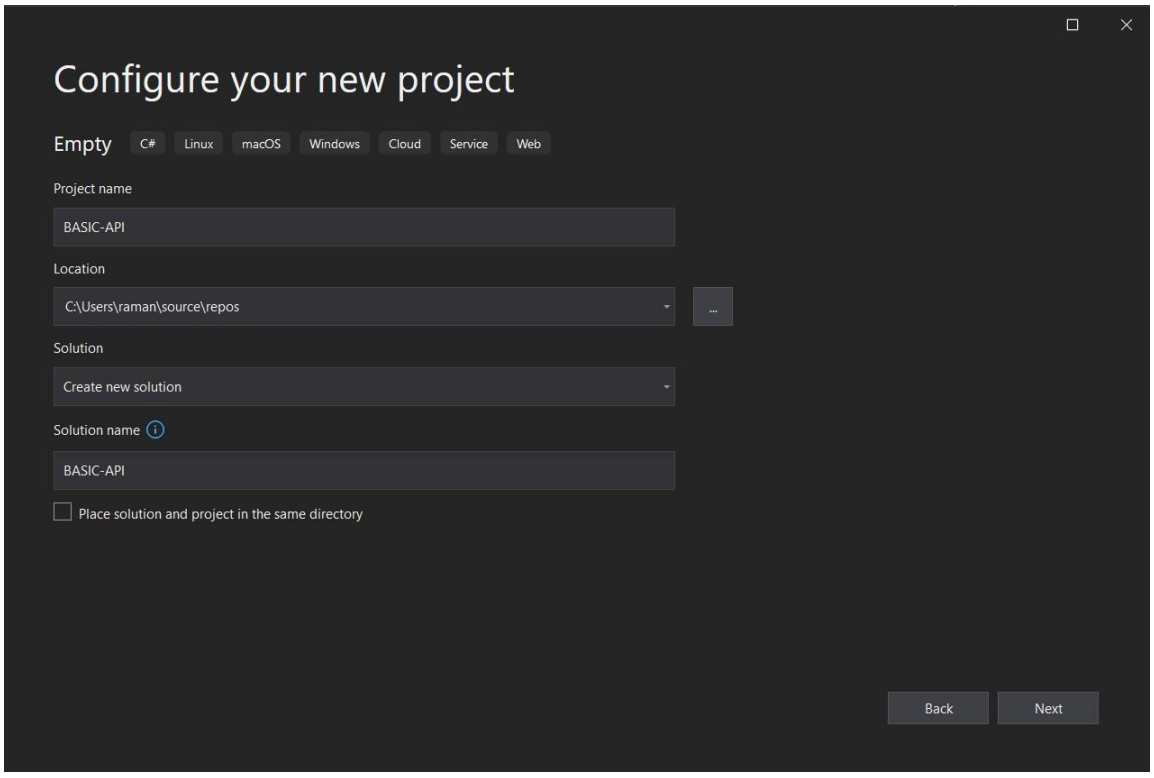
PRATICAL 6: Creating Backing Service with ASP.NET Core.

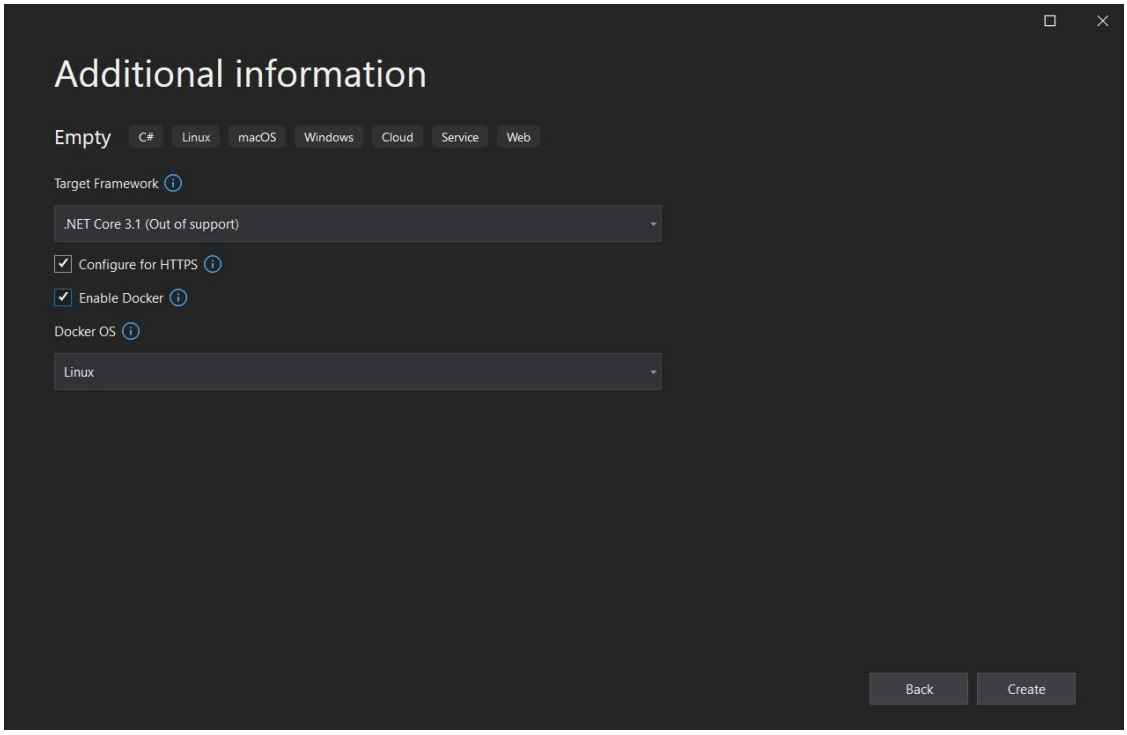
Requirement:

- Microsoft Visual Studio 2019 or higher
- ASP.Net Core 3.1

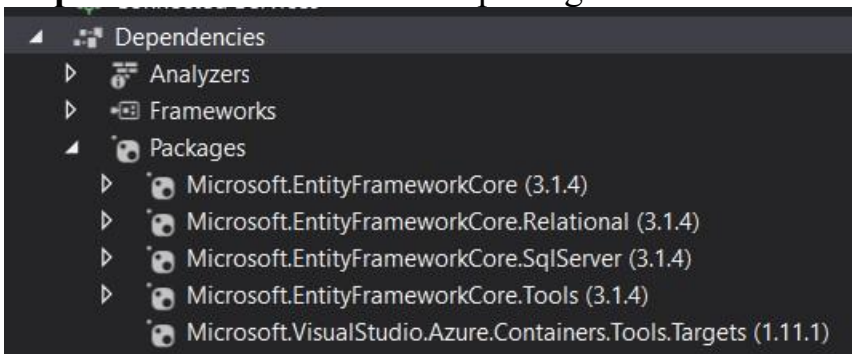
Step 1: Create and Configure a new empty project



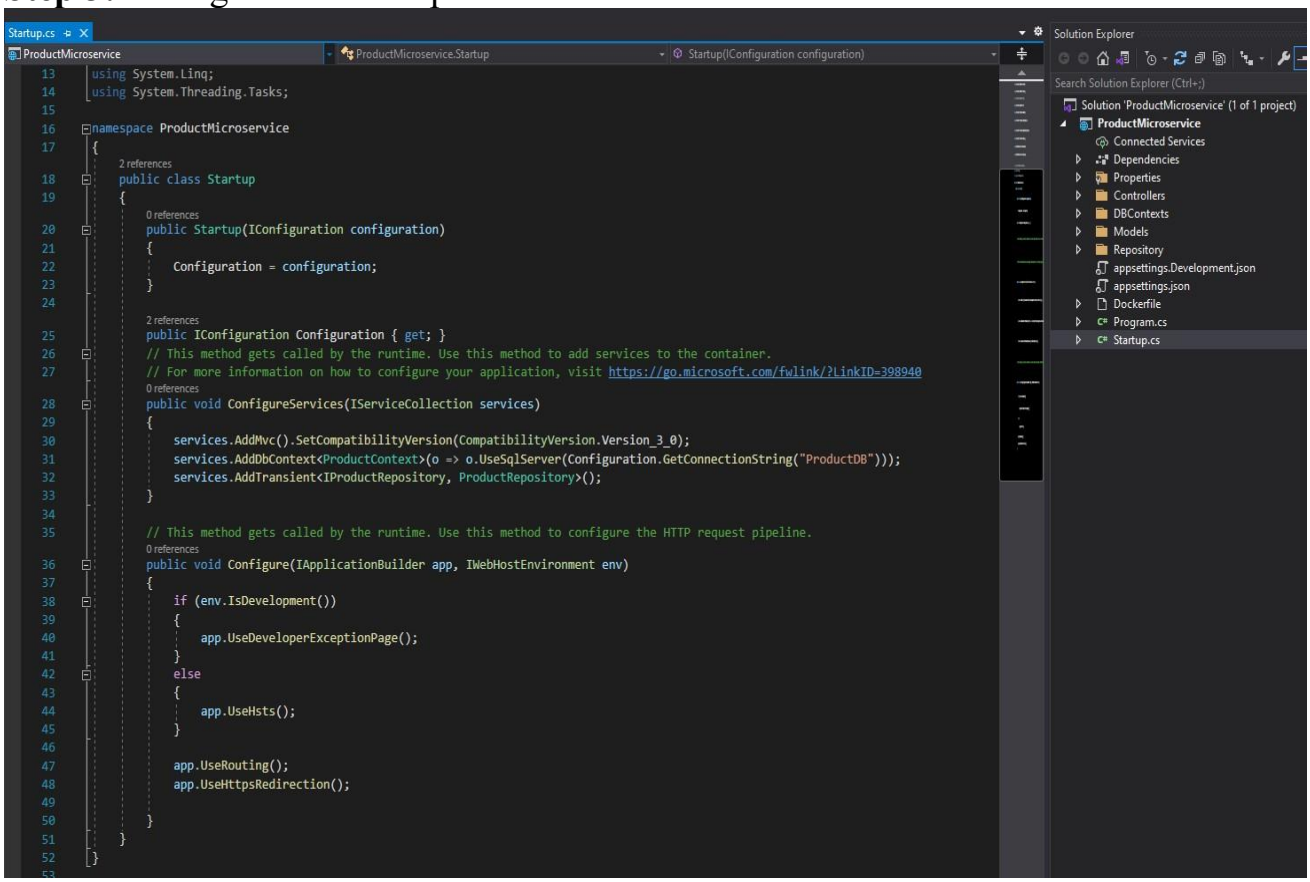




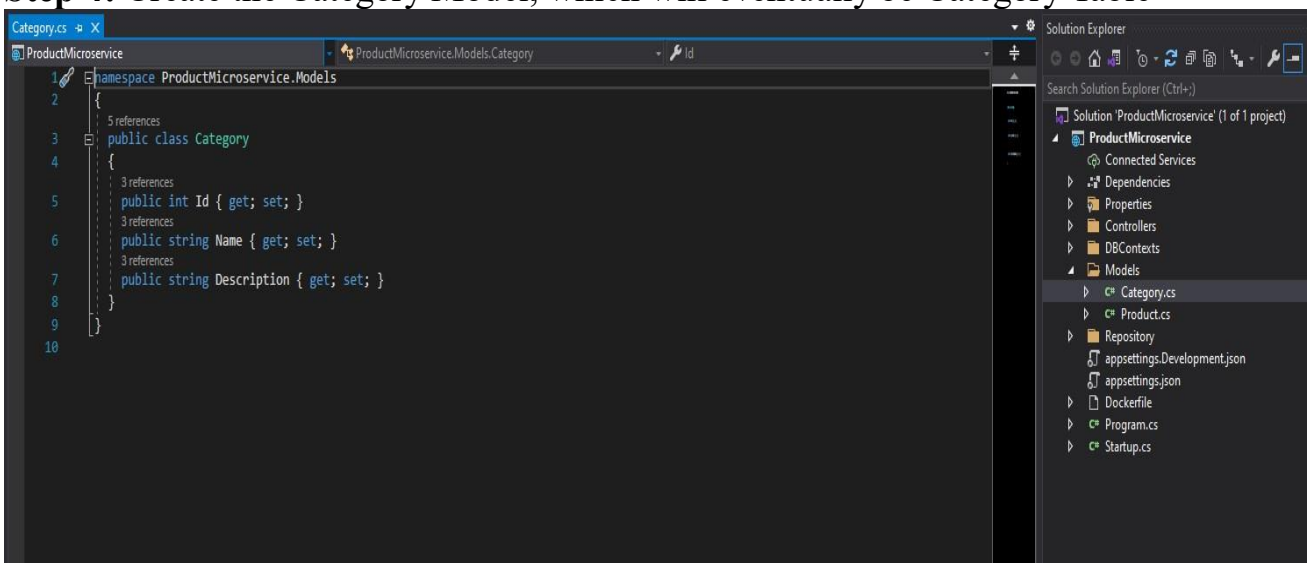
Step 2: Make sure to add these packages via NuGet Package Manager.



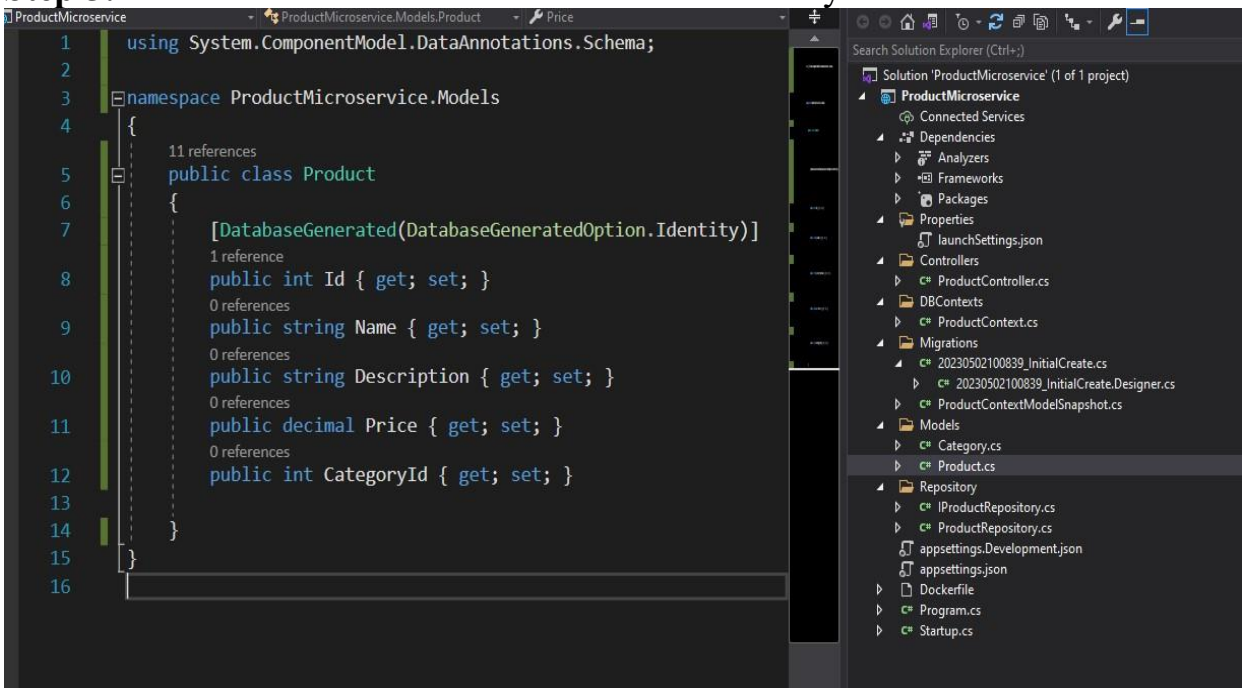
Step 3: Configure the Startup.cs file



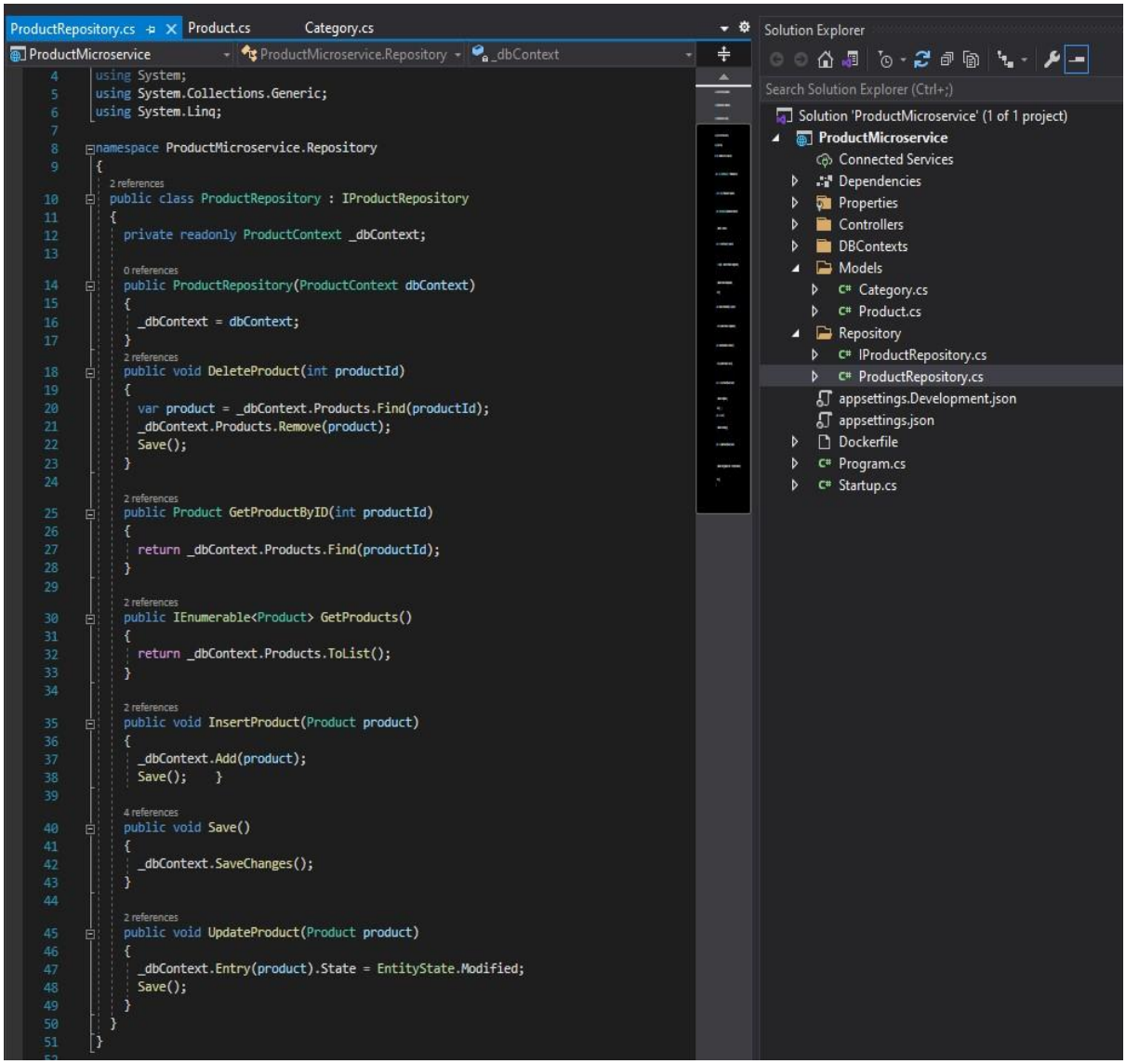
Step 4: Create the Category Model, which will eventually be Category Table



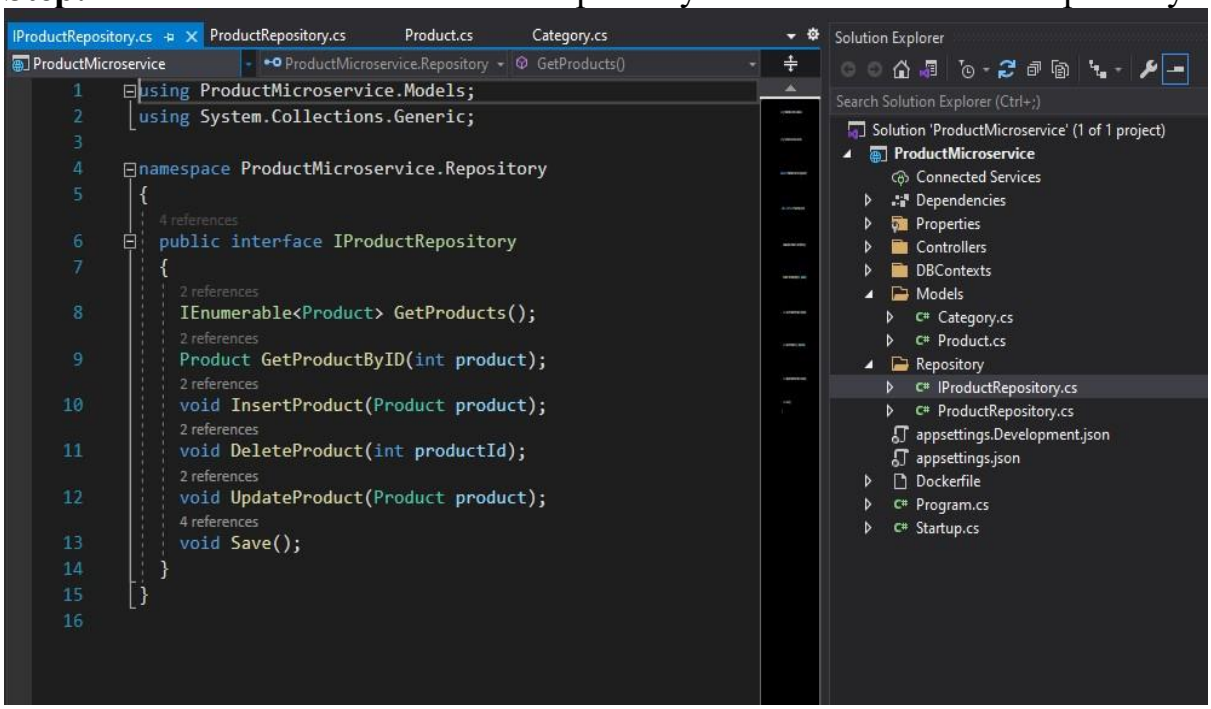
Step 5: Create the Product Model which will eventually be the Product Table.



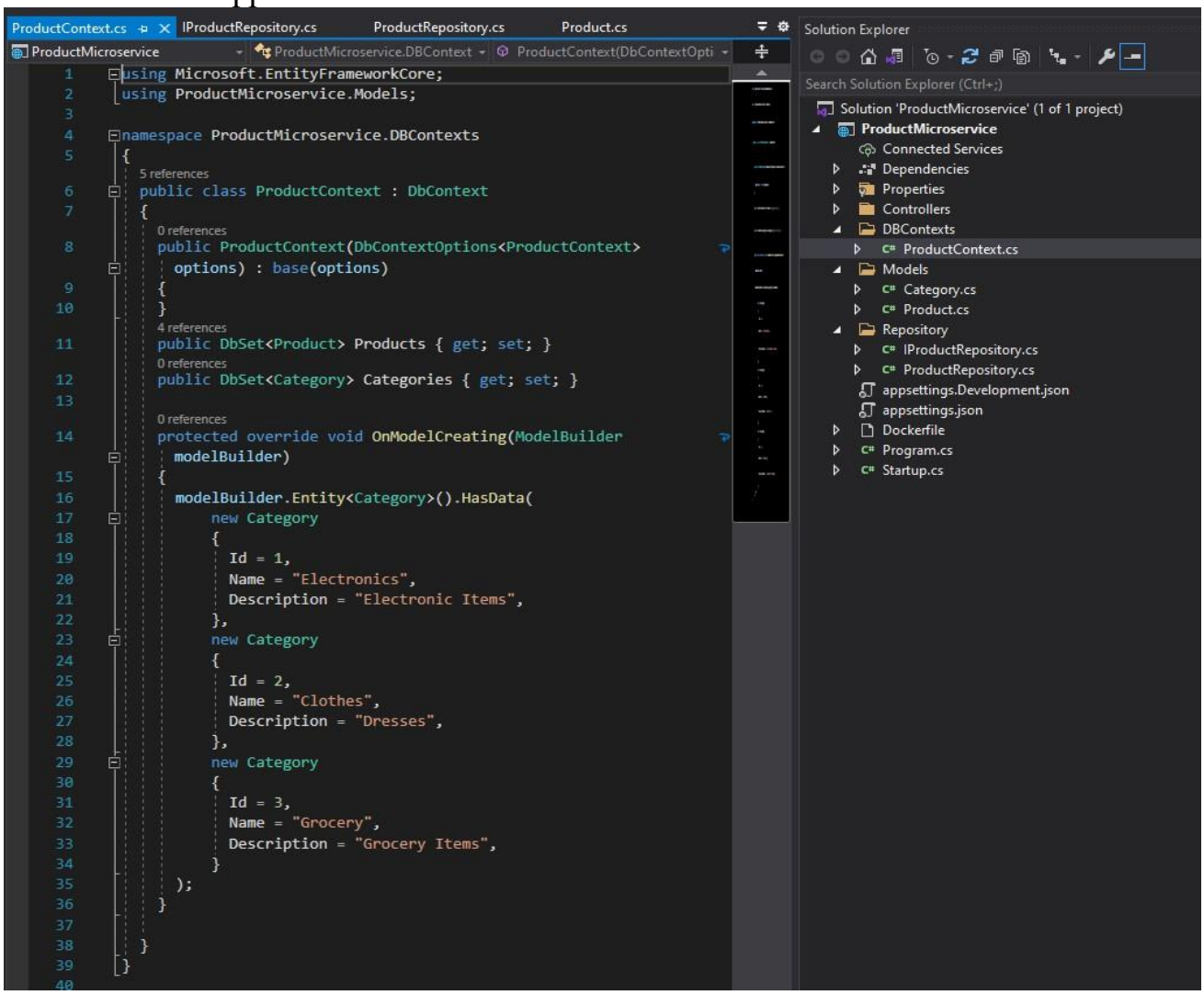
Step 6: Create a ProductRepository for all our operations.



Step:7 Create an interface IProductRepository to access the ProductRepository.

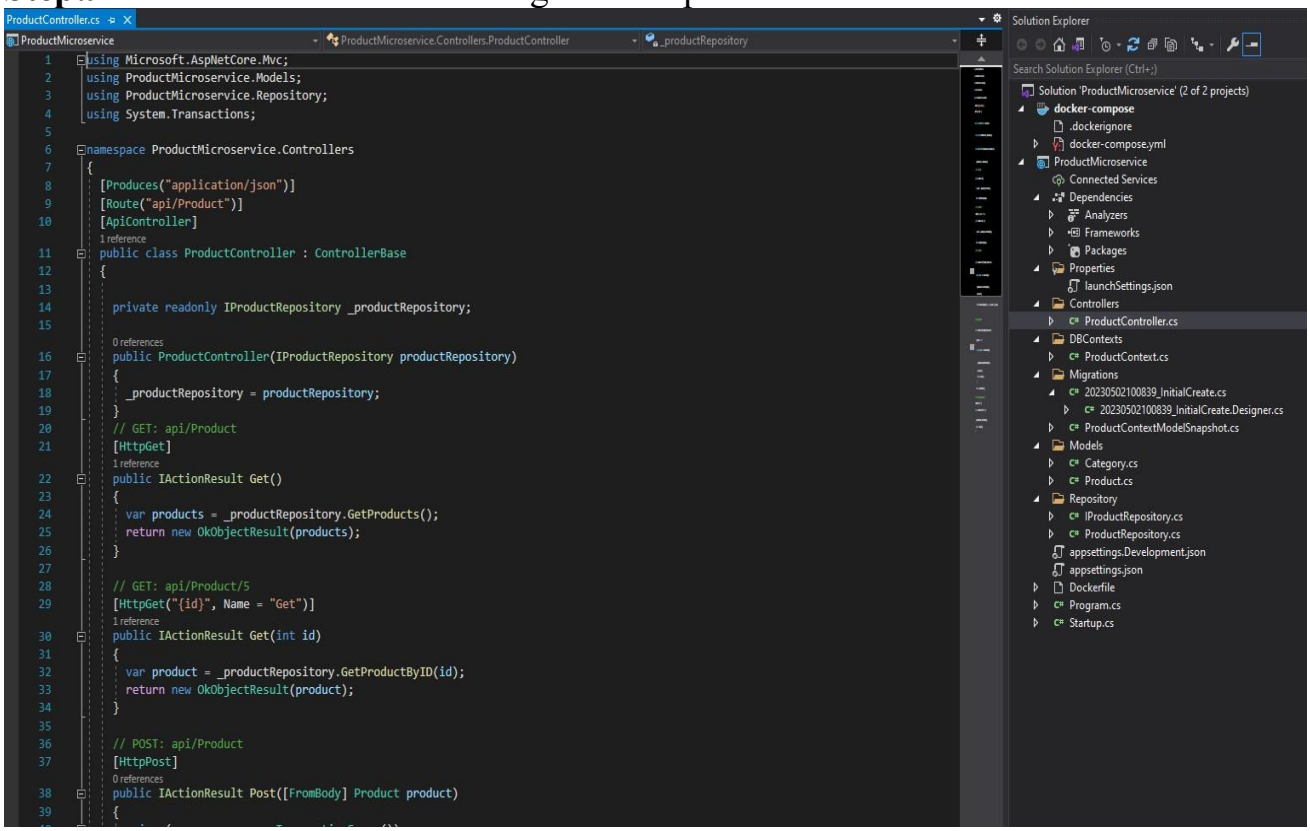


Step:8 Create a ProductContext to build the model for the database; this will be a Code First Database Approach



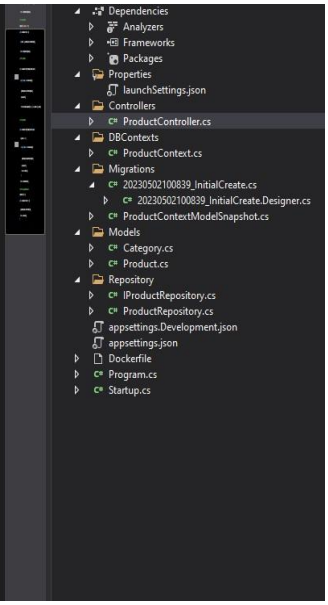
```
using Microsoft.EntityFrameworkCore;
using ProductMicroservice.Models;

namespace ProductMicroservice.DBContexts
{
    public class ProductContext : DbContext
    {
        public ProductContext(DbContextOptions<ProductContext>
            options) : base(options)
        {
        }
        public DbSet<Product> Products { get; set; }
        public DbSet<Category> Categories { get; set; }

        protected override void OnModelCreating(ModelBuilder
            modelBuilder)
        {
            modelBuilder.Entity<Category>().HasData(
                new Category
                {
                    Id = 1,
                    Name = "Electronics",
                    Description = "Electronic Items",
                },
                new Category
                {
                    Id = 2,
                    Name = "Clothes",
                    Description = "Dresses",
                },
                new Category
                {
                    Id = 3,
                    Name = "Grocery",
                    Description = "Grocery Items",
                }
            );
        }

    }
}
```

Step:9 Add a ProductController to get an endpoint for the APIs



```
using Microsoft.AspNetCore.Mvc;
using ProductMicroservice.Models;
using ProductMicroservice.Repository;
using System.Transactions;

namespace ProductMicroservice.Controllers
{
    [Produces("application/json")]
    [Route("api/Product")]
    [ApiController]
    public class ProductController : ControllerBase
    {

        private readonly IProductRepository _productRepository;

        public ProductController(IProductRepository productRepository)
        {
            _productRepository = productRepository;
        }
        // GET: api/Product
        [HttpGet]
        public IActionResult Get()
        {
            var products = _productRepository.GetProducts();
            return new OkObjectResult(products);
        }

        // GET: api/Product/5
        [HttpGet("{id}", Name = "Get")]
        public IActionResult Get(int id)
        {
            var product = _productRepository.GetProductByID(id);
            return new OkObjectResult(product);
        }

        // POST: api/Product
        [HttpPost]
        public IActionResult Post([FromBody] Product product)
        {
```

```
public IActionResult Post([FromBody] Product product)
{
  using (var scope = new TransactionScope())
  {
    _productRepository.InsertProduct(product);
    scope.Complete();
    return CreatedAtAction(nameof(Get), new { id = product.Id }, product);
  }
}

// PUT: api/Product/5
[HttpPut]
0 references
public IActionResult Put([FromBody] Product product)
{
  if (product != null)
  {
    using (var scope = new TransactionScope())
    {
      _productRepository.UpdateProduct(product);
      scope.Complete();
      return new OkResult();
    }
  }
  return new NoContentResult();
}

// DELETE: api/ApiWithActions/5
[HttpDelete("{id}")]
0 references
public IActionResult Delete(int id)
{
  _productRepository.DeleteProduct(id);
  return new OkResult();
}
}
}
```



Step 10: Add the connection string settings to connect the project to the SQLSERVER database. Once added run add-migration command to run the migration

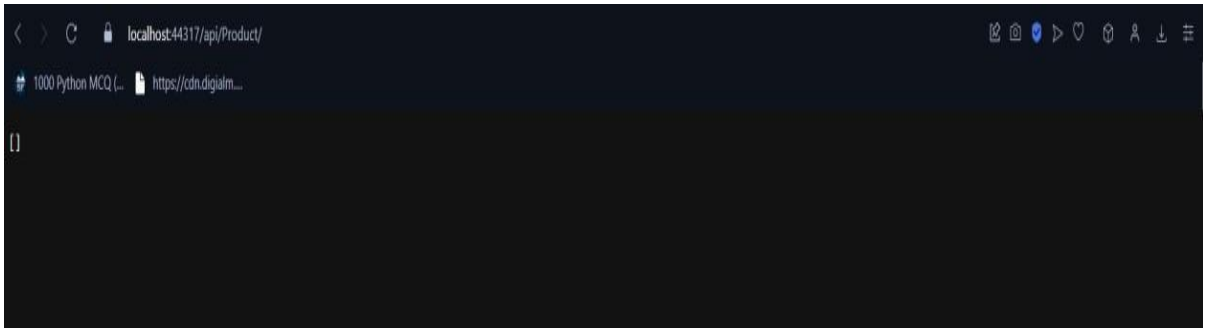
```
{
"Logging": {
  "LogLevel": {
    "Default": "Information",
    "Microsoft": "Warning",
    "Microsoft.Hosting.Lifetime": "Information"
  }
},
"AllowedHosts": "*",
"ConnectionStrings": {
  "ProductDB": "Server=LAPTOP-5A4DVJ0D\\MYSQLSERVER2022;Database=ProductDB;Trusted_Connection=True;MultipleActiveResultSets=true;"
}
}
```

```
PM> add-migration IntitalCreate -verbose
Using project 'BASIC-API'.
Using startup project 'BASIC-API'.
Build started...
Build succeeded.
```

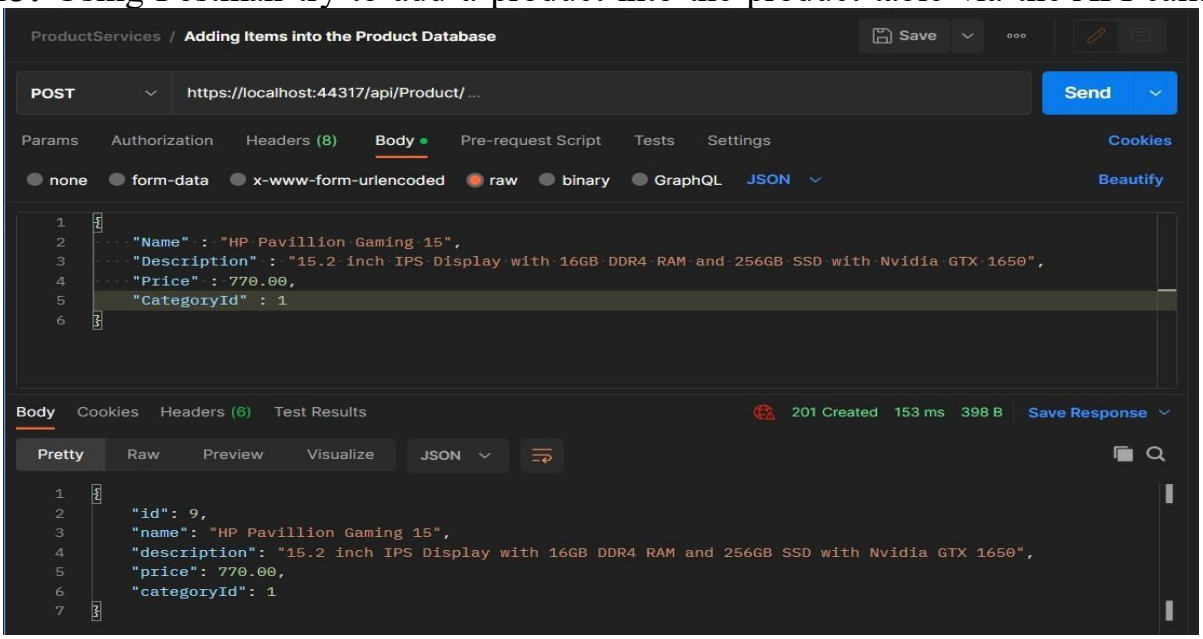
Step 11: Run the update-database command to reflect the model changes to the database.

```
PM> update-database -verbose
Using project 'BASIC-API'.
Using startup project 'BASIC-API'.
Build started...
Build succeeded.
```

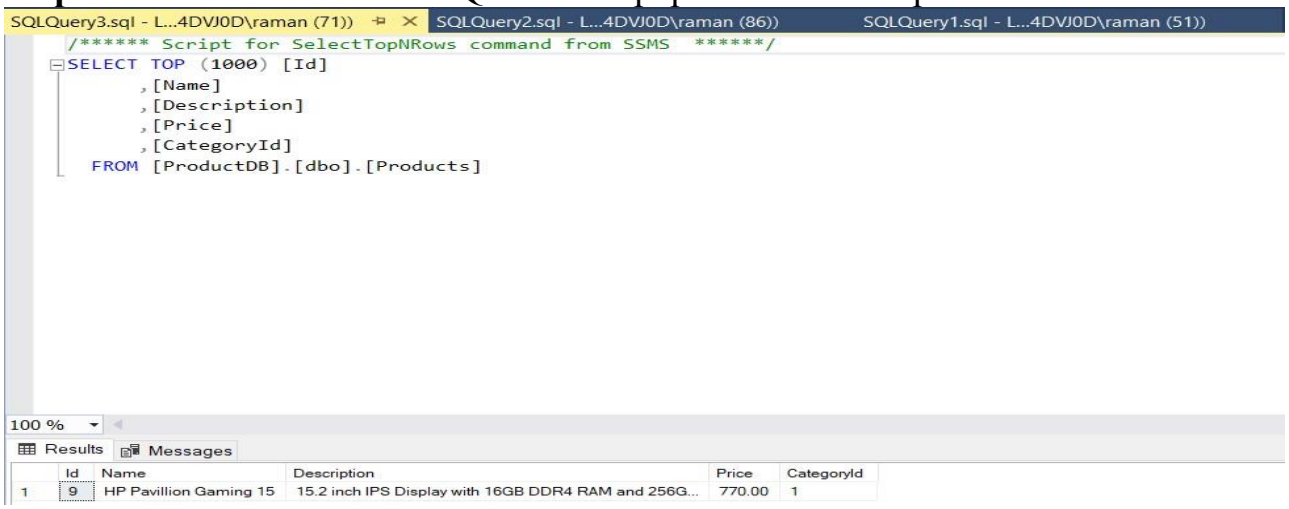
Step 12: Try to run the project, you will see that a browser window opens with an empty list.



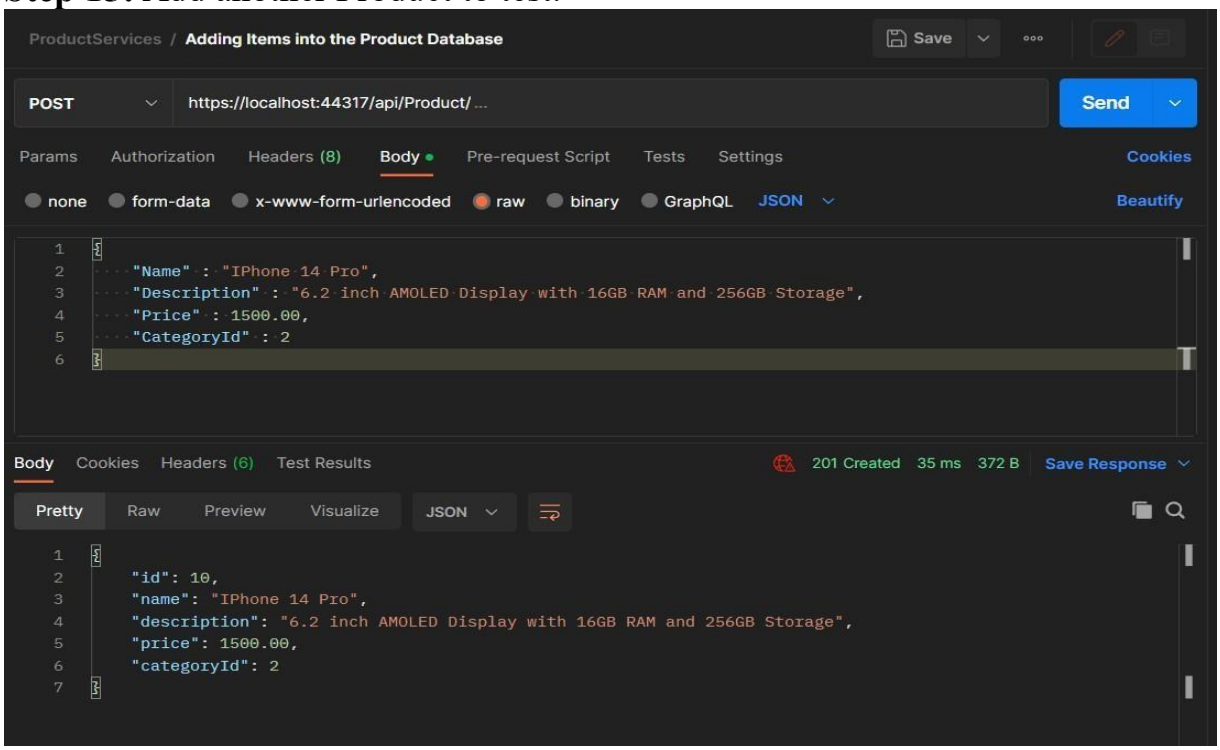
Step 13: Using Postman try to add a product into the product table via the API call.



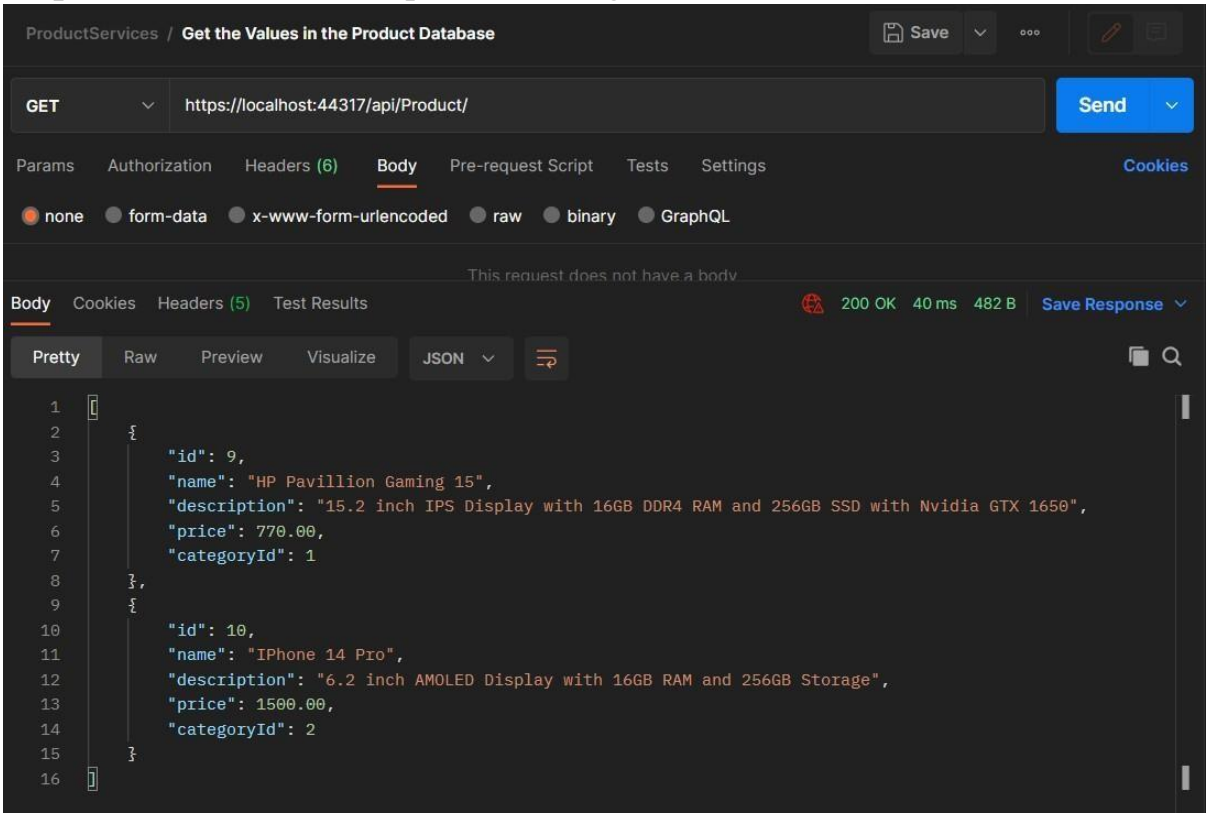
Step 14: You will notice the SQL table to populate with that product.



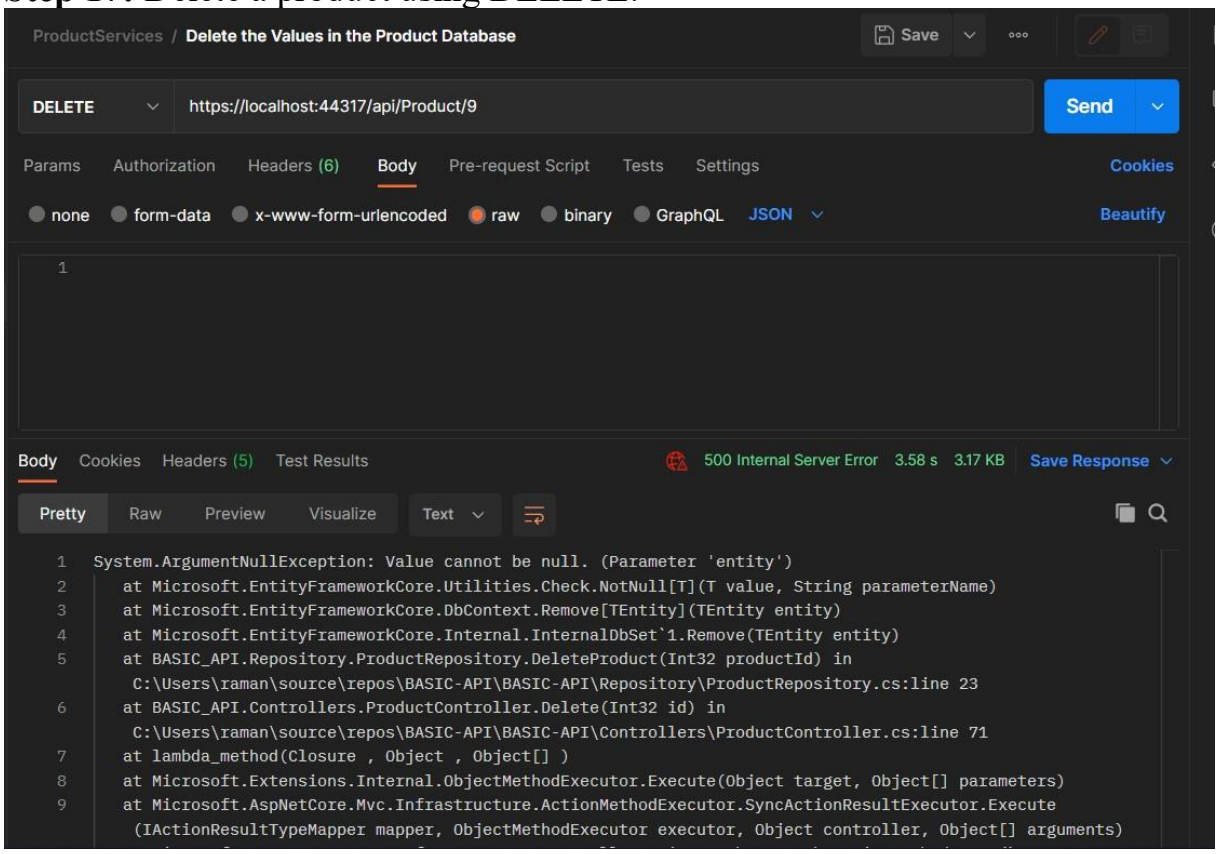
Step 15: Add another Product to test.



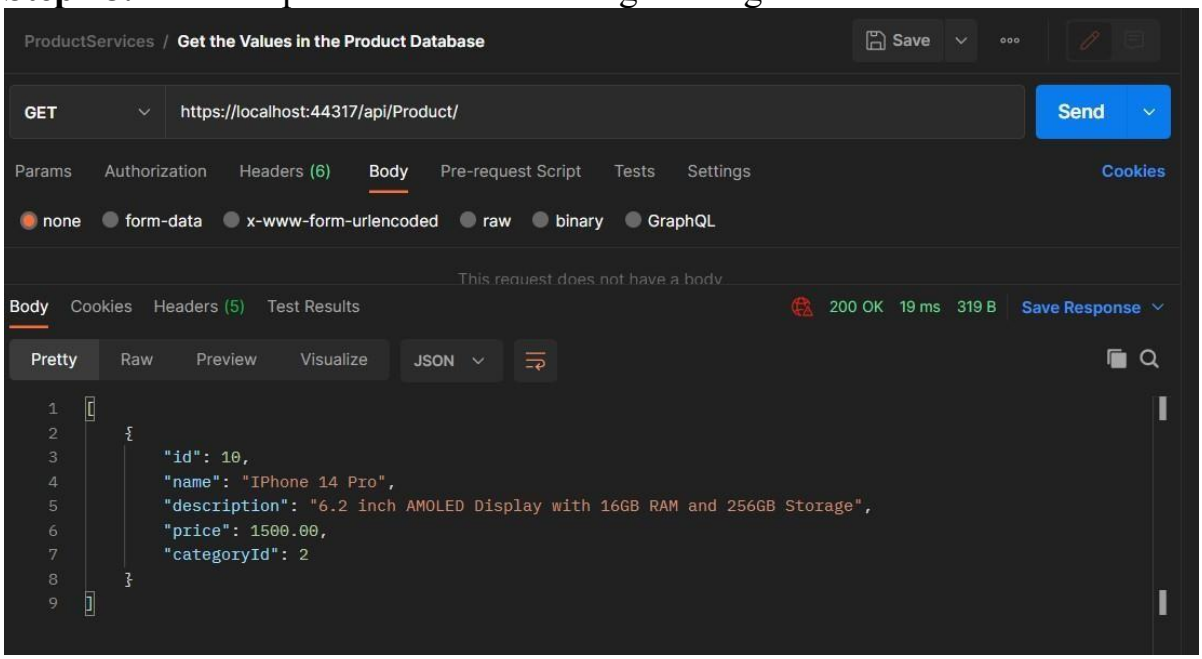
Step 16: Get all the added products using GET.



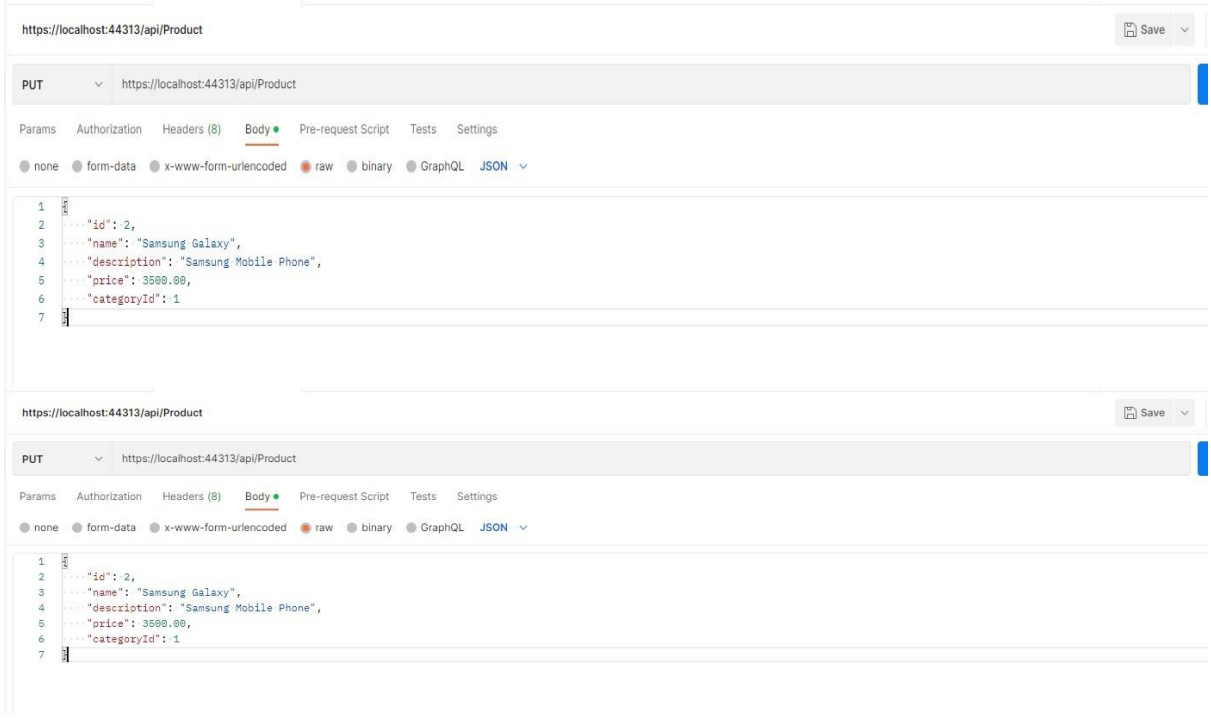
Step 17: Delete a product using DELETE.

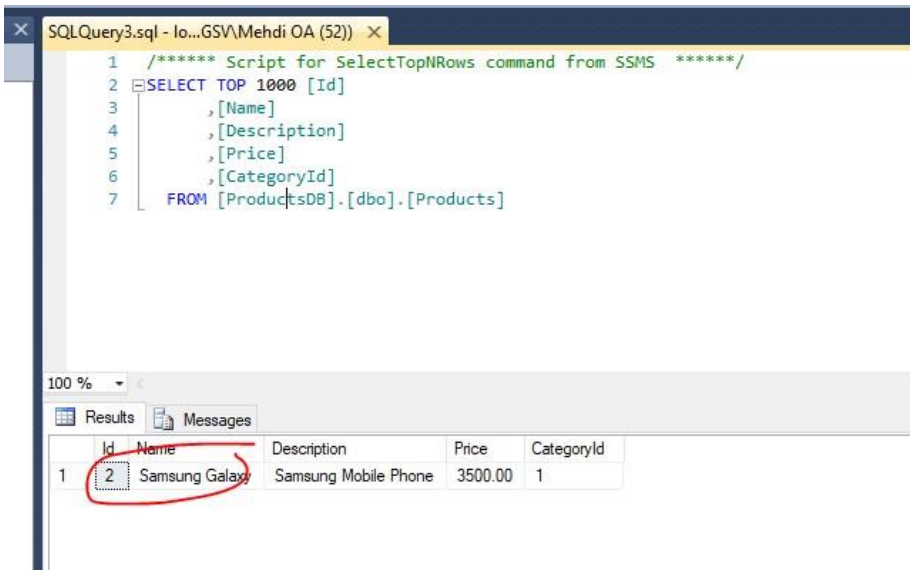


Step 18: Check is product is deleted using GET again

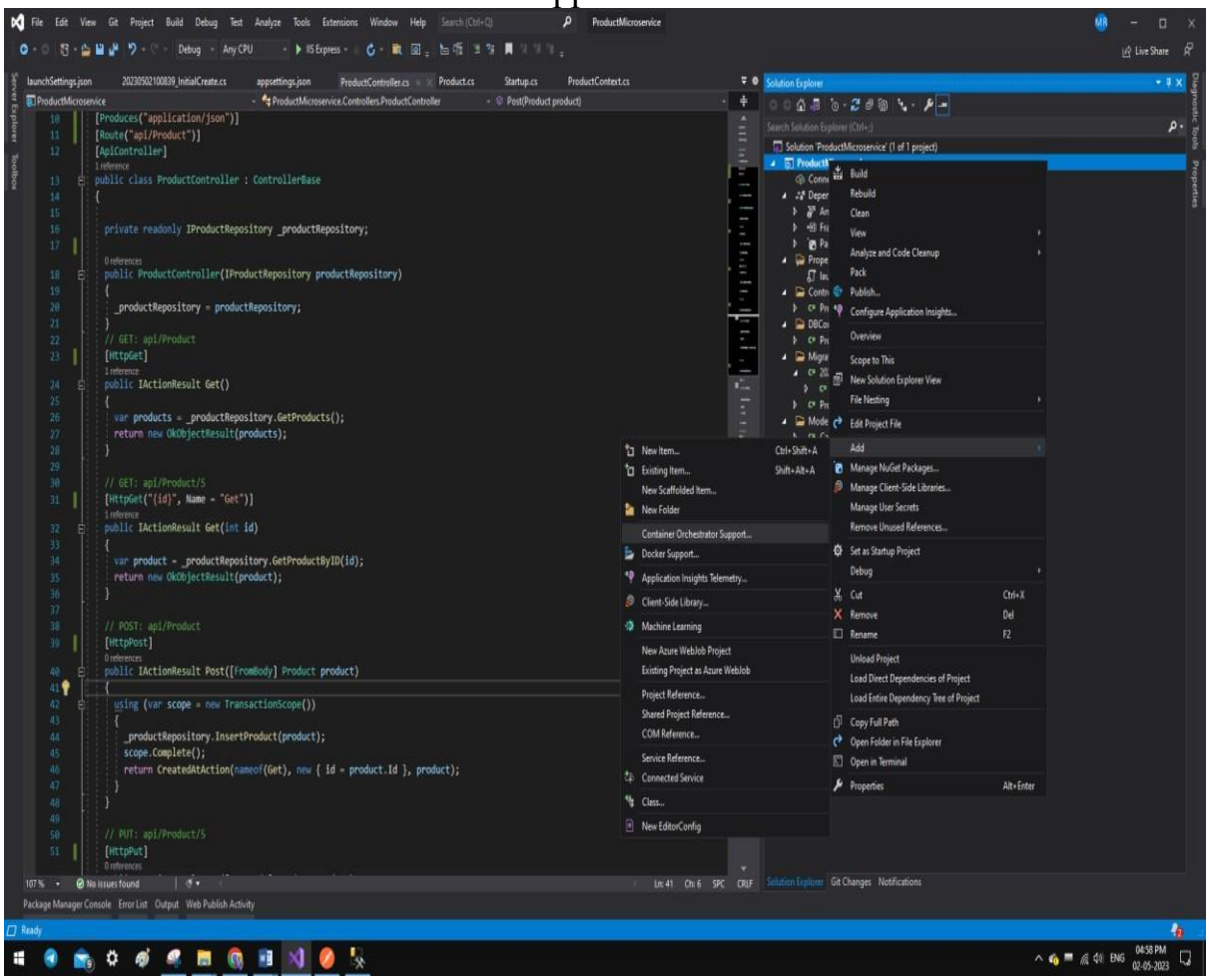


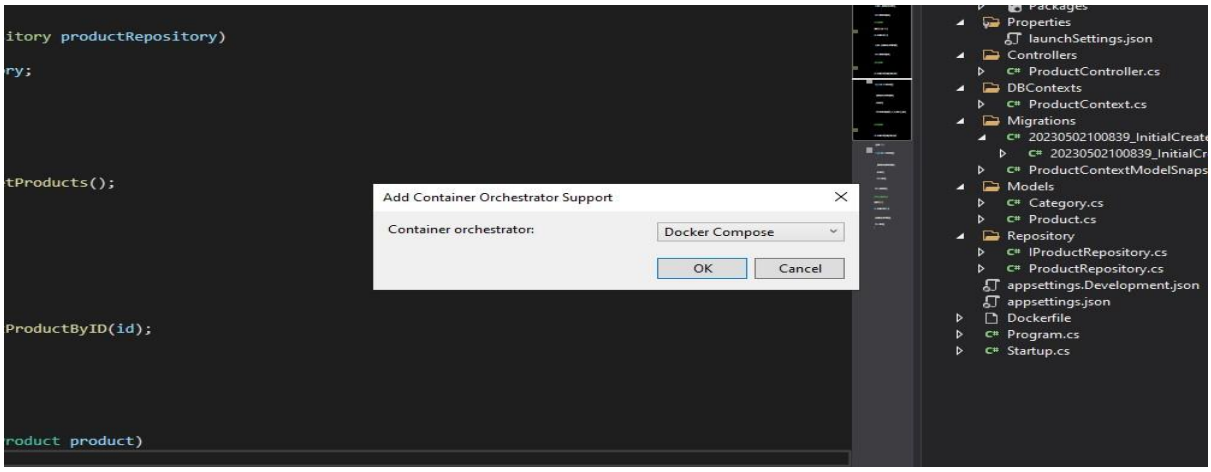
Step 19: Update a product using the PUT



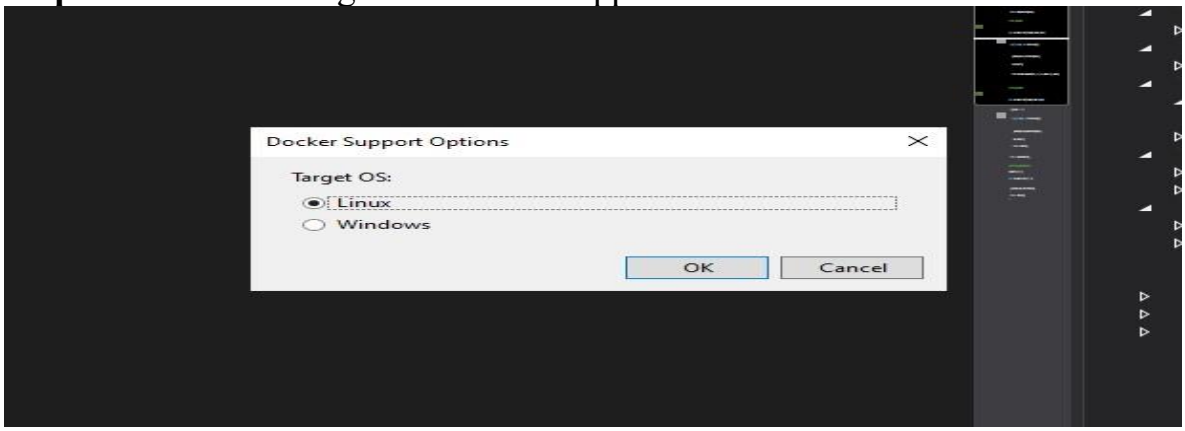


Step 20: Now create a docker container and image of the product. Right click the project and select Container Orchestrator Support and select Ok.





Step 21: Select the target docker OS support which will be LINUX

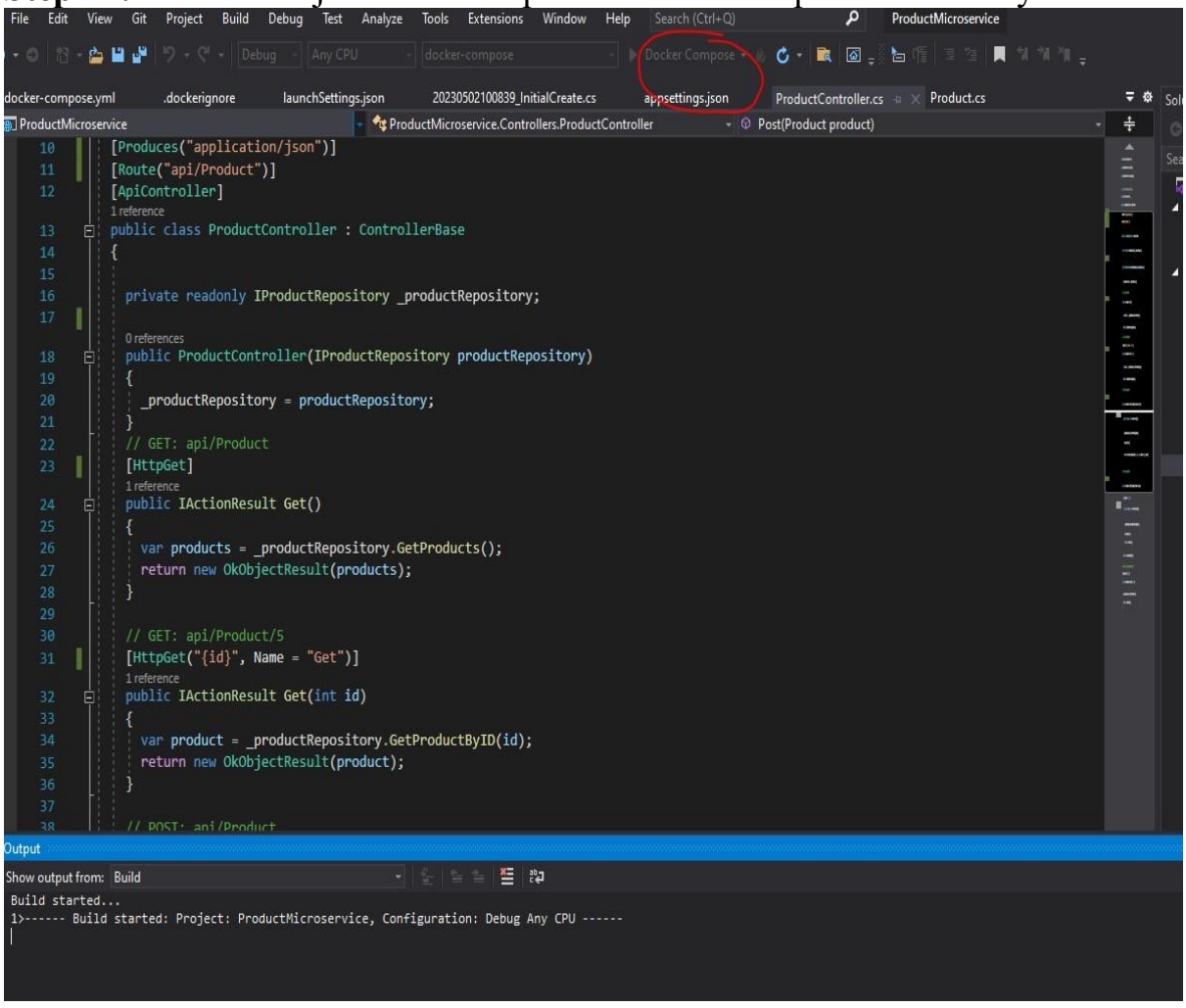


Step 22: You can select Yes to run the Docker desktop too.

Step 23: Run the docker commands 'docker images' to see the created image of your project.

```
C:\Users\raman>docker images
REPOSITORY                          TAG       IMAGE ID       CREATED         SIZE
webapi1                             dev       30aee5f1f9c9   7 weeks ago     208MB
webapplication1                     dev       872e14f1554c   7 weeks ago     208MB
<none>                              <none>    bae4e11c4bb1   7 weeks ago     208MB
productservice                      dev       2a9886bd1404   7 weeks ago     208MB
basicapi                            dev       591cee1a5e44   7 weeks ago     208MB
<none>                              <none>    7a5deb8e122b   7 weeks ago     208MB
webapplication2                     dev       5a5a5c6434b9   7 weeks ago     208MB
<none>                              <none>    ba2df3366bff   7 weeks ago     208MB
mcr.microsoft.com/dotnet/aspnet     3.1       454a55eab920   4 months ago    208MB
```

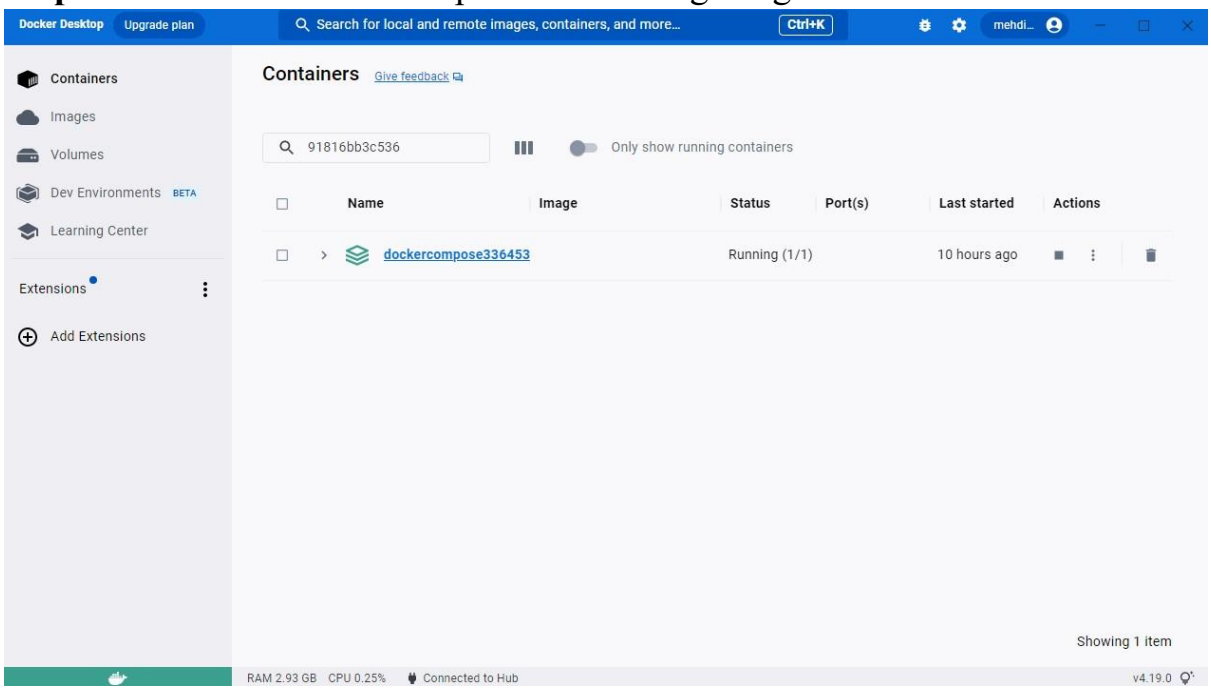
Step 24: Run the Project with the option Docker Compose on the Play.



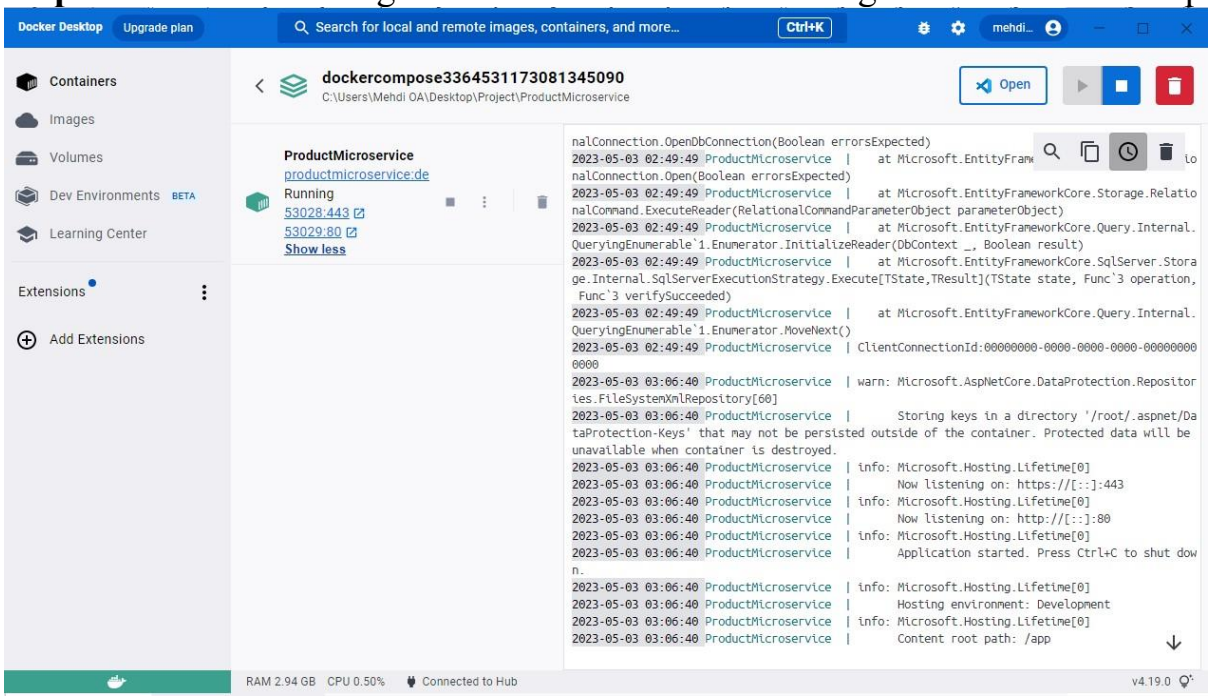
Step 25: Run the **docker ps** command to check the running docker container.

```
C:\Users\raman\source\repos\ProductService>docker ps
CONTAINER ID   IMAGE                 COMMAND                  CREATED          STATUS          PORTS                                                NAMES
6d946df292ed   productservice:dev    "tail -f /dev/null"      11 minutes ago   Up 11 minutes   0.0.0.0:52030->80/tcp, 0.0.0.0:52029->443/tcp        ProductService_1
```

Step 26: Check docker desktop for the running image.



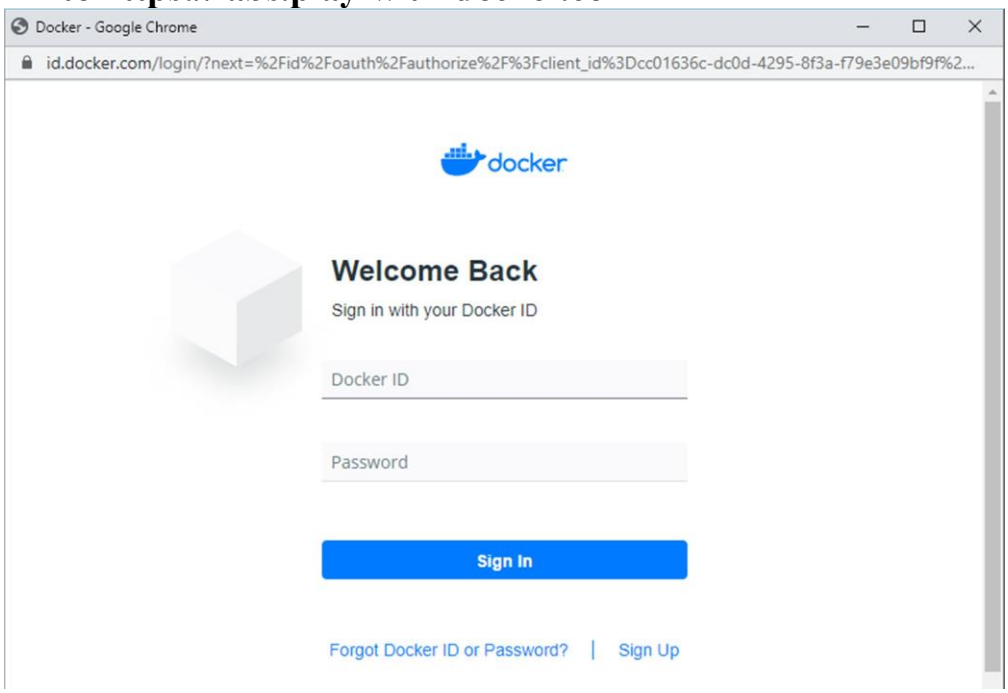
Step 27: The docker image and containers will be running in the docker desktop app.



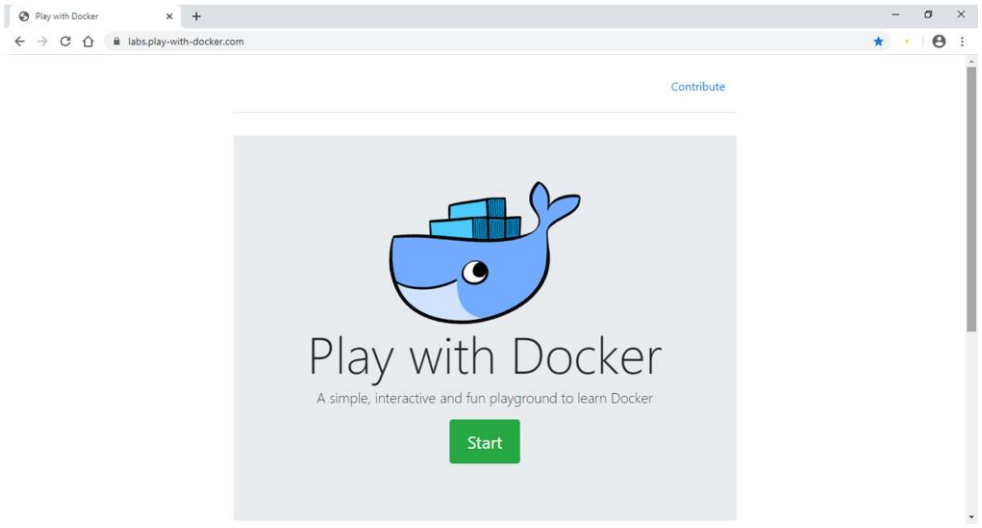
PRATICAL 7: Creating (Backing Service) Running Location Service in Docker

(create docker hub login first to use it in play with docker)

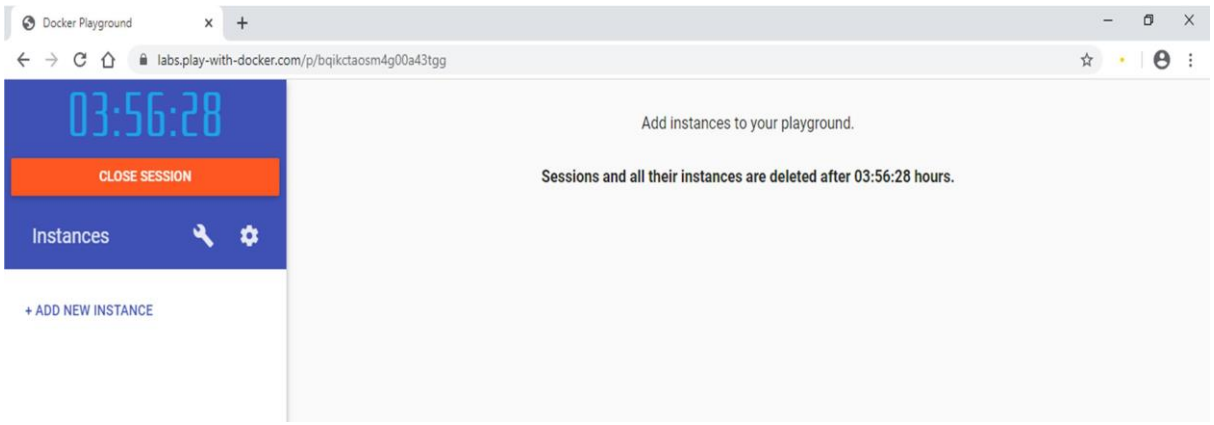
Now login in to https://labs.play-with-docker.com

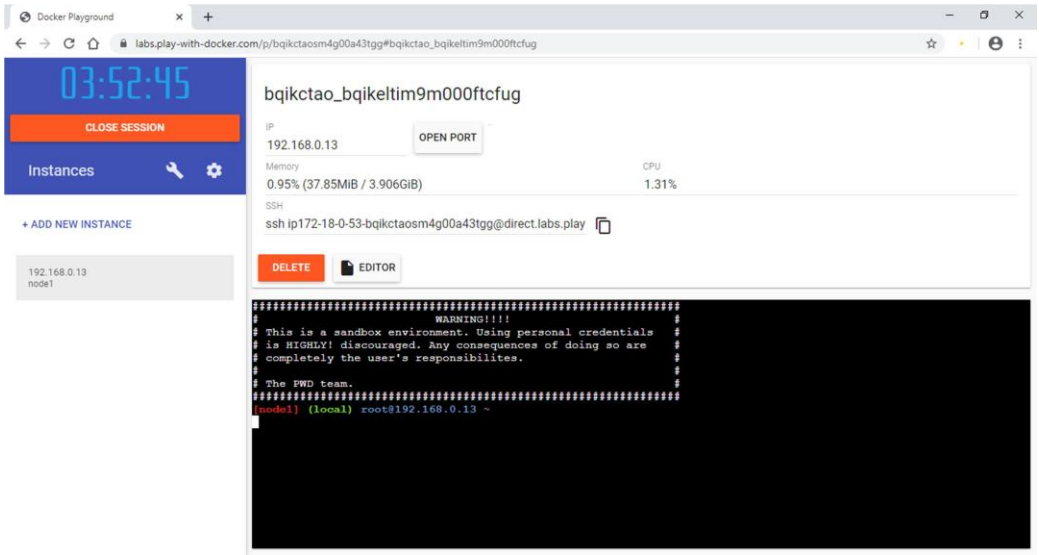


Click on Start

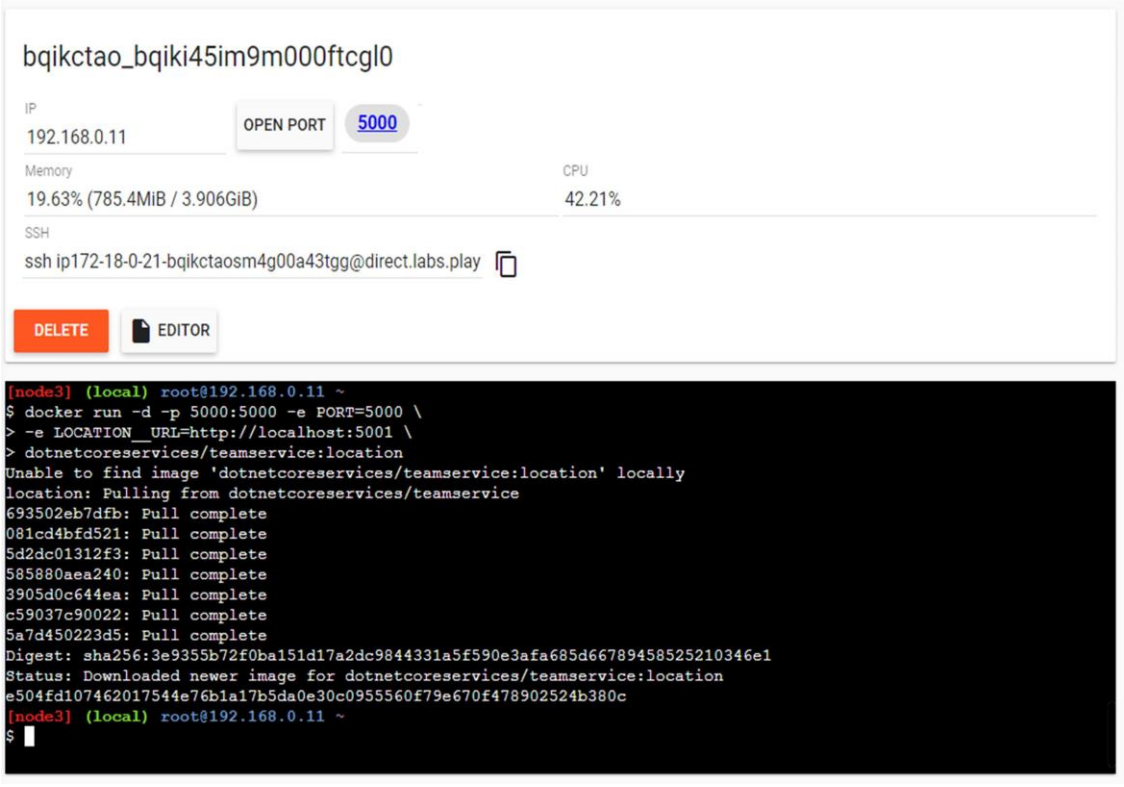


Click on Add New Instance

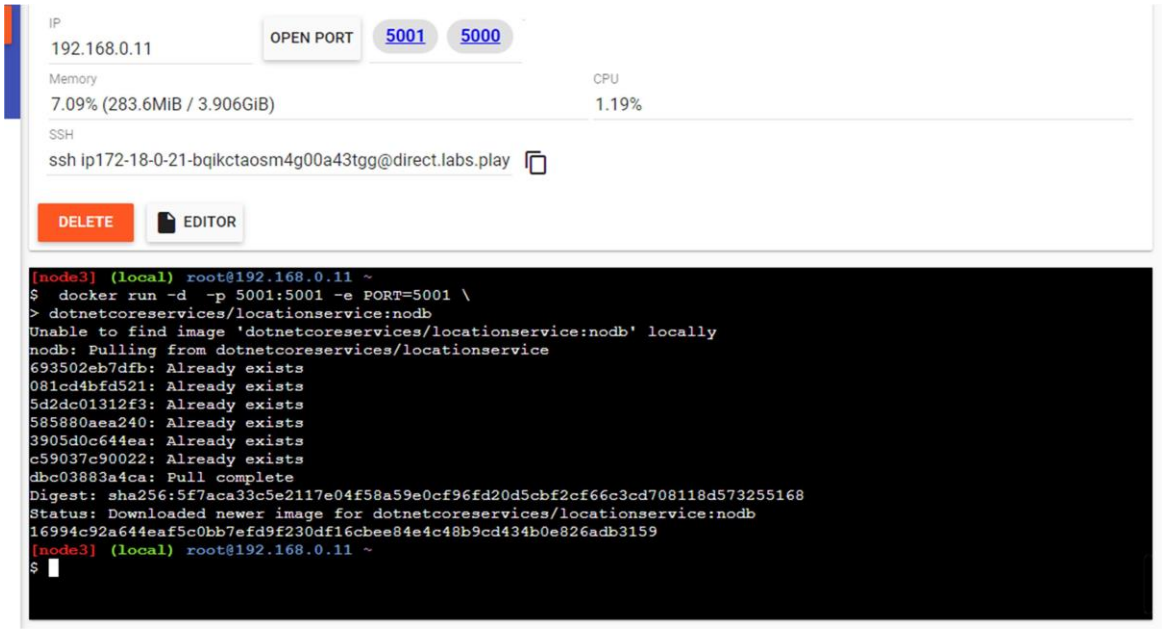




Start typing following commands
Command : To run teamservice
**docker run -d -p 5000:5000 -e PORT=5000 **
**-e LOCATION__URL=http://localhost:5001 **
dotnetcoreservices/teamservice:location
output: (you can observe that it has started port 5000 on top)

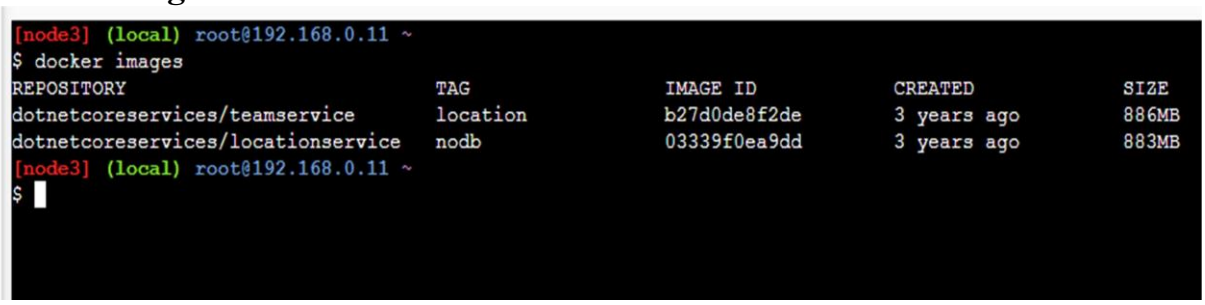


Command: to run location service
**docker run -d -p 5001:5001 -e PORT=5001 **
dotnetcoreservices/locationservice:nodb
output: (now it has started one more port that is 5001 for location service)



Command : to check running images in docker

docker images



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

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:

```
[node1] (local) root@192.168.0.23 ~
$ 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
{"teamID":"e52baa63-d511-417e-9e54-7aab04286281","memberID":"63e7acf8-8fae-42ce-9349-3c8593ac8292"}[node1] (local
) root@192.168.0.23 ~
$
```

Command :To confirm member added

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

```
[node1] (local) root@192.168.0.23 ~
$ curl http://localhost:5000/teams/e52baa63-d511-417e-9e54-7aab04286281
{"name":"KC","id":"e52baa63-d511-417e-9e54-7aab04286281","members":[null,{"id":"63e7acf8-8fae-42ce-9349-3c8593ac8
292","firstName":"Kirti","lastName":"Bhatt"}]}[node1] (local) root@192.168.0.23 ~
$
```

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:

```
[node1] (local) root@192.168.0.23 ~
$ 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-9349-3c859
3ac8292
{"id":"64c3e69f-1580-4b2f-a9ff-2c5f3b8f0e1f","latitude":12.0,"longitude":12.0,"altitude":10.0,"timestamp":0,"memb
erID":"63e7acf8-8fae-42ce-9349-3c8593ac8292"}[node1] (local) root@192.168.0.23 ~
$
```

Command : To confirm location is added in member

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

Output:

```
[node1] (local) root@192.168.0.23 ~
$ curl http://localhost:5001/locations/63e7acf8-8fae-42ce-9349-3c8593ac8292
[{"id":"64c3e69f-1580-4b2f-a9ff-2c5f3b8f0e1f","latitude":12.0,"longitude":12.0,"altitude":10.0,"timestamp":0,"mem
berID":"63e7acf8-8fae-42ce-9349-3c8593ac8292"}][node1] (local) root@192.168.0.23 ~
$
```

IMAGE PROCESSING

INDEX

PRATICAL 1A): Program to calculate number of samples required for an image.

Code:

```
clc;
close;
m =4;
n=6;
N=400;
N2=2*N;
Fs= m*N2*n*N2;
disp('Number of samples required to preserve the information in the images=',Fs)
```

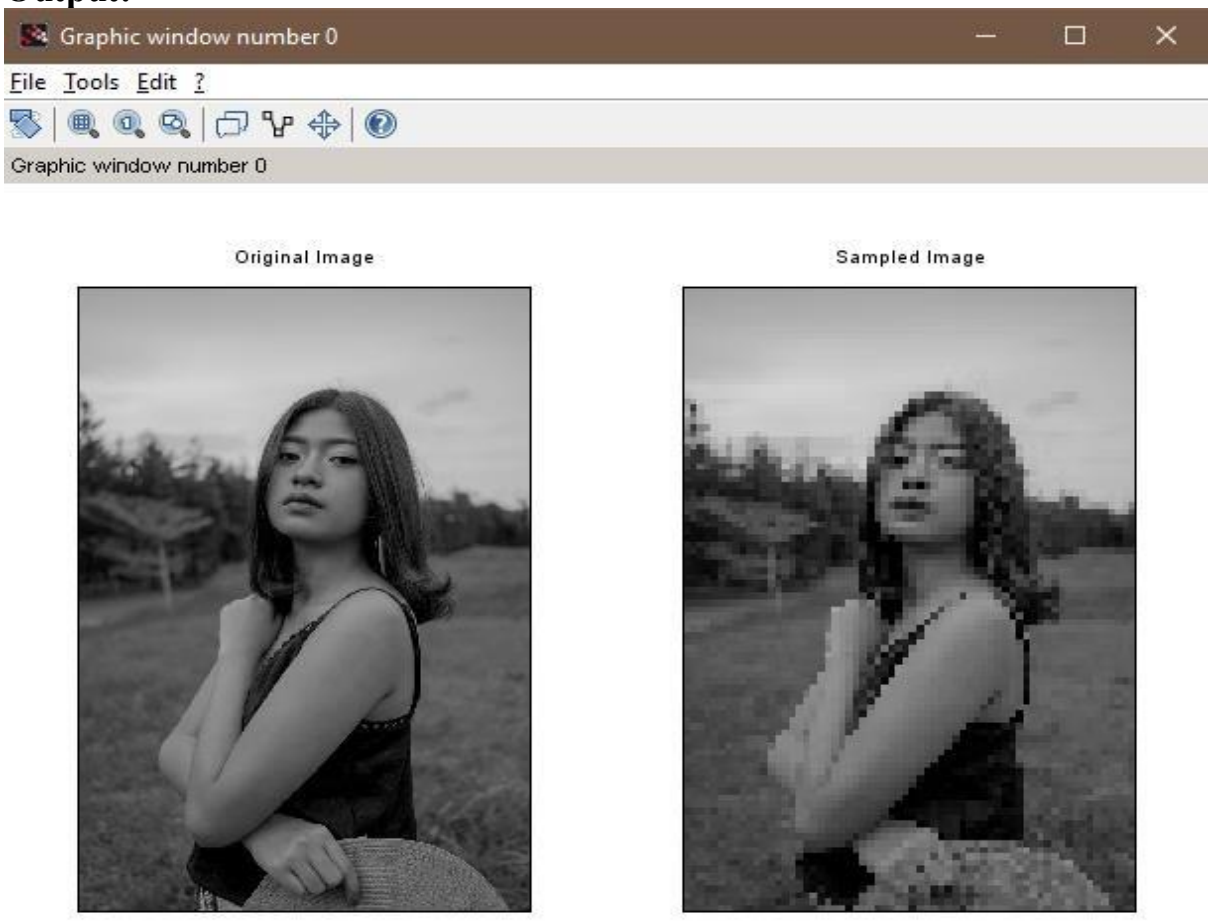
Output:

```
"Number of samples required to preserve the information in the image="

 15360000.
```

B) Program to study the effects of reducing the spatial resolution of a digital image.

```
clc;
clear all;
n = input ('Enter the input sample');
img=imread("C:\Users\bnnco\abc.jpg");
a=size(img);
w=a(2);
h=a(1);
im=zeros(100);
for i=1:n:h
for j=1:n:w
for k=0:n-1
for l=0:n-1
   im(i+k,j+l)=img(i,j);
end
end
end
end
subplot(1,2,1);
imshow(uint8(img));title('original image');
subplot(1,2,2);
imshow(uint8(im));title('sampled image');
```

Output:

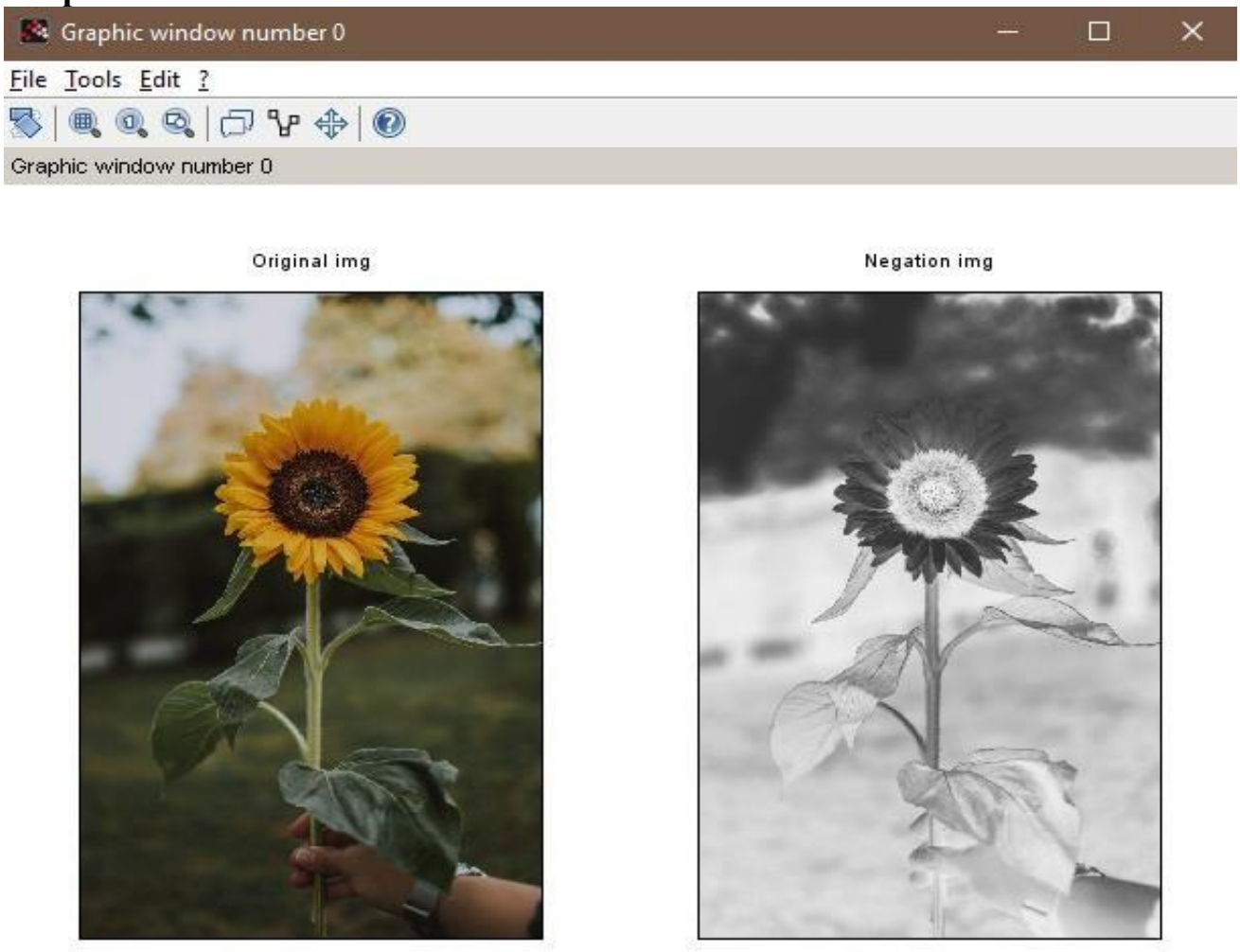


PRATICAL 2: Basic Intensity Transformation functions

A) Program to perform Image negation

```
clc;
clear all;
a=imread("C:\Users\bnnco\xyz1.jpg");
subplot(1,2,1);
imshow(a)
title('original image')
[m,n]=size(a);
for i=1:m
for j=1:n
c(i,j)=255-a(i,j)
end
end
subplot(1,2,2);
imshow(c)
title('negation img')
```

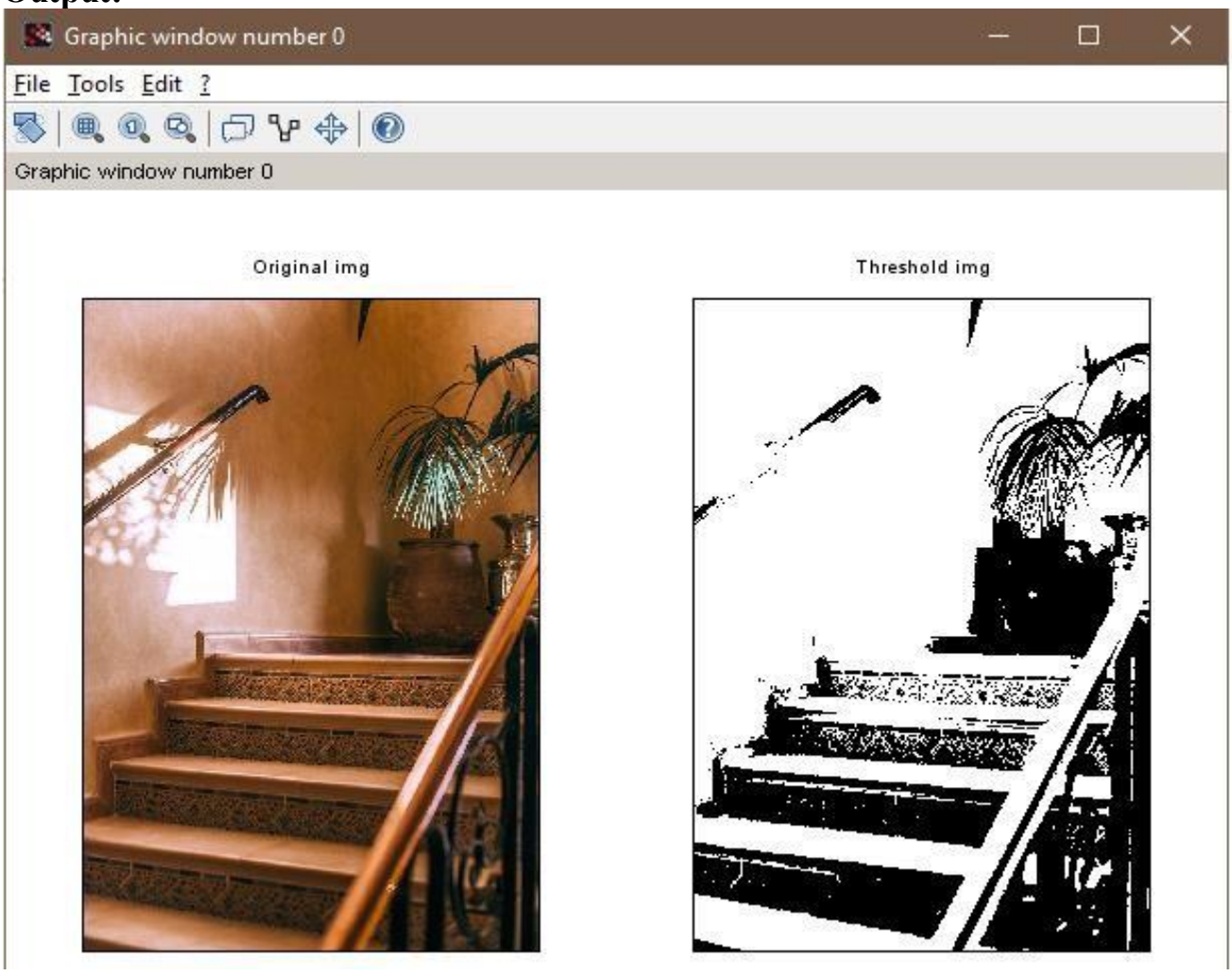
Output:



B) Program to perform threshold on an image.

```
clc;
clear all;
a=imread("C:\Users\bnnco\xyz2.jpg");
b=double(a)
subplot(1,2,1);
imshow(a);
title('original img');
t=100;
[m,n]=size(b);
for i=1:m
for j=1:n
if (b(i,j)<t)
c(i,j)=0;
else
c(i,j)=255;
end
end
end
subplot(1,2,2);
imshow(c);
title('threshold img');
```

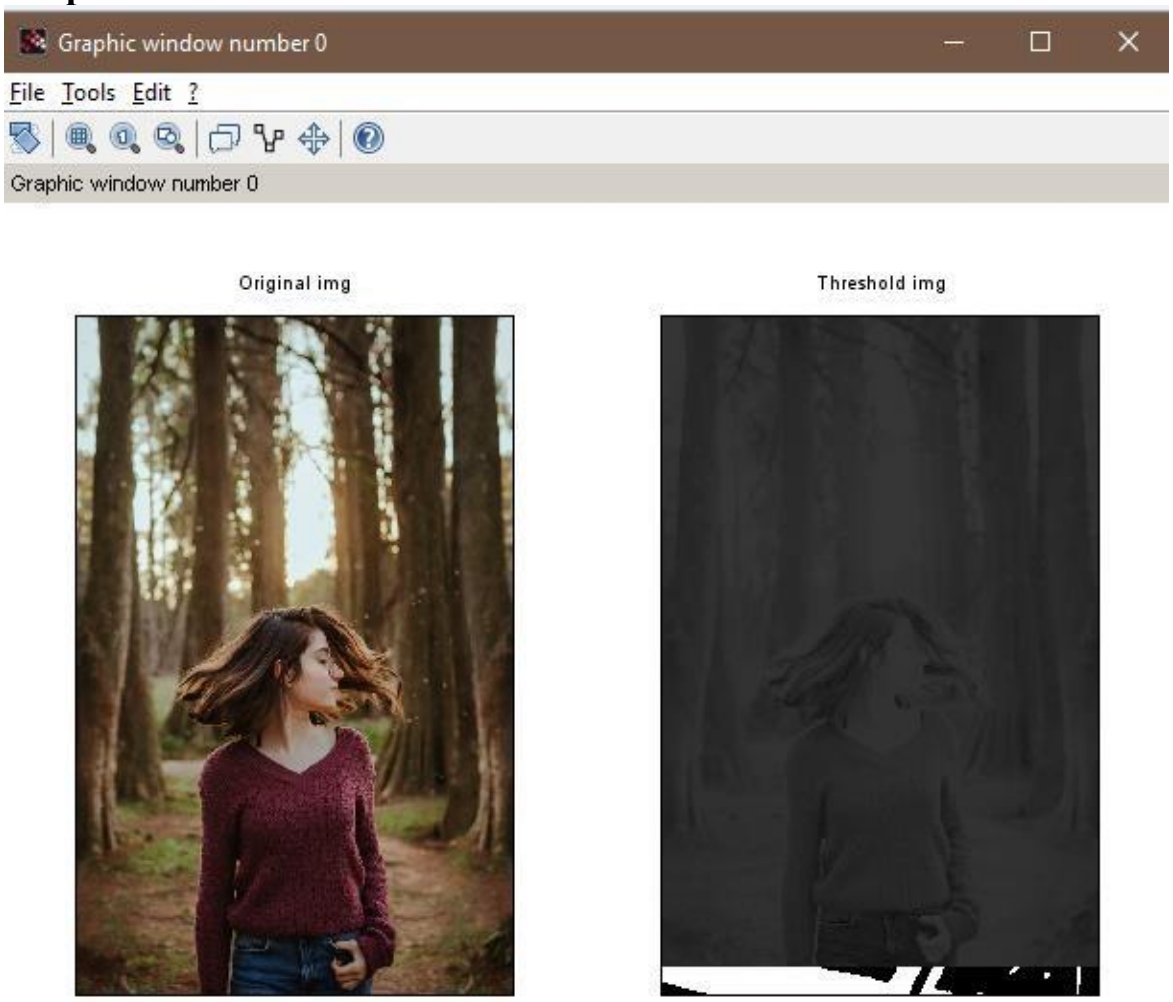
Output:



C) Program to perform Log transformation

```
clc;
clear all;
a=imread("C:\Users\bnnco\xyz3.jpg");
b=double(a)
subplot(1,2,1);
imshow(a);
title('original img');
t=10;
[m,n]=size(b);
for i=1:m
for j=1:n
c(i,j)=t*log(1+b(i,j))
end
end
subplot(1,2,2);
imshow(uint8(c));
title('threshold img');
```

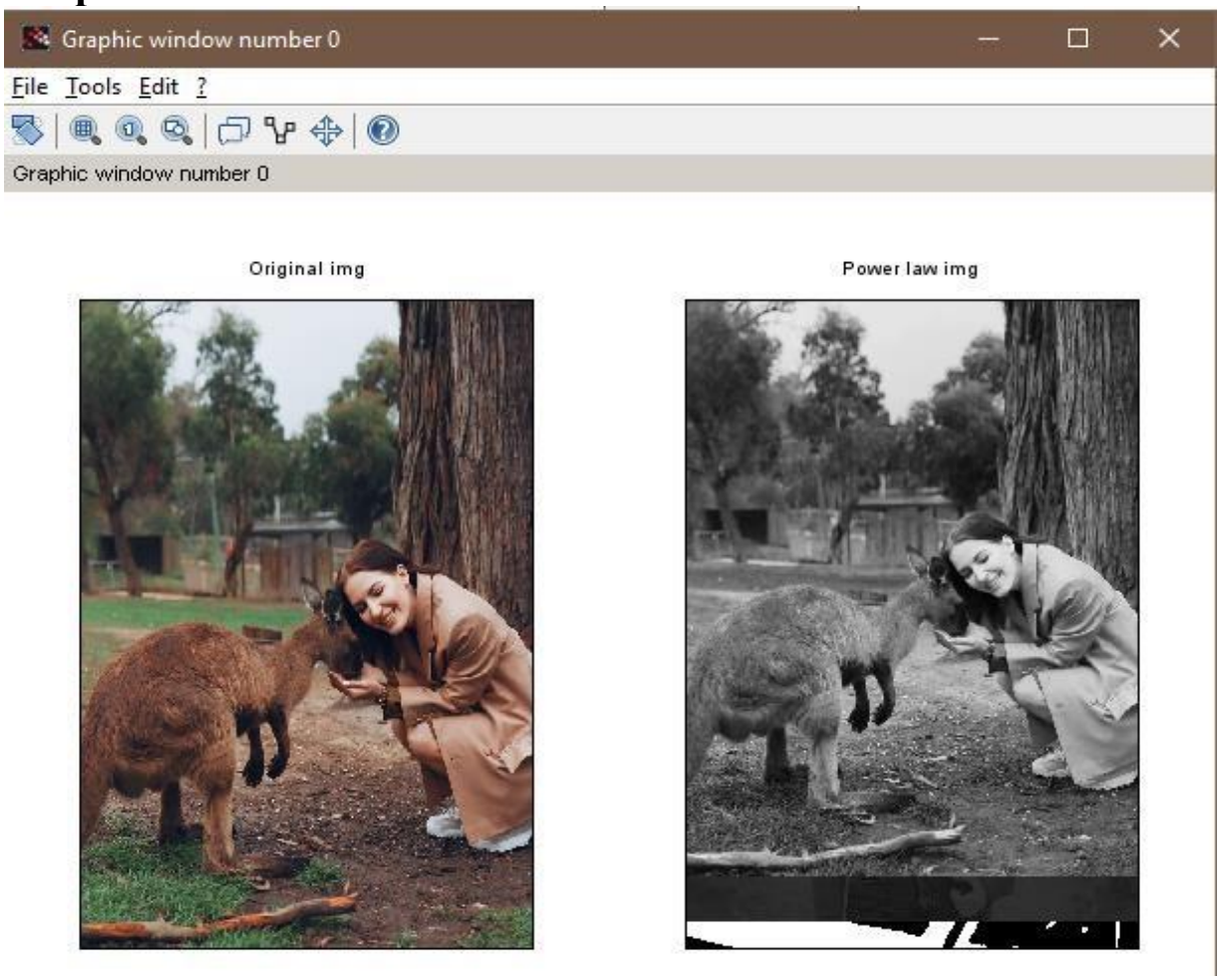
Output:



D) Power-law transformations

```
clc;
clear all;
a=imread("C:\Users\bnnco\xyz3.jpg");
b=double(a)
subplot(1,2,1);
imshow(a);
title('original img');
k=1;
gamma=1;
[m,n]=size(b);
for i=1:m
for j=1:n
c(i,j)=k*(b(i,j)^gamma);
end
end
subplot(1,2,2);
imshow(uint8(c));
title('power law img');
```

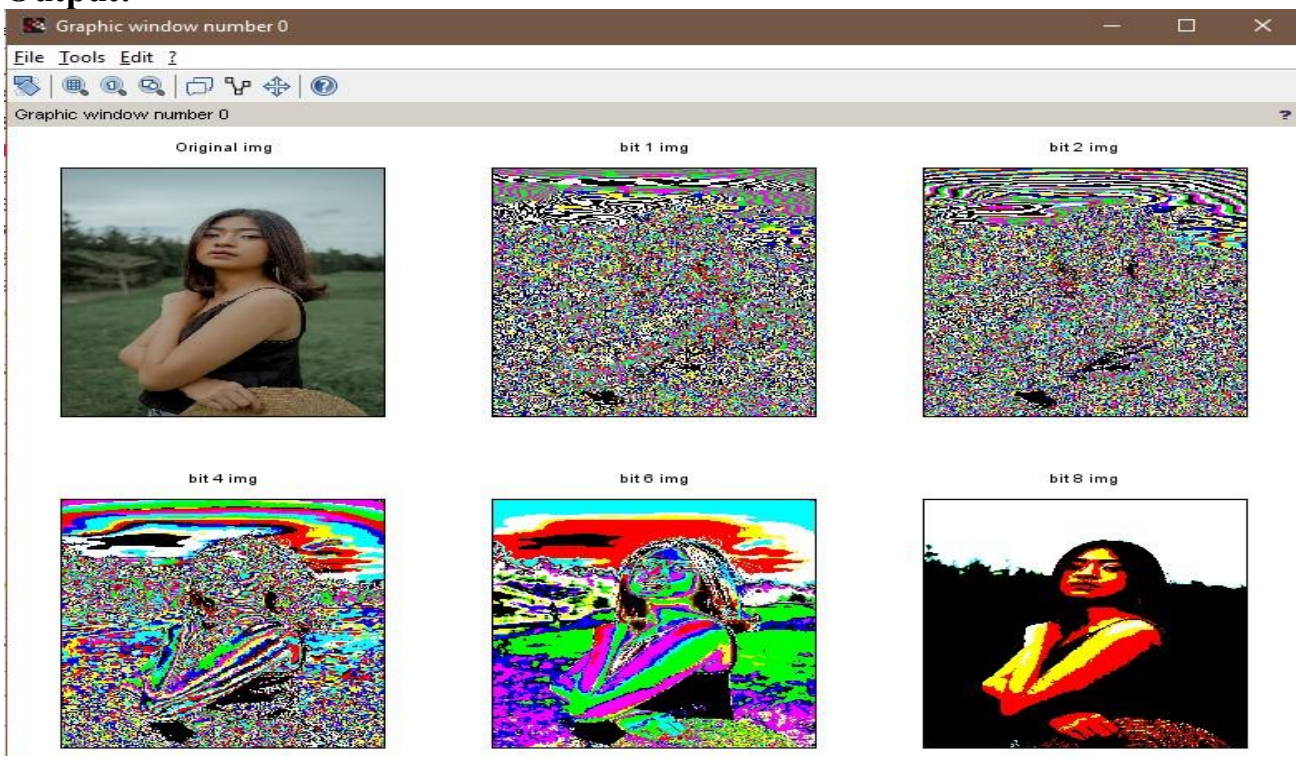
Output:



E) Piecewise linear transformations

```
clc;
clear all;
a=imread("C:\Users\bnnco\xyz.jpg");
b=double(a)
subplot(2,3,1);
imshow(a);
title('original img');
f1=bitget(b,1);
subplot(2,3,2);
imshow(f1);
title('bit 1 img');
f2=bitget(b,2);
subplot(2,3,3);
imshow(f2);
title('bit 2 img');
f3=bitget(b,4);
subplot(2,3,4);
imshow(f3);
title('bit 4 img');
f4=bitget(b,6);
subplot(2,3,5);
imshow(f4);
title('bit 6 img');
f5=bitget(b,8);
subplot(2,3,6);
imshow(f5);
title('bit 8 img');
```

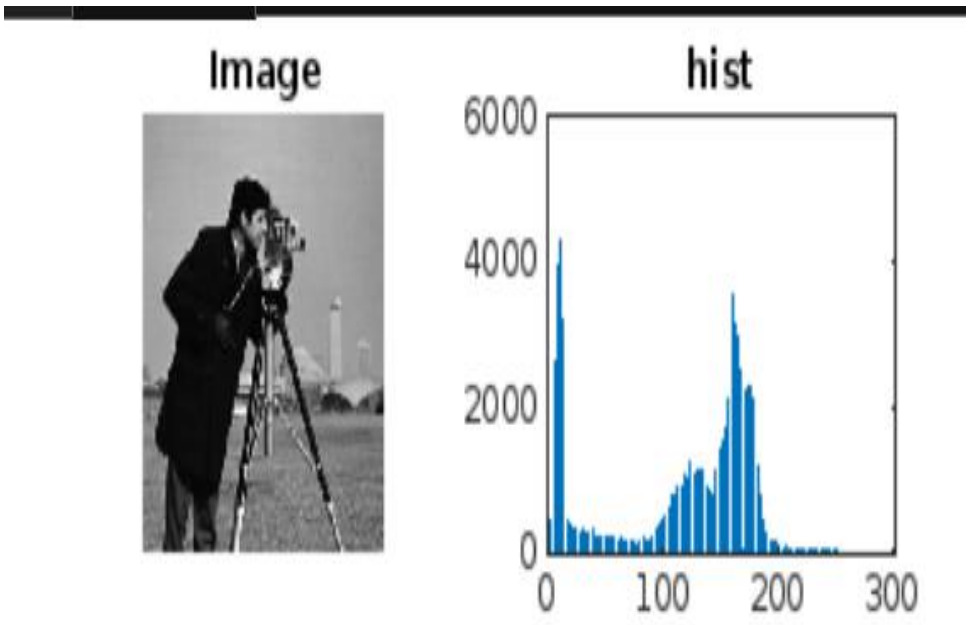
Output:



PRATICAL 3: Program to plot the histogram of an image and categorise

```
a = imread("cm.tif");
a =double(a);
big=256;
[row ,col ] = size(a);
c=row*col;
h=zeros(1,300);
z = zeros(1,300);
for n = 1:1:row
for m = 1:1:col
if a(n,m)==0
a(n,m) = 1;
end
end
end
for n = 1:1:row
for m = 1:1:col
t= a(n,m);
h(t) = h(t)+1;
end
end
subplot(1,2,1);imshow(uint8(a));title('Image');
subplot(1,2,2);bar(h);title('hist');
```

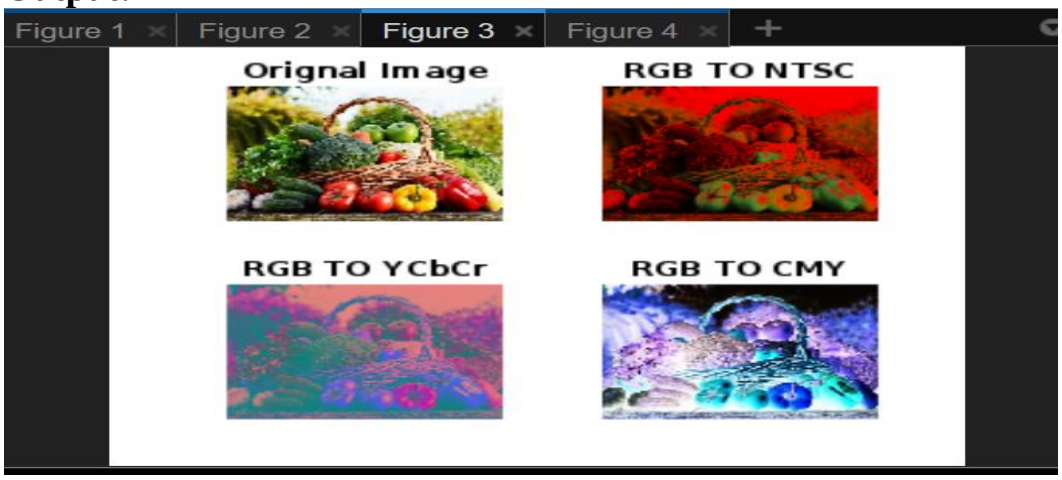
Output:



PRATICAL 4: Program to read a color image and segment into RGB planes , histogram of color image

```
a = imread('vege.jpg');
figure(3),subplot(2,2,1),imshow(a);
title('Orignal Image');
k=rgb2ntsc(a);
figure(3),subplot(2,2,2),imshow(k);
title('RGB TO NTSC');
l=rgb2ycbcr(a);
figure(3),subplot(2,2,3),imshow(l);
title('RGB TO YCbCr');
m=imcomplement(a);
figure(3),subplot(2,2,4),imshow(m);
title('RGB TO CMY');
imr=a(:,:,1);
img=a(:,:,2);
imb=a(:,:,3);
figure(4),subplot(2,2,1),imshow(a),title('original image');
figure(4),subplot(2,2,2),imshow(imr),title('Red');
figure(4),subplot(2,2,3),imshow(img),title('Green');
figure(4),subplot(2,2,4),imshow(imb),title('Blue');
```

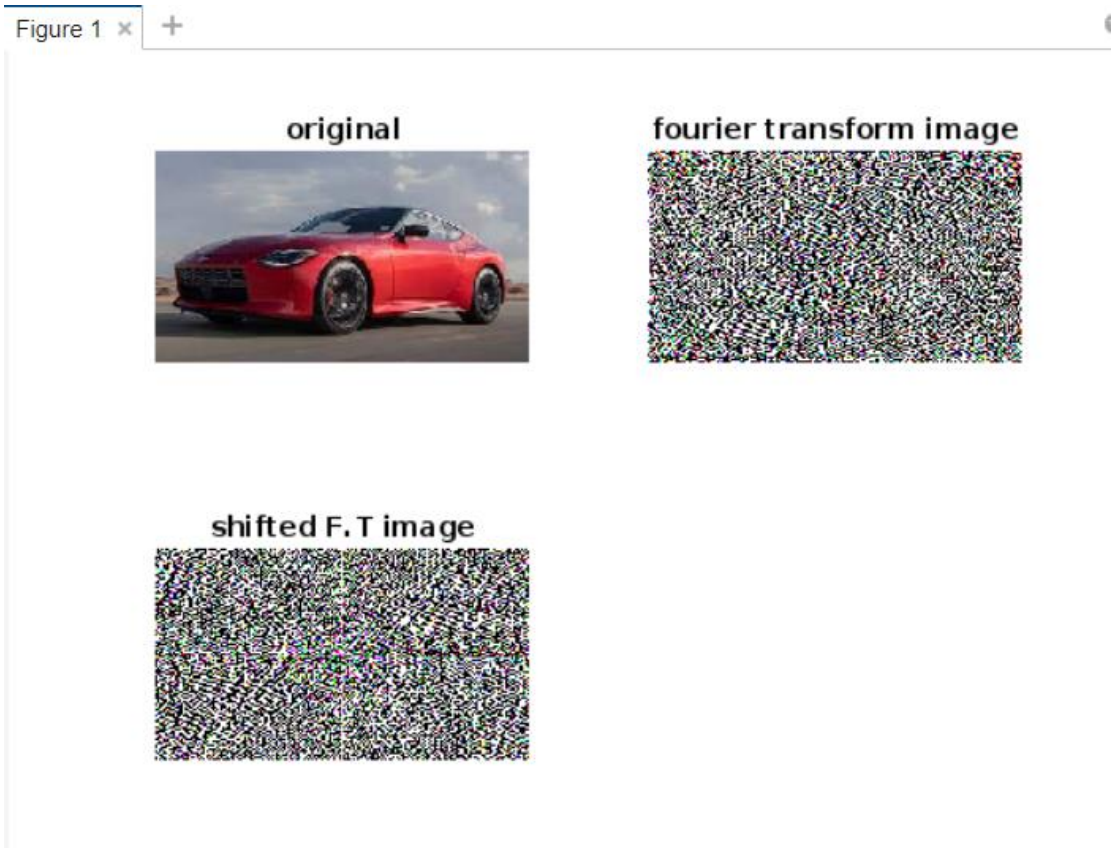
Output:



PRATICAL 5: Program to apply Discrete Fourier Transform on an image

```
I=imread("car.jpg");
subplot(2,2,1)
imshow(I)
title("original")
I=double(I);
J=fft2(I);
subplot(2,2,2)
imshow(J)
title("fourier transform image")
L=fftshift(real(J));
subplot(2,2,3)
imshow(L)
title("shifted F.T image")
```

Output:



PRATICAL 6 : Write a program to apply following morphological operations on the image

A) OPENING:-

```
img=imread('cameraman.tif');
se1 = strel('square',11);
im2 = imerode(img,se1);
im3 = imdilate(im2,se1);
subplot(1,2,1),imshow(img),title('orignal image');
subplot(1,2,2),imshow(im3),title('opening image');
```

Output:

B) CLOSING:-

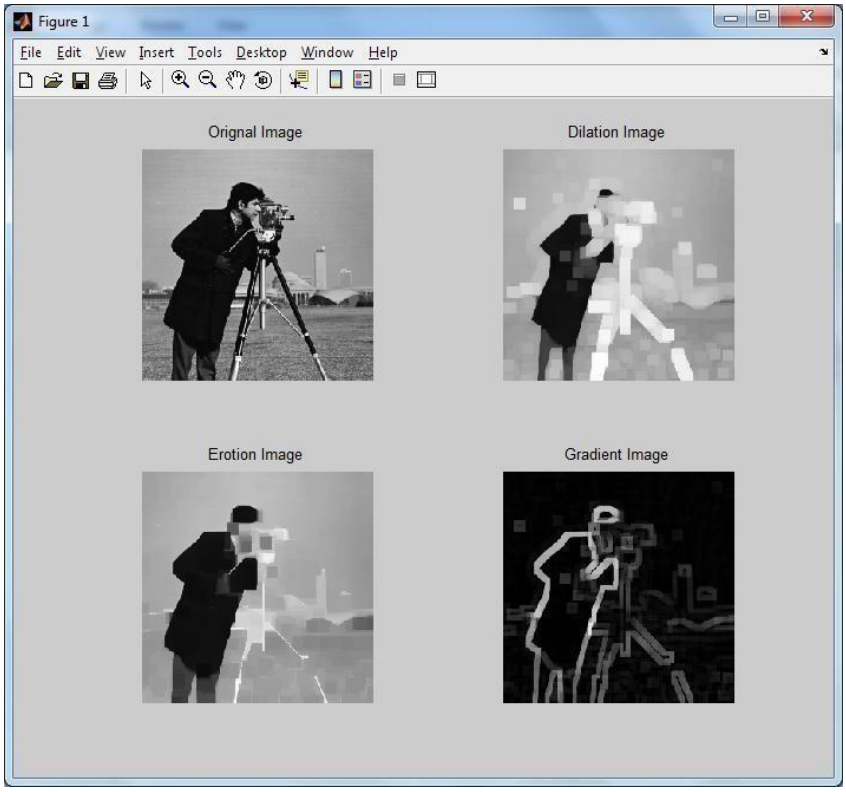
```
aa=imread('cameraman.tif');
se1=strel('square',11);
IM2=imdilate(aa,se1);
IM3=imerode(IM2,se1);
subplot(1,2,1),imshow(aa),title('Original Image');
subplot(1,2,2),imshow(IM3),title('Closed Image');
```

Output:

C) MORPHOLOGICAL GRADIENT:-

```
img=imread('cameraman.tif');
se1=strel('square',12);
im1=imdilate(img,se1);
im2=imerode(im1,se1);
g=im1-im2;
subplot(2,2,1),imshow(img),title('Orignal Image');
subplot(2,2,2),imshow(im1),title('Dilation Image');
subplot(2,2,3),imshow(im2),title('Erotion Image');
subplot(2,2,4),imshow(g),title('Gradient Image');
```

Output:



D) TOP-HAT TRANSFORMATION:-

```
i=imread('cameraman.tif');
se1=strel('square',22);
im1=imerode(i,se1);
im2=imdilate(im1,se1);
h=i-im2;
subplot(2,2,1),imshow(i),title('Orignal Image');
subplot(2,2,2),imshow(im1),title('Erotion Image');
subplot(2,2,3),imshow(im2),title('Dilation Image');
subplot(2,2,4),imshow(h),title('Top Hat Transformation Image');
```

Output: