Shri Vile Parle Kelavani Mandal's
USHA PRAVIN GANDHI COLLEGE OF ARTS, SCIENCE AND COMMERCE
(Autonomous Affiliated to University of Mumbai)
Bhaktivedanta Swami Marg, Juhu Scheme, Vile Parle (West), Mumbai - 400056
Tel.: 42332041/42/43/44/45/46 ● Website: www.upgcm.ac.in ● Email: info@upgcm.ac.in
NAAC RE-ACCREDITED " A+ " GRADE WITH CGPA 3.27

CERTIFICATE

M.Sc.(I.T.) Part – I, Semester - II
2024 - 2025

This is to certify that

Mr. / Ms. : _____,

student of Master of Science (Information Technology) Part - I, Semester - II Roll No: _____ *has successfully completed practical and project work in* _____,

in partial fulfillment as per the syllabus for the academic year 2024 – 2025.

The practical and project work are completed successfully under the supervision of the undersigned.

Internal Examiner	***External Examiner***
Name:	***Name:***
Date:	***Date:***
Program Coordinator	***Head of Department***
Name: Dr. Swapnali Lotlikar	***Name: Dr. Smruti Nanavaty***
Date:	***Date:***

INDEX

Practical 1

Aim: TO Configure and verify the IP SLA feature.

TO Configure and verify the IP SLA feature on a router using GNS3.

Objectives:

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

Required Resources:

• GNS3 version 6.x

• Cisco Routers (IOS 15.x or equivalent)

• Ethernet cables for connectivity

• Two or more networks to demonstrate path control

Network Topology:

The network consists of four routers (R1, R2, R3, R4) connected in a triangle with R3 acting as the central node.

Step 1: Configure loopbacks and assign addresses.

Router R1

```
R1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#interface fastEthernet 0/0
R1(config-if)#ip add 10.0.0.1 255.255.255.0
R1(config-if)#no shutdown
R1(config-if)#
*Mar 19 13:53:43.147: %LINK-3-UPDOWN: Interface FastEthernet0/0, changed state t
o up
```

```
R1(config-if)#exit
R1(config)#router rip
R1(config-router)#version 2
R1(config-router)#network 10.0.0.0
R1(config-router)#exit
```

```
R1(config-sla-monitor)#
R1#
*Mar 19 14:01:04.303: %SYS-5-CONFIG_I: Configured from console by console
R1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#rtr 1
R1(config-sla-monitor)#type echo protocol ipIcmpEcho 10.0.1.2
R1(config-sla-monitor-echo)#timeout 1000
R1(config-sla-monitor-echo)#frequency 3
R1(config-sla-monitor-echo)#exit
```

```
R1(config)#rtr schedule 1 start-time now life forever
R1(config)#track 1 rtr 1 reachability
R1(config-track)#exit
R1(config)#ip route 0.0.0.0 0.0.0.0 10.0.0.2 track 1
R1(config)#ip route 0.0.0.0 0.0.0.0 10.0.1.2 10
R1(config)#exit
R1#
*Mar 19 14:07:39.687: %SYS-5-CONFIG_I: Configured from console by console
```

```
R1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#rtr 1
R1(config-sla-monitor)#type echo protocol ipIcmpEcho 10.0.1.2
R1(config-sla-monitor-echo)#timeout 1000
R1(config-sla-monitor-echo)#frequency 3
R1(config-sla-monitor-echo)#exit
```

Router ISP1 (R2)

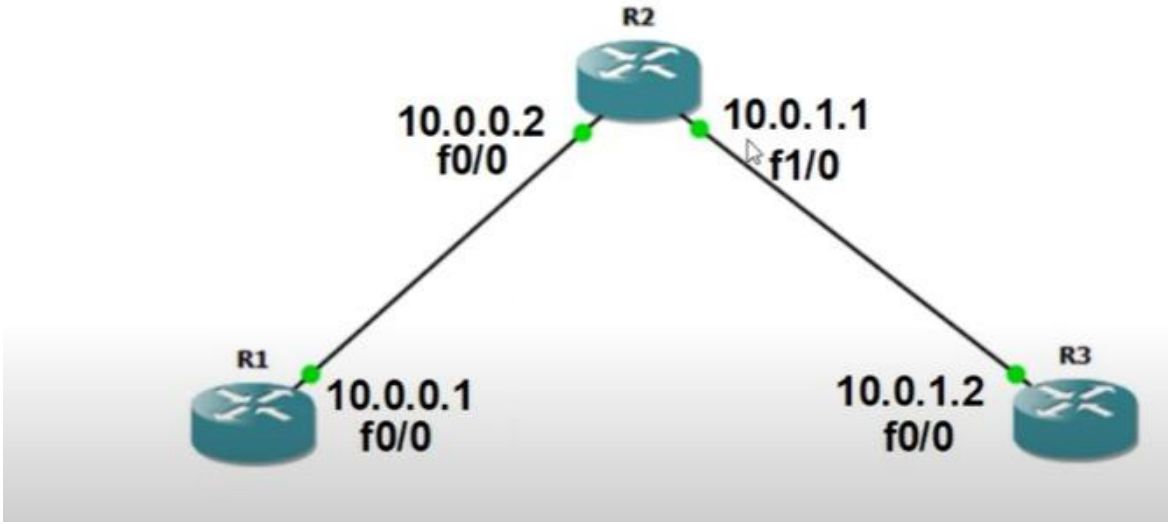
```
R2#
R2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#interface fastethernet 0/0
R2(config-if)#ip add 10.0.0.2 255.255.255.0
R2(config-if)#no shutdown
R2(config-if)#
*Mar 19 14:14:08.931: %LINK-3-UPDOWN: Interface FastEthernet0/0, changed state t
o up
```

```
R2(config)#interface fastethernet 1/0
R2(config-if)#ip add 10.0.1.1 255.255.255.0
R2(config-if)#no shutdown
R2(config-if)#
*Mar 19 14:15:05.303: %LINK-3-UPDOWN: Interface FastEthernet1/0, changed state t
o up
R2(config-if)#
*Mar 19 14:15:05.303: %ENTITY_ALARM-6-INFO: CLEAR INFO Fa1/0 Physical Port Admin
istrative State Down
*Mar 19 14:15:06.303: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthern
et1/0, changed state to up
R2(config-if)#exit
R2(config)#
R2(config)#router rip
R2(config-router)#version 2
R2(config-router)#network 10.0.0.0
R2(config-router)#network 10.0.1.0
R2(config-router)#exit
R2(config)#
```

Router ISP2 (R3)

```
R3#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#interface fastethernet 0/0
R3(config-if)#ip add 10.0.1.2 255.255.255.0
R3(config-if)#no shutdown
R3(config-if)#
*Mar 19 14:17:44.707: %LINK-3-UPDOWN: Interface FastEthernet0/0, changed state t
o up
R3(config-if)#
*Mar 19 14:17:44.707: %ENTITY_ALARM-6-INFO: CLEAR INFO Fa0/0 Physical Port Admin
istrative State Down
*Mar 19 14:17:45.707: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthern
et0/0, changed state to up
R3(config-if)#exit
R3(config)#router rip
R3(config-router)#version 2
R3(config-router)#network 10.0.1.0
R3(config-router)#exit
R3(config)#
```

OUTPUTS



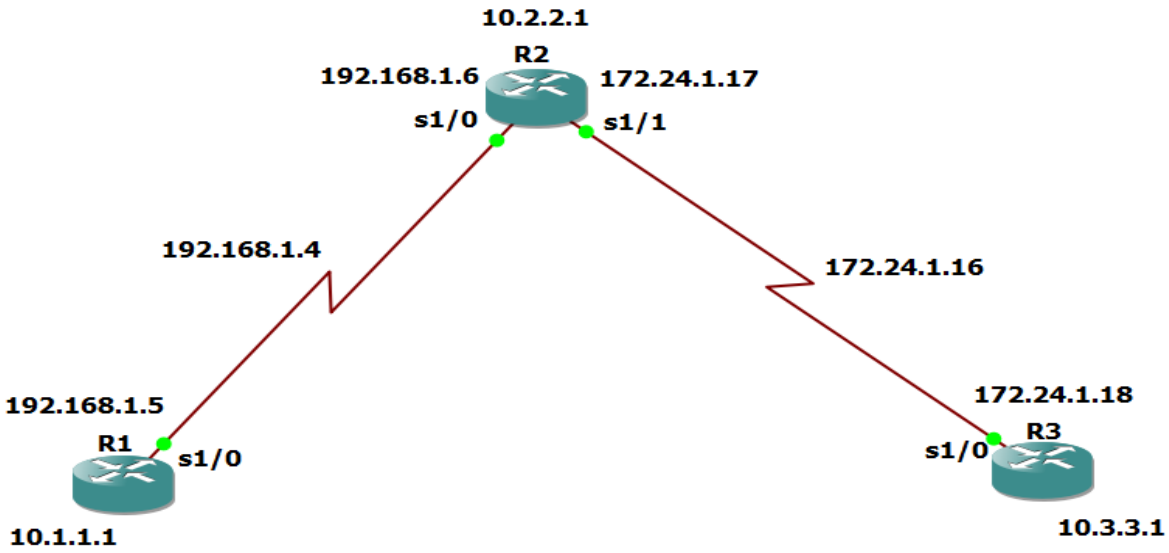

```
R1#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, 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 10.0.0.2 to network 0.0.0.0

     10.0.0.0/24 is subnetted, 2 subnets
C       10.0.0.0 is directly connected, FastEthernet0/0
R       10.0.1.0 [120/1] via 10.0.0.2, 00:00:18, FastEthernet0/0
S*   0.0.0.0/0 [1/0] via 10.0.0.2
R1#show track
Track 1
  Response Time Reporter 1 reachability
  Reachability is Up
    2 changes, last change 00:01:01
  Latest operation return code: OK
  Latest RTT (millisecs) 79
  Tracked by:
    STATIC-IP-ROUTING 0
```

```
R1#show track
Track 1
  Response Time Reporter 1 reachability
  Reachability is Up
    2 changes, last change 00:01:01
  Latest operation return code: OK
  Latest RTT (millisecs) 79
  Tracked by:
    STATIC-IP-ROUTING 0
```

```
R1#show rtr operational-state
Entry number: 1
Modification time: *14:05:54.411 UTC Wed Mar 19 2025
Number of Octets Used by this Entry: 2272
Number of operations attempted: 306
Number of operations skipped: 0
Current seconds left in Life: Forever
Operational state of entry: Active
Last time this entry was reset: Never
Connection loss occurred: FALSE
Timeout occurred: FALSE
Over thresholds occurred: FALSE
Latest RTT (milliseconds): 88
Latest operation start time: *14:21:09.411 UTC Wed Mar 19 2025
Latest operation return code: OK
RTT Values:
RTTAvg: 88      RTTMin: 88      RTTMax: 88
NumOfRTT: 1     RTTSum: 88      RTTSum2: 7744


R1#show track 1
Track 1
  Response Time Reporter 1 reachability
  Reachability is Up
    2 changes, last change 00:03:14
  Latest operation return code: OK
  Latest RTT (millisecs) 88
  Tracked by:
    STATIC-IP-ROUTING 0
R1#show running-config | section rtr
track 1 rtr 1 reachability
R1#ping 10.0.0.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.0.0.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 32/49/60 ms
R1#
```

Practical 2

Aim: Using the AS_PATH Attribute



Objectives:

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

Background:

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

Required Resources

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

Step 1: Configure Interfaces Addresses

SanJose(R1)

```
SanJose(config)# interface Loopback0
```

```
SanJose(config-if)# ip address 10.1.1.1 255.255.255.0

SanJose(config-if)# exit

SanJose(config)# interface Serial0/0/0

SanJose(config-if)# ip address 192.168.1.5 255.255.255.252

SanJose(config-if)# clock rate 128000

SanJose(config-if)# no shutdown

SanJose(config-if)# end

SanJose#
```

ISP(R2)

```
ISP(config)# interface Loopback0

ISP(config-if)# ip address 10.2.2.1 255.255.255.0

ISP(config-if)# interface Serial0/0/0

ISP(config-if)# ip address 192.168.1.6 255.255.255.252

ISP(config-if)# no shutdown

ISP(config-if)# exit

ISP(config)# interface Serial0/0/1

ISP(config-if)# ip address 172.24.1.17 255.255.255.252

ISP(config-if)# clock rate 128000

ISP(config-if)# no shutdown

ISP(config-if)# end

ISP#
```

CustRtr (R3)

```
CustRtr(config)# interface Loopback0

CustRtr(config-if)# ip address 10.3.3.1 255.255.255.0

CustRtr(config-if)# exit

CustRtr(config)# interface Serial0/0/1

CustRtr(config-if)# ip address 172.24.1.18 255.255.255.252

CustRtr(config-if)# no shutdown

CustRtr(config-if)# end

CustRtr#
```

Step 2: Configure BGP

SanJose(R1)

```
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(R2)

```
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 (R3)

```
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
```

ISP(R2)

ISP# show ip bgp neighbors

BGP neighbor is 172.24.1.18, remote AS 65000, external link

BGP version 4, remote router ID 10.3.3.1

BGP state = Established, up for 00:00:28

Last read 00:00:28, last write 00:00:28, hold time is 180, keepalive interval is 60 seconds

<output omitted>

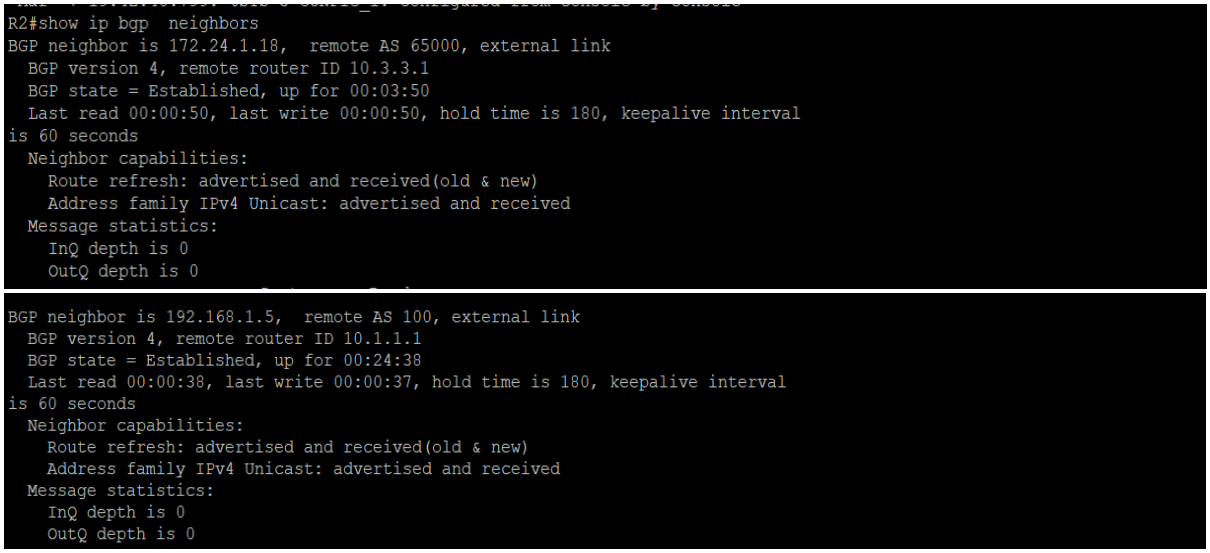
BGP neighbor is 192.168.1.5, remote AS 100, external link

BGP version 4, remote router ID 10.1.1.1

BGP state = Established, up for 00:01:34

Last read 00:00:33, last write 00:00:06, hold time is 180, keepalive interval is 60 seconds

<output omitted>



Step 3: Remove the private AS

SanJose(R1)

SanJose#show ip route

Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP

D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area

N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2

E1 - OSPF external type 1, E2 - OSPF external type 2

i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2

ia - IS-IS inter area, * - candidate default, U - per-user static route

o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP

a - application route

+ - replicated route, % - next hop override

Gateway of last resort is not set

10.0.0.0/8 is variably subnetted, 4 subnets, 2 masks

C 10.1.1.0/24 is directly connected, Loopback0

L 10.1.1.1/32 is directly connected, Loopback0

B 10.2.2.0/24 [20/0] via 192.168.1.6, 00:04:22

B 10.3.3.0/24 [20/0] via 192.168.1.6, 00:03:14

192.168.1.0/24 is variably subnetted, 2 subnets, 2 masks

C 192.168.1.4/30 is directly connected, Serial0/0/0

L 192.168.1.5/32 is directly connected, Serial0/0/0

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

Gateway of last resort is not set

     10.0.0.0/24 is subnetted, 3 subnets
B       10.3.3.0 [20/0] via 192.168.1.6, 00:18:36
B       10.2.2.0 [20/0] via 192.168.1.6, 00:39:27
C       10.1.1.0 is directly connected, Loopback0
     192.168.1.0/30 is subnetted, 1 subnets
C       192.168.1.4 is directly connected, Serial1/0
R1#
```

SanJose# ping 10.3.3.1 source 10.1.1.1

```
R1#
R1#ping 10.3.3.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.3.3.1, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
R1#ping 10.3.3.1 source 10.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.3.3.1, timeout is 2 seconds:
Packet sent with a source address of 10.1.1.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 64/98/128 ms
R1#
```

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

ISP(R2)

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

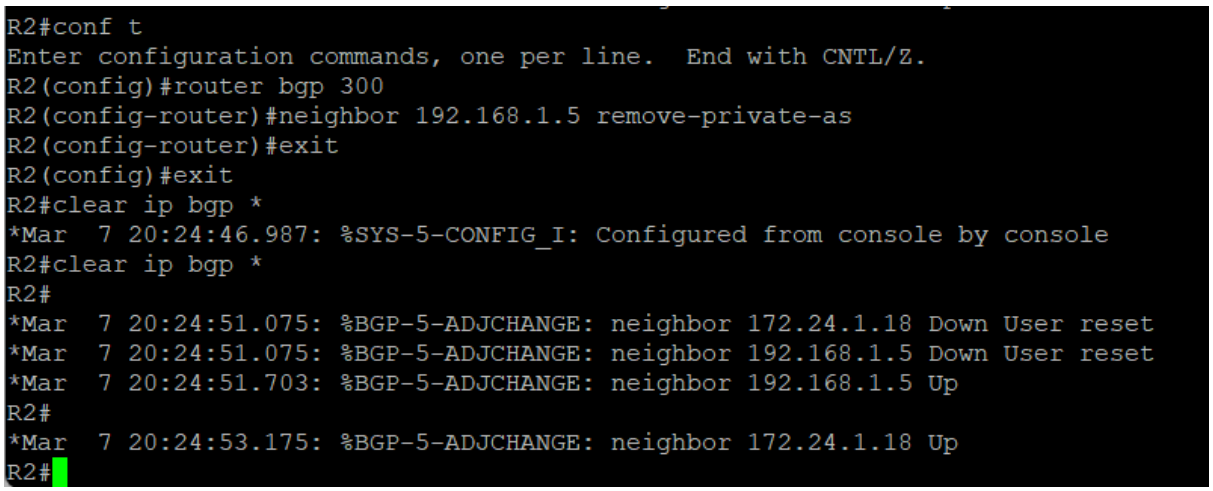
*Sep 8 18:40:03.551: %BGP_SESSION-5-ADJCHANGE: neighbor 192.168.1.5 IPv4 Unicast topology base removed from session User reset

*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(R1)

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,

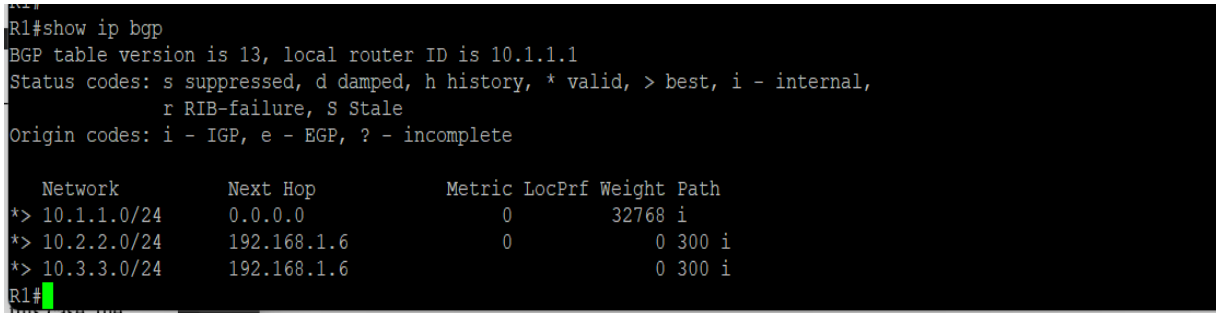
x best-external, a additional-path, c RIB-compressed,

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



Step 4: Use the AS_PATH attribute to filter routes

ISP(R2)

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

ISP(config)# ip as-path access-list 1 permit .*

ISP(config)# router bgp 300

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

ISP# clear ip bgp *

ISP#

*Sep 8 18:48:04.915: %BGP-5-ADJCHANGE: neighbor 172.24.1.18 Down User reset

*Sep 8 18:48:04.915: %BGP_SESSION-5-ADJCHANGE: neighbor 172.24.1.18 IPv4 Unicast topology base removed from session User reset

*Sep 8 18:48:04.915: %BGP-5-ADJCHANGE: neighbor 192.168.1.5 Down User reset

*Sep 8 18:48:04.915: %BGP_SESSION-5-ADJCHANGE: neighbor 192.168.1.5 IPv4 Unicast topology base removed from session User reset

*Sep 8 18:48:04.951: %BGP-5-ADJCHANGE: neighbor 172.24.1.18 Up

*Sep 8 18:48:04.955: %BGP-

ISP#5-ADJCHANGE: neighbor 192.168.1.5 Up

```
ISP# show ip route

     10.0.0.0/8 is variably subnetted, 4 subnets, 2 masks

B       10.1.1.0/24 [20/0] via 192.168.1.5, 00:00:29

C       10.2.2.0/24 is directly connected, Loopback0

L       10.2.2.1/32 is directly connected, Loopback0

B       10.3.3.0/24 [20/0] via 172.24.1.18, 00:00:29

     172.24.0.0/16 is variably subnetted, 2 subnets, 2 masks

C       172.24.1.16/30 is directly connected, Serial0/0/1

L       172.24.1.17/32 is directly connected, Serial0/0/1

     192.168.1.0/24 is variably subnetted, 2 subnets, 2 masks

C       192.168.1.4/30 is directly connected, Serial0/0/0

L       192.168.1.6/32 is directly connected, Serial0/0/0

ISP#
```

```
R2#clear ip bgp *
R2#
*Mar  7 22:19:42.062: %BGP-5-ADJCHANGE: neighbor 172.24.1.18 Down User reset
*Mar  7 22:19:42.062: %BGP-5-ADJCHANGE: neighbor 192.168.1.5 Down User reset
*Mar  7 22:19:42.394: %BGP-5-ADJCHANGE: neighbor 192.168.1.5 Up
R2#
*Mar  7 22:19:43.234: %BGP-5-ADJCHANGE: neighbor 172.24.1.18 Up
R2#
```

```
R2#
R2#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     172.24.0.0/30 is subnetted, 1 subnets
C       172.24.1.16 is directly connected, Serial1/1
     10.0.0.0/24 is subnetted, 3 subnets
B       10.3.3.0 [20/0] via 172.24.1.18, 00:06:58
C       10.2.2.0 is directly connected, Loopback0
B       10.1.1.0 [20/0] via 192.168.1.5, 00:06:58
     192.168.1.0/30 is subnetted, 1 subnets
C       192.168.1.4 is directly connected, Serial1/0
R2#
```

CustRtr(R3)

```
CustRtr# show ip route

      10.0.0.0/8 is variably subnetted, 3 subnets, 2 masks

B        10.2.2.0/24 [20/0] via 172.24.1.17, 00:00:32

C        10.3.3.0/24 is directly connected, Loopback0

L        10.3.3.1/32 is directly connected, Loopback0

      172.24.0.0/16 is variably subnetted, 2 subnets, 2 masks

C        172.24.1.16/30 is directly connected, Serial0/0/1

L        172.24.1.18/32 is directly connected, Serial0/0/1

CustRtr#
```

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

Gateway of last resort is not set

     172.24.0.0/30 is subnetted, 1 subnets
C       172.24.1.16 is directly connected, Serial1/0
     10.0.0.0/24 is subnetted, 2 subnets
C       10.3.3.0 is directly connected, Loopback0
B       10.2.2.0 [20/0] via 172.24.1.17, 00:20:15
R3#
```

ISP(R2)

```
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

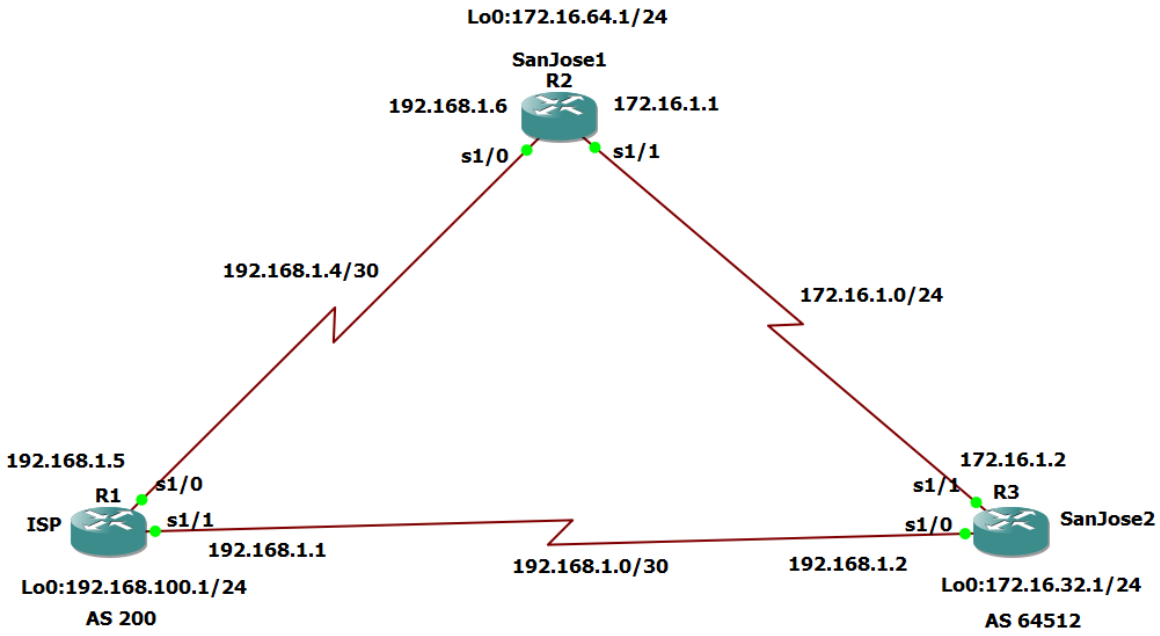
```
R2#show ip bgp regexp ^100$
BGP table version is 4, local router ID is 10.2.2.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 10.1.1.0/24      192.168.1.5              0             0 100 i
R2#
R2#
```

```
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 }
```

```
R2#ping 10.1.1.1

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 = 28/37/44 ms
R2#ping 10.2.2.1

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 = 1/2/4 ms
R2#ping 10.3.3.1

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 = 32/36/48 ms
```

Practical 3

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



Objectives:

• For IBGP peers to correctly exchange routing information, use the next-hop-self command with the Local-Preference and MED attributes.

• Ensure that the flat-rate, unlimited-use T1 link is used for sending and receiving data to and from the AS 200 on ISP and that the metered T1 only be used in the event that the primary T1 link has failed.

Background:

The International Travel Agency runs BGP on its SanJose1 and SanJose2 routers externally with the ISP router in AS 200. IBGP is run internally between SanJose1 and SanJose2. Your job is to configure both EBGP and IBGP for this internetwork to allow for redundancy. The metered T1 should only be used in the event that the primary T1 link has failed. Traffic sent across the metered T1 link offers the same bandwidth of the primary link but at a huge expense. Ensure that this link is not used unnecessarily.

Required Resources

• 3 routers (Cisco IOS Release 15.2 or comparable)

• Serial and Ethernet cables

Step 1: Configure Interface Addresses

ISP(R1)

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

SanJose1(R2)

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

```
SanJose1(config-if)# clock rate 128000
SanJose1(config-if)# no shutdown
SanJose1(config-if)# end
SanJose1#
```

SanJose2(R3)

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

Step 2: Configure EIGRP

SanJose1(R2)

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

SanJose2(R3)

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

Step 3: Configure IBGP and verify BGP neighbors

SanJose1(R2)

```
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
```

SanJose2(R3)

```
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
```

SanJose2(R3)

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

```
R3#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 00:00:25
  Last read 00:00:25, last write 00:00:25, hold time is 180, keepalive interval is 60 seconds
  Neighbor capabilities:
    Route refresh: advertised and received(old & new)
    Address family IPv4 Unicast: advertised and received
  Message statistics:
    InQ depth is 0
    OutQ depth is 0
                         Sent       Rcvd
    Opens:                  1          1
    Notifications:          0          0
    Updates:                0          0
    Keepalives:             1          1
    Route Refresh:          0          0
    Total:                  2          2
```

Step 4: Configure EBGP and verify BGP neighbors

ISP(R1)

```
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
```

SanJose1(R2)

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

SanJose1(R2)

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

SanJose1(R2)

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


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

```
R2#show ip bgp neighbors
BGP neighbor is 172.16.32.1,  remote AS 64512, internal link
  BGP version 4, remote router ID 172.16.32.1
  BGP state = Established, up for 00:04:41
  Last read 00:00:41, last write 00:00:14, hold time is 180, keepalive interval is 60 seconds
  Neighbor capabilities:
    Route refresh: advertised and received(old & new)
    Address family IPv4 Unicast: advertised and received
  Message statistics:
    InQ depth is 0
    OutQ depth is 0
                         Sent       Rcvd
    Opens:                  1          1
    Notifications:          0          0
    Updates:                1          0
    Keepalives:             6          6
    Route Refresh:          0          0
    Total:                  8          7
  Default minimum time between advertisement runs is 0 seconds
```

SanJose2(R3)

```
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

SanJose2(R3)

```
SanJose2# show ip bgp summary
BGP router identifier 172.16.32.1, local AS number 64512
BGP table version is 6, main routing table version 6
2 network entries using 288 bytes of memory
4 path entries using 320 bytes of memory
4/2 BGP path/bestpath attribute entries using 640 bytes of memory
1 BGP AS-PATH entries using 24 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 1272 total bytes of memory
BGP activity 2/0 prefixes, 4/0 paths, scan interval 60 secs
```

```
Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down
State/PfxRcd

172.16.64.1     4        64512      27      26        6    0    0 00:18:15
2

192.168.1.1     4          200      10       7        6    0    0 00:01:42
1

SanJose2#
```

```
R3#show ip bgp summary
BGP router identifier 172.16.32.1, local AS number 64512
BGP table version is 4, main routing table version 4
1 network entries using 117 bytes of memory
2 path entries using 104 bytes of memory
3/1 BGP path/bestpath attribute entries using 372 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 593 total bytes of memory
BGP activity 1/0 prefixes, 2/0 paths, scan interval 60 secs

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
172.16.64.1     4 64512      11      11        4    0    0 00:07:05        1
192.168.1.1     4   200       5       5        3    0    0 00:00:22        0
R3#
```

Step 6: Verify which path the traffic takes

ISP(R1)

```
ISP# clear ip bgp *

ISP#

*Nov 9 22:05:32.427: %BGP-5-ADJCHANGE: neighbor 192.168.1.2 Down User
reset

*Nov 9 22:05:32.427: %BGP_SESSION-5-ADJCHANGE: neighbor 192.168.1.2
IPv4 Unicast topology base removed from session User reset

*Nov 9 22:05:32.427: %BGP-5-ADJCHANGE: neighbor 192.168.1.6 Down User
reset

*Nov 9 22:05:32.427: %BGP_SESSION-5-ADJCHANGE: neighbor 192.168.1.6
IPv4 Unicast topology base removed from session User reset

*Nov 9 22:05:32.851: %BGP-5-ADJCHANGE: neighbor 192.168.1.2 Up

*Nov 9 22:05:32.851: %BGP-

ISP#5-ADJCHANGE: neighbor 192.168.1.6 Up

ISP#
```

```
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#

```
R1#clear ip bgp *
R1#
*Mar  7 14:17:14.819: %BGP-5-ADJCHANGE: neighbor 192.168.1.2 Down User reset
*Mar  7 14:17:14.819: %BGP-5-ADJCHANGE: neighbor 192.168.1.6 Down User reset
*Mar  7 14:17:15.779: %BGP-5-ADJCHANGE: neighbor 192.168.1.6 Up
R1#
*Mar  7 14:17:17.295: %BGP-5-ADJCHANGE: neighbor 192.168.1.2 Up
R1#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)
R1#
```

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#

```
R1#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 = 28/38/64 ms
R1#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 = 24/32/44 ms
R1#
```

ISP# show ip bgp

BGP table version is 3, local router ID is 192.168.100.1

Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,

r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,

x best-external, a additional-path, c RIB-compressed,

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

RPKI validation codes: V valid, I invalid, N Not found

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

ISP#

ISP# show ip bgp

ISP# ping 172.16.1.1 source 192.168.100.1

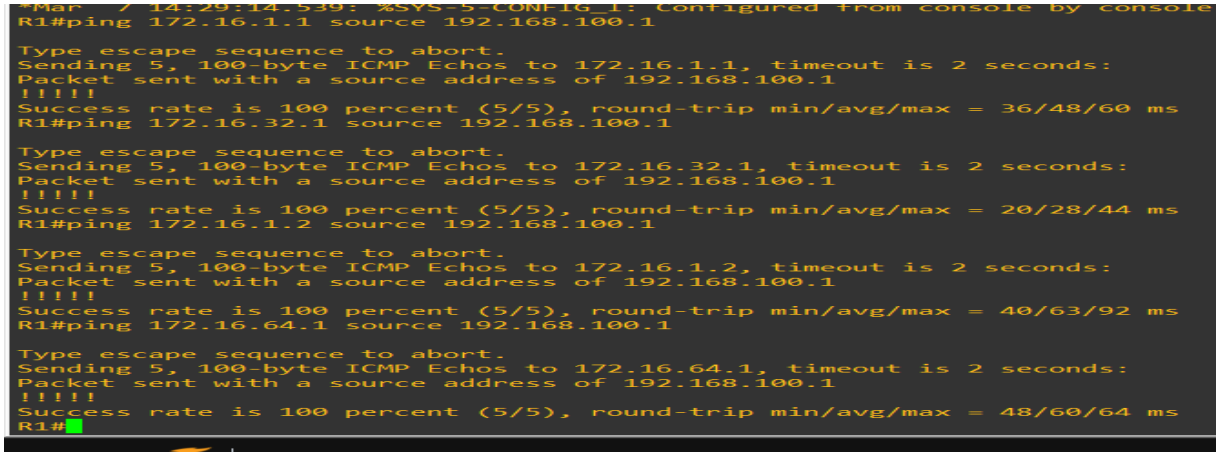
Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 172.16.1.1, timeout is 2 seconds:

```
Packet sent with a source address of 192.168.100.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 20/21/24 ms

ISP# ping 172.16.32.1 source 192.168.100.1

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

ISP# ping 172.16.1.2 source 192.168.100.1

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

ISP# ping 172.16.64.1 source 192.168.100.1

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

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

```
ISP# ping

Protocol [ip]:

Target IP address: 172.16.64.1

Repeat count [5]:

Datagram size [100]:

Timeout in seconds [2]:

Extended commands [n]: y

Source address or interface: 192.168.100.1

Type of service [0]:

Set DF bit in IP header? [no]:

Validate reply data? [no]:

Data pattern [0xABCD]:

Loose, Strict, Record, Timestamp, Verbose[none]:

Sweep range of sizes [n]:

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 172.16.64.1, timeout is 2 seconds:

Packet sent with a source address of 192.168.100.1

!!!!!

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

ISP#
```



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

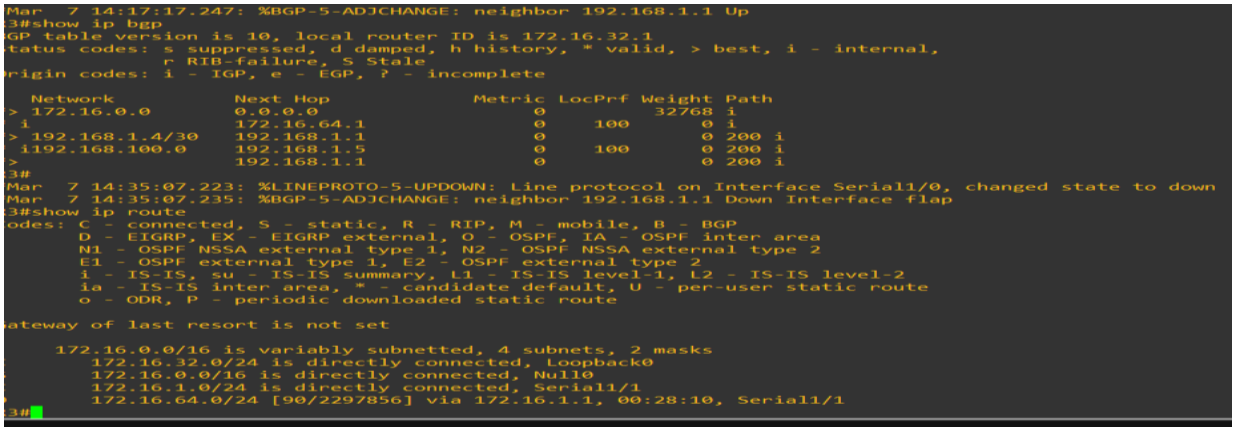
ISP(R1)

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

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

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

SanJose2(R3)

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

Gateway of last resort is not set

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



ISP(R1)

```
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)#
```

SanJose2(R3)

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

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

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

Gateway of last resort is not set

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

SanJose1(R2)

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

SanJose2(R3)

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

SanJose1(R2)

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

SanJose2(R3)

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

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

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

```
3#clear ip bgp *
3#
Mar  7 14:38:19.367: %BGP-5-ADJCHANGE: neighbor 172.16.64.1 Down User reset
3#
3#
Mar  7 14:38:21.907: %BGP-5-ADJCHANGE: neighbor 172.16.64.1 Up
3#show ip bgp
GP table version is 1, local router ID is 172.16.32.1
tatus codes: s suppressed, d damped, h history, * valid, > best, i - internal,
             r RIB-failure, S Stale
rigin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
 i192.168.100.0     172.16.64.1              0    100      0 200 i
3#show ip route
odes: 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

ateway of last resort is not set

    172.16.0.0/16 is variably subnetted, 4 subnets, 2 masks
       172.16.32.0/24 is directly connected, Loopback0
       172.16.0.0/16 is directly connected, Null0
       172.16.1.0/24 is directly connected, Serial1/1
       172.16.64.0/24 [90/2297856] via 172.16.1.1, 00:31:42, Serial1/1
3#
```

SanJose2# show ip route

Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP

D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area

N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2

E1 - OSPF external type 1, E2 - OSPF external type 2

i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2

ia - IS-IS inter area, * - candidate default, U - per-user static route

o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP

a - application route

+ - replicated route, % - next hop override

Gateway of last resort is not set

172.16.0.0/16 is variably subnetted, 6 subnets, 3 masks

S 172.16.0.0/16 is directly connected, Null0

C 172.16.1.0/24 is directly connected, Serial0/0/1

L 172.16.1.2/32 is directly connected, Serial0/0/1

C 172.16.32.0/24 is directly connected, Loopback0

L 172.16.32.1/32 is directly connected, Loopback0

D 172.16.64.0/24 [90/2297856] via 172.16.1.1, 04:27:19, Serial0/0/1

```
B     192.168.100.0/24 [200/0] via 172.16.64.1, 00:00:46
SanJose2#
```

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

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

Gateway of last resort is not set

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

```
L       192.168.1.2/32 is directly connected, Serial0/0/0
B     192.168.100.0/24 [20/0] via 192.168.1.1, 00:01:35
SanJose2#
```

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

Gateway of last resort is not set

     172.16.0.0/16 is variably subnetted, 4 subnets, 2 masks
C       172.16.32.0/24 is directly connected, Loopback0
S       172.16.0.0/16 is directly connected, Null0
C       172.16.1.0/24 is directly connected, Serial1/1
D       172.16.64.0/24 [90/2297856] via 172.16.1.1, 00:32:53, Serial1/1
     192.168.1.0/30 is subnetted, 1 subnets
C       192.168.1.0 is directly connected, Serial1/0
R3#
*Mar  7 14:40:07.223: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial1/0, changed state to up
R3#
*Mar  7 14:40:33.179: %BGP-5-ADJCHANGE: neighbor 192.168.1.1 Up
R3#
```

Step 8: Set BGP local preference

SanJose1(R2)

```
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(R3)

```
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
```

SanJose1(R2)

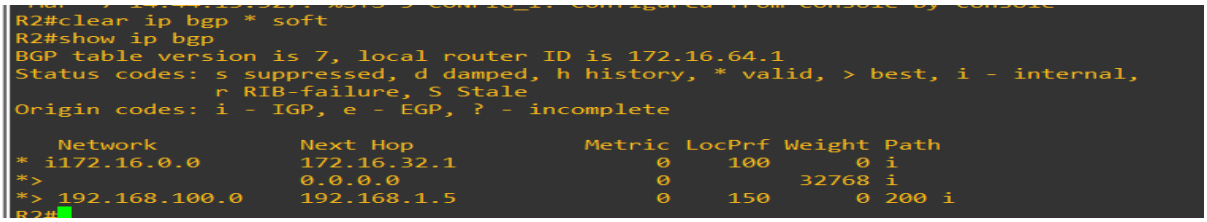
```
SanJose1# clear ip bgp * soft
```

SanJose2(R3)

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

SanJose1(R2)

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

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



SanJose2(R3)

```
SanJose2# show ip bgp
BGP table version is 7, local router ID is 172.16.32.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
```

```
RPKI validation codes: V valid, I invalid, N Not found

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

```
R3#
R3#show ip bgp
BGP table version is 5, local router ID is 172.16.32.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 172.16.0.0       0.0.0.0                  0         32768 i
* i                 172.16.64.1              0    100      0 i
*  192.168.100.0    192.168.1.1              0    125      0 200 i
*>i                 172.16.64.1              0    150      0 200 i
R3#
```

Step 9: Set BGP MED

ISP(R1)

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

    Network          Next Hop            Metric LocPrf Weight Path
 *   172.16.0.0       192.168.1.6              0             0 64512 i
 *>                   192.168.1.2              0             0 64512 i
 *>  192.168.100.0    0.0.0.0                  0         32768 i
ISP# show ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
```

D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area

N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2

E1 - OSPF external type 1, E2 - OSPF external type 2

i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2

ia - IS-IS inter area, * - candidate default, U - per-user static route

o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP

a - application route

+ - replicated route, % - next hop override

Gateway of last resort is not set

B 172.16.0.0/16 [20/0] via 192.168.1.2, 00:12:45

192.168.1.0/24 is variably subnetted, 4 subnets, 2 masks

C 192.168.1.0/30 is directly connected, Serial0/0/1

L 192.168.1.1/32 is directly connected, Serial0/0/1

C 192.168.1.4/30 is directly connected, Serial0/0/0

L 192.168.1.5/32 is directly connected, Serial0/0/0

192.168.100.0/24 is variably subnetted, 2 subnets, 2 masks

C 192.168.100.0/24 is directly connected, Loopback0

L 192.168.100.1/32 is directly connected, Loopback0

ISP#

```
R1#show ip bgp
BGP table version is 12, local router ID is 192.168.1.5
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       192.168.1.2              0             0 64512 i
*>                  192.168.1.6              0             0 64512 i
*> 192.168.100.0    0.0.0.0                  0         32768 i
R1#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

B    172.16.0.0/16 [20/0] via 192.168.1.6, 00:07:11
     192.168.1.0/30 is subnetted, 2 subnets
C       192.168.1.0 is directly connected, Serial1/1
C       192.168.1.4 is directly connected, Serial1/0
C    192.168.100.0/24 is directly connected, Loopback0
R1#
```

SanJose2(R3)

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

(0.0.0.0)

(0.0.0.0)

Reply to request 0 (20 ms). Received packet has options

Total option bytes= 40, padded length=40

Record route:

(172.16.1.2)

(192.168.1.6)

(192.168.100.1)

(192.168.1.1)

(172.16.32.1) <*>

(0.0.0.0)

(0.0.0.0)

(0.0.0.0)

(0.0.0.0)

End of list

Reply to request 1 (20 ms). Received packet has options

Total option bytes= 40, padded length=40

Record route:

(172.16.1.2)

(192.168.1.6)

(192.168.100.1)

(192.168.1.1)

(172.16.32.1) <*>

(0.0.0.0)

(0.0.0.0)

(0.0.0.0)

(0.0.0.0)

```
End of list

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

Reply to request 3 (24 ms).  Received packet has options
 Total option bytes= 40, padded length=40
 Record route:
   (172.16.1.2)
   (192.168.1.6)
   (192.168.100.1)
   (192.168.1.1)
   (172.16.32.1) <*>
   (0.0.0.0)
   (0.0.0.0)
   (0.0.0.0)
   (0.0.0.0)
 End of list
```

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

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

SanJose1(R2)

```
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(R3)

```
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
```

SanJose1(R2)

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

SanJose2(R3)

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

SanJose1(R2)

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

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

SanJose2(R3)

```
SanJose2# show ip bgp
BGP table version is 8, local router ID is 172.16.32.1
```

```
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

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

Step 10: Establish a default route

ISP(R1)

```
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)#
```

SanJose1(R2)

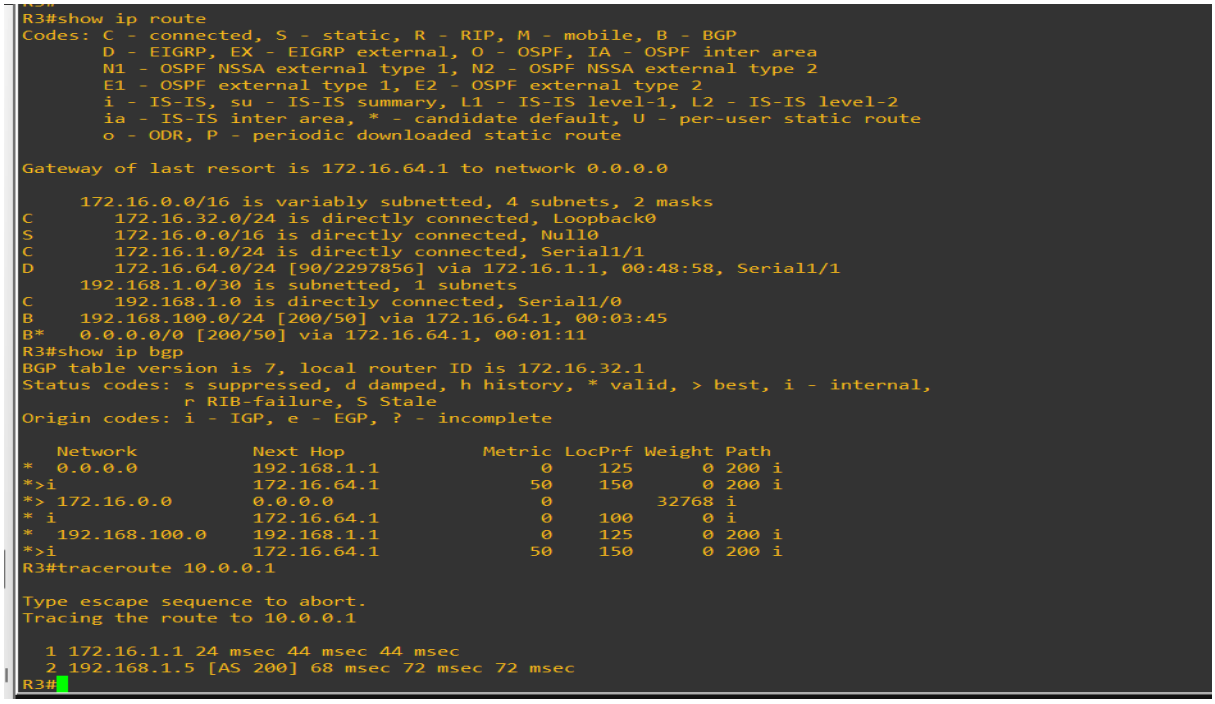
```
SanJose1# show ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
```

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

Gateway of last resort is 192.168.1.5 to network 0.0.0.0

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

SanJose2(R3)

```
SanJose2# show ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
      D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
      N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
      E1 - OSPF external type 1, E2 - OSPF external type 2
      i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
      ia - IS-IS inter area, * - candidate default, U - per-user static route
```

```
      o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
      a - application route
      + - replicated route, % - next hop override

Gateway of last resort is 172.16.64.1 to network 0.0.0.0

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



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

Gateway of last resort is 172.16.64.1 to network 0.0.0.0

     172.16.0.0/16 is variably subnetted, 4 subnets, 2 masks
C       172.16.32.0/24 is directly connected, Loopback0
S       172.16.0.0/16 is directly connected, Null0
C       172.16.1.0/24 is directly connected, Serial1/1
D       172.16.64.0/24 [90/2297856] via 172.16.1.1, 00:48:58, Serial1/1
     192.168.1.0/30 is subnetted, 1 subnets
C       192.168.1.0 is directly connected, Serial1/0
B    192.168.100.0/24 [200/50] via 172.16.64.1, 00:03:45
B*   0.0.0.0/0 [200/50] via 172.16.64.1, 00:01:11
R3#show ip bgp
BGP table version is 7, local router ID is 172.16.32.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*  0.0.0.0          192.168.1.1              0    125      0 200 i
*>i                 172.16.64.1             50    150      0 200 i
*> 172.16.0.0       0.0.0.0                  0         32768 i
* i                 172.16.64.1              0    100      0 i
*  192.168.100.0    192.168.1.1              0    125      0 200 i
*>i                 172.16.64.1             50    150      0 200 i
R3#traceroute 10.0.0.1

Type escape sequence to abort.
Tracing the route to 10.0.0.1

  1 172.16.1.1 24 msec 44 msec 44 msec
  2 192.168.1.5 [AS 200] 68 msec 72 msec 72 msec
R3#
```

SanJose2(R3)

SanJose2# show ip bgp

BGP table version is 38, local router ID is 172.16.32.1

Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,

r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,

x best-external, a additional-path, c RIB-compressed,

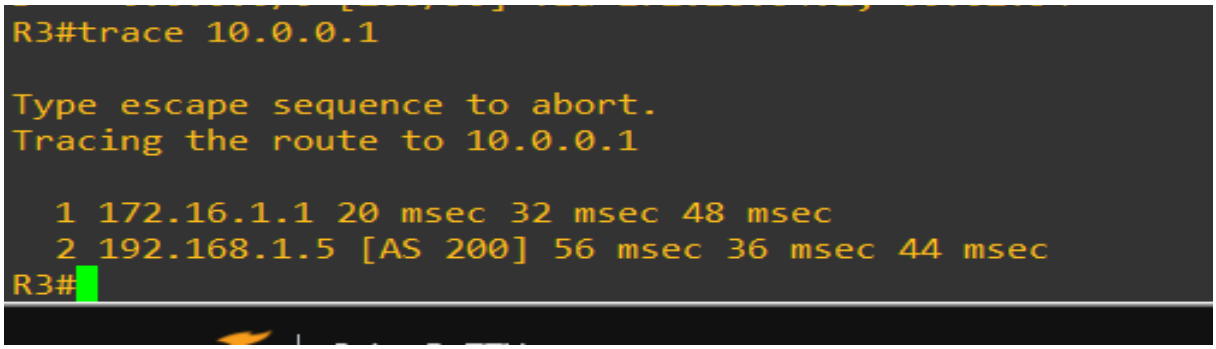
Origin codes: i - IGP, e - EGP, ? - incomplete

RPKI validation codes: V valid, I invalid, N Not found

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

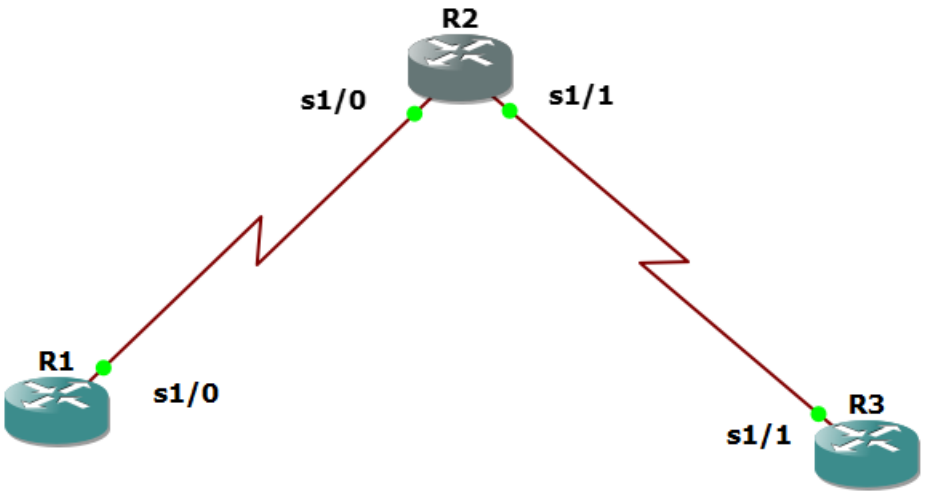
SanJose2#

```
SanJose2# traceroute 10.0.0.1

Type escape sequence to abort.

Tracing the route to 10.0.0.1

VRF info: (vrf in name/id, vrf out name/id)

  1 172.16.1.1 8 msec 4 msec 8 msec

  2 192.168.1.5 [AS 200] 12 msec * 12 msec

SanJose2#
```



Practical 4

Aim- Preform Secure Management Plane



Objectives:

Secure management access.

Configure enhanced username password security.

Enable AAA RADIUS authentication.

Enable secure remote management.

Background:

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

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

Required Resources:

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

Step 1: Configure Interfaces Addresses

R1

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

R2

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

```
!
end
```

R3

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

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

Step 2: Configure Static Routes

R1

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

R2

```
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
```

R3

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

R1

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

R1# tclsh

R1(tcl)#foreach address {

+>(tcl)#192.168.1.1

+>(tcl)#10.1.1.1

+>(tcl)#10.1.1.2

+>(tcl)#10.2.2.1

+>(tcl)#10.2.2.2

+>(tcl)#192.168.3.1

+>(tcl)#} { ping $address }
```

```
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 64/84/104 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 = 24/41/48 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 = 16/33/44 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 = 32/72/112 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 = 56/74/108 ms
R1(tcl)#foreach address {
+>(tcl)#192.168.1.1
+>(tcl)#10.1.1.1
+>(tcl)#10.1.1.2
+>(tcl)#10.2.2.1
+>(tcl)#10.2.2.2
+>(tcl)#192.168.3.1
+>(tcl)#} { ping $address }
```

Step 3: Secure Management Access

R1

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

R1(config)# enable  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)# secret class12345

R1(config)# line vty 0 4

R1(config-line)# password ciscovtypass

R1(config-line)# exec-timeout 5 0

R1(config-line)# login

R1(config-line)# exit

R1(config)#

R1(config)# line aux 0

R1(config-line)# no exec

R1(config-line)# end

R1#

R1(config)# service password-encryption

R1(config)#

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

R1(config)# exit
```

R3

```
R3(config)# security passwords min-length 10

R3(config)# enable  R1(config)# line console 0

R3(config-line)# password ciscoconpass

R3(config-line)# exec-timeout 5 0

R3(config-line)# login

R3(config-line)# logging synchronous
```

```
R3(config-line)# exit

R3(config)# secret class12345

R3(config)# line vty 0 4

R3(config-line)# password ciscovtypass

R3(config-line)# exec-timeout 5 0

R3(config-line)# login

R3(config-line)# exit

R3(config)#

R3(config)# line aux 0

R3(config-line)# no exec

R3(config-line)# end

R3#

R3(config)# service password-encryption

R3(config)#

R3(config)# banner motd $Unauthorized access strictly prohibited!$

R3(config)# exit
```

Step 4: Configure enhanced username password security

R1

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

R1(config)# username ADMIN secret class54321

R1(config)# line console 0

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

```
R1(config-line)# exit

R1(config)#

R1(config)# line vty 0 4

R1(config-line)# login local

R1(config-line)# end

R1(config)#
```

R3

```
R3(config)# username JR-ADMIN secret class12345

R3(config)# username ADMIN secret class54321

R3(config)# line console 0

R3(config-line)# login local

R3(config-line)# exit

R3(config)#

R3(config)# line vty 0 4

R3(config-line)# login local

R3(config-line)# end

R3(config)#
```

R1

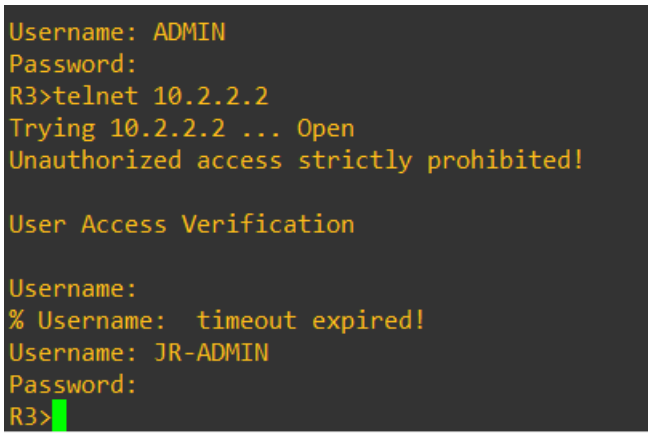
```
R1# telnet 10.2.2.2

Trying 10.2.2.2 ... Open

Unauthorized access strictly prohibited!
```

User Access Verification

Username: ADMIN

Password:

R3>



Practical 5:

Aim: Configure and Verify Path Control Using PBR

To configure and verify path control using Policy-Based Routing (PBR) on a router using GNS3.

Objectives:

• Implement Policy-Based Routing (PBR) to influence path selection for specific traffic.

• Use route maps to set conditions for forwarding packets based on source or other criteria.

• Verify PBR functionality using ping and traceroute commands.

Required Resources:

• GNS3 version 6.x

• Cisco Routers (IOS 15.x or equivalent)

• Ethernet cables for connectivity

• Two or more networks to demonstrate path control

Network Topology:

The network consists of four routers (R1, R2, R3, R4) connected in a triangle with R3 acting as the central node. The network uses the following subnets:

• 172.16.12.0/29 between R1 and R2

• 172.16.23.0/29 between R1 and R3

• 172.16.13.0/29 between R2 and R3

• 172.16.34.0/29 between R3 and R4

Step 1: Configure Interface Addresses

```
R1(config)# interface Serial1/0
R1(config-if)# ip address 172.16.12.2 255.255.255.248
R1(config-if)# no shutdown

R1(config)# interface Serial1/1
R1(config-if)# ip address 172.16.23.2 255.255.255.248
R1(config-if)# no shutdown

R2(config)# interface Serial1/0
R2(config-if)# ip address 172.16.12.1 255.255.255.248
```

```
R2(config-if)# no shutdown

R2(config)# interface Serial1/2
R2(config-if)# ip address 172.16.13.1 255.255.255.248
R2(config-if)# no shutdown

R3(config)# interface Serial1/1
R3(config-if)# ip address 172.16.23.3 255.255.255.248
R3(config-if)# no shutdown

R3(config)# interface Serial1/2
R3(config-if)# ip address 172.16.13.3 255.255.255.248
R3(config-if)# no shutdown

R3(config)# interface Serial1/0
R3(config-if)# ip address 172.16.34.3 255.255.255.248
R3(config-if)# no shutdown

R4(config)# interface Serial1/0
R4(config-if)# ip address 172.16.34.4 255.255.255.248
R4(config-if)# no shutdown
```

Step 2: Configure PBR using Route Maps

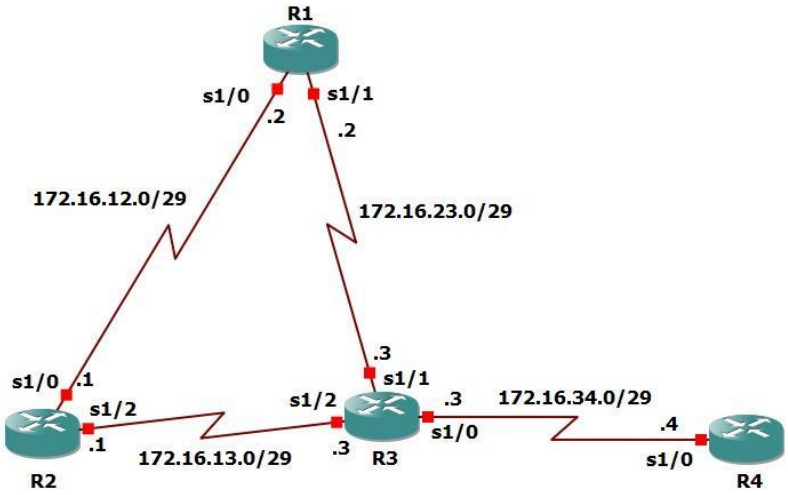
```
R2(config)# access-list 100 permit ip 172.16.12.0 0.0.0.7 any

R2(config)# route-map PBR_POLICY permit 10
R2(config-route-map)# match ip address 100
R2(config-route-map)# set ip next-hop 172.16.13.3

R2(config)# interface Serial1/0
R2(config-if)# ip policy route-map PBR_POLICY
```

Step 3: Verify PBR Configuration

```
R2# show ip policy
R2# show route-map
R2# show ip route

PC1# traceroute 8.8.8.8
```

Screenshots:

The following screenshots provide a step-by-step visual guide to implementing PBR:




```
R1(config)#interface Lo1
R1(config-if)#interface Lo1
*Mar 17 22:17:00.095: %LINEPROTO-5-UPDOWN: Line protocol on Interface Loopback1,
 changed state to up
R1(config-if)#ip address 192.168.1.1 255.255.255.0
R1(config-if)#interface s1/0
R1(config-if)#ip address 172.16.12.1 255.255.255.248
R1(config-if)#clock rate 128000
R1(config-if)#bandwidth 128
R1(config-if)#no shutdown
R1(config-if)#
*Mar 17 22:19:02.151: %LINK-3-UPDOWN: Interface Serial1/0, changed state to up
R1(config-if)#
*Mar 17 22:19:02.151: %ENTITY_ALARM-6-INFO: CLEAR INFO Se1/0 Physical Port Admin
istrative State Down
R1(config-if)#
*Mar 17 22:19:03.155: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial1/0,
 changed state to up
R1(config-if)#interface s1/2
R1(config-if)#ip address 172.16.13.1 255.255.255.248
R1(config-if)#bandwidth 64
R1(config-if)#no shutdown
R1(config-if)#end
*Mar 17 22:19:29.735: %LINK-3-UPDOWN: Interface Serial1/2, changed state to up
*Mar 17 22:19:29.735: %ENTITY_ALARM-6-INFO: CLEAR INFO Se1/2 Physical Port Admin
istrative State Down
R1(config-if)#end
R1#
*Mar 17 22:19:30.207: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial1/0,
 changed state to down
*Mar 17 22:19:30.531: %SYS-5-CONFIG_I: Configured from console by console
*Mar 17 22:19:30.739: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial1/2,
 changed state to up
R1#
```

```
R2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#interface Lo2
R2(config-if)#
*Mar 17 22:22:51.599: %LINEPROTO-5-UPDOWN: Line protocol on Interface Loopback2,
 changed state to up
R2(config-if)#ip address 192.168.2.1 255.255.255.0
R2(config-if)#interface s1/0
R2(config-if)#
R2(config-if)#
R2(config-if)#
R2(config-if)#ip address 172.16.12.2 255.255.255.248
R2(config-if)#bandwidth 128
R2(config-if)#no shutdown
R2(config-if)#
*Mar 17 22:24:13.275: %LINK-3-UPDOWN: Interface Serial1/0, changed state to up
R2(config-if)#
*Mar 17 22:24:13.275: %ENTITY_ALARM-6-INFO: CLEAR INFO Se1/0 Physical Port Admin
istrative State Down
R2(config-if)#
*Mar 17 22:24:14.279: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial1/0,
 changed state to up
R2(config-if)#interface s1/1
R2(config-if)#ip address 172.16.23.2 255.255.255.248
R2(config-if)#clock rate 128000
R2(config-if)#bandwidth 128
R2(config-if)#no shutdown
R2(config-if)#en
*Mar 17 22:24:43.367: %LINK-3-UPDOWN: Interface Serial1/1, changed state to up
R2(config-if)#end
R2#
*Mar 17 22:24:43.367: %ENTITY_ALARM-6-INFO: CLEAR INFO Se1/1 Physical Port Admin
istrative State Down
*Mar 17 22:24:44.367: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial1/1,
 changed state to up
R2#
*Mar 17 22:24:44.971: %SYS-5-CONFIG_I: Configured from console by console
R2#
```

```
R3#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#interface Lo3
R3(config-if)#
*Mar 17 22:27:13.511: %LINEPROTO-5-UPDOWN: Line protocol on Interface Loopback3,
 changed state to up
R3(config-if)#ip address 192.168.3.1 255.255.255.0
R3(config-if)#interface s1/2
R3(config-if)#ip address 172.16.13.3 255.255.255.248
R3(config-if)#clock rate 64000
R3(config-if)#bandwidth 64
R3(config-if)#no shutdown
R3(config-if)#
*Mar 17 22:27:49.119: %LINK-3-UPDOWN: Interface Serial1/2, changed state to up
R3(config-if)#
*Mar 17 22:27:49.119: %ENTITY_ALARM-6-INFO: CLEAR INFO Se1/2 Physical Port Admin
istrative State Down
R3(config-if)#i
*Mar 17 22:27:50.123: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial1/2,
 changed state to up
```

```
R3(config-if)#interface s1/1
R3(config-if)#
*Mar 17 22:28:40.219: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial1/0,
 changed state to down
R3(config-if)#ip address 172.16.23.3 255.255.255.248
R3(config-if)#bandwidth 128
R3(config-if)#no shutdown
R3(config-if)#
*Mar 17 22:28:45.887: %LINK-3-UPDOWN: Interface Serial1/1, changed state to up
R3(config-if)#
*Mar 17 22:28:45.887: %ENTITY_ALARM-6-INFO: CLEAR INFO Se1/1 Physical Port Admin
istrative State Down
R3(config-if)#
*Mar 17 22:28:46.891: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial1/1,
 changed state to up
R3(config-if)#s1/0
             ^
% Invalid input detected at '^' marker.

R3(config-if)#interface s1/0
R3(config-if)#ip address 172.16.34.3 255.255.255.248
R3(config-if)#clock rate 64000
R3(config-if)#bandwidth 64
R3(config-if)#no shutdown
R3(config-if)#end
R3#
*Mar 17 22:29:17.779: %SYS-5-CONFIG_I: Configured from console by console
R3#
```

```
R4#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R4(config)#interface Lo4
R4(config-if)#
*Mar 17 22:31:22.583: %LINEPROTO-5-UPDOWN: Line protocol on Interface Loopback4,
 changed state to up
R4(config-if)#ip address 192.168.4.1 255.255.255.128
R4(config-if)#interface Lo5
R4(config-if)#
*Mar 17 22:31:40.975: %LINEPROTO-5-UPDOWN: Line protocol on Interface Loopback5,
 changed state to up
R4(config-if)#ip address 192.168.4.129 255.255.255.128
R4(config-if)#interface s1/0
R4(config-if)#ip address 172.16.34.4 255.255.255.248
R4(config-if)#bandwidth 64
R4(config-if)#no shutdown
R4(config-if)#end
R4#
*Mar 17 22:32:12.743: %LINK-3-UPDOWN: Interface Serial1/0, changed state to up
*Mar 17 22:32:12.743: %ENTITY_ALARM-6-INFO: CLEAR INFO Se1/0 Physical Port Admin
istrative State Down
R4#
*Mar 17 22:32:13.183: %SYS-5-CONFIG_I: Configured from console by console
*Mar 17 22:32:13.747: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial1/0,
 changed state to up
R4#
```

```
R3#show ip interface brief | include up
Serial1/0                  172.16.34.3     YES manual up                    up

Serial1/1                  172.16.23.3     YES manual up                    up

Serial1/2                  172.16.13.3     YES manual up                    up

Loopback3                  192.168.3.1     YES manual up                    up

R3#show protocols
Global values:
  Internet Protocol routing is enabled
FastEthernet0/0 is administratively down, line protocol is down
Serial1/0 is up, line protocol is up
  Internet address is 172.16.34.3/29
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 172.16.13.3/29
Serial1/3 is administratively down, line protocol is down
Serial1/4 is administratively down, line protocol is down
Serial1/5 is administratively down, line protocol is down
Serial1/6 is administratively down, line protocol is down
Serial1/7 is administratively down, line protocol is down
Loopback3 is up, line protocol is up
  Internet address is 192.168.3.1/24
R3#
R3#
R3#show interfaces description | include up
Se1/0                          up             up
Se1/1                          up             up
Se1/2                          up             up
Lo3                            up             up
R3#
```

```
R1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#router eigrp 1
R1(config-router)#network 192.168.1.0
R1(config-router)#network 172.16.12.0 0.0.0.7
R1(config-router)#network 172.16.13.0 0.0.0.7
R1(config-router)#no auto-summary
```

```
R2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#router eigrp 1
R2(config-router)#network 192.168.2.0
R2(config-router)#network 172.16.12.0 0.0.0.7
R2(config-router)#network 172.16.23.0 0.0.0.7
R2(config-router)#no auto-summary
R2(config-router)#
*Mar 17 22:38:57.019: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 1: Neighbor 172.16.12.1 (
rial1/0) is up: new adjacency
```

```
R3#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#router eigrp 1
R3(config-router)#network 192.168.3.0
R3(config-router)#network 172.16.13.0 0.0.0.7
R3(config-router)#network 172.16.23.0 0.0.0.7
R3(config-router)#network 172.16.34.0 0.0.0.7
R3(config-router)#no auto-summary
R3(config-router)#
*Mar 17 22:39:41.859: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 1: Neighbor 172.16.23.2 (Se
rial1/1) is up: new adjacency
*Mar 17 22:39:41.863: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 1: Neighbor 172.16.13.1 (Se
rial1/2) is up: new adjacency
*Mar 17 22:39:42.711: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 1: Neighbor 172.16.13.1 (Se
rial1/2) is resync: summary configured
*Mar 17 22:39:42.711: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 1: Neighbor 172.16.23.2 (Se
rial1/1) is resync: summary configured
R3(config-router)#
```

```
R4#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R4(config)#router eigrp 1
R4(config-router)#network 192.168.4.0
R4(config-router)#network 172.16.34.0 0.0.0.7
R4(config-router)#no auto-summary
R4(config-router)#
*Mar 17 22:40:39.727: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 1: Neighbor 172.16.34.3 (Se
rial1/0) is up: new adjacency
*Mar 17 22:40:40.615: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 1: Neighbor 172.16.34.3 (Se
rial1/0) is resync: summary configured
R4(config-router)#
```

```
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            14 00:01:29   42  2280  0  28
R4#
```

```
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             Se1/0            11 00:02:16   32  2280  0  7
1   172.16.13.1             Se1/2            11 00:03:13   35  2280  0  22
0   172.16.23.2             Se1/1            10 00:03:13   35  1140  0  21
R3#
```

```
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            12 00:04:00   38  1140  0  33
0   172.16.12.1             Se1/0            11 00:04:45   41  1140  0  23
R2#
```

```
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             Se1/2            13 00:04:45   32  2280  0  34
0   172.16.12.2             Se1/0            14 00:05:30   33  1140  0  26
R1#
```

```
R1#show ip route | begin Gateway
Gateway of last resort is not set

     172.16.0.0/29 is subnetted, 4 subnets
D       172.16.34.0 [90/41024000] via 172.16.13.3, 00:05:38, Serial1/2
D       172.16.23.0 [90/21024000] via 172.16.12.2, 00:05:40, Serial1/0
C       172.16.12.0 is directly connected, Serial1/0
C       172.16.13.0 is directly connected, Serial1/2
     192.168.4.0/25 is subnetted, 2 subnets
D       192.168.4.0 [90/41152000] via 172.16.13.3, 00:04:42, Serial1/2
D       192.168.4.128 [90/41152000] via 172.16.13.3, 00:04:42, Serial1/2
C    192.168.1.0/24 is directly connected, Loopback1
D    192.168.2.0/24 [90/20640000] via 172.16.12.2, 00:05:38, Serial1/0
D    192.168.3.0/24 [90/21152000] via 172.16.12.2, 00:05:38, Serial1/0
R1#
```

```
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            14 00:01:29   42  2280  0  28
R4#traceroute 192.168.1.1 source 192.168.4.1

Type escape sequence to abort.
Tracing the route to 192.168.1.1

  1 172.16.34.3 48 msec 32 msec 44 msec
  2 172.16.23.2 96 msec 96 msec 108 msec
  3 172.16.12.1 108 msec 92 msec 108 msec
R4#
```

```
R4# traceroute 192.168.1.1 source 192.168.4.129

Type escape sequence to abort.
Tracing the route to 192.168.1.1

  1 172.16.34.3 52 msec 40 msec 56 msec
  2 172.16.23.2 92 msec 76 msec 84 msec
  3 172.16.12.1 104 msec 136 msec 128 msec
```

```
R3#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
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)#
```

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

```
R3(config)#interface s1/0
R3(config-if)#ip policy route-map R3-to-R1
R3(config-if)#end
```

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

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

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

```
  3 172.16.12.1 108 msec 92 msec 108 msec
R4#
R4# traceroute 192.168.1.1 source 192.168.4.129

Type escape sequence to abort.
Tracing the route to 192.168.1.1

  1 172.16.34.3 52 msec 40 msec 56 msec
  2 172.16.23.2 92 msec 76 msec 84 msec
  3 172.16.12.1 104 msec 136 msec 128 msec
R4#traceroute 192.168.1.1 source 192.168.4.1

Type escape sequence to abort.
Tracing the route to 192.168.1.1

  1 172.16.34.3 52 msec 44 msec 32 msec
  2 172.16.23.2 108 msec 72 msec 84 msec
  3 172.16.12.1 120 msec 92 msec 128 msec
R4#traceroute 192.168.1.1 source 192.168.4.129

Type escape sequence to abort.
Tracing the route to 192.168.1.1

  1 172.16.34.3 40 msec 24 msec 48 msec
  2 172.16.13.1 88 msec 52 msec 60 msec
R4#
```

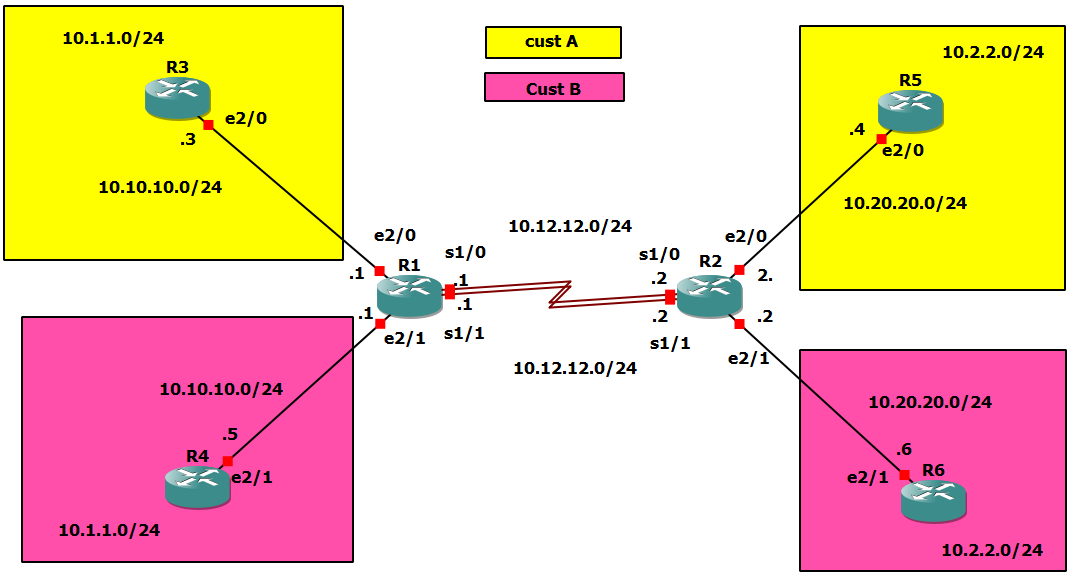
```
R3#show route-map
route-map R3-to-R1, permit, sequence 10
  Match clauses:
    ip address (access-lists): PBR-ACL
  Set clauses:
    ip next-hop 172.16.13.1
  Policy routing matches: 3 packets, 96 bytes
R3#
```

Conclusion:

Policy-Based Routing (PBR) successfully directs traffic based on defined rules, influencing path control in a network.

Practical 6

Aim- Preform Virtual Routing and Forwarding (VRF)



Objectives:

Connect 2 companies in the same network connected to the same network but separated using VRF.

Background:

Here we have 2 companies Cust-A and Cust-B. Cust-A is on routers R3 and R5 and Cust-B R4 and R6. R1 is connected to R3 and R4 via ethernet cable whereas R2 is connected to R5 and R6 via ethernet. R1 is connected to R2 via 2 serial ports. Use VRF to connect both companies to themselves without interrupting the connection to the other companies.

Required Resources:

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

Step 1: Configure Interfaces Addresses

Confiure R1

```
R1#conf t
R1(config)#ip vrf cust-A
R1(config- vrf)#exit
R1(config)#ip vrf cust-B
```

```
R1(config- vrf)# exit
R1(config)#int e2/0
R1(config-if)#ip vrf forwarding cust-A
R1(config-if)#ip address 10.10.10.1 255.255.255.0
R1(config-if)#no shut
R1(config-if)#exit
R1(config)#int e2/1
R1(config-if)#ip vrf forwarding cust-B
R1(config-if)#ip address 10.10.10.1 255.255.255.0
R1(config-if)#no shut
R1(config-if)#exit
R1(config)#int s1/0
R1(config-if)#ip vrf forwarding cust-A
R1(config-if)#ip address 10.12.12.1 255.255.255.0
R1(config-if)#no shut
R1(config-if)#exit
R1(config)#int s1/1
R1(config-if)#ip vrf forwarding cust-B
R1(config-if)#ip address 10.12.12.1 255.255.255.0
R1(config-if)#no shut
R1(config-if)#exit
```

Configure R2

```
R2#conf t
R2(config)#ip vrf cust-A
R2(config- vrf)#exit
R2(config)#ip vrf cust-B
R2(config- vrf)#exit
R2(config)#int e2/0
```

```
R2(config-if)#ip vrf forwarding cust-A
R2(config-if)#ip address 10.20.20.2 255.255.255.0
R2(config-if)#no shut
R2(config-if)#exit
R2(config)#int e2/1
R2(config-if)#ip vrf forwarding cust-B
R2(config-if)#ip address 10.20.20.2 255.255.255.0
R2(config-if)#no shut
R2(config-if)#exit
R2(config)#int s1/0
R2(config-if)#ip vrf forwarding cust-A
R2(config-if)#ip address 10.12.12.2 255.255.255.0
R2(config-if)#no shut
R2(config-if)#exit
R2(config)#int s1/1
R2(config-if)#ip vrf forwarding cust-B
R2(config-if)#ip address 10.12.12.2 255.255.255.0
R2(config-if)#no shut
R2(config-if)#exit
```

Configure R3

```
R3#conf t
R3(config)#int e2/0
R3(config-if)#ip address 10.10.10.3 255.255.255.0
R3(config-if)# no shut
R3(config-if)#exit
R3(config)#int lo0
R3(config-if)#ip address 10.1.1.3 255.255.255.0
R3(config-if)# no shut
```

```
R3(config-if)#exit
```

Configure R4

```
R4#conf t
R4(config)#int e2/1
R4(config-if)#ip address 10.10.10.5 255.255.255.0
R4(config-if)# no shut
R4(config-if)#exit
R4(config)#int lo0
R4(config-if)#ip address 10.1.1.5 255.255.255.0
R4(config-if)# no shut
R4(config-if)#exit
```

Configure R5

```
R5#conf t
R5(config)#int e2/0
R5config-if)#ip address 10.20.20.4 255.255.255.0
R5(config-if)# no shut
R5(config-if)#exit
R5(config)#int lo0
R5(config-if)#ip address 10.2.2.4 255.255.255.0
R5(config-if)# no shut
R5(config-if)#exit
```

Configure R6

```
R6#conf t
R6(config)#int e2/1
R6(config-if)#ip address 10.20.20.6 255.255.255.0
R6(config-if)# no shut
R6(config-if)#exit
```

```
R6(config)#int lo0
R6(config-if)#ip address 10.2.2.6 255.255.255.0
R6(config-if)# no shut
R6(config-if)#exit
```

Step 2: set routing

On R3

```
R3(config)#Router eigrp 100
R3(config-router)#no auto
R3(config-router)#net 10.0.0.0
```

On R4

```
R4(config)#Router eigrp 100
R4(config-router)#no auto
R4(config-router)#net 10.0.0.0
```

On R5

```
R5(config)#Router eigrp 100
R5(config-router)#no auto
R5(config-router)#net 10.0.0.0
```

On R6

```
R6(config)#Router eigrp 100
R6(config-router)#no auto
R6(config-router)#net 10.0.0.0
```

On R1

```
R1(config)#router eigrp 1
R1(config-router)#!
```

```
R1(config-router)# address-family ipv4 vrf cust-A
R1(config-router-af)# autonomous-system 100
R1(config-router-af)# no auto
R1(config-router-af)#network 10.0.0.0
R1(config-router-af)#exit
R1(config)#router eigrp 1
R1(config-router)#!
R1(config-router)#address-family ipv4 vrf cust-B
R1(config-router-af)# autonomous-system 100
R1(config-router-af)# no auto
R1(config-router-af)#network 10.0.0.0
R1(config-router-af)#exit
```

On R2

```
R2(config)#router eigrp 1
R2(config-router)#!
R2(config-router)#address-family ipv4 vrf cust-A
R2(config-router-af)# autonomous-system 100
R2(config-router-af)# no auto
R2(config-router-af)#network 10.0.0.0
R2(config-router-af)#exit
R2(config)#router eigrp 1
R2(config-router)#!
R2(config-router)#address-family ipv4 vrf cust-B
R2(config-router-af)# autonomous-system 100
R2(config-router-af)# no auto
R2(config-router-af)#network 10.0.0.0
R2(config-router-af)#exit
```

Step 3: check routing and ping

On R1

```
R1#show ip route
R1#show ip route vrf cust-A
R1#show ip int br
R1#ping vrf cust-A 10.1.1.3
R1#show ip route
R1#show ip route vrf cust-B
R1#show ip int br
R1#ping vrf cust-B 10.1.1.5
```

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

Gateway of last resort is not set

R1#show ip route vrf cust-A

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

Gateway of last resort is not set

     10.0.0.0/24 is subnetted, 5 subnets
D       10.20.20.0 [90/2195456] via 10.12.12.2, 00:18:14, Serial1/0
C       10.12.12.0 is directly connected, Serial1/0
C       10.10.10.0 is directly connected, Ethernet2/0
D       10.2.2.0 [90/2323456] via 10.12.12.2, 00:18:12, Serial1/0
O       10.1.1.0 [90/409600] via 10.10.10.3, 00:22:32, Ethernet2/0
R1#do show ip int br
   ^
% Invalid input detected at '^' marker.

R1#show ip int br
Interface                  IP-Address      OK? Method Status                Protocol
FastEthernet0/0            unassigned      YES unset  administratively down down
Serial1/0                  10.12.12.1      YES manual up                    up
Serial1/1                  10.12.12.1      YES manual up                    up
Serial1/2                  unassigned      YES unset  administratively down down
Serial1/3                  unassigned      YES unset  administratively down down
Ethernet2/0                10.10.10.1      YES manual up                    up
Ethernet2/1                10.10.10.1      YES manual up                    up
Ethernet2/2                unassigned      YES unset  administratively down down
```

On R3

```
R3#ping 10.2.2.4
```

```
R3#ping 10.2.2.4

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.2.2.4, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 240/331/444 ms
R3#
```

On R2

R2#show ip route

```
R2#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/24 is subnetted, 1 subnets
C       10.12.12.0 is directly connected, Serial1/1
```

Conclusion

Using VRF we can connection multiple different companies to same network path through router but still differentiate their paths allowing multiple companies and connections to be situated in same router

Practical 7

Aim: MPLS

To configure and verify Inter-VLAN Routing using a Layer 3 device in GNS3.

Objectives:

• Configure VLANs on a Layer 2 switch.

• Implement Inter-VLAN Routing using a Layer 3 device.

• Verify connectivity between different VLANs.

Required Resources:

• GNS3 version 6.x

• Cisco Layer 3 Switch (IOS 15.x or equivalent)

• Cisco Layer 2 Switch

• PCs for testing VLAN communication

Network Topology:

The network consists of a Layer 3 switch and a Layer 2 switch with multiple VLANs configured. The VLANs use different subnets and require routing through the Layer 3 switch.

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 are 0

int f0/0

ip add 10.0.0.2 255.255.255.0

no shut

ip ospf 1 area 0

int f0/1

ip add 10.0.1.2 255.255.255.0

no shut

ip ospf 1 area 0

R3

hostname R3

int lo0

ip add 3.3.3.3 255.255.255.255

ip ospf 1 are 0

int f0/0

ip add 10.0.1.3 255.255.255.0

no shut

ip ospf 1 area 0

You should now have full ip connectivity between R1, R2, R3 to verify 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#

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#

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.

So to review we have now configured IP addresses on the MPLS core, enabled OSPF and full IP connectivity between all routers and finally enabled mpls on all the interfaces in the core and have established ldp neighbors between all routers.

The next step is to configure MP-BGP between R1 and R3

This is when you start to see the layer 3 vpn configuration come to life

Step 3 – MPLS BGP Configuration between R1 and R3

We need to establish a Multi Protocol BGP session between R1 and R3 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
```

You should see log messages showing the BGP sessions coming up.

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

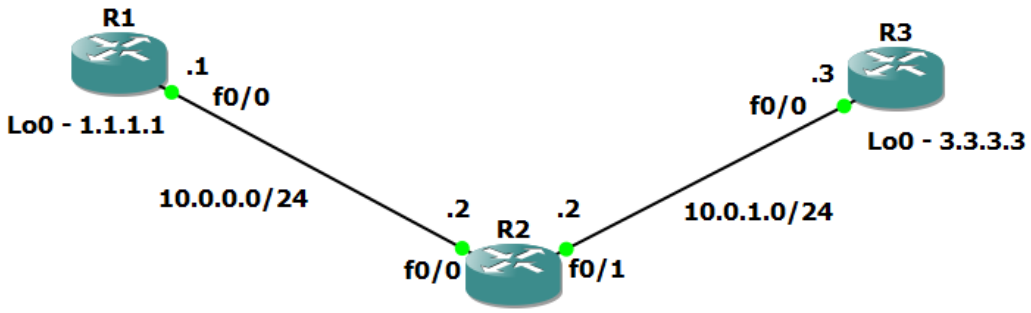
BGP router identifier 1.1.1.1, local AS number 1

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

You can see here that we do have a bgp vpnv4 peering to R3 – looking at the PfxRcd you can see it says 0 this is because we have not got any routes in BGP. We are now going to add two more routers to the topology. These will be the customer sites connected to R1 and R3. We will then create a VRF on each router and put the interfaces connected to each site router into that VRF.

Screenshots:




```
R3#
R3#ping 1.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 1.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 16/28/44 ms
R3#ping 2.2.2.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2.2.2.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 16/20/32 ms
R3#
R3#
R3#
```

```
R2(config-router)#
*Mar  1 00:05:44.815: %LDP-5-NBRCHG: LDP Neighbor 1.1.1.1:0 (1) is UP
R2(config-router)#
*Mar  1 00:06:00.463: %LDP-5-NBRCHG: LDP Neighbor 3.3.3.3:0 (2) is UP
R2(config-router)#end
R2#sh mpls inter
*Mar  1 00:06:33.003: %SYS-5-CONFIG_I: Configured from console by console
R2#sh mpls interface
Interface              IP            Tunnel   Operational
FastEthernet0/0        Yes (ldp)     No       Yes
FastEthernet0/1        Yes (ldp)     No       Yes
R2#
R2#
R2#
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 - 2.2.2.2.56154
        State: Oper; Msgs sent/rcvd: 9/9; Downstream
        Up time: 00:01:05
        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: 3.3.3.3.45511 - 2.2.2.2.646
        State: Oper; Msgs sent/rcvd: 8/9; Downstream
        Up time: 00:00:50
        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
R2#
```

```
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2.2.2.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 16/20/32 ms
R3#
R3#
R3#ping 1.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 1.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 16/19/20 ms
R3#sh ip ospf neigh

Neighbor ID     Pri   State           Dead Time   Address         Interface
2.2.2.2           1   FULL/DR         00:00:33    10.0.1.2        FastEthernet0/0
R3#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#router ospf 1
R3(config-router)#mpls ldp autoconfig
R3(config-router)#
R3(config-router)#
R3(config-router)#
*Mar  1 00:05:40.451: %LDP-5-NBRCHG: LDP Neighbor 2.2.2.2:0 (1) is UP
R3(config-router)#end
R3#sh mpl
*Mar  1 00:06:01.039: %SYS-5-CONFIG_I: Configured from console by console
R3#sh mpls interface
Interface              IP            Tunnel   Operational
FastEthernet0/0        Yes (ldp)     No       Yes
R3#
```

```
R3(config-router)#
R3(config-router)#
*Mar  1 00:05:40.451: %LDP-5-NBRCHG: LDP Neighbor 2.2.2.2:0 (1) is UP
R3(config-router)#end
R3#sh mpl
*Mar  1 00:06:01.039: %SYS-5-CONFIG_I: Configured from console by console
R3#sh mpls interface
Interface              IP            Tunnel   Operational
FastEthernet0/0        Yes (ldp)     No       Yes
R3#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#router bgp 1
R3(config-router)# neighbor 1.1.1.1 remote-as 1
R3(config-router)# neighbor 1.1.1.1 update-source Loopback0
R3(config-router)# no auto-summary
R3(config-router)# !
R3(config-router)# address-family vpnv4
R3(config-router-af)#  neighbor 1.1.1.1 activate
R3(config-router-af)#end
R3#
R3#
R3#
*Mar  1 00:09:04.871: %SYS-5-CONFIG_I: Configured from console by console
R3#
*Mar  1 00:09:23.635: %BGP-5-ADJCHANGE: neighbor 1.1.1.1 Up
R3#sh bgp vpnv4 unicast all summary
BGP router identifier 3.3.3.3, 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
1.1.1.1         4     1       2       2        0    0    0 00:00:16        0
R3#
```

Practical 8

Aim: Inter-VLAN Routing

To configure and verify Inter-VLAN Routing using a Layer 3 device in GNS3.

Objectives:

• Configure VLANs on a Layer 2 switch.

• Implement Inter-VLAN Routing using a Layer 3 device.

• Verify connectivity between different VLANs.

Required Resources:

• GNS3 version 6.x

• Cisco Layer 3 Switch (IOS 15.x or equivalent)

• Cisco Layer 2 Switch

• PCs for testing VLAN communication

Network Topology:

The network consists of a Layer 3 switch and a Layer 2 switch with multiple VLANs configured. The VLANs use different subnets and require routing through the Layer 3 switch.

Step 1: Configure VLANs on the Switch

```
Switch(config)# vlan 10
Switch(config-vlan)# name SALES
Switch(config-vlan)# exit

Switch(config)# vlan 20
Switch(config-vlan)# name HR
Switch(config-vlan)# exit

Switch(config)# vlan 30
Switch(config-vlan)# name IT
Switch(config-vlan)# exit

Switch(config)# interface FastEthernet0/1
Switch(config-if)# switchport mode access
Switch(config-if)# switchport access vlan 10

Switch(config)# interface FastEthernet0/2
Switch(config-if)# switchport mode access
Switch(config-if)# switchport access vlan 20

Switch(config)# interface FastEthernet0/3
```

```
Switch(config-if)# switchport mode access
Switch(config-if)# switchport access vlan 30
```

Step 2: Configure Inter-VLAN Routing on the Layer 3 Switch

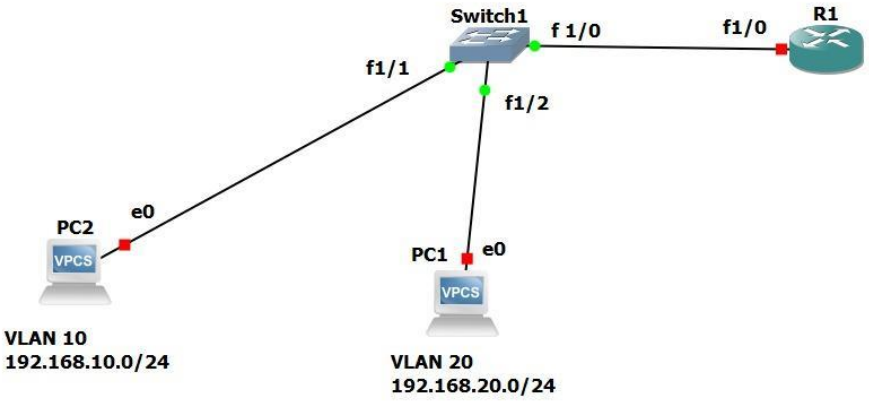
```
L3Switch(config)# interface vlan 10
L3Switch(config-if)# ip address 192.168.10.1 255.255.255.0
L3Switch(config-if)# no shutdown

L3Switch(config)# interface vlan 20
L3Switch(config-if)# ip address 192.168.20.1 255.255.255.0
L3Switch(config-if)# no shutdown

L3Switch(config)# interface vlan 30
L3Switch(config-if)# ip address 192.168.30.1 255.255.255.0
L3Switch(config-if)# no shutdown

L3Switch(config)# ip routing
```

Step 3: Verify Inter-VLAN Communication

```
PC1> ping 192.168.20.1
PC1> ping 192.168.30.1

PC2> ping 192.168.10.1
PC2> ping 192.168.30.1

PC3> ping 192.168.10.1
PC3> ping 192.168.20.1
```

Screenshots:

The following screenshots provide a step-by-step visual guide to implementing Inter-VLAN Routing:




```
PC1> sh

NAME   IP/MASK              GATEWAY           MAC                LPORT  RHOST:PORT
PC1    0.0.0.0/0            0.0.0.0           00:50:79:66:68:00  10014  127.0.0.1:10015
       fe80::250:79ff:fe66:6800/64

PC1> ip 192.168.10.10/24 192.168.10.1
Checking for duplicate address...
PC1 : 192.168.10.10 255.255.255.0 gateway 192.168.10.1

PC1> save
Saving startup configuration to startup.vpc
.  done

PC1> sh
```

```
PC2> sh

NAME   IP/MASK              GATEWAY           MAC                LPORT  RHOST:PORT
PC2    0.0.0.0/0            0.0.0.0           00:50:79:66:68:01  10016  127.0.0.1:10017
       fe80::250:79ff:fe66:6801/64

PC2> ip 192.168.20.10/24 192.168.20.1
Checking for duplicate address...
PC1 : 192.168.20.10 255.255.255.0 gateway 192.168.20.1

PC2> save
Saving startup configuration to startup.vpc
.  done

PC2> sh

NAME   IP/MASK              GATEWAY           MAC                LPORT  RHOST:PORT
PC2    192.168.20.10/24     192.168.20.1      00:50:79:66:68:01  10016  127.0.0.1:10017
       fe80::250:79ff:fe66:6801/64
```

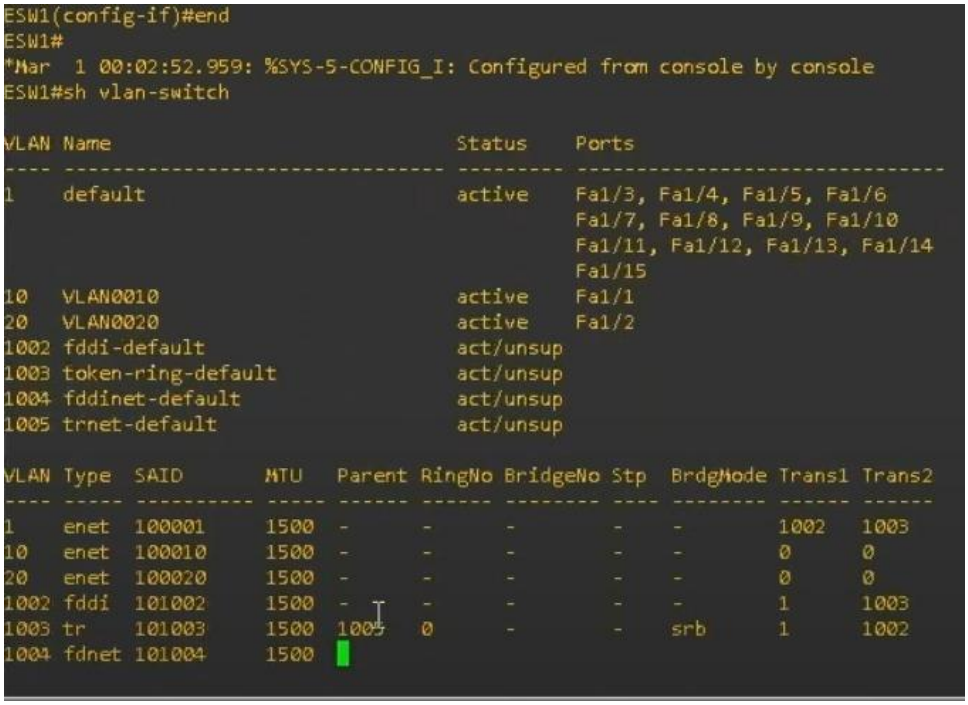
```
ESW1#config t
Enter configuration commands, one per line.  End with CNTL/Z.
ESW1(config)#int f1/0
ESW1(config-if)#switchport trunk encapsulation dot1q
ESW1(config-if)#switchport mode trunk
ESW1(config-if)#d
*Mar  1 00:01:15.427: %DTP-5-TRUNKPORTON: Port Fa1/0 has become dot1q trunk
ESW1(config-if)#do wr
Building configuration...
[OK]
ESW1(config-if)#int f1/1
ESW1(config-if)#switchport mode access
ESW1(config-if)#switchport access vlan 10
% Access VLAN does not exist. Creating vlan 10
ESW1(config-if)#do wr
Building configuration...
[OK]
ESW1(config-if)#int f1/2
ESW1(config-if)#switchport mode access
ESW1(config-if)#switchport access vlan 20
% Access VLAN does not exist. Creating vlan 20
ESW1(config-if)#do wr
Building configuration...
[OK]
ESW1(config-if)#exit
ESW1(config)#exit
ESW1#
*Mar  1 00:02:46.083: %SYS-5-CONFIG_I: Configured from console by console
ESW1#sh swi
```

```
R1(config)#int f1/0
R1(config-if)#no sh
R1(config-if)#int
*Mar  1 00:01:10.415: %LINK-3-UPDOWN: Interface FastEthernet1/0, changed state to up
*Mar  1 00:01:11.415: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet1/0, changed state to up
R1(config-if)#int f1/0.10
R1(config-subif)#encapsulation dot1Q 10
R1(config-subif)#ip add 192.168.10.1 255.255.255.0
R1(config-subif)#do wr
Building configuration...
[OK]
R1(config-subif)#end
R1#
*Mar  1 00:02:04.071: %SYS-5-CONFIG_I: Configured from console by console
R1#config t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#ing f1/0
           ^
% Invalid input detected at '^' marker.

R1(config)#int f1/0
R1(config-if)#encapsulation dot1Q 20
                            ^
% Invalid input detected at '^' marker.

R1(config-if)#ex
R1(config)#int f1/0.20
R1(config-subif)#encapsulation dot1Q 20
R1(config-subif)#ip add 192.168.20.1 255.255.255.0
R1(config-subif)#do wr
Building configuration...
[OK]
R1(config-subif)#
```

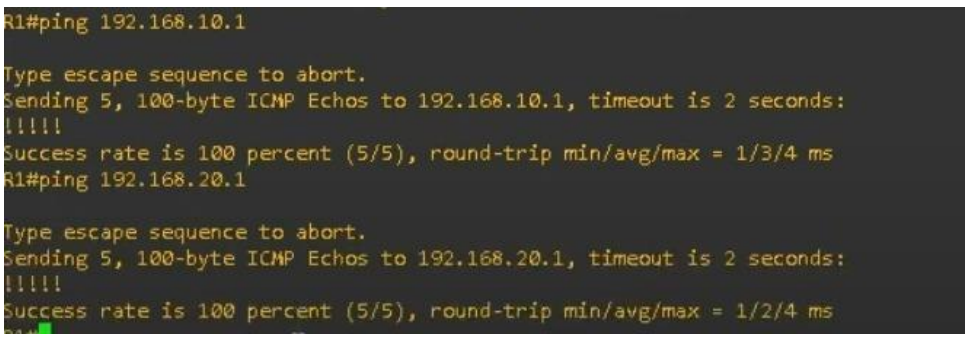
```
R1(config-subif)#end
R1#sh
*Mar  1 00:05:01.563: %SYS-5-CONFIG_I: Configured from console by console
R1#sh ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

C    192.168.10.0/24 is directly connected, FastEthernet1/0.10
C    192.168.20.0/24 is directly connected, FastEthernet1/0.20
R1#
```

```
et1/0, changed state to up

**********************************************************
This is a normal Router with a Switch module inside (NM-16ESW)
It has been pre-configured with hard-coded speed and duplex

To create vlans use the command "vlan database" in exec mode
After creating all desired vlans use "exit" to apply the config

To view existing vlans use the command "show vlan-switch brief"

Alias(exec)      : vl    - "show vlan-switch brief" command
Alias(configure): va X - macro to add vlan X
Alias(configure): vd X - macro to delete vlan X
**********************************************************


ESW1#config t
```

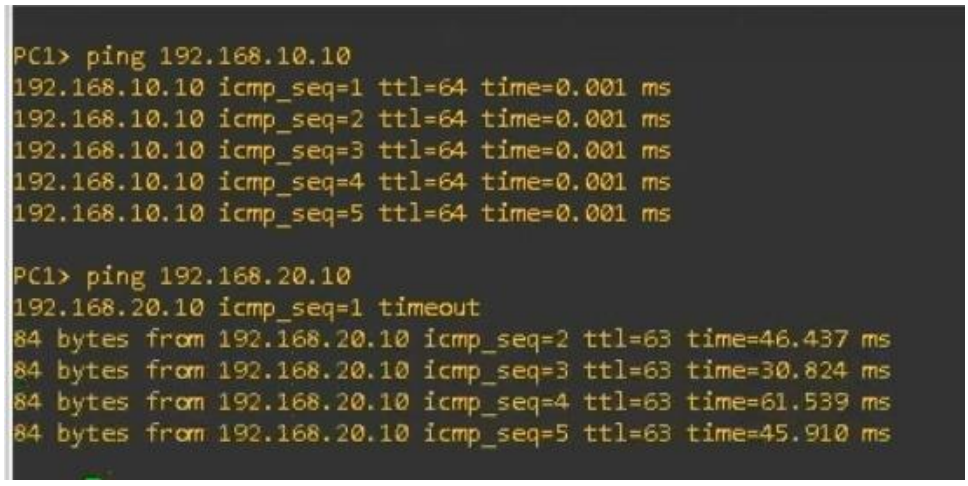
```
ESW1(config)#int f1/0
ESW1(config-if)#no sh
ESW1(config-if)#int f1/2
ESW1(config-if)#no sh
ESW1(config-if)#int f1/1
ESW1(config-if)#no sh
ESW1(config-if)#do wr
Building configuration...
```



```
ESW1(config-if)#end
ESW1#
*Mar  1 00:02:52.959: %SYS-5-CONFIG_I: Configured from console by console
ESW1#sh vlan-switch

VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
1    default                          active    Fa1/3, Fa1/4, Fa1/5, Fa1/6
                                                Fa1/7, Fa1/8, Fa1/9, Fa1/10
                                                Fa1/11, Fa1/12, Fa1/13, Fa1/14
                                                Fa1/15
10   VLAN0010                         active    Fa1/1
20   VLAN0020                         active    Fa1/2
1002 fddi-default                     act/unsup
1003 token-ring-default               act/unsup
1004 fddinet-default                  act/unsup
1005 trnet-default                    act/unsup

VLAN Type  SAID       MTU   Parent RingNo BridgeNo Stp  BrdgMode Trans1 Trans2
---- ----- ---------- ----- ------ ------ -------- ---- -------- ------ ------
1    enet  100001     1500  -      -      -        -    -        1002   1003
10   enet  100010     1500  -      -      -        -    -        0      0
20   enet  100020     1500  -      -      -        -    -        0      0
1002 fddi  101002     1500  -      -      -        -    -        1      1003
1003 tr    101003     1500  1005   0      -        -    srb      1      1002
1004 fdnet 101004     1500
```



```
R1#ping 192.168.10.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.10.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-round-trip min/avg/max = 1/3/4 ms
R1#ping 192.168.20.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.20.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
```



```
PC1> ping 192.168.10.10
192.168.10.10 icmp_seq=1 ttl=64 time=0.001 ms
192.168.10.10 icmp_seq=2 ttl=64 time=0.001 ms
192.168.10.10 icmp_seq=3 ttl=64 time=0.001 ms
192.168.10.10 icmp_seq=4 ttl=64 time=0.001 ms
192.168.10.10 icmp_seq=5 ttl=64 time=0.001 ms

PC1> ping 192.168.20.10
192.168.20.10 icmp_seq=1 timeout
84 bytes from 192.168.20.10 icmp_seq=2 ttl=63 time=46.437 ms
84 bytes from 192.168.20.10 icmp_seq=3 ttl=63 time=30.824 ms
84 bytes from 192.168.20.10 icmp_seq=4 ttl=63 time=61.539 ms
84 bytes from 192.168.20.10 icmp_seq=5 ttl=63 time=45.910 ms
```

Conclusion:

Inter-VLAN Routing successfully enables communication between different VLANs using a Layer 3 device.

Shri Vile Parle Kelavani Mandal's
USHA PRAVIN GANDHI COLLEGE OF ARTS, SCIENCE AND COMMERCE
(Autonomous Affiliated to University of Mumbai)
Bhaktivedanta Swami Marg, Juhu Scheme, Vile Parle (West), Mumbai - 400056
Tel.: 42332041/42/43/44/45/46 ● Website: www.upgcm.ac.in ● Email: info@upgcm.ac.in
NAAC RE-ACCREDITED " A+ " GRADE WITH CGPA 3.27

CERTIFICATE

M.Sc.(I.T.) Part – I, Semester - II
2024 - 2025

This is to certify that

Mr. / Ms. : _____,

student of Master of Science (Information Technology) Part - I, Semester - II Roll No: _____ *has successfully completed practical and project work in* _____,

in partial fulfillment as per the syllabus for the academic year 2024 – 2025.

The practical and project work are completed successfully under the supervision of the undersigned.

Internal Examiner	***External Examiner***
Name:	***Name:***
Date:	***Date:***
Program Coordinator	***Head of Department***
Name: Dr. Swapnali Lotlikar	***Name: Dr. Smruti Nanavaty***
Date:	***Date:***

INDEX

Practical 1

Aim:- Implement Decision tree classification techniques

A decision tree is a graphical representation of different options for solving a problem and show how different factors are related. It has a hierarchical tree structure starts with one main question at the top called a node which further branches out into different possible outcomes where:

- **Root Node** is the starting point that represents the entire dataset.
- **Branches**: These are the lines that connect nodes. It shows the flow from one decision to another.
- **Internal Nodes** are Points where decisions are made based on the input features.
- **Leaf Nodes**: These are the terminal nodes at the end of branches that represent final outcomes or predictions

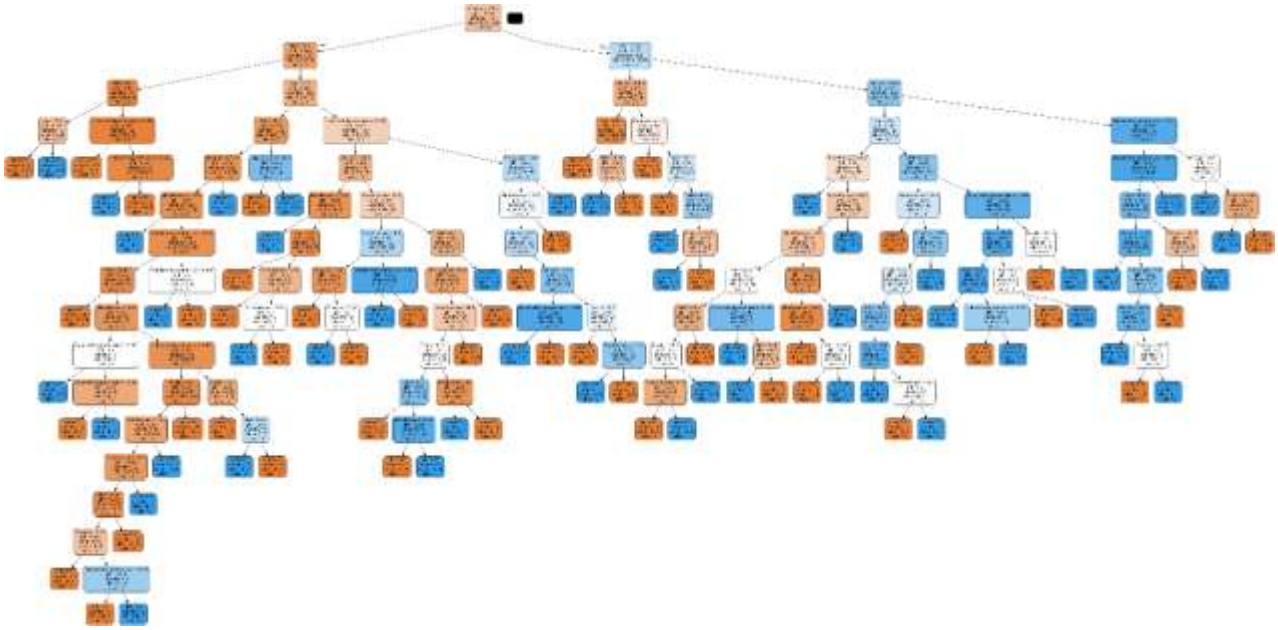
Code:-

Decision Tree without using PySpark

```
from google.colab import drive
drive.mount("/content/drive")

import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics

col_names=['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','Diabete
sPedigreeFunction','Age','Outcome']

pima = pd.read_csv("/content/drive/MyDrive/diabetes.csv", header = 0, names = col_names)
pima.head()

feature_cols=['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','Diabe
tesPedigreeFunction','Age']

X = pima[feature_cols]
y = pima.Outcome.astype('int')
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 1)
X_train = X_train.astype('float')
X_test = X_test.astype('float')
clf = DecisionTreeClassifier()
```

```
clf = clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))


!pip install graphviz
!pip install pydotplus

from sklearn.tree import export_graphviz
from io import StringIO
from IPython.display import Image
import pydotplus

dot_data = StringIO()
export_graphviz(clf, out_file = dot_data, filled = True, rounded = True, special_characters =
True, feature_names = feature_cols, class_names = ['0','1'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('diabetes.png')
Image(graph.create_png())
```

Output:-



Decision Tree using PySpark

```
from pyspark.sql import SparkSession
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.classification import DecisionTreeClassifier
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.ml.classification import DecisionTreeClassificationModel
import matplotlib.pyplot as plt
from graphviz import Source

from google.colab import drive
drive.mount("/content/drive")

spark = SparkSession.builder.appName("DiabetesPrediction").getOrCreate()
data = spark.read.csv("/content/drive/MyDrive/diabetes.csv", header=True, inferSchema=True)
feature_cols=['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPedigreeFunction','Age']
assembler = VectorAssembler(inputCols=feature_cols, outputCol="features")
data = assembler.transform(data)
data = data.select("features", "Outcome")
train_data, test_data = data.randomSplit([0.8, 0.2], seed=42)
dt = DecisionTreeClassifier(labelCol="Outcome", featuresCol="features")
model = dt.fit(train_data)
predictions = model.transform(test_data)
evaluator=MulticlassClassificationEvaluator(labelCol="Outcome",metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
print(f"Test Accuracy: {accuracy}")
predictions.select("features", "Outcome", "prediction").show(10)
```

Output:-

```
Test Accuracy: 0.7967479674796748
```

```
+--------------------+-------+----------+
|            features|Outcome|prediction|
+--------------------+-------+----------+
|(8,[0,1,6,7],[3.0...|      0|       0.0|
|(8,[1,5,6,7],[73....|      0|       0.0|
|(8,[1,5,6,7],[117...|      0|       1.0|
|(8,[1,5,6,7],[145...|      1|       1.0|
|[0.0,78.0,88.0,29...|      0|       0.0|
|[0.0,91.0,68.0,32...|      0|       0.0|
|[0.0,95.0,64.0,39...|      0|       0.0|
|[0.0,100.0,88.0,6...|      0|       1.0|
|[0.0,104.0,64.0,2...|      0|       0.0|
|[0.0,104.0,64.0,3...|      1|       0.0|
+--------------------+-------+----------+
only showing top 10 rows
```

Practical 2

Aim:- Implement SVM classification technique.

Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification and regression tasks. While it can handle regression problems, SVM is particularly well-suited for classification tasks.

SVM aims to find the optimal hyperplane in an N-dimensional space to separate data points into different classes. The algorithm maximizes the margin between the closest points of different classes.

SVM without using PySpark

```
# Mount Google Drive
from google.colab import drive
drive.mount('/content/drive')

# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn import metrics

# Load dataset
col_names = ["Pregnancies", "Glucose", "BloodPressure", "SkinThickness", "Insulin",
             "BMI", "DiabetesPedigreeFunction", "Age", "Outcome"]
pima = pd.read_csv("/content/drive/MyDrive/computer_vision/diabetes.csv",
                   header=0, names=col_names)

# Select two features for visualization
feature_cols = ["Glucose", "BMI"]  # Only 2 features for 2D plotting
X = pima[feature_cols]
y = pima.Outcome.astype("int")

# Split data into train & test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)

# Convert data types
X_train = X_train.astype("float")
X_test = X_test.astype("float")

# Train SVM model
clf = SVC(kernel='linear')  # Using a linear kernel for simplicity
clf.fit(X_train, y_train)

# Make predictions
y_pred = clf.predict(X_test)
```

```
# Print accuracy
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))
```

Output:-

Accuracy: 0.7662337662337663

SVM using PySpark

```
# Import necessary libraries
from pyspark.sql import SparkSession
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.classification import LinearSVC
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
import matplotlib.pyplot as plt
import seaborn as sns

# Mount Google Drive
from google.colab import drive
drive.mount("/content/drive")

# Create Spark session
spark = SparkSession.builder.appName("DiabetesPrediction_SVM").getOrCreate()

# Load dataset
data = spark.read.csv("/content/drive/MyDrive/computer_vision/diabetes.csv",
header=True, inferSchema=True)

# Define feature columns
feature_cols = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
            'BMI', 'DiabetesPedigreeFunction', 'Age']

# Convert features into a single vector
assembler = VectorAssembler(inputCols=feature_cols, outputCol="features")
data = assembler.transform(data)

# Select relevant columns
data = data.select("features", "Outcome")

# Split data into training (80%) and testing (20%) sets
train_data, test_data = data.randomSplit([0.8, 0.2], seed=42)

# Train SVM model
svm = LinearSVC(labelCol="Outcome", featuresCol="features", maxIter=100,
regParam=0.01)
model = svm.fit(train_data)
```

```
# Make predictions on test data
predictions = model.transform(test_data)

# Evaluate accuracy
evaluator = MulticlassClassificationEvaluator(labelCol="Outcome",
metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
print(f"Test Accuracy: {accuracy:.4f}")

# Show some predictions
predictions.select("features", "Outcome", "prediction").show(10)
```

Output:-

```
Test Accuracy: 0.8130
+--------------------+-------+----------+
|            features|Outcome|prediction|
+--------------------+-------+----------+
|(8,[0,1,6,7],[3.0...|      0|       0.0|
|(8,[1,5,6,7],[73....|      0|       0.0|
|(8,[1,5,6,7],[117...|      0|       1.0|
|(8,[1,5,6,7],[145...|      1|       1.0|
|[0.0,78.0,88.0,29...|      0|       0.0|
|[0.0,91.0,68.0,32...|      0|       0.0|
|[0.0,95.0,64.0,39...|      0|       0.0|
|[0.0,100.0,88.0,6...|      0|       0.0|
|[0.0,104.0,64.0,2...|      0|       0.0|
|[0.0,104.0,64.0,3...|      1|       0.0|
+--------------------+-------+----------+
only showing top 10 rows
```

Practical 3

Aim:- Implement Random Forest classification technique.

Random Forest algorithm is a powerful tree learning technique in Machine Learning to make predictions and then we do voting of all the tress to make prediction. They are widely used for classification and regression task.

- It is a type of classifier that uses many decision trees to make predictions.
- It takes different random parts of the dataset to train each tree and then it combines the results by averaging them. This approach helps improve the accuracy of predictions. Random Forest is based on ensemble learning.

Random Forest without using PySpark

```
# Mount Google Drive
from google.colab import drive
drive.mount('/content/drive')

# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestClassifier  # Using Random Forest
from sklearn.model_selection import train_test_split
from sklearn import metrics

# Load dataset
col_names = ["Pregnancies", "Glucose", "BloodPressure", "SkinThickness", "Insulin",
             "BMI", "DiabetesPedigreeFunction", "Age", "Outcome"]
pima = pd.read_csv("/content/drive/MyDrive/computer_vision/diabetes.csv",
                   header=0, names=col_names)

# Define features and target variable
feature_cols = ["Pregnancies", "Glucose", "BloodPressure", "SkinThickness", "Insulin",
                "BMI", "DiabetesPedigreeFunction", "Age"]
X = pima[feature_cols]
y = pima.Outcome.astype("int")

# Split into train & test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)

# Convert data types
X_train = X_train.astype("float")
X_test = X_test.astype("float")

# Train Random Forest model
clf = RandomForestClassifier(n_estimators=100, random_state=42)  # 100 trees
clf.fit(X_train, y_train)
```

```
# Make predictions
y_pred = clf.predict(X_test)

# Print accuracy
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))
```

Output:-

Accuracy: 0.8138528138528138

Random Forest using PySpark

```
# Import necessary libraries
from pyspark.sql import SparkSession
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
import matplotlib.pyplot as plt
import seaborn as sns

# Mount Google Drive
from google.colab import drive
drive.mount("/content/drive")

# Create Spark session
spark = SparkSession.builder.appName("DiabetesPrediction_RF").getOrCreate()

# Load dataset
data = spark.read.csv("/content/drive/MyDrive/computer_vision/diabetes.csv",
header=True, inferSchema=True)

# Define feature columns
feature_cols = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
            'BMI', 'DiabetesPedigreeFunction', 'Age']

# Convert features into a single vector
assembler = VectorAssembler(inputCols=feature_cols, outputCol="features")
data = assembler.transform(data)

# Select relevant columns
data = data.select("features", "Outcome")

# Split data into training (80%) and testing (20%) sets
train_data, test_data = data.randomSplit([0.8, 0.2], seed=42)

# Train Random Forest model
```

```
rf = RandomForestClassifier(labelCol="Outcome", featuresCol="features", numTrees=100)
model = rf.fit(train_data)

# Make predictions on test data
predictions = model.transform(test_data)

# Evaluate accuracy
evaluator = MulticlassClassificationEvaluator(labelCol="Outcome",
metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
print(f"Test Accuracy: {accuracy:.4f}")

# Show some predictions
predictions.select("features", "Outcome", "prediction").show(10)
```

Output:-

```
Test Accuracy: 0.8211
+--------------------+-------+----------+
|            features|Outcome|prediction|
+--------------------+-------+----------+
|(8,[0,1,6,7],[3.0...|      0|       0.0|
|(8,[1,5,6,7],[73....|      0|       0.0|
|(8,[1,5,6,7],[117...|      0|       1.0|
|(8,[1,5,6,7],[145...|      1|       1.0|
|[0.0,78.0,88.0,29...|      0|       0.0|
|[0.0,91.0,68.0,32...|      0|       0.0|
|[0.0,95.0,64.0,39...|      0|       0.0|
|[0.0,100.0,88.0,6...|      0|       1.0|
|[0.0,104.0,64.0,2...|      0|       0.0|
|[0.0,104.0,64.0,3...|      1|       0.0|
+--------------------+-------+----------+
only showing top 10 rows
```

Practical 4

Aim:- Implement the K-means clustering technique.

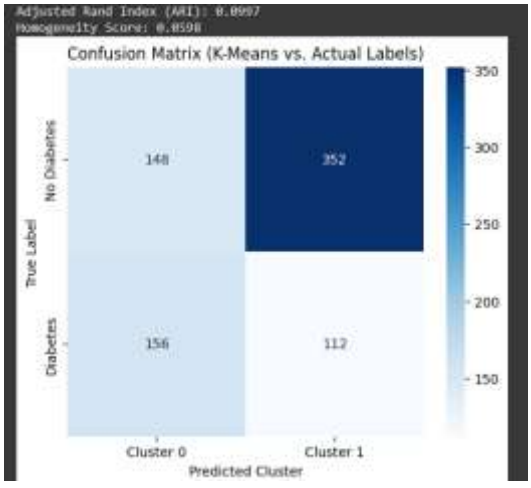
K-means clustering works by first randomly picking some central points called centroids and each data point is then assigned to the closest centroid forming a cluster. After all the points are assigned to a cluster the centroids are updated by finding the average position of the points in each cluster. This process repeats until the centroids stop changing forming clusters. The goal of clustering is to divide the data points into clusters so that similar data points belong to same group.

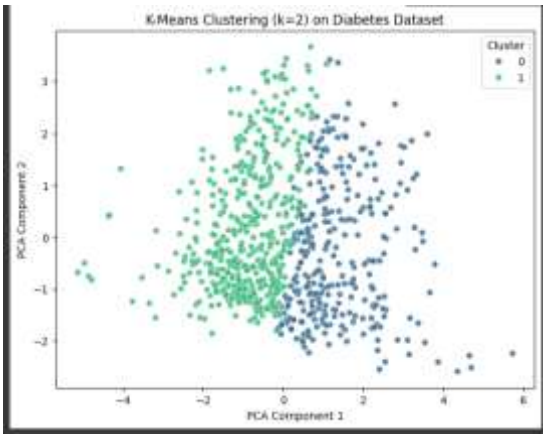
K-means without using PySpark

```
# Mount Google Drive
from google.colab import drive
drive.mount('/content/drive')

# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import adjusted_rand_score, homogeneity_score, confusion_matrix

# Load dataset
col_names = ["Pregnancies", "Glucose", "BloodPressure", "SkinThickness", "Insulin",
             "BMI", "DiabetesPedigreeFunction", "Age", "Outcome"]
pima = pd.read_csv("/content/drive/MyDrive/computer_vision/diabetes.csv",
                   header=0, names=col_names)

# Select features (exclude Outcome since it's unsupervised learning)
feature_cols = ["Pregnancies", "Glucose", "BloodPressure", "SkinThickness", "Insulin",
                "BMI", "DiabetesPedigreeFunction", "Age"]
X = pima[feature_cols]
y_true = pima["Outcome"]  # Actual diabetes labels

# Standardize features (important for K-Means)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Apply K-Means Clustering
k = 2  # We expect two groups: diabetic & non-diabetic
kmeans = KMeans(n_clusters=k, random_state=42)
clusters = kmeans.fit_predict(X_scaled)

# Add cluster labels to the dataset
pima["Cluster"] = clusters
```

```
# 📊 Accuracy Metrics (How well does clustering match actual labels?)
ari_score = adjusted_rand_score(y_true, clusters)
homogeneity = homogeneity_score(y_true, clusters)

print(f"Adjusted Rand Index (ARI): {ari_score:.4f}")  # Closer to 1 is better
print(f"Homogeneity Score: {homogeneity:.4f}")  # Closer to 1 is better

# 📊 Confusion Matrix
plt.figure(figsize=(6, 5))
cm = confusion_matrix(y_true, clusters)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=["Cluster 0", "Cluster 1"],
            yticklabels=["No Diabetes", "Diabetes"])
plt.xlabel("Predicted Cluster")
plt.ylabel("True Label")
plt.title("Confusion Matrix (K-Means vs. Actual Labels)")
plt.show()

# 📊 Visualize Clusters using PCA (2D plot)
pca = PCA(n_components=2)  # Reduce to 2 dimensions for visualization
X_pca = pca.fit_transform(X_scaled)

plt.figure(figsize=(8, 6))
sns.scatterplot(x=X_pca[:, 0], y=X_pca[:, 1], hue=pima["Cluster"], palette="viridis", alpha=0.8)
plt.xlabel("PCA Component 1")
plt.ylabel("PCA Component 2")
plt.title("K-Means Clustering (k=2) on Diabetes Dataset")
plt.legend(title="Cluster")
plt.show()
```

Output:-

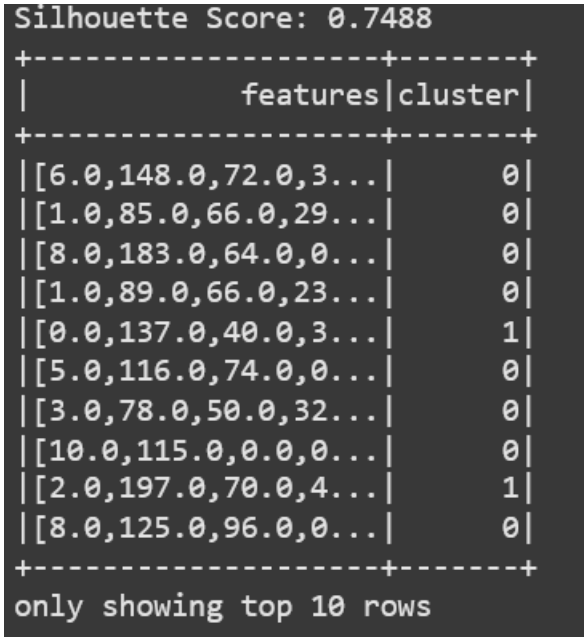




K-means using PySpark

```
# Import necessary libraries
from pyspark.sql import SparkSession
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.clustering import KMeans
from pyspark.ml.evaluation import ClusteringEvaluator
import matplotlib.pyplot as plt
import seaborn as sns

# Mount Google Drive
from google.colab import drive
drive.mount("/content/drive")

# Create Spark session
spark = SparkSession.builder.appName("Diabetes_KMeans").getOrCreate()

# Load dataset
data = spark.read.csv("/content/drive/MyDrive/computer_vision/diabetes.csv",
header=True, inferSchema=True)

# Define feature columns
feature_cols = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
            'BMI', 'DiabetesPedigreeFunction', 'Age']

# Convert features into a single vector
assembler = VectorAssembler(inputCols=feature_cols, outputCol="features")
data = assembler.transform(data)

# Select only feature vector (KMeans does not use labels)
data = data.select("features")

# Train K-Means model (Choosing 2 clusters since we have binary classification data)
kmeans = KMeans(k=2, seed=42, featuresCol="features", predictionCol="cluster")
model = kmeans.fit(data)

# Make predictions
predictions = model.transform(data)

# Evaluate clustering using Silhouette Score
evaluator = ClusteringEvaluator(featuresCol="features", predictionCol="cluster",
metricName="silhouette")
silhouette_score = evaluator.evaluate(predictions)
print(f"Silhouette Score: {silhouette_score:.4f}")

# Show some clustered data
predictions.select("features", "cluster").show(10)
```

Output:-



```
Silhouette Score: 0.7488
+--------------------+-------+
|            features|cluster|
+--------------------+-------+
|[6.0,148.0,72.0,3...|      0|
|[1.0,85.0,66.0,29...|      0|
|[8.0,183.0,64.0,0...|      0|
|[1.0,89.0,66.0,23...|      0|
|[0.0,137.0,40.0,3...|      1|
|[5.0,116.0,74.0,0...|      0|
|[3.0,78.0,50.0,32...|      0|
|[10.0,115.0,0.0,0...|      0|
|[2.0,197.0,70.0,4...|      1|
|[8.0,125.0,96.0,0...|      0|
+--------------------+-------+
only showing top 10 rows
```

Practical 5

Aim:- Implement Linear Regression model using python.

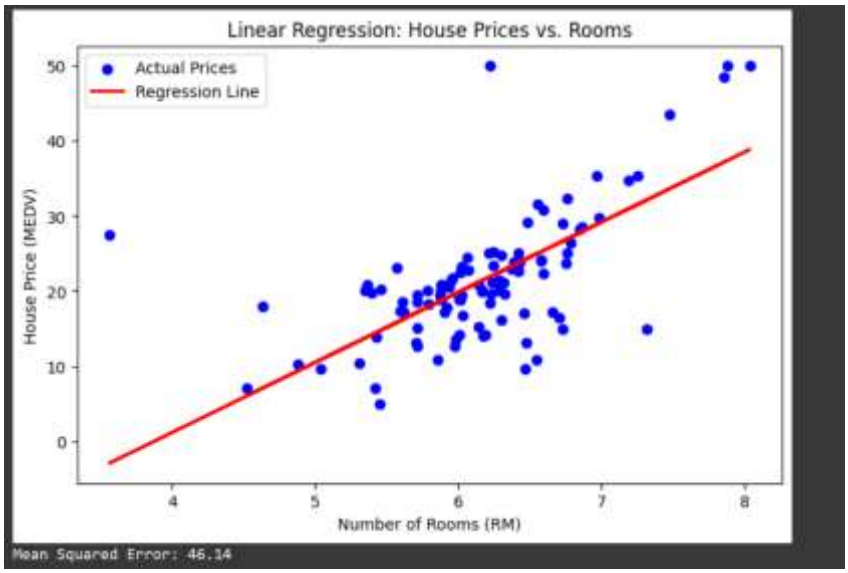
Linear regression is also a type of supervised machine-learning algorithm that learns from the labelled datasets and maps the data points with most optimized linear functions which can be used for prediction on new datasets. It computes the linear relationship between the dependent variable and one or more independent features by fitting a linear equation with observed data. It predicts the continuous output variables based on the independent input variable.

Linear Regression without using PySpark

```
# Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.datasets import fetch_openml

# Load dataset
boston = fetch_openml(name='boston', version=1, as_frame=True)
df = boston.frame

# Select features and target variable
X = df[['RM']]  # Number of rooms as a feature
y = df['MEDV']  # Median house price

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Predictions
y_pred = model.predict(X_test)

# 📊 Plot Regression Line
plt.figure(figsize=(8, 5))
plt.scatter(X_test, y_test, color='blue', label='Actual Prices')
plt.plot(X_test, y_pred, color='red', linewidth=2, label='Regression Line')
plt.xlabel("Number of Rooms (RM)")
plt.ylabel("House Price (MEDV)")
plt.title("Linear Regression: House Prices vs. Rooms")
plt.legend()
plt.show()
```

```
# Print Mean Squared Error
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse:.2f}")
from sklearn.metrics import accuracy_score
```

Output:-



Linear Regression using PySpark

```
from pyspark.sql import SparkSession
from pyspark.ml.regression import LinearRegression
from pyspark.ml.feature import VectorAssembler
from sklearn.datasets import fetch_california_housing
import pandas as pd

# Initialize Spark Session
spark = SparkSession.builder.appName("LinearRegression").getOrCreate()

# Load California Housing Dataset
california = fetch_california_housing(as_frame=True)
df = pd.DataFrame(california.data, columns=california.feature_names)
df["MedHouseVal"] = california.target  # Target variable

# Convert to PySpark DataFrame
spark_df = spark.createDataFrame(df)

# Assemble features into a single column
feature_cols = california.feature_names
assembler = VectorAssembler(inputCols=feature_cols, outputCol="features")
data = assembler.transform(spark_df).select("features", "MedHouseVal")
```

```
# Train-Test Split
train_data, test_data = data.randomSplit([0.8, 0.2], seed=42)

# Train Linear Regression Model
lr = LinearRegression(featuresCol="features", labelCol="MedHouseVal")
model = lr.fit(train_data)

# Model Evaluation
predictions = model.transform(test_data)
print(f"RMSE: {model.summary.rootMeanSquaredError:.4f}")
print(f"R² Score: {model.summary.r2:.4f}")

# Show some predictions
predictions.select("features", "MedHouseVal", "prediction").show(10)
```

Output:-

```
RMSE: 0.7231
R² Score: 0.6095
+--------------------+-----------+--------------------+
|            features|MedHouseVal|          prediction|
+--------------------+-----------+--------------------+
|[0.4999,28.0,7.67...|    5.00001| 0.8636611002081835|
|[0.4999,52.0,2.6,...|      0.906| 1.0207615146533868|
|[0.536,16.0,4.5,1...|        3.5|  1.091494549716252|
|[0.6775,52.0,1.74...|        3.5| 1.5128374133887448|
|[0.7403,37.0,4.49...|      0.686| 1.1569189315147597|
|[0.76,10.0,2.6515...|      1.625| 1.0887481760842306|
|[0.8026,23.0,5.36...|      1.125| 0.9722254546464413|
|[0.8185,13.0,2.49...|      1.875| 0.9692636883896455|
|[0.8585,15.0,4.82...|       0.45|-0.1640116290411484|
|[0.8639,28.0,4.28...|      0.494| 0.7026276336935879|
+--------------------+-----------+--------------------+
only showing top 10 rows
```

Practical 6

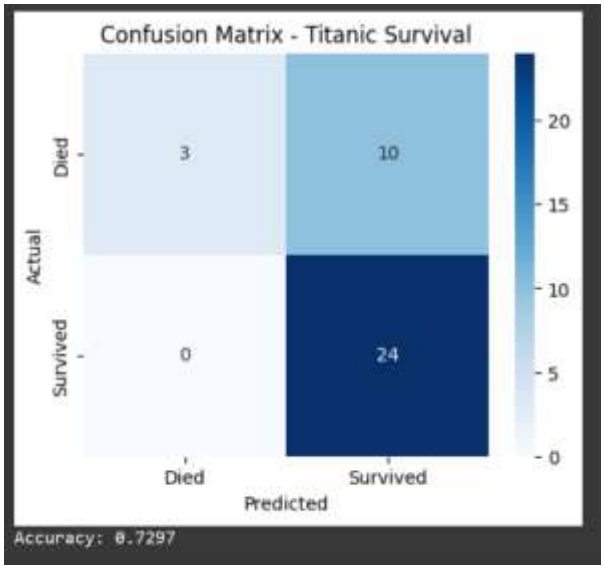
Aim:- Implement Logistic Regression model using python.

Logistic regression is a supervised machine learning algorithm used for classification tasks where the goal is to predict the probability that an instance belongs to a given class or not. Logistic regression is a statistical algorithm which analyze the relationship between two data factors.

Logistic Regression without using PySpark

```
# Import necessary libraries
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix

# Load dataset
df = sns.load_dataset('titanic').dropna()

# Select features and target variable
X = df[['pclass', 'age', 'fare', 'sibsp', 'parch']]
y = df['survived']

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train Logistic Regression model
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

# Predictions
y_pred = model.predict(X_test)

# 📊 Confusion Matrix
plt.figure(figsize=(5, 4))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, cmap="Blues", fmt="d",
xticklabels=['Died', 'Survived'], yticklabels=['Died', 'Survived'])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix - Titanic Survival")
plt.show()

# Print Accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.4f}")
```

Output:-



Logistic Regression using PySpark

```
from pyspark.sql import SparkSession
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.feature import VectorAssembler
from sklearn.datasets import load_breast_cancer
import pandas as pd

# Initialize Spark Session
spark = SparkSession.builder.appName("LogisticRegression").getOrCreate()

# Load Breast Cancer Dataset
cancer = load_breast_cancer(as_frame=True)
df = pd.DataFrame(cancer.data, columns=cancer.feature_names)
df["target"] = cancer.target  # Target variable (0 or 1)

# Convert to PySpark DataFrame
spark_df = spark.createDataFrame(df)

# Assemble features into a single column
feature_cols = cancer.feature_names
assembler = VectorAssembler(inputCols=feature_cols, outputCol="features")
data = assembler.transform(spark_df).select("features", "target")

# Train-Test Split
train_data, test_data = data.randomSplit([0.8, 0.2], seed=42)
```

```
# Train Logistic Regression Model
lr = LogisticRegression(featuresCol="features", labelCol="target")
model = lr.fit(train_data)

# Model Evaluation
predictions = model.transform(test_data)
accuracy = predictions.filter(predictions.prediction == predictions.target).count() /
test_data.count()
print(f"Test Accuracy: {accuracy:.4f}")

# Show some predictions
predictions.select("features", "target", "prediction").show(10)
```

Output:-

```
Test Accuracy: 0.9368
+--------------------+------+----------+
|            features|target|prediction|
+--------------------+------+----------+
|[8.219,20.7,53.27...|     1|       1.0|
|[8.726,15.83,55.8...|     1|       1.0|
|[8.95,15.76,58.74...|     1|       1.0|
|[9.567,15.91,60.2...|     1|       1.0|
|[9.787,19.94,62.1...|     1|       1.0|
|[10.08,15.11,63.7...|     1|       1.0|
|[10.49,19.29,67.4...|     1|       1.0|
|[10.75,14.97,68.2...|     1|       1.0|
|[11.31,19.04,71.8...|     1|       1.0|
|[11.32,27.08,71.7...|     1|       1.0|
+--------------------+------+----------+
only showing top 10 rows
```

Practical 7

Aim:- Implement Multiple Regression model using python.

Linear regression is a basic and commonly used for predictive analysis. It is a statistical approach for modeling relationship between a dependent variable and one independent variables. Multiple Linear Regression is a extended version of it only. It attempts to model relationship between two or more features to fit a linear equation to predict one dependent variable.

The equation for multiple linear regression is:

y=β0+β1X1+β2X2+⋯+βnXny=β0+β1X1+β2X2+⋯+βnXn

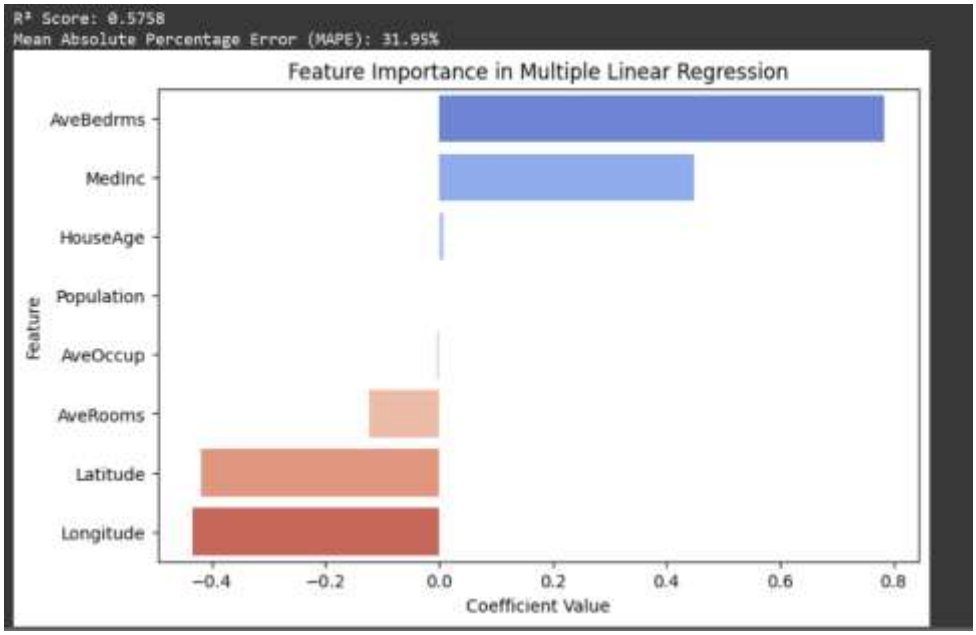
- yyis the dependent variable
- X1,X2,⋯XnX1,X2,⋯Xn are the independent variables
- β0β0 is the intercept
- β1,β2,⋯βnβ1,β2,⋯βnare the slopes

Multiple Regression without using PySpark

```
# ✅ Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_absolute_percentage_error

#  Load California Housing dataset (built-in)
data = fetch_california_housing(as_frame=True)
df = data.frame  # Convert to Pandas DataFrame

#  Select features and target variable
X = df.drop(columns=["MedHouseVal"])  # Features
y = df["MedHouseVal"]  # Target (Median House Value)

# Split into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train Multiple Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)

#  Make predictions
y_pred = model.predict(X_test)
```

```
#  Evaluate performance
r2 = r2_score(y_test, y_pred)
mape = mean_absolute_percentage_error(y_test, y_pred) * 100  # Convert to %

print(f"R² Score: {r2:.4f}")  # How well the model explains variance
print(f"Mean Absolute Percentage Error (MAPE): {mape:.2f}%")  # Lower is better

#  Feature Importance Visualization (Based on Coefficients)
feature_importance = pd.DataFrame({"Feature": X.columns, "Coefficient": model.coef_})
feature_importance = feature_importance.sort_values(by="Coefficient", ascending=False)

#  Plot feature importance
plt.figure(figsize=(8, 5))
sns.barplot(x="Coefficient", y="Feature", hue="Feature", data=feature_importance,
            palette="coolwarm", legend=False)
plt.xlabel("Coefficient Value")
plt.ylabel("Feature")
plt.title("Feature Importance in Multiple Linear Regression")
plt.show()
```

Output:-



Multiple Regression using PySpark

```
from pyspark.sql import SparkSession
from pyspark.ml.regression import LinearRegression
from pyspark.ml.feature import VectorAssembler
from sklearn.datasets import load_diabetes
import pandas as pd
```

```
# Initialize Spark Session
spark = SparkSession.builder.appName("MultipleLinearRegression").getOrCreate()

# Load Diabetes Dataset
diabetes = load_diabetes(as_frame=True)
df = pd.DataFrame(diabetes.data, columns=diabetes.feature_names)
df["target"] = diabetes.target  # Target variable

# Convert to PySpark DataFrame
spark_df = spark.createDataFrame(df)

# Assemble features into a single column
feature_cols = diabetes.feature_names
assembler = VectorAssembler(inputCols=feature_cols, outputCol="features")
data = assembler.transform(spark_df).select("features", "target")

# Train-Test Split
train_data, test_data = data.randomSplit([0.8, 0.2], seed=42)

# Train Multiple Linear Regression Model
mlr = LinearRegression(featuresCol="features", labelCol="target")
model = mlr.fit(train_data)

# Model Evaluation
predictions = model.transform(test_data)
print(f"RMSE: {model.summary.rootMeanSquaredError:.4f}")
print(f"R² Score: {model.summary.r2:.4f}")

# Show some predictions
predictions.select("features", "target", "prediction").show(10)
```

Output:-

```
RMSE: 52.1752
R² Score: 0.5384
+--------------------+------+------------------+
|            features|target|        prediction|
+--------------------+------+------------------+
|[-0.0999605547053...|  55.0| 75.33250271834474|
|[-0.0963280162542...| 200.0| 80.07849393946874|
|[-0.0926954770032...|  97.0|105.90136220615642|
|[-0.0854304089012...|  60.0|117.81688255328885|
|[-0.0745327855481...| 103.0| 144.7652237520766|
|[-0.0709002470971...| 310.0|  215.066950242682|
|[-0.0636351701951...| 101.0|  95.2968361705268|
|[-0.0563700932930...| 128.0|234.51396387075857|
|[-0.0491050163910...|  68.0|124.60475420943342|
|[-0.0491050163910...| 200.0|159.47600114277665|
+--------------------+------+------------------+
only showing top 10 rows
```

Practical 8

Aim:- Install, configure, and run Hadoop and HDFS and explore HDFS.

Step 1: Download Java from oracle website

- Search for java SE development kit 8 download
- https://www.oracle.com/java/technologies/downloads/#java8-windows

Step 2: Download Hadoop for the local system

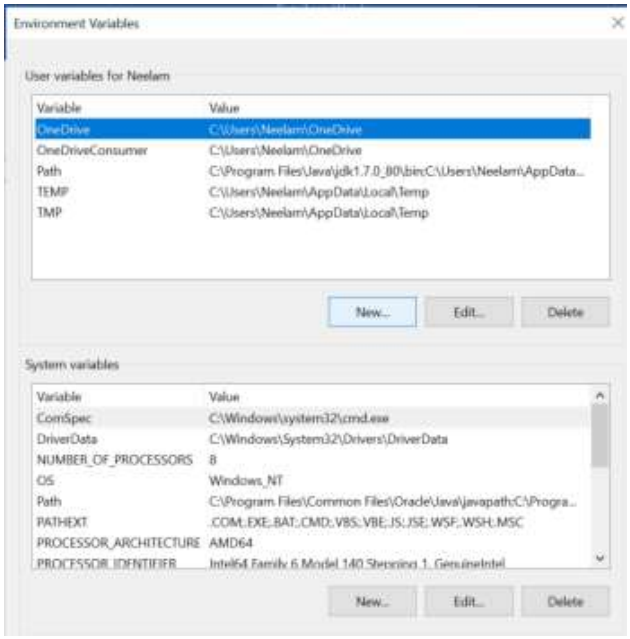
- https://hadoop.apache.org/releases.html

- Move files to c drive

Step 3: Install Java

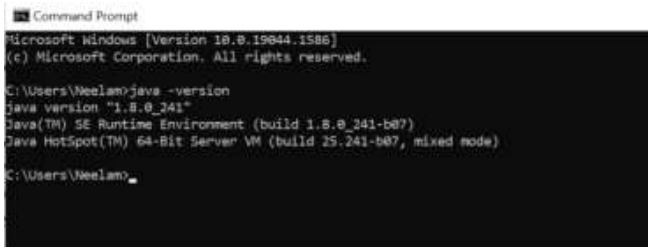
Step 4: Setting environment variables for java

- Windows->settings->system->environment variables for system->edit the system environment variables
- Set java home and path for java, Set user variables and system variables.



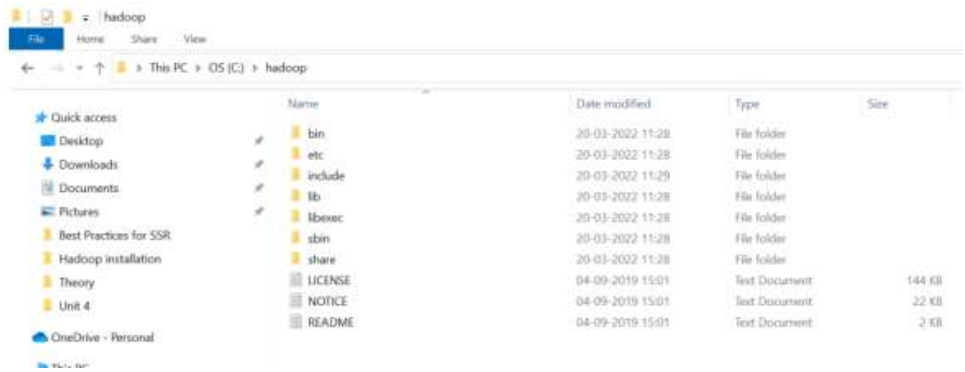
Step 5: Java version checking

- Go to command prompt



Step 6: Install Hadoop now to our local system

- Unzip Hadoop setup file
- Rename Hadoop-3.1.3 folder as Hadoop



Step 7: Configuration of Hadoop

- Hadoop->etc-> Hadoop
- Important files: Core-site.xml, Hdfs-site.xml, Mapred-site.xml, Yarn-site.xml, Hadoop-env windows command prompt file
- Open all these files in notepad++

Core-site.xml

<configuration>

<property>

<name>fs.defaultFS</name>

<value>hdfs://localhost:9000</value>

</property

</configuration>

Mapred-site.xml

<configuration>

<property>

<name>mapred.framework.name</name>

<value>yarn</value>

</property>

</configuration>

Yarn-site.xml

<configuration>

<!-- Site specific YARN configuration properties -->

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce_shuffle</value>

</property>

<property>

<name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>

<value>org.apache.hadoop.mapred.ShuffleHandler</value>

</property>

</configuration>

Create data directory and subdirectories in Hadoop

Hdfs-site.xml file

<configuration>

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

<property>

<name>dfs.namenode.name.dir</name>

<value>file:///C:/hadoop/data/namenode</value>

</property>

<property>

<name>dfs.datanode.data.dir</name>

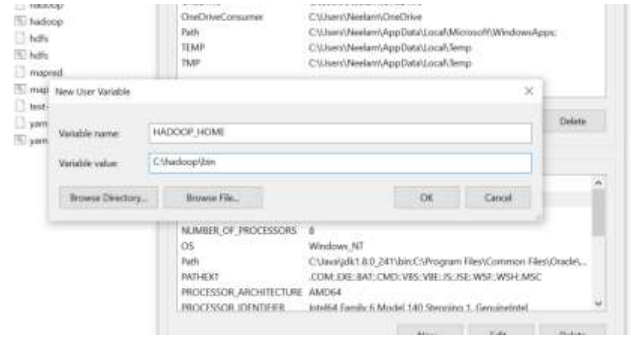
<value>file:///C:/hadoop/data/datanode</value>

</property>

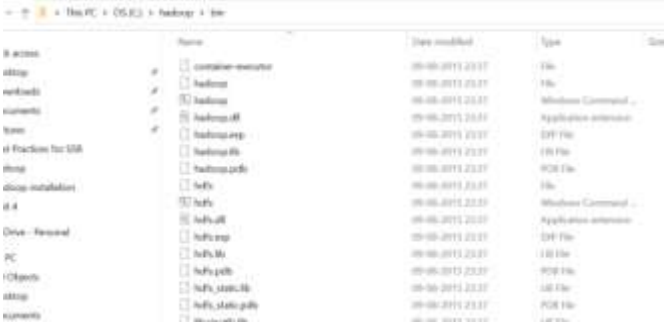
</configuration>

Hadoop-env.xml

- set JAVA_HOME=C:\Java\jdk1.8.0_241
- Set home and path for Hadoop



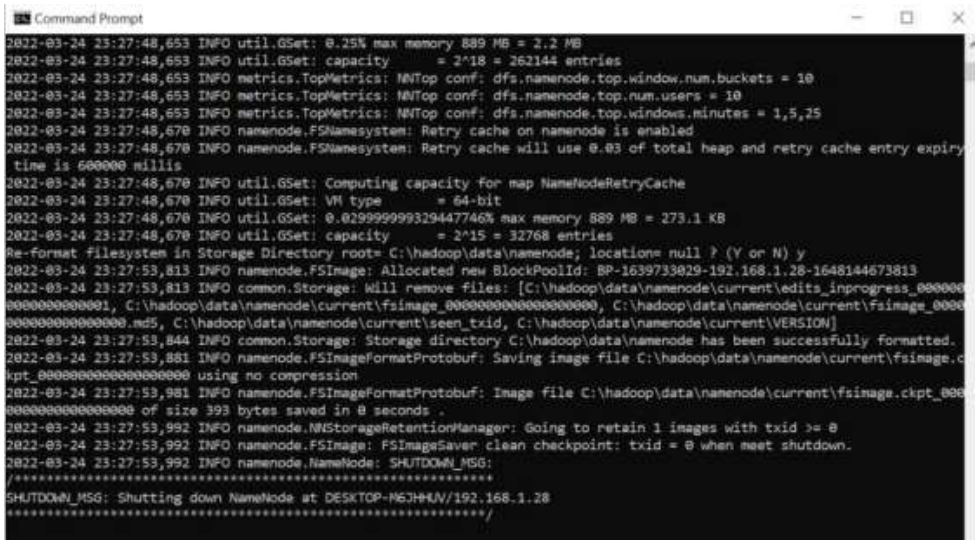
- Other configuration files
- Extract HadoopConfiguration-FIXbin
- Copy bin folder from HadoopConfiguration-Fixbin folder and replace Hadoop\bin folder with this



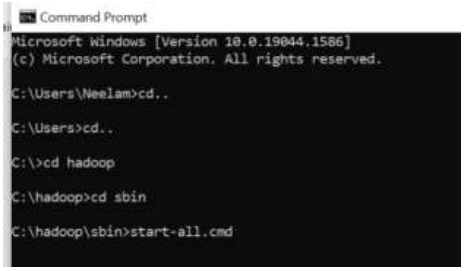
Step 8: Verification of Hadoop installation

- Go to command prompt.

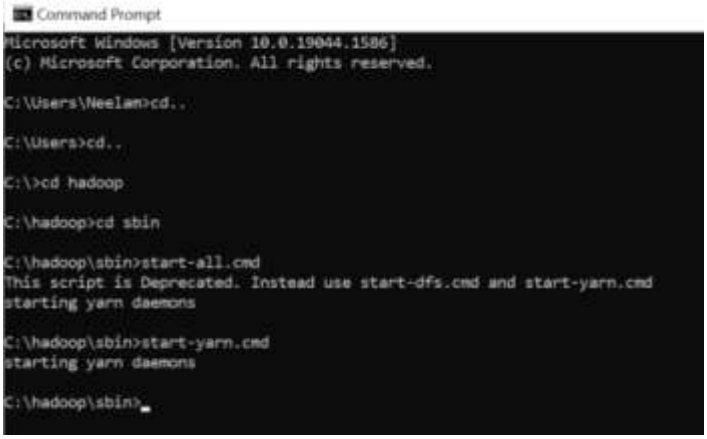
- Type:
- hdfs namenode -format
- set of files pop up on the terminal.
- That means successful installation of Hadoop.



- Open new terminal and start all Hadoop daemons
- Go to Hadoop location.
- C:\hadoop\sbin
- Type the command
- start-all.cmd



- All the nodes will start successfully.
- Also try the following command to start yarn daemons.
- start-yarn.cmd



Practical 9

Aim:- Implement word count / frequency programs using Map Reduce

Code

```
!pip install pyspark
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("WordCount").getOrCreate()
from google.colab import drive
drive.mount("/content/drive")
rdd = spark.sparkContext.textFile("/content/drive/MyDrive/big-p2-wordcount.txt")
words = rdd.flatMap(lambda line: line.split(" ")).map(lambda word: (word,1))
word_counts = words.reduceByKey(lambda a, b: a + b)
for word, count in word_counts.collect():
  print(f"{word}: {count}")
top_5_words = word_counts.sortBy(lambda item: item[1], ascending=False).take(5)
for word, count in top_5_words:
    print(f"{word}: {count}")
spark.stop()
```

Output:-

```
Streaming output truncated to the last 5000 lines.
ridicule,: 2
forfeit!": 3
Nizhni.: 1
Contributions.: 1
Gallicism,": 1
author:: 1
"Quand: 2
regiment,": 1
caustic,: 1
target: 4
unreasonable,: 1
Razumovskis: 1
going:: 1
besides--I: 1
```

```
the: 71744
: 69285
of: 39169
and: 35968
to: 27895
```

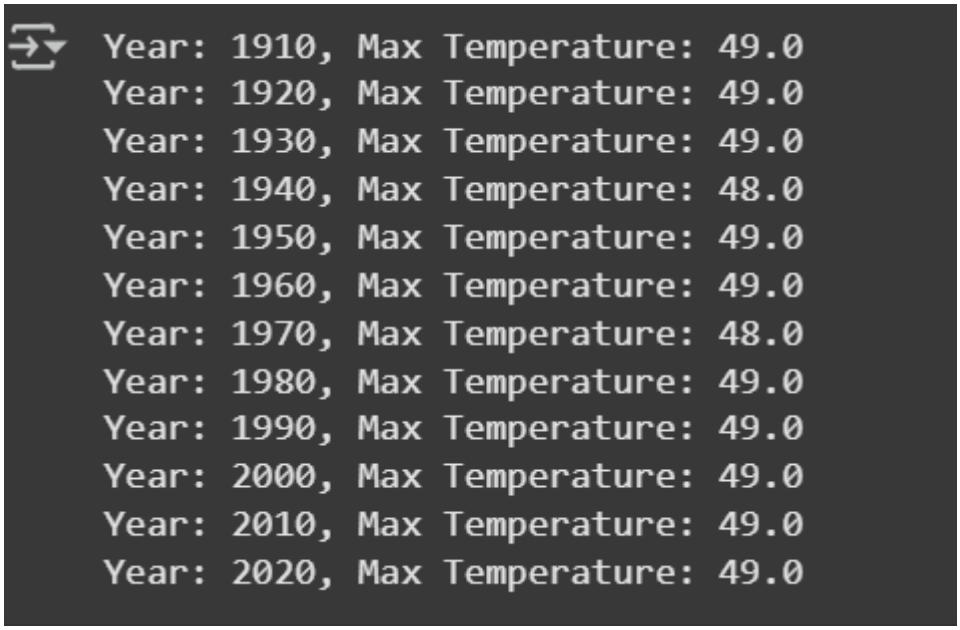
Practical 10

Aim:- Implement Map Reduce program fro finding the highest temperature in a decade.

Code

```
from google.colab import drive
drive.mount("/content/drive")
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("Temperature").getOrCreate()
rdd = spark.sparkContext.textFile("/content/drive/MyDrive/Temperature.txt")
def funct_temp(line):
  parts = line.split(" ")
  if len(parts) >= 2:
    try:
      year = int(parts[0])
      temp = float(parts[1])
      fiveyear = ((year // 10) * 10) + 10
      return (fiveyear, temp)
    except ValueError:
      print(f"Skipping line: {line} - Invalid data format")
      return (None, None)
  else:
    print(f"Skipping line: {line} - Insufficient data")
    return (None, None)
temp_rdd = rdd.map(funct_temp).filter(lambda x: x[0] is not None)
max_temparature = temp_rdd.reduceByKey(lambda x, y: max(x, y))
for year, max_temp in max_temparature.collect():
  print(f"Year: {year}, Max Temperature: {max_temp}")
spark.stop()
```

Output:-


Year: 1910, Max Temperature: 49.0
Year: 1920, Max Temperature: 49.0
Year: 1930, Max Temperature: 49.0
Year: 1940, Max Temperature: 48.0
Year: 1950, Max Temperature: 49.0
Year: 1960, Max Temperature: 49.0
Year: 1970, Max Temperature: 48.0
Year: 1980, Max Temperature: 49.0
Year: 1990, Max Temperature: 49.0
Year: 2000, Max Temperature: 49.0
Year: 2010, Max Temperature: 49.0
Year: 2020, Max Temperature: 49.0

Practical 11

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

Theory: MongoDB is an open-source and the leading NoSQL database. It is a document-oriented database that offers high performance, easy scalability, and high availability. It uses documents and collections to organize data rather than relations. This makes it an ideal database management system for the storage of unstructured data.

Steps of the installation:

Step 1: Download MongoDB

- Go to official website: MongoDB Community server https://www.mongodb.com/try/download/community
- Click on download

Step 2: Install MongoDB.

- It will be the .msi file. Click on it and start the installation.

Step 3: Verify MongoDB Installation

- Now go to C:\Program Files\MongoDB\Server\5.0\bin
- Start command prompt from this location
- To start mongo db server
- Enter mongod command
- It says c:\data\db directory not found
- So create c:\data\db
- To open the mongo shell
- Go to C:\Program Files\MongoDB\Server\5.0\bin Start command prompt from this location
- Fire the command: mongo
- To see all the default databases:
- >show dbs
- To set path of MongoDB server and shell Copy in clipboard:
- C:\Program Files\MongoDB\Server\5.0\bin Go to environment variables Add mongodb path to system path variable

Step 4: Install MongoDB Python on Windows

- We will be performing a few key basic operations on a MongoDB database in Python using the PyMongo library.
- Install package to use MongoDB by running command: conda install -c anaconda pymongo

Step 5: Verify MongoDB Python Connection

- To retrieve the data from a MongoDB database, we will first connect to it. Write and execute the below code in your spider anaconda

- import pymongo mongo_uri = "mongodb://localhost:27017/"
- client = pymongo.MongoClient(mongo_uri)

Aim- mongodb code for creating databse with data of shop name shop location and shop category and then choose to display all the details of a specific location and data to display data of a specific category and also display loaction of area which appears most number of times

```
from pymongo import MongoClient
from collections import Counter

# Connect to MongoDB server (default: localhost:27017)
client = MongoClient("mongodb://localhost:27017/")

# Create or access the database
db = client["ShopDatabase"]

# Create or access the collection
shop_collection = db["Shops"]

# Sample data
shops = [
    {"name": "ABC Supermarket", "location": "New York", "category": "Grocery"},
    {"name": "Tech World", "location": "San Francisco", "category": "Electronics"},
    {"name": "Fashion Hub", "location": "Los Angeles", "category": "Clothing"},
    {"name": "Gadget Zone", "location": "New York", "category": "Electronics"},
    {"name": "Food Mart", "location": "New York", "category": "Grocery"}
]

# Insert data into the collection
shop_collection.insert_many(shops)

print("Database and shop data created successfully!")

# Display shops by location
location = "New York"
results = shop_collection.find({"location": location})
print(f"Shops in {location}:")
for shop in results:
    print(shop)
```

```
# Display shops by category
category = "Electronics"
results = shop_collection.find({"category": category})
print(f"Shops in category {category}:")
for shop in results:
    print(shop)

# Find the most common shop location
locations = [shop["location"] for shop in shop_collection.find({}, {"location": 1, "_id": 0})]
location_counts = Counter(locations)
most_common = location_counts.most_common(1)
if most_common:
    print(f"Most common shop location: {most_common[0][0]} (Count: {most_common[0][1]})")
else:
    print("No locations found.")
```

Output:

```
Database and shop data created successfully!
Shops in New York:
{'_id': ObjectId('67d56112d81b5d90e24441e3'), 'name': 'ABC Supermarket', 'location': 'New York', 'category': 'Grocery'}
{'_id': ObjectId('67d56112d81b5d90e24441e6'), 'name': 'Gadget Zone', 'location': 'New York', 'category': 'Electronics'}
{'_id': ObjectId('67d56112d81b5d90e24441e7'), 'name': 'Food Mart', 'location': 'New York', 'category': 'Grocery'}
Shops in category Electronics:
{'_id': ObjectId('67d56112d81b5d90e24441e4'), 'name': 'Tech World', 'location': 'San Francisco', 'category': 'Electronics'}
{'_id': ObjectId('67d56112d81b5d90e24441e6'), 'name': 'Gadget Zone', 'location': 'New York', 'category': 'Electronics'}
Most common shop location: New York (Count: 3)
```

Aim- create a databse wich collectes the data of what item an person purchased and quantity purchased. With the cost displayed. And the calculate the total money spent

```
from pymongo import MongoClient

# Connect to MongoDB server (default: localhost:27017)
client = MongoClient("mongodb://localhost:27017/")

# Create or access the database
db = client["PurchaseDatabase"]

# Create or access the collection
purchase_collection = db["Purchases"]

# Sample data
purchases = [
    {"person": "John Doe", "items": [
        {"name": "Laptop", "quantity": 1, "cost": 1200},
        {"name": "Mouse", "quantity": 1, "cost": 50}
    ]},
    {"person": "Alice Smith", "items": [
        {"name": "Phone", "quantity": 2, "cost": 800},
        {"name": "Keyboard", "quantity": 1, "cost": 100}
    ]},
    {"person": "Bob Johnson", "items": [
        {"name": "Headphones", "quantity": 3, "cost": 150}
    ]}
]

# Insert data into the collection
purchase_collection.insert_many(purchases)

print("Database and purchase data created successfully!")

# Display all purchase records
results = purchase_collection.find()
print("All Purchase Records:")
for record in results:
    print(record)

# Calculate total purchase amount for each person
print("\nTotal Purchase Amount:")
results = purchase_collection.find()
for record in results:
    total_amount = sum(item["quantity"] * item["cost"] for item in record["items"])
    print(f"{record['person']}: ${total_amount}")
```

Output:

```
Database and purchase data created successfully!
All Purchase Records:
{'_id': ObjectId('67d56413d81b5d90e24441e9'), 'person': 'John Doe', 'items': [{'name': 'Laptop', 'quantity': 1, 'cost': 1200}, {'name': 'Mouse', 'quantity': 1, 'cost': 50}]}
{'_id': ObjectId('67d56413d81b5d90e24441ea'), 'person': 'Alice Smith', 'items': [{'name': 'Phone', 'quantity': 2, 'cost': 800}, {'name': 'Keyboard', 'quantity': 1, 'cost': 100}]}
{'_id': ObjectId('67d56413d81b5d90e24441eb'), 'person': 'Bob Johnson', 'items': [{'name': 'Headphones', 'quantity': 3, 'cost': 150}]}

Total Purchase Amount:
John Doe: $1250
Alice Smith: $1700
Bob Johnson: $450
```