CREATING ROUTER USING LINUX NAMESPACE

A comprehensive guide

Abstract

Utilize the power of one of the Linux Namespaces to create an isolated network space in Linux to act as a router using iptables and route rules

Rezuanul Hoque

hoque.shamil@gmail.com

+8801860241352

Table of Contents

Objective 1

Scope of Work 1

Prerequisites 1

Solution Diagram 2

Implementation 2

- **IP Address Scheme Table** 3
- **Routing Scheme Table** 3
- **Create Network Namespaces** 3
- **Create and Configure Bridges** 4
  - **For br0 bridge** 4
  - **For br1 bridge** 5
- **Create and configure veth cables** 6
  - **ns1_br0 -- br0_ns1** 6
  - **br0_rt -- rt_br0** 7
  - **rt_br1 -- br1_rt** 8
  - **br1_ns2 -- ns2_br1** 10
- **Assign Forwarding Rules** 11
  - **In root namespace** 11
  - **In ns1 namespace** 12
  - **In ns2 namespace** 13
  - **In router namespace** 13

Testing Connections 14

- **From ns1:** 14
- **From ns2:** 15

Bonus Challenge: Bash script and systemd file 15

- **Bash script file** 16
- **Systemd File** 18

Clean Up Function as Command 19

Objective

The objective of this project is to leverage Linux namespaces to create an isolated network namespace that functions as a router between two other network namespaces. This will involve setting up a network bridge and using virtual Ethernet (veth) pairs to establish connectivity between the namespaces. The project aims to demonstrate network isolation, routing, and inter-namespace communication using Linux networking primitives, providing a practical understanding of namespace-based networking and its applications.

Scope of Work

- Create three network namespaces: **ns1, ns2, router-ns**
- Create two bridges: **br0, br1**
- Create four pairs of veth cable: **(ns1_br0 -- br0_ns1), (br0_rt -- rt_br0), (rt_br1 -- br1_rt), (br1_ns2 -- ns2_br1)**
- Define Iptables forward rules and route rules in both root and router-ns

Prerequisites

- A computing machine with Linux operating system
- Sudo access to run the commands

Solution Diagram

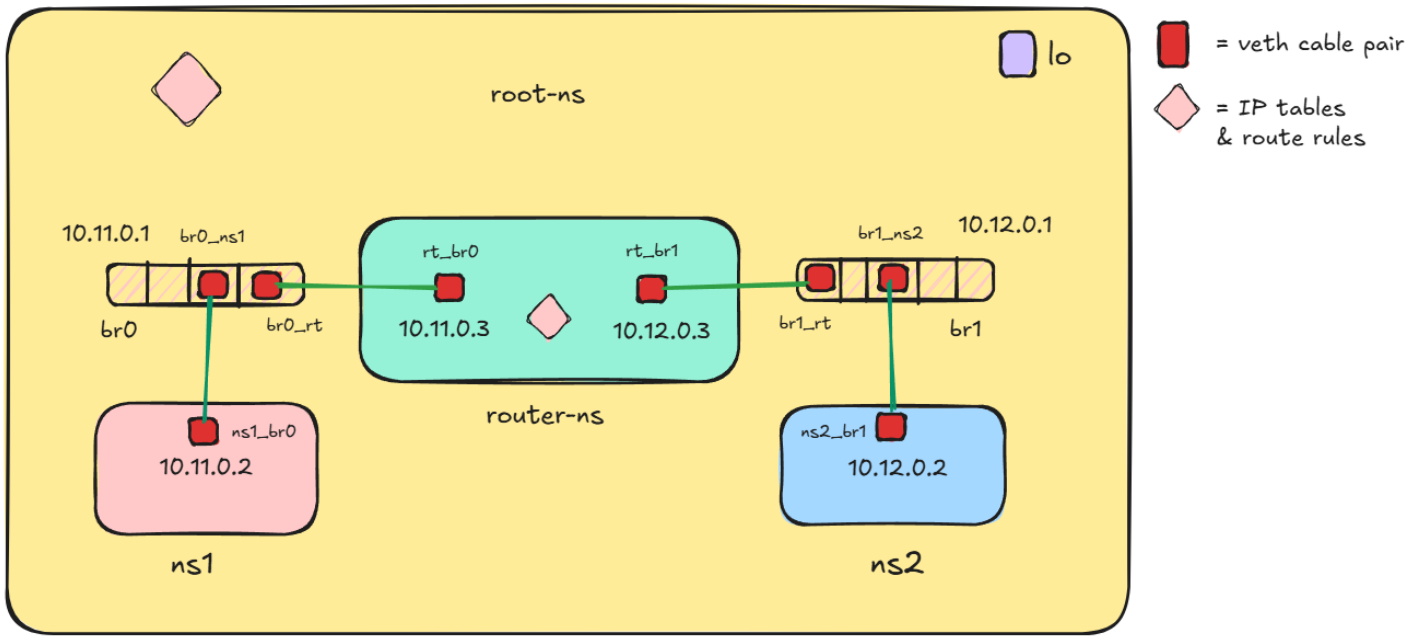


Figure 1.0: Solution Diagram of Proposed Model

Implementation

Turn on the Linux machine and attain privilege access mode to run the commands. Please refer to the IP addressing scheme table and routing scheme table below to configure setup:

IP Address Scheme Table

Components	IP addresses
br0	10.11.0.1/16
br1	10.12.0.1/16
ns1_br0	10.11.0.2/16
rt_br0	10.11.0.3/16
rt_br1	10.12.0.3/16
ns2_br1	10.12.0.2/16

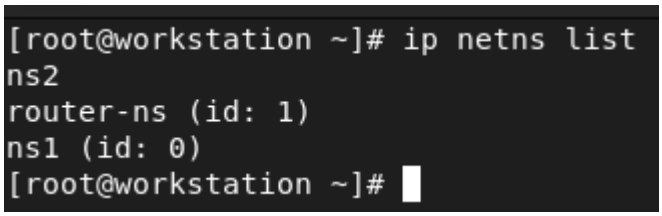
Routing Scheme Table

Network	Gateway
ns1	10.11.0.3
ns2	10.12.0.3

Create Network Namespaces

Run the following commands in the terminal:

```
#ip netns add ns1

#ip netns add ns2

#ip netns add router-ns

#ip netns list
```

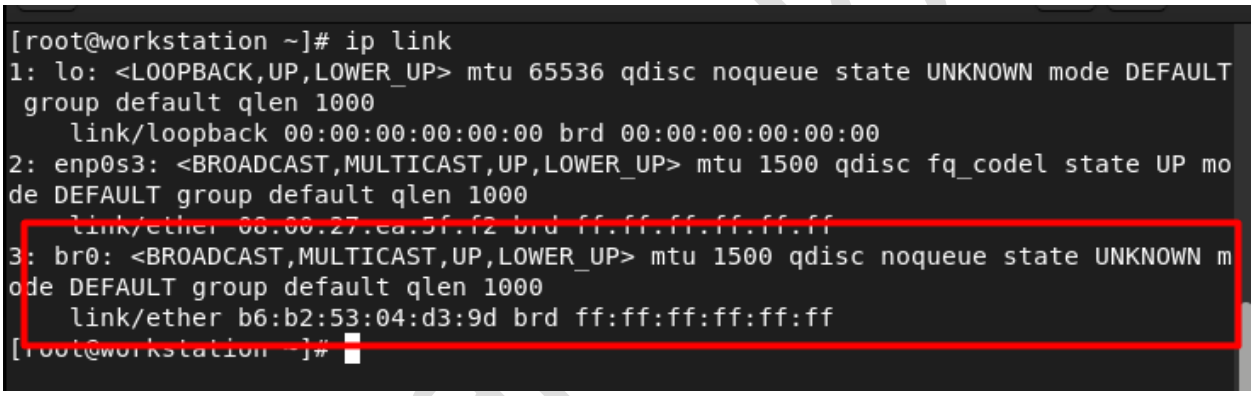


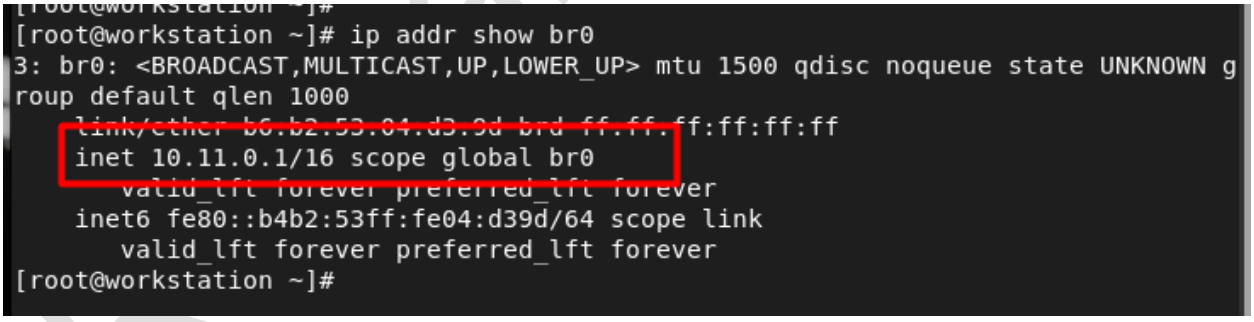
Create and Configure Bridges

Run the following commands in the terminal:

For br0 bridge

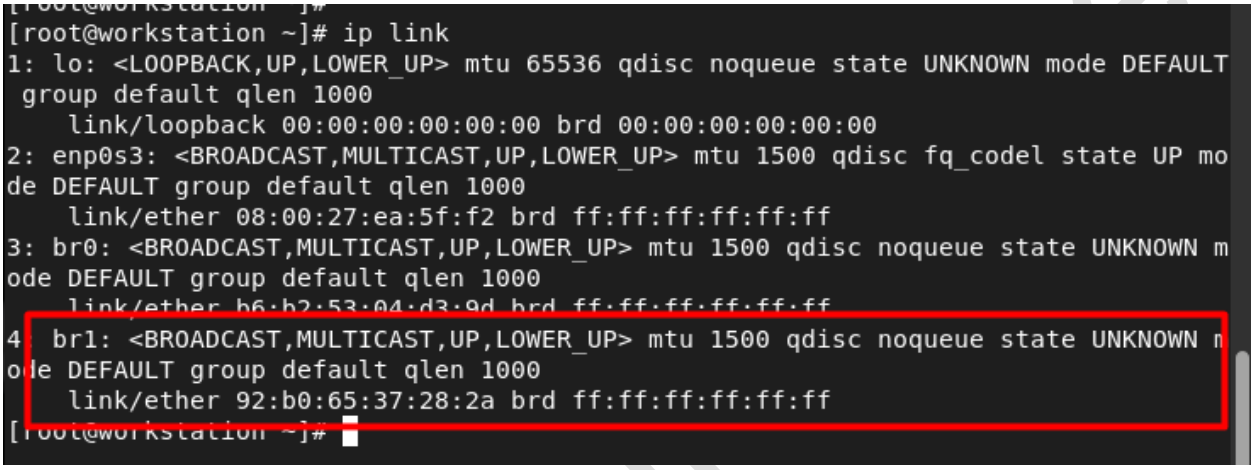
```
#ip link add br0 type bridge

#ip addr add 10.11.0.1/16 dev br0

#ip link set br0 up
```

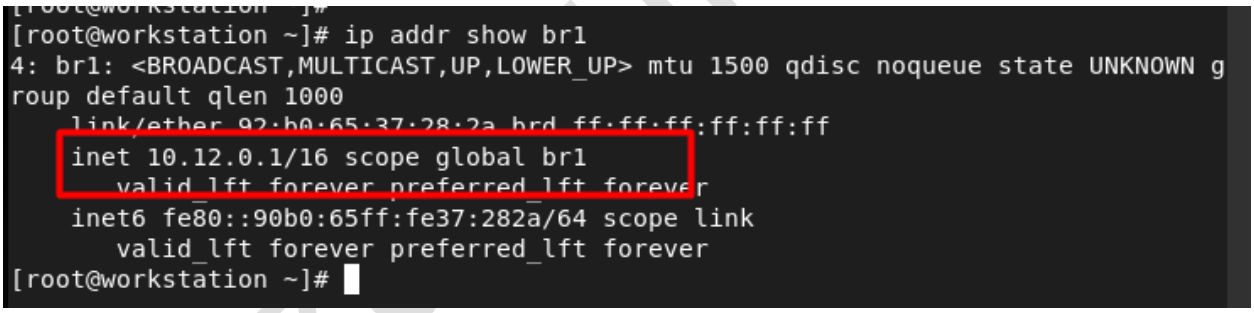




For br1 bridge

```
#ip link add br1 type bridge

#ip addr add 10.12.0.1/16 dev br0

#ip link set br1 up
```



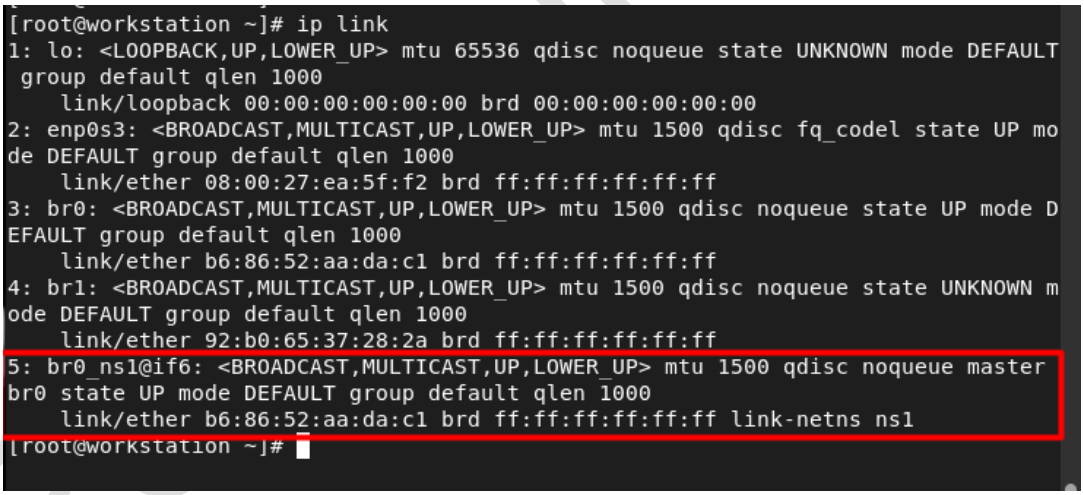


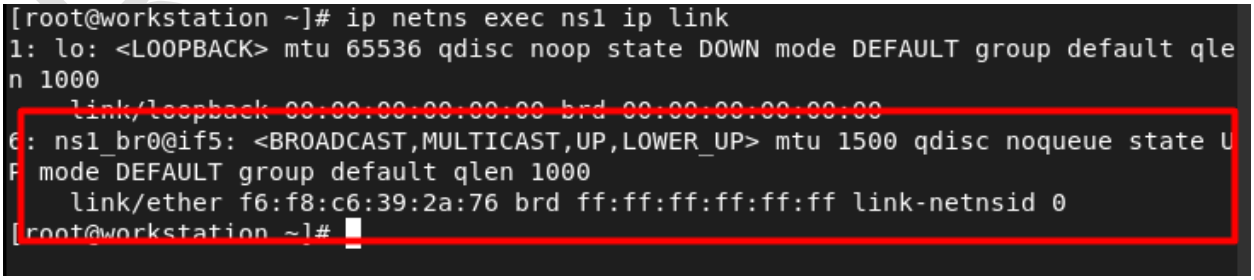
Create and configure veth cables

Run the following commands in the terminal:

ns1_br0 -- br0_ns1

```
#ip link add ns1_br0 type veth peer name br0_ns1

#ip link set br0_ns1 up

#ip link set br0_ns1 master br0

#ip link set ns1_br0 netns ns1

#ip netns exec ns1 ip addr add 10.11.0.2/16 dev ns1_br0

#ip netns exec ns1 ip link set ns1_br0 up
```





```
[root@workstation ~]# ip netns exec ns1 ip addr show ns1_br0
6: ns1_br0@if5: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state U
P group default qlen 1000
    link/ether f6:f8:c6:39:2a:76 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.11.0.2/16 scope global ns1_br0
       valid_lft forever preferred_lft forever
    inet6 fe80::f4f8:c6ff:fe39:2a76/64 scope link
       valid_lft forever preferred_lft forever
[root@workstation ~]#
```

br0_rt -- rt_br0

#ip link add br0_rt type veth peer name rt_br0

#ip link set br0_rt up

#ip link set br0_rt master br0

#ip link set rt_br0 netns router-ns

#ip netns exec router-ns ip addr add 10.11.0.3/16 dev rt_br0

#ip netns exec router-ns ip link set rt_br0 up

```
[root@workstation ~]# ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
 group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mo
de DEFAULT group default qlen 1000
    link/ether 08:00:27:ea:5f:f2 brd ff:ff:ff:ff:ff:ff
3: br0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode D
EFAULT group default qlen 1000
    link/ether 5e:e9:9d:0b:27:98 brd ff:ff:ff:ff:ff:ff
4: br1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN m
ode DEFAULT group default qlen 1000
    link/ether 92:b0:65:37:28:2a brd ff:ff:ff:ff:ff:ff
5: br0_ns1@if6: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master
br0 state UP mode DEFAULT group default qlen 1000
    link/ether b6:86:52:aa:da:c1 brd ff:ff:ff:ff:ff:ff link-netns ns1
8: br0_rt@if7: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master b
r0 state UP mode DEFAULT group default qlen 1000
    link/ether 5e:e9:9d:0b:27:98 brd ff:ff:ff:ff:ff:ff link-netns router-ns
[root@workstation ~]#
```

```
[root@workstation ~]# ip netns exec router-ns ip link
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN mode DEFAULT group default qle
n 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
7: rt_br0@if8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
 mode DEFAULT group default qlen 1000
    link/ether 9e:c1:ef:ea:34:c3 brd ff:ff:ff:ff:ff:ff link-netnsid 0
[root@workstation ~]#
```

```
[root@workstation ~]# ip netns exec router-ns ip addr show rt_br0
7: rt_br0@if8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
 group default qlen 1000
    link/ether 9e:c1:ef:ea:34:c3 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.11.0.3/16 scope global rt_br0
       valid_lft forever preferred_lft forever
    inet6 fe80::9cc1:efff:feea:34c3/64 scope link
       valid_lft forever preferred_lft forever
[root@workstation ~]#
```

rt_br1 -- br1_rt

#ip link add rt_br1 type veth peer name br1_rt

#ip link set br1_rt up

#ip link set br1_rt master br1

#ip link set rt_br1 netns router-ns

#ip netns exec router-ns ip addr add 10.12.0.3/16 dev rt_br1

#ip netns exec router-ns ip link set rt_br1 up

```
[root@workstation ~]# ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
 group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mo
de DEFAULT group default qlen 1000
    link/ether 08:00:27:ea:5f:f2 brd ff:ff:ff:ff:ff:ff
3: br0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode D
EFAULT group default qlen 1000
    link/ether 5e:e9:9d:0b:27:98 brd ff:ff:ff:ff:ff:ff
4: br1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode D
EFAULT group default qlen 1000
    link/ether a2:5f:7b:fe:80:e8 brd ff:ff:ff:ff:ff:ff
5: br0_ns1@if6: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master
br0 state UP mode DEFAULT group default qlen 1000
    link/ether b6:86:52:aa:da:c1 brd ff:ff:ff:ff:ff:ff link-netns ns1
8: br0_rt@if7: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master b
r0 state UP mode DEFAULT group default qlen 1000
    link/ether 5e:e9:9d:0b:27:98 brd ff:ff:ff:ff:ff:ff link-netns router-ns
9: br1_rt@if10: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master
br1 state UP mode DEFAULT group default qlen 1000
    link/ether a2:5f:7b:fe:80:e8 brd ff:ff:ff:ff:ff:ff link-netns router-ns
[root@workstation ~]#
```

```
[root@workstation ~]# ip netns exec router-ns ip link
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN mode DEFAULT group default qle
n 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
7: rt_br0@if8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
 mode DEFAULT group default qlen 1000
    link/ether 9e:c1:ef:ea:34:c3 brd ff:ff:ff:ff:ff:ff link-netnsid 0
10: rt_br1@if9: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state U
P mode DEFAULT group default qlen 1000
    link/ether 42:1e:c3:91:4d:53 brd ff:ff:ff:ff:ff:ff link-netnsid 0
[root@workstation ~]#
```

```
[root@workstation ~]# ip netns exec router-ns ip addr show rt_br1
10: rt_br1@if9: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state U
P group default qlen 1000
    link/ether 42:1e:c3:91:4d:53 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.12.0.3/16 scope global rt_br1
       valid_lft forever preferred_lft forever
    inet6 fe80::401e:c3ff:fe91:4d53/64 scope link
       valid_lft forever preferred_lft forever
[root@workstation ~]#
```

br1_ns2 -- ns2_br1

```
#ip link add br1_ns2 type veth peer name ns2_br1

#ip link set br1_ns2 up

#ip link set br1_ns2 master br1

#ip link set ns2_br1 netns ns2

#ip netns exec ns2 ip addr add 10.12.0.2/16 dev ns2_br1

#ip netns exec ns2 ip link set ns2_br1 up
```

```
5: br0_ns1@if6: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master
br0 state UP mode DEFAULT group default qlen 1000
    link/ether b6:86:52:aa:da:c1 brd ff:ff:ff:ff:ff:ff link-netns ns1
8: br0_rt@if7: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master b
r0 state UP mode DEFAULT group default qlen 1000
    link/ether 5e:e9:9d:0b:27:98 brd ff:ff:ff:ff:ff:ff link-netns router-ns
9: br1_rt@if10: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master
br1 state UP mode DEFAULT group default qlen 1000
12: br1_ns2@if11: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue maste
r br1 state UP mode DEFAULT group default qlen 1000
    link/ether f2:24:8c:d7:b3:a1 brd ff:ff:ff:ff:ff:ff link-netns ns2
[root@workstation ~]#
```

```
[root@workstation ~]# ip netns exec ns2 ip link
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN mode DEFAULT group default qle
n 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
11: ns2_br1@if12: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
 UP mode DEFAULT group default qlen 1000
    link/ether de:31:f4:c2:de:20 brd ff:ff:ff:ff:ff:ff link-netnsid 0
[root@workstation ~]#
```

```
[root@workstation ~]# ip netns exec ns2 ip addr show ns2_br1
11: ns2_br1@if12: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
 UP group default qlen 1000
    link/ether de:31:f4:c2:de:20 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.12.0.2/16 scope global ns2_br1
       valid_lft forever preferred_lft forever
    inet6 fe80::dc31:f4ff:fec2:de20/64 scope link
       valid_lft forever preferred_lft forever
[root@workstation ~]#
```

Assign Forwarding Rules

Run the following commands in the terminal:

In root namespace

#sysctl -w net.ipv4.ip_forward=1

```
[root@workstation ~]# sysctl -w net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
[root@workstation ~]#
```

#iptables --append FORWARD --in-interface br0 --jump ACCEPT

#iptables --append FORWARD --out-interface bro --jump ACCEPT

```
[root@workstation ~]#
[root@workstation ~]# iptables -L FORWARD -n -v
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

    0     0 ACCEPT     all  --  br0    *       0.0.0.0/0            0.0.0.0/0

    0     0 ACCEPT     all  --  *      br0     0.0.0.0/0            0.0.0.0/0

[root@workstation ~]#
```

#iptables --append FORWARD --in-interface br1 --jump ACCEPT

#iptables --append FORWARD --out-interface br1 --jump ACCEPT

```
[root@workstation ~]# iptables -L FORWARD -n -v
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

    0     0 ACCEPT     all  --  br0    *       0.0.0.0/0            0.0.0.0/0

    0     0 ACCEPT     all  --  *      br0     0.0.0.0/0            0.0.0.0/0

    0     0 ACCEPT     all  --  br1    *       0.0.0.0/0            0.0.0.0/0

    0     0 ACCEPT     all  --  *      br1     0.0.0.0/0            0.0.0.0/0

[root@workstation ~]#
```

In ns1 namespace

Let us confirm that the gateway is reachable from the ns1 namespace

#ip netns exec ns1 ping 10.11.0.3 -c 3

```
[root@workstation ~]# ip netns exec ns1 ping 10.11.0.3 -c 3
PING 10.11.0.3 (10.11.0.3) 56(84) bytes of data.
64 bytes from 10.11.0.3: icmp_seq=1 ttl=64 time=1.54 ms
64 bytes from 10.11.0.3: icmp_seq=2 ttl=64 time=0.107 ms
64 bytes from 10.11.0.3: icmp_seq=3 ttl=64 time=0.106 ms

--- 10.11.0.3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2045ms
rtt min/avg/max/mdev = 0.106/0.582/1.535/0.673 ms
[root@workstation ~]#
```

Now add the default gateway route for ns1 namespace:

#ip netns exec ns1 ip route add default via 10.11.0.3

```
[root@workstation ~]# ip netns exec ns1 ip route add default via 10.11.0.3
[root@workstation ~]# ip netns exec ns1 ip route
default via 10.11.0.3 dev ns1_br0
10.11.0.0/16 dev ns1_br0 proto kernel scope link src 10.11.0.2
[root@workstation ~]#
```

In ns2 namespace

Let us confirm that the gateway is reachable from the ns1 namespace

#ip netns exec ns2 ping 10.12.0.3 -c 3

```
[root@workstation ~]# ip netns exec ns2 ping 10.12.0.3 -c 3
PING 10.12.0.3 (10.12.0.3) 56(84) bytes of data.
64 bytes from 10.12.0.3: icmp_seq=1 ttl=64 time=1.39 ms
64 bytes from 10.12.0.3: icmp_seq=2 ttl=64 time=0.137 ms
64 bytes from 10.12.0.3: icmp_seq=3 ttl=64 time=0.103 ms

--- 10.12.0.3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2026ms
rtt min/avg/max/mdev = 0.103/0.541/1.385/0.596 ms
[root@workstation ~]#
```

Now add the default gateway route for ns2 namespace:

#ip netns exec ns2 ip route add default via 10.12.0.3

```
[root@workstation ~]# ip netns exec ns2 ip route add default via 10.12.0.3
[root@workstation ~]# ip netns exec ns2 ip route
default via 10.12.0.3 dev ns2_br1
10.12.0.0/16 dev ns2_br1 proto kernel scope link src 10.12.0.2
[root@workstation ~]#
```

In router namespace

The routing rules for forwarding traffic to specific networks has been automatically got created in the router-ns route table:

```
[root@workstation ~]# ip netns exec router-ns ip route
10.11.0.0/16 dev rt_br0 proto kernel scope link src 10.11.0.3
10.12.0.0/16 dev rt_br1 proto kernel scope link src 10.12.0.3
[root@workstation ~]#
```

But we have to add the ipv4 traffic forwarding rule to the router-ns:

```
#ip netns exec router-ns sysctl -w net.ipv4.ip_forward=1
```

```
[root@workstation ~]# ip netns exec router-ns sysctl -w net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
```

Testing Connections

Let us now test the ping connectivity from ns1 to ns2 and vice versa:

(Unit testing of reachability to respective gateways has been done previously when we set the default gateway for both of the namespaces. Here we are going to test the integration testing)

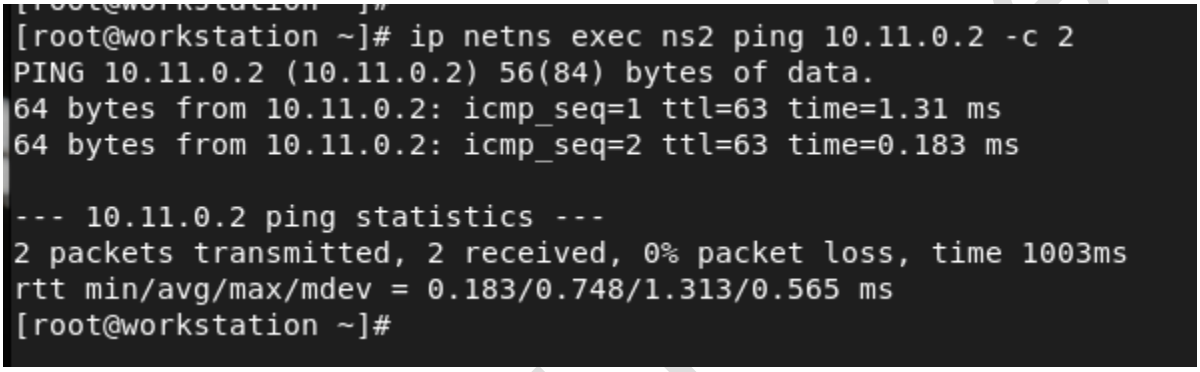
From ns1:

```
#ip netns exec ns1 ping 10.12.0.2 -c 2
```

```
[root@workstation ~]# ip netns exec ns1 ping 10.12.0.2 -c 2
PING 10.12.0.2 (10.12.0.2) 56(84) bytes of data.
64 bytes from 10.12.0.2: icmp_seq=1 ttl=63 time=2.11 ms
64 bytes from 10.12.0.2: icmp_seq=2 ttl=63 time=0.126 ms

--- 10.12.0.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1008ms
rtt min/avg/max/mdev = 0.126/1.115/2.105/0.989 ms
[root@workstation ~]#
```

From ns2:

#ip netns exec ns2 ping 10.11.0.2 -c 2



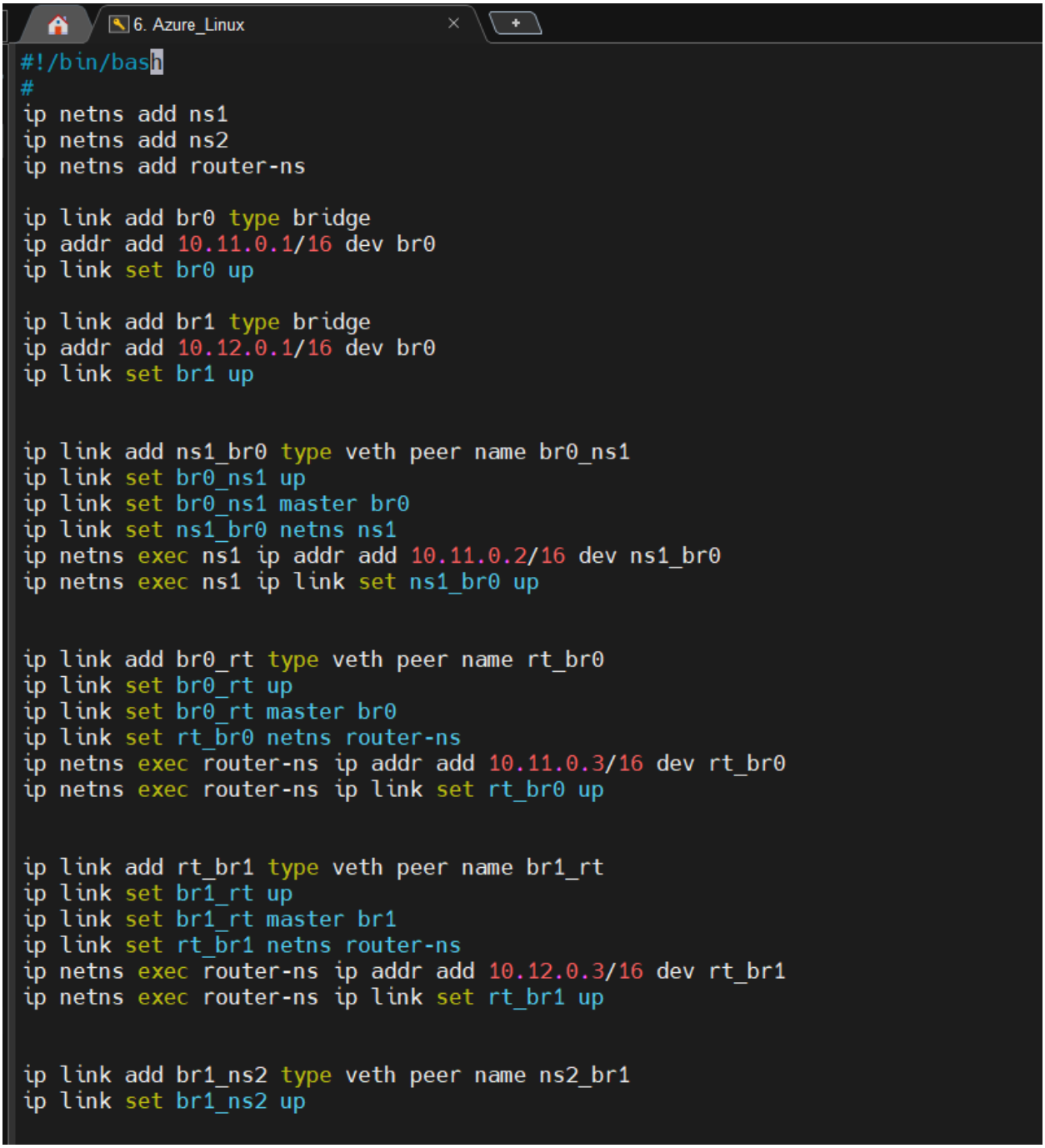
Bonus Challenge: Bash script and systemd file

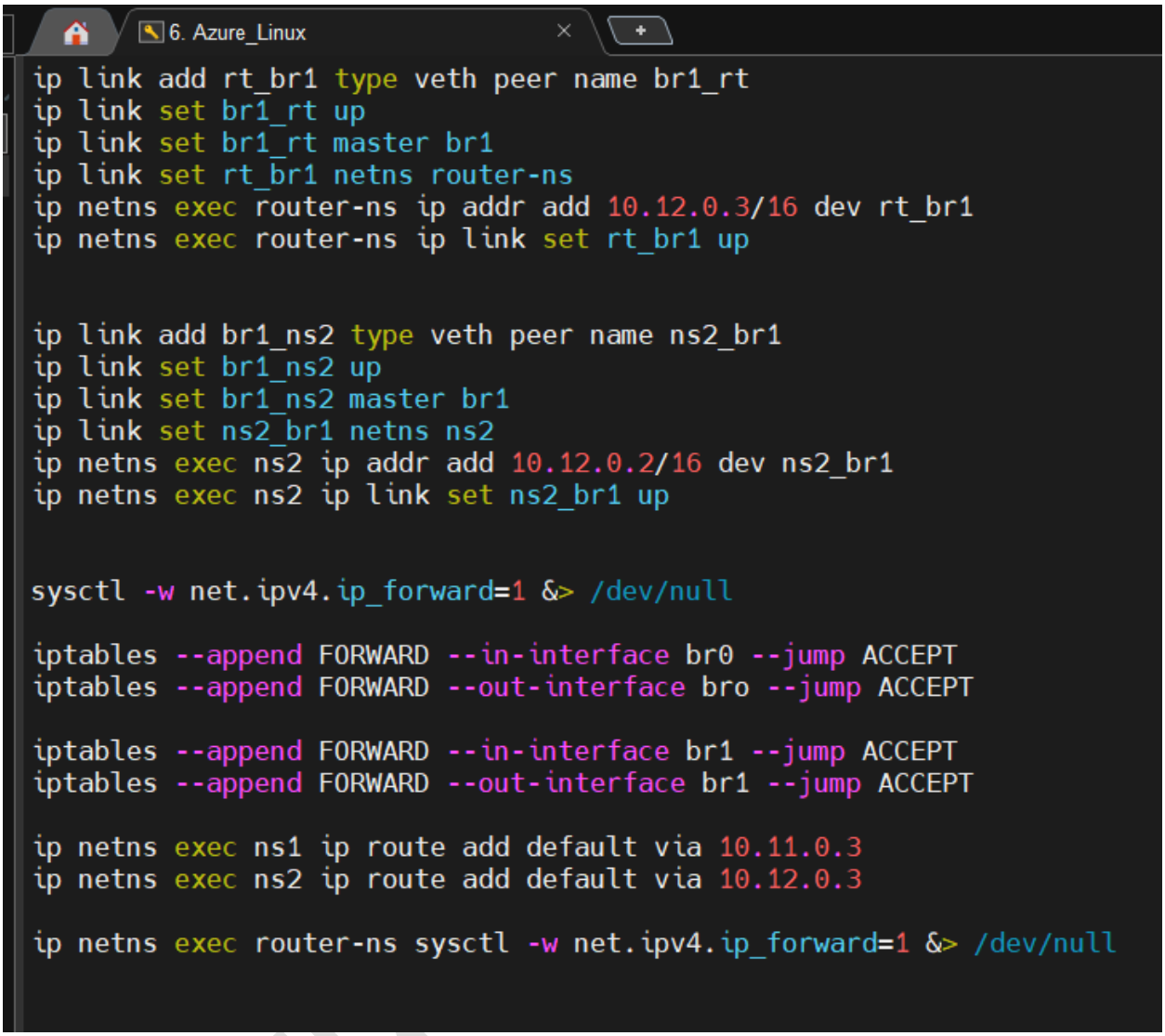
As Linux Namespaces are stored in memory, every time we boot up or reboot the system, all the configurations will get wiped out. To solve this problem, we will create:

- A bash script containing all the commands
- A systemd service file to run the script file automatically when the system gets online

Bash script file

Add the all commands to a file and make it executable with chmod +x file_name.sh:




6. Azure_Linux
ip link add rt_br1 type veth peer name br1_rt
ip link set br1_rt up
ip link set br1_rt master br1
ip link set rt_br1 netns router-ns
ip netns exec router-ns ip addr add 10.12.0.3/16 dev rt_br1
ip netns exec router-ns ip link set rt_br1 up
ip link add br1_ns2 type veth peer name ns2_br1
ip link set br1_ns2 up
ip link set br1_ns2 master br1
ip link set ns2_br1 netns ns2
ip netns exec ns2 ip addr add 10.12.0.2/16 dev ns2_br1
ip netns exec ns2 ip link set ns2_br1 up
sysctl -w net.ipv4.ip_forward=1 &> /dev/null
iptables --append FORWARD --in-interface br0 --jump ACCEPT
iptables --append FORWARD --out-interface bro --jump ACCEPT
iptables --append FORWARD --in-interface br1 --jump ACCEPT
iptables --append FORWARD --out-interface br1 --jump ACCEPT
ip netns exec ns1 ip route add default via 10.11.0.3
ip netns exec ns2 ip route add default via 10.12.0.3
ip netns exec router-ns sysctl -w net.ipv4.ip_forward=1 &> /dev/null

Systemd File

Move the script file to /usr/local/bin directory using mv or cp command

```
root@my-linux:~# ls /usr/local/bin/
router_ns.sh
root@my-linux:~#
```

Go to /etc/systemd/system/ directory and create a service file using any text editor:

```
6. Azure_Linux
root@my-linux:~# cd /etc/systemd/system/
root@my-linux:/etc/systemd/system# ls | grep -i route
router_ns.service
root@my-linux:/etc/systemd/system#
```

Add the following systemd configurations in the file:

```
6. Azure_Linux
[Unit]
Description=Network Namespace Router Setup
After=network.target

[Service]
Type=simple
ExecStart=/usr/local/bin/router_ns.sh
Restart=always
User=root

[Install]
WantedBy=multi-user.target
~
```

Now run these commands to enable the new systemd service file:

```
#systemctl daemon-reload
#systemctl enable router_ns.service
#systemctl start router_ns.service
```

Note: *If you run the systemctl status command it might show error as the namespaces and IP addresses are already assigned. When you do a reboot of the system, the service file will automatically run the script and your namespaces will get created with all the configurations.*

Clean Up Function as Command

Open /etc/bashrc with a text editor and append this script at the end of the file:

```
fi
# vim:ts=4:sw=4

cleanup()
{
        for ns in $(ip netns list | awk '{print $1}'); do
        echo "Deleting the ns "$ns""
        sudo ip netns del "$ns"
done


# delete bridges
for br in $(ip -br link show type bridge | awk '{print $1}'); do
        echo "Deleting the bridges "$br""
        sudo ip link set "$br" down
        sudo ip link del "$br"
done

echo "Clean up complete"
}
```

After adding the script type:

```
#source /etc/bashrc

#cleanup
```

(This will declare ‘cleanup’ as a command for the Linux system to delete all the network namespaces and bridges)

```
[root@workstation ~]# source /etc/bashrc
[root@workstation ~]# cleanup
Deleting the ns router-ns
Deleting the ns ns2
Deleting the ns ns1
Deleting the bridges br0
Deleting the bridges br1
Clean up complete
[root@workstation ~]#
```