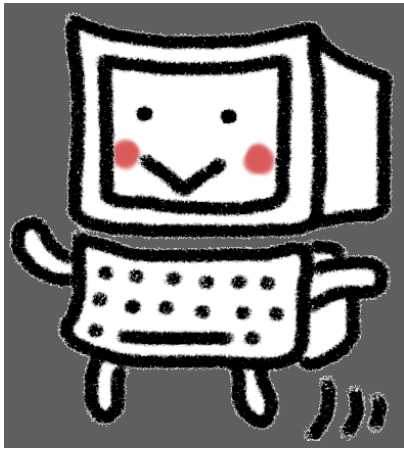


Project 5

VNF – Virtual Router

Date: 2020/04/30

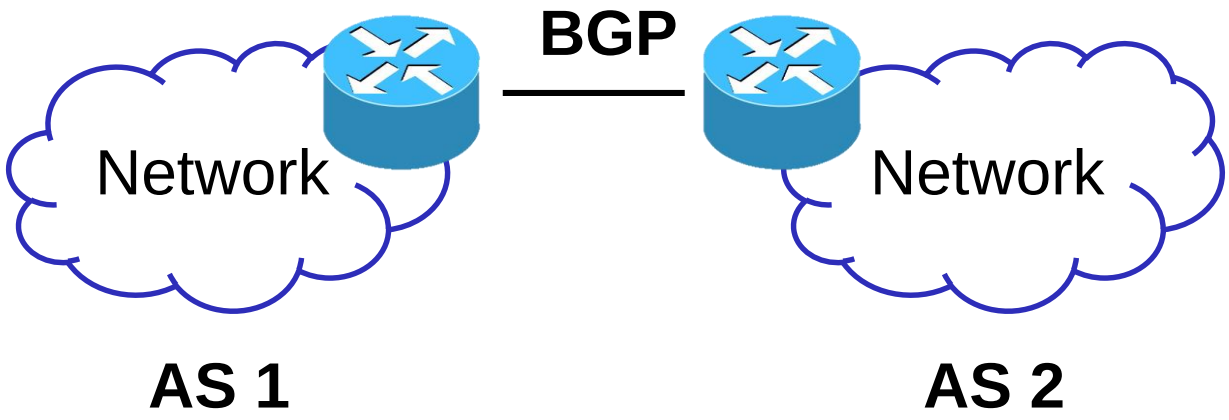
Deadline: 2020/05/21

Outline

- Scenario
- Quagga Introduction
- Docker Introduction
- Environment Setup
- Target Topology
- Submit to e3
- References

Scenario

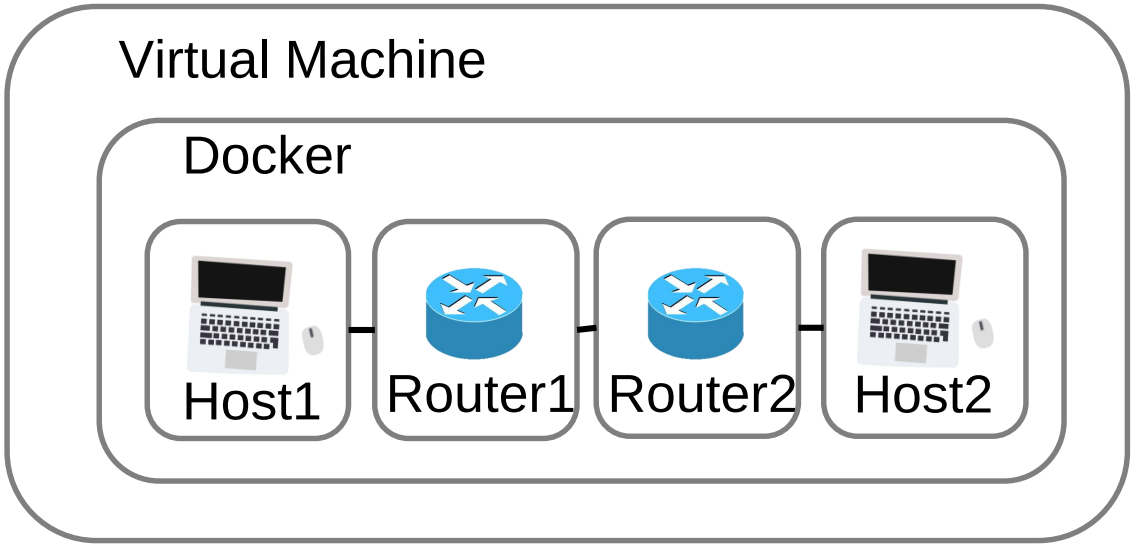
- Interconnection of two networks



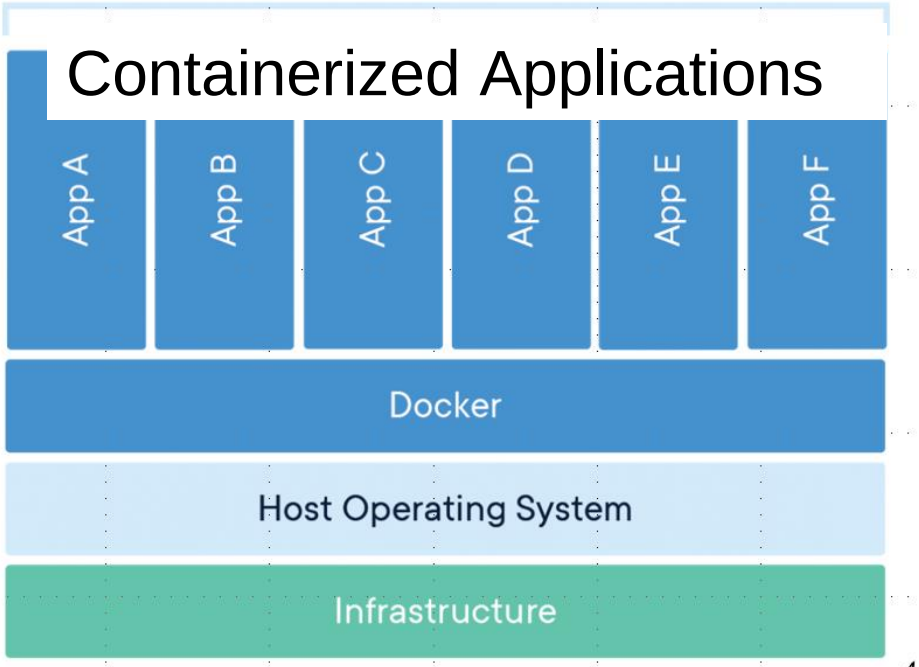
- BGP: Border Gateway Protocol
- AS: Autonomous System

Environment

- VirtualBox
  - Ubuntu Desktop 16.04



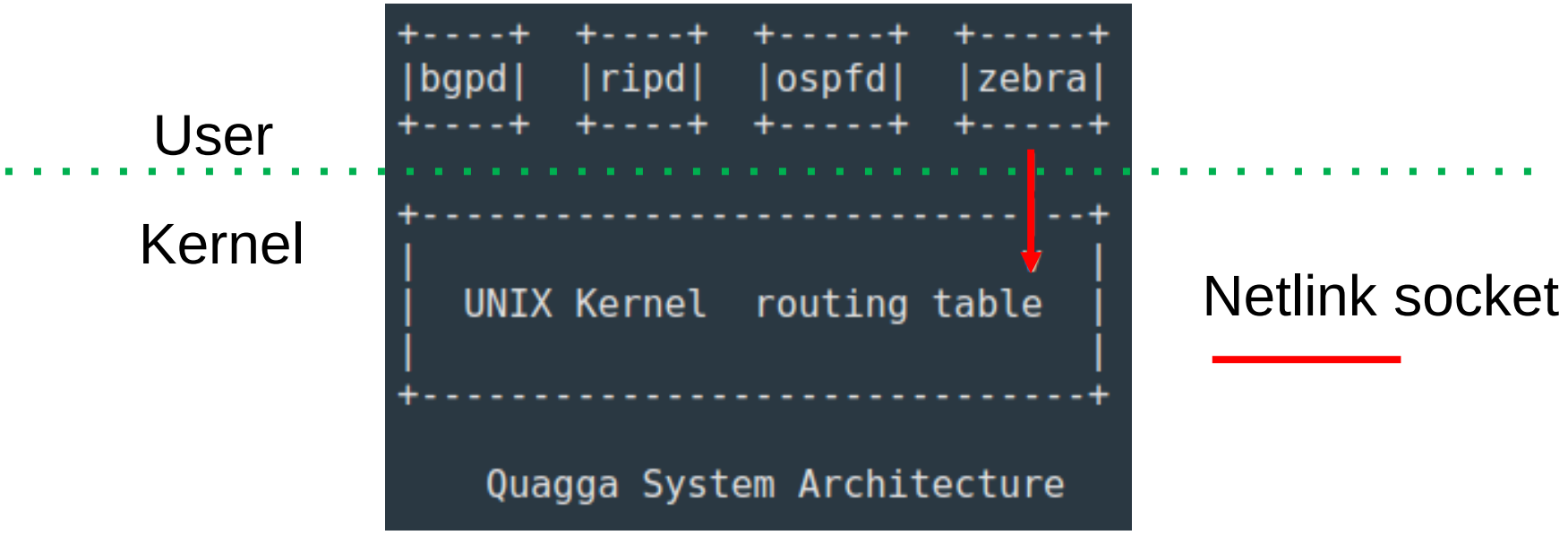
- Use Docker to build Container in VM



4

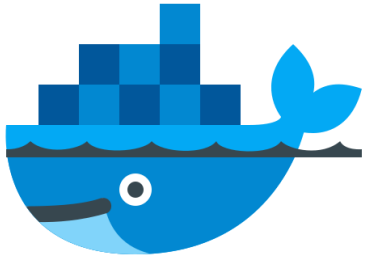
Introduction of Quagga

- Quagga is an open source software that provides routing services
  - Supports common routing protocols
    - BGP, OSPF, RIP and IS-IS
- Choose the routing protocol
- Routing protocol will modify the kernel routing table through the kernel routing manager zebra daemon



Introduction of Docker

- Three Basic Concepts of Docker
  - Built Docker images of the desired OS and applications
  - Store the images in Docker registry
    - Public (Docker Hub)
    - Private
  - Run Docker to build containers of images



Installation of Docker

- Update apt

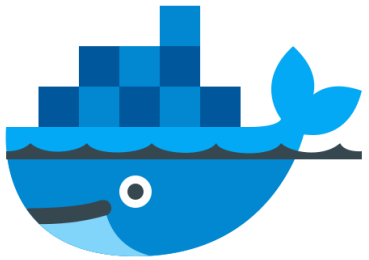
```
~$ sudo apt-get update # Confirm to install the latest package
```

- Install curl to transfer data

```
~$ sudo apt-get install -y curl
```

- Get Docker installation script and install Docker

```
~$ sudo curl –ssl https://get.docker.com/ | sh
```



Pull Image from Docker Hub

- Usage

```
~$ sudo docker pull NAME[:TAG]
```

- Pull an image from the Docker Hub registry.

```
~$ sudo docker pull ubuntu:16.04
```

- List images

```
~$ sudo docker images
```

```
lepg5487@lepg5487-Aspire-VN7-791G:~$ sudo docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu              16.04               77be327e4b63        8 weeks ago         124MB
```

Commands – Create a Docker Container of an Image

- Usage

```
~$ sudo docker run [OPTIONS] IMAGE[:TAG]
```

- Create container

```
~$ sudo docker run -d -it --name test ubuntu:16.04
```

  - -d: Detached (like a daemon in background)
  - -it: Interactive processes (like a shell)
  - --name: Assign a name to the container

- List containers

```
~$ sudo docker ps –a
```

  - “--all”,“-a“: Show all containers

```
lepg5487@lepg5487-Aspire-VN7-791G:~$ sudo docker run -d -it --name test ubuntu:16.04
e36a60782047ba37a8d063176228641ec5e50d70923d0182e9fa60ec067f8566
lepg5487@lepg5487-Aspire-VN7-791G:~$ sudo docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
e36a60782047        ubuntu:16.04        "/bin/bash"         5 seconds ago       Up 4 seconds                            test
```

Commands – Activate a Container and Enter Command Mode

- Usage

```
~$ sudo docker exec [OPTIONS] CONTAINER COMMAND
```

- Run a test container and use bash to enter the command mode of the container

```
~$ sudo docker exec -it test bash
```

```
lepg5487@lepg5487-Aspire-VN7-791G:~$ sudo docker exec -it test bash
root@e36a60782047:/#
```

Commands – Create a Docker Network

- Usage

```
~$ sudo docker network create [OPTIONS] NETWORK
```

- Create a docker bridge: e.g., `testbr`

```
~$ sudo docker network create testbr
```

  - testbr: Bridge name

- List networks

```
~$ sudo docker network ls
```

```
lepg5487@lepg5487-Aspire-VN7-791G:~$ sudo docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
a93c207acfee        bridge              bridge              local
097f2d9b795d        host                host                local
30ea76858491        none                null                local
a117784e1c61        testbr              bridge              local
lepg5487@lepg5487-Aspire-VN7-791G:~$
```

Commands – Connect a Container to a Docker Network

- Usage

```
~$ sudo docker network connect NETWORK CONTAINER
```

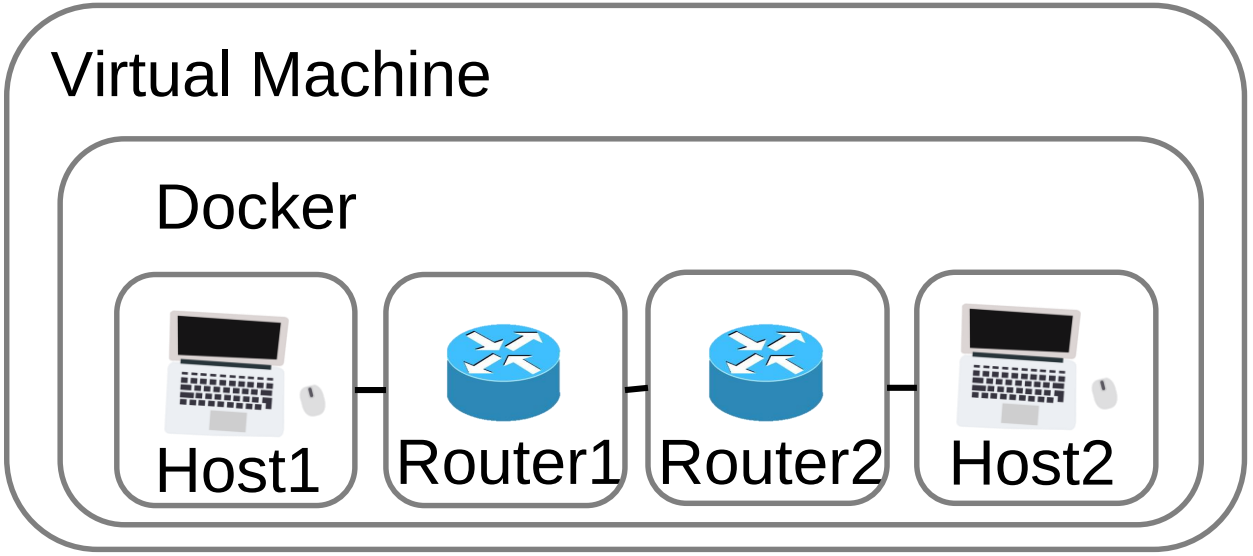
- Connect a container to a docker bridge

```
~$ sudo docker network connect testbr test
```

- Container will add an interface and assign an IP to the container (e.g., test)

Four Steps of Environment Setting

- Create Containers
- Container Network Setup
- Set default gateways of Host Containers
- Configure Router Containers



Create a Container of Ubuntu

```
~$ sudo docker run --privileged \
--cap-add NET_ADMIN --cap-add NET_BROADCAST \
-d –it --name <ContainerName> ubuntu:16.04
```

- --privileged: Give extended privileges to this container
- --cap-add: Add Linux capabilities
  - NET_ADMIN: Perform various network-related operations
  - NET_BROADCAST: Make socket broadcasts, and listen to multicasts

Host and Virtual Router Containers Creation

- Create a container as a host (h1)

```
~$ sudo docker run --privileged \
--cap-add NET_ADMIN --cap-add NET_BROADCAST \
-d –it --name h1 ubuntu:16.04
```

- Create a container as a virtual router (Q1)

```
~$ sudo docker run --privileged \
--cap-add NET_ADMIN --cap-add NET_BROADCAST \
-d –it --name Q1 ubuntu:16.04
```

Domain Network Setup in Docker

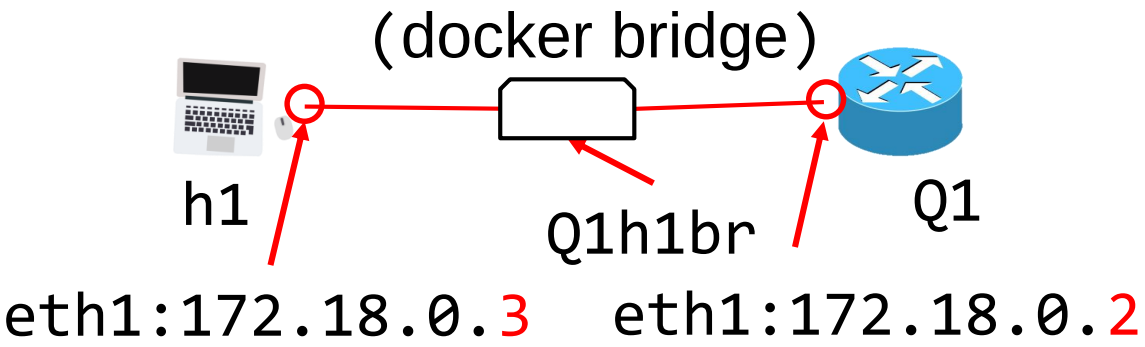
- Create host (h1) and virtual router (Q1) containers first
- Create docker bridge

```
~$ sudo docker network create Q1h1br
```

  - Q1h1br: Bridge name

- Connect containers

```
~$ sudo docker network connect Q1h1br Q1
~$ sudo docker network connect Q1h1br h1
```



- Repeat network setup procedure for each domain

Network Setup - Connect two domains

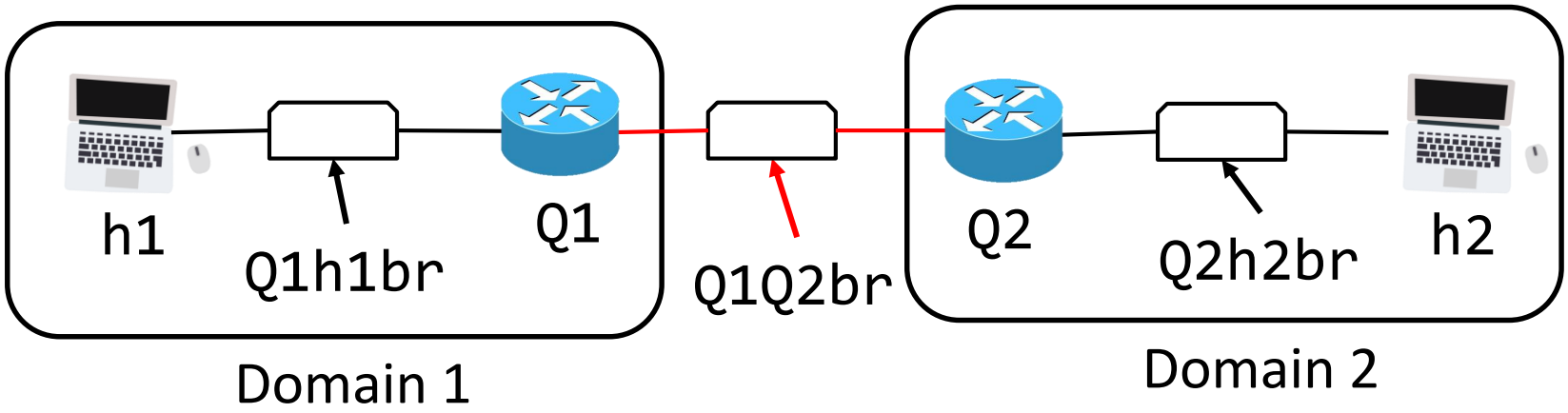
- Create inter domain bridge

```
~$ sudo docker network create Q1Q2br
```

  - Q1Q2br: Bridge name

- Connect containers

```
~$ sudo docker network connect Q1Q2br Q1
~$ sudo docker network connect Q1Q2br Q2
```



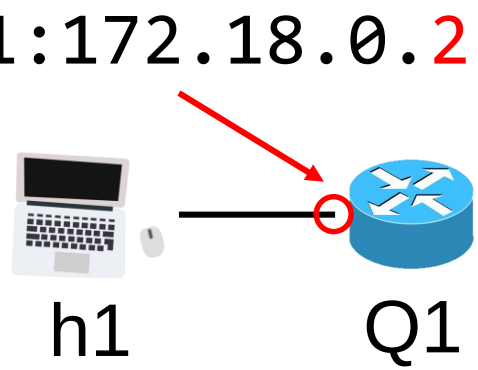
Host gateway configuration

- Use bash to enter the h1 container

```
~$ sudo docker exec -it h1 bash
```

- Install iproute2 to modify ip

```
/# apt-get update
/# apt-get install -y iproute2
```



- Set Q1 as the default gateway in h1

```
/# ip route del default
/# ip route add default via 172.18.0.2
```

- Check route in h1

```
/# route
```

```
root@23fea982ef40:/# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         Q1.Q1h1br       0.0.0.0         UG    0      0        0 eth1
172.17.0.0      *               255.255.0.0     U     0      0        0 eth0
172.18.0.0      *               255.255.0.0     U     0      0        0 eth1
```

Install Quagga in Container

- Use bash to enter the Q1 container

```
~$ sudo docker exec -it Q1 bash
```

- Install vim and quagga in container

```
/# apt-get update
/# apt-get install -y vim
/# apt-get install -y quagga
```

- Enable IP forwarding
  - Edit system control configuration file

```
/# vim /etc/sysctl.conf
```

    - Add “net.ipv4.ip_forward = 1” in sysctl.conf
  - Load in sysctl settings from `/etc/sysctl.conf`

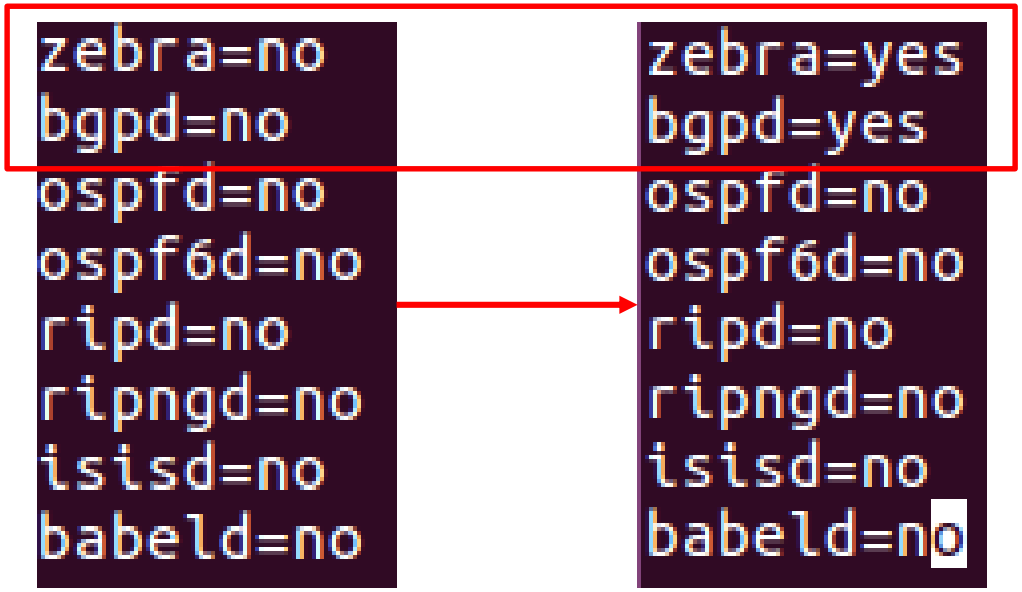
```
/# sysctl -p
```

Activate Quagga Daemons

- Activate zebra and bgpd daemons
- Edit quagga daemons

```
/# vim /etc/quagga/daemons
```

  - Change zebra and bgpd to Yes



zebra.conf configuration in Q1 Container

- Edit configuration file of zebra.conf

```
/# vim /etc/quagga/zebra.conf
```

  - Add router name and password

```
hostname R1zebra
password sdnip
log stdout
```

    - hostname and password will be used by telnet

bgpd.conf configuration in Q1 Container

- Edit configuration file of bgpd.conf

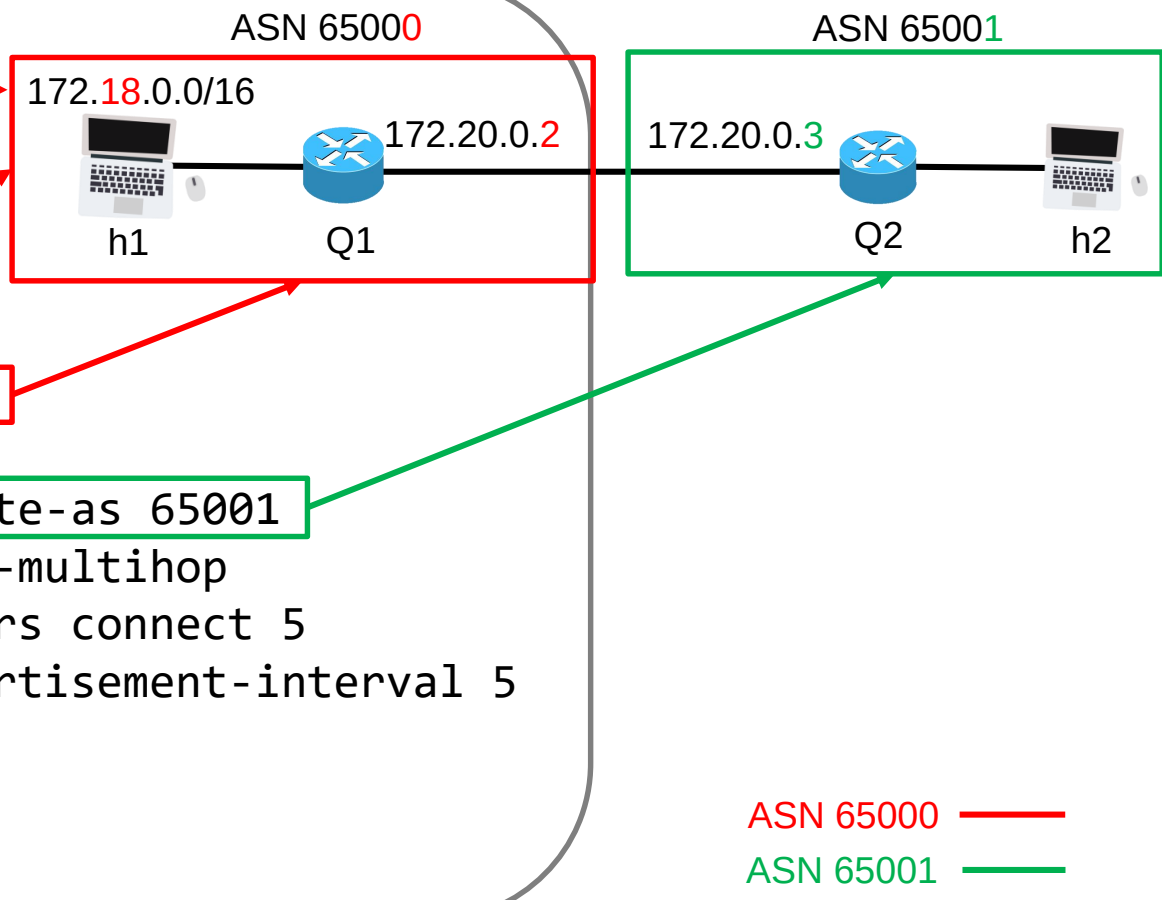
```
/# vim /etc/quagga/bgpd.conf
```

```
! BGP configuration for R1
!
hostname R1
password sdnip
!
router bgp 65000
  bgp router-id 172.20.0.2
  timers bgp 3 9
  neighbor 172.20.0.3 remote-as 65001
  neighbor 172.20.0.3 ebgp-multihop
  neighbor 172.20.0.3 timers connect 5
  neighbor 172.20.0.3 advertisement-interval 5
  network 172.18.0.0/16
!
log stdout
```

illustration of bgpd.conf configuration

Q1

```
! BGP configuration for R1
!
hostname R1
password sdnip
!
router bgp 65000
  bgp router-id 172.20.0.2
  timers bgp 3 9
  neighbor 172.20.0.3 remote-as 65001
  neighbor 172.20.0.3 ebgp-multihop
  neighbor 172.20.0.3 timers connect 5
  neighbor 172.20.0.3 advertisement-interval 5
  network 172.18.0.0/16
!
log stdout
```



bgpd.conf configuration in Q2 Container

- Edit configuration file of bgpd.conf

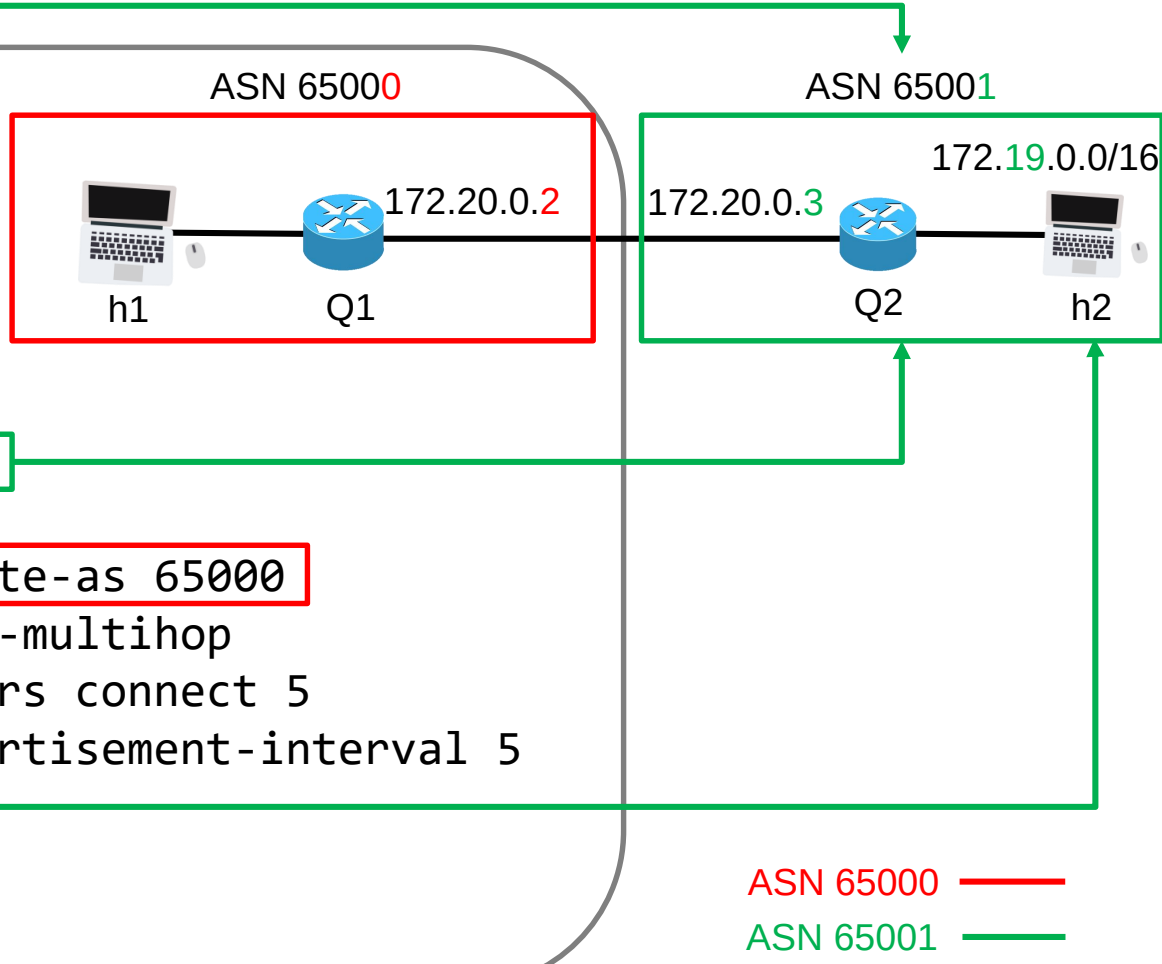
```
/# vim /etc/quagga/bgpd.conf
```

```
! BGP configuration for R2
!
hostname R2
password sdnip
!
router bgp 65001
  bgp router-id 172.20.0.3
  timers bgp 3 9
  neighbor 172.20.0.2 remote-as 65000
  neighbor 172.20.0.2 ebgp-multihop
  neighbor 172.20.0.2 timers connect 5
  neighbor 172.20.0.2 advertisement-interval 5
  network 172.19.0.0/16
!
log stdout
```

illustration of bgpd.conf configuration

Q2

```
! BGP configuration for R2
!
hostname R2
password sdnip
!
router bgp 65001
  bgp router-id 172.20.0.3
  timers bgp 3 9
  neighbor 172.20.0.2 remote-as 65000
  neighbor 172.20.0.2 ebgp-multihop
  neighbor 172.20.0.2 timers connect 5
  neighbor 172.20.0.2 advertisement-interval 5
  network 172.19.0.0/16
!
log stdout
```



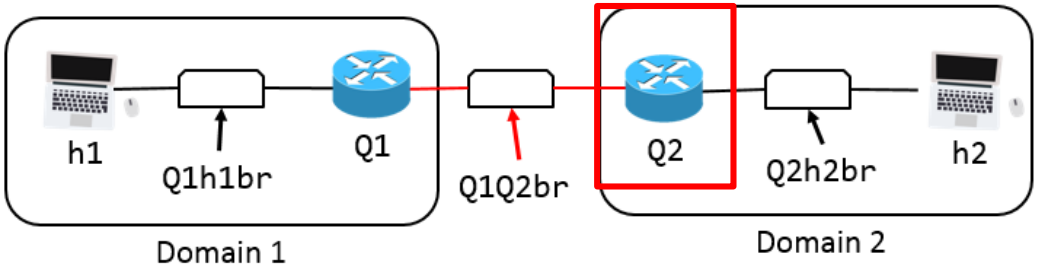
Restart Quagga and Check Route

- Use bash to enter the quagga container first
- Restart Quagga

```
/# /etc/init.d/quagga restart
```

- Check Route

```
/# route
```




```
root@b1369477c3f0:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         172.18.0.1      0.0.0.0         UG    0      0        0 eth1
172.17.0.0      *               255.255.0.0     U     0      0        0 eth0
172.18.0.0      *               255.255.0.0     U     0      0        0 eth1
172.19.0.0      (Q2.Q1Q2br      255.255.0.0     UG    0      0        0 eth2
172.20.0.0      *               255.255.0.0     U     0      0        0 eth2
```

Telnet zebra daemons in quagga container

- Use bash to enter the quagga container first
- Telnet zebra daemons
  - zebra listens on port 2601

```
~# telnet localhost 2601
```

- Show bgp route

```
R1zebra> show ip route bgp
```

```
zebra> show ip route bgp
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, P - PIM, A - Babel,
       > - selected route, * - FIB route

B>* 172.19.0.0/16 [20/0] via 172.20.0.3, eth2, 06:46:17
B>* 172.18.0.0/16 [20/0] via 172.20.0.4, eth2, 06:41:05
```

Telnet bgpd daemons in quagga container

- Use bash to enter the quagga container first
- Telnet bgpd daemons
  - bgpd listens on port 2605

```
~# telnet localhost 2605
```

- Show bgp summary

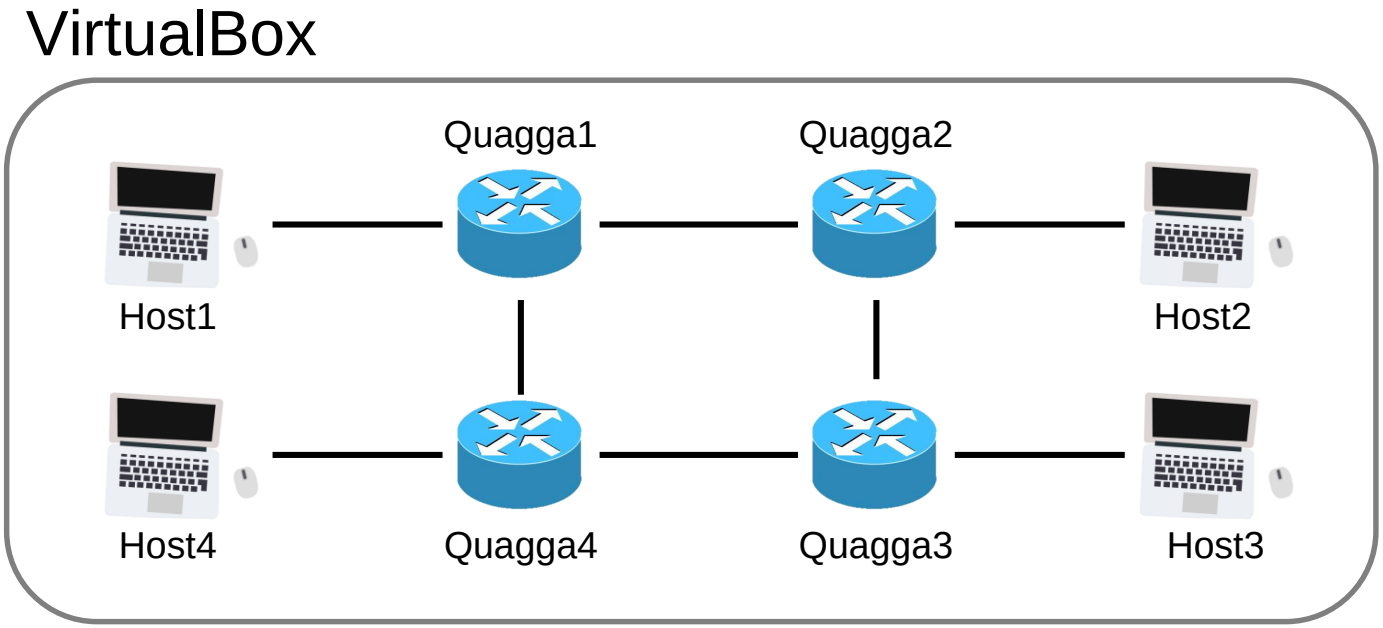
```
R1> show ip bgp summary
```

```
r1> show ip bgp summary
BGP router identifier 172.20.0.2, local AS number 65000
RIB entries 7, using 784 bytes of memory
Peers 2, using 9136 bytes of memory

Neighbor        V         AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
172.20.0.3      4 65001    8151    8154        0    0    0 06:47:36        3

Total number of neighbors 1
```

Target Topology of Project 5



Report Submission

- Files
  - A report: project5_<studentID>.pdf
    - Show topology with IP addresses, interfaces and ASNs
    - Capture one BGP packet from wireshark and show screenshots
    - Telnet zebra and bgpd daemons of each route and show route screenshots
    - Write down what you have learned or solved.
- Submit
  - Upload project5_<studentID>.pdf to e3
  - Report with incorrect file name or format subjects to not scoring.

References

- Docker overview
  - https://docs.docker.com/engine/docker-overview/
- Docker commandline reference
  - https://docs.docker.com/engine/reference/commandline/run/
- Learn Docker Browser-Based
  - https://www.katacoda.com/courses/docker

Thank You!

謝謝您們的聆聽