



Lab 3

Network Function Virtualization: Software Router and Containerization

Deadline: 2023/11/29 (WED) 23:59



Outline

- Introduction
- Docker Installation
- Docker Usage
- Example Scenario Setup
- Introduction to Docker Compose
- Lab 3 Overview
- Submission & Scoring Criteria
- References



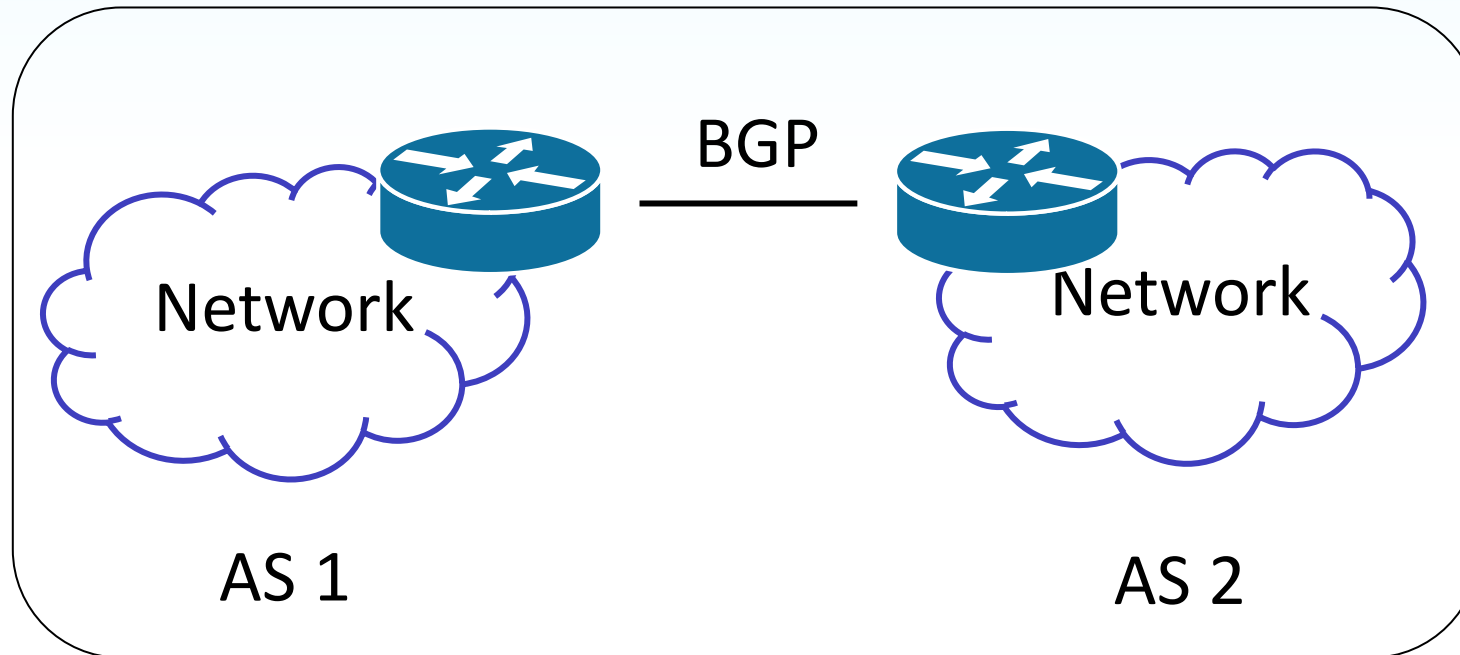
Outline

- Introduction
- Docker Installation
- Docker Usage
- Example Scenario Setup
- Introduction to Docker Compose
- Lab 3 Overview
- Submission & Scoring Criteria
- References



Example Scenario

- Interconnection of two networks



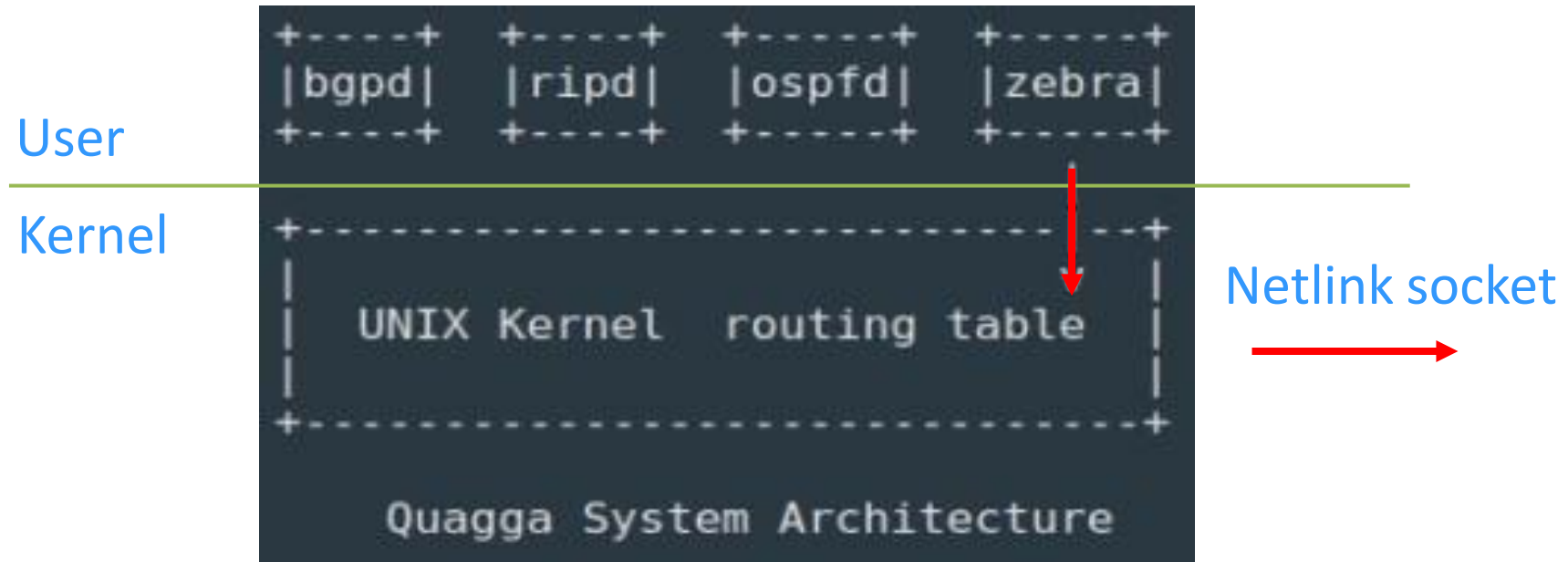
- **BGP**: Broder Gateway Protocol
- **AS**: Autonomous System



Introduction to Quagga

Quagga is an open-source software that **provides routing services**

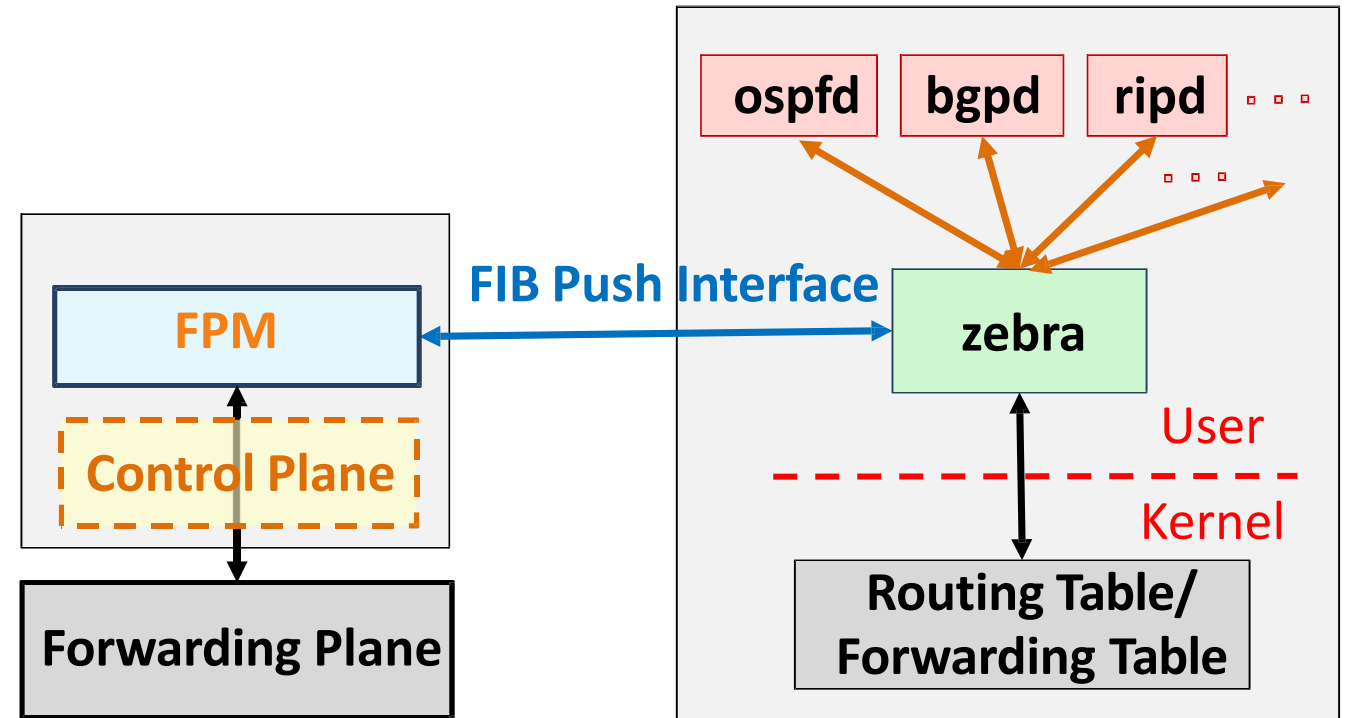
- Supports common routing protocols: BGP, OSPF, RIP, and IS-IS
- Consists of a **core daemon Zebra** and separate **routing protocol** daemons
- Routing Protocols (daemons) communicate their best routes to Zebra
- Zebra computes best routes and modifies **kernel routing table** through netlink





Introduction to FIB Pushing

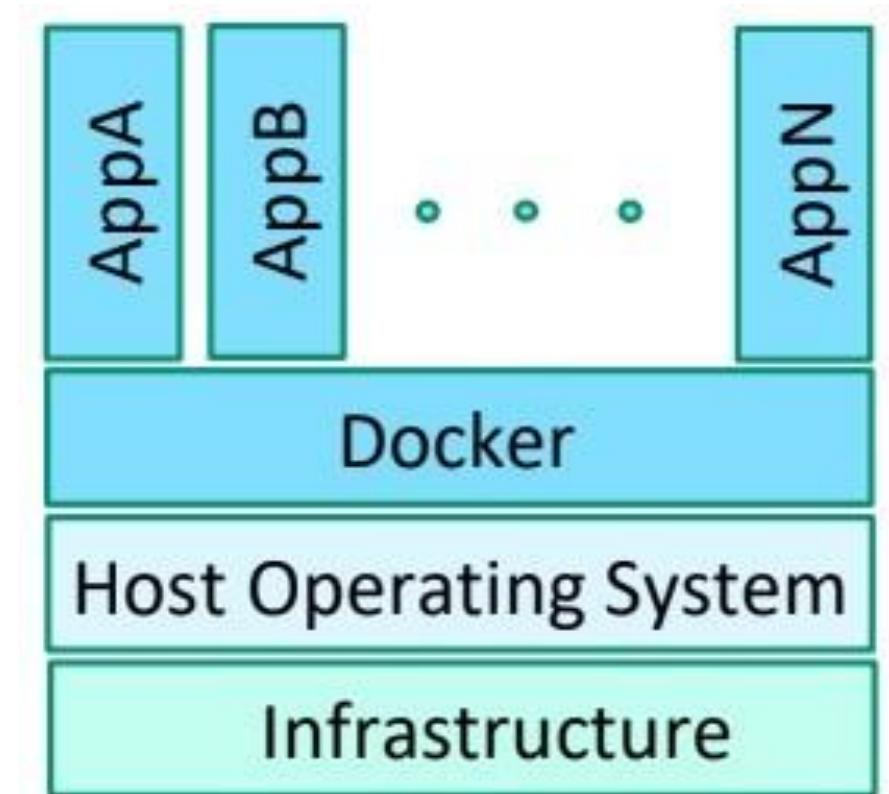
- Zebra supports a FIB Push Interface (FPI)
 - FPI allows an **external component** to **learn** the forwarding information
- Forwarding Plane Manager (FPM):
 - Receives FIB
 - Decode FIB into routes
 - Programs forwarding plane
- FIB Pushing:
 - FPM establishes a TCP connection with zebra
 - Zebra pushes FIB to FPM





Introduction to Docker

- Docker is a software platform that allows you to build, test, and deploy applications quickly in packages called containers
- Typical steps for creating Docker containers:
 1. **Build Docker images** of the desired OS distribution and applications
 2. Store the images in a Docker Registry
 - Public (Docker Hub)
 - Private
 3. **Run Docker to build containers** of images





Outline

- Introduction
- **Docker Installation**
- Docker Usage
- Example Scenario Setup
- Introduction to Docker Compose
- Lab 3 Overview
- Submission & Scoring Criteria
- References



Installation of Docker

- Update apt (confirm to install the latest package)

```
$ sudo apt-get update
```

- Install curl for data transfer

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

- Retrieve Docker installation script and install Docker

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

- Manage Docker as a non-root user

```
$ sudo groupadd docker
```

```
$ sudo usermod -aG docker $USER
```

```
$ newgrp docker
```



Outline

- Introduction
- Docker Installation
- **Docker Usage**
- Example Scenario Setup
- Introduction to Docker Compose
- Lab 3 Overview
- Submission & Scoring Criteria
- References



Pull Image

- Usage

```
$ docker pull <name>:<tag>
```

- E.g., Pull ubuntu:22.04 image from Docker hub registry

```
$ docker pull ubuntu:22.04
```

- List images

```
$ docker images
```

```
cu@SDN-NFV:~$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
ubuntu        22.04     e4c58958181a   2 weeks ago   77.8MB
```



Docker run (1/2)

- Run a command in a new container
 - Create and run a container
 - Execute a command in the container
- Usage

```
$ docker run <options> <image>:<tag> <command><arg..>
```

Create and Run a container

Execute a command in the container

- E.g., Create and Run a container “test”

```
$ docker run -d -it --name test ubuntu:22.04
```

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



Docker run (2/2)

- List containers

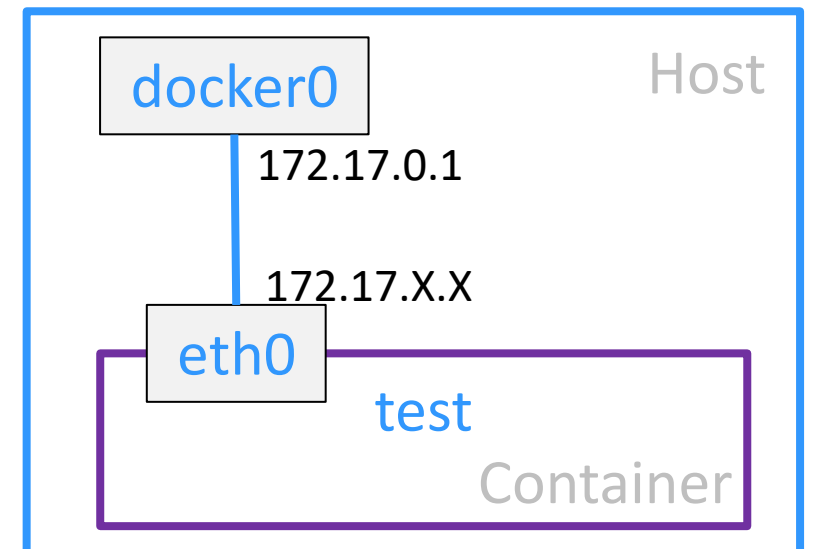
```
$ docker ps -a
```

- “--all”, “-a”: Show all containers

```
cu@SDN-NFV:~$ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
23497ece301e	ubuntu:22.04	"/bin/bash"	4 seconds ago	Up 3 seconds		test

- Container is connected to docker0 bridge by default
- Default IP for docker0 bridge is 172.17.0.1/16
- Docker will assign an IP for the container





Docker exec

- Execute a command in a running container
- Usage

```
$ docker exec <options> <container> <command>
```

- E.g., Exec *bash* command in a **running** container “test”

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

```
cu@SDN-NFV:~$ docker exec -it test bash  
root@23497ece301e:/#
```



Docker network - Create

- Create a bridge network

- Usage

```
$ docker network create <bridge>
```

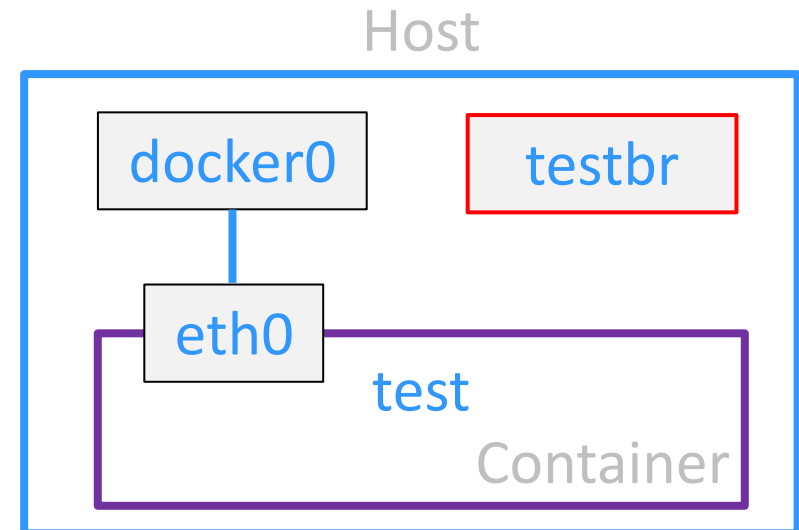
- E.g., Create a network bridge “testbr”

```
$ docker network create testbr
```

- List networks

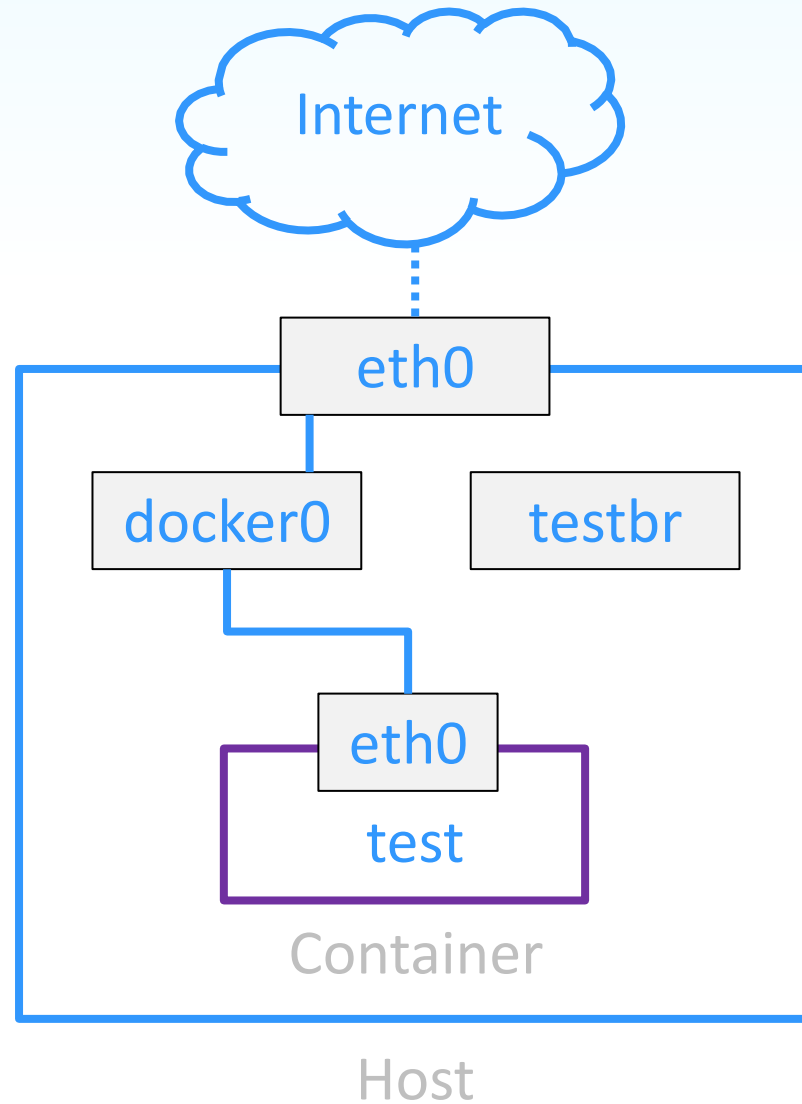
```
$ docker network ls
```

```
cu@SDN-NFV:~$ docker network ls
NETWORK ID          NAME       DRIVER  SCOPE
35c1f8ddd132        bridge    bridge  local
b9d64cd68f56        host      host    local
5e0125ebf294        none      null    local
376b09c5a977        testbr    bridge  local
```





Network Environment after testbr Creation





Docker network - Connect

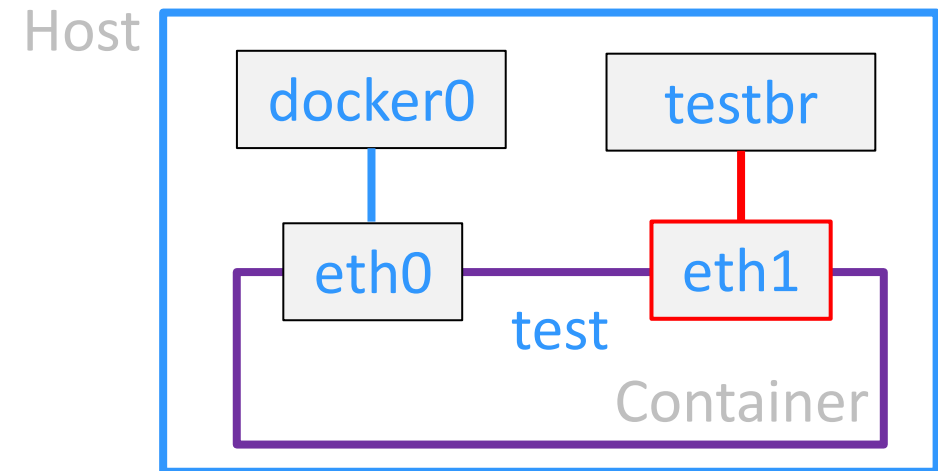
- Connect to a bridge network
- Usage

```
$ docker network connect <network> <container>
```

- E.g., Connect container “test” to bridge “testbr”

```
$ docker network connect testbr test
```

- Docker will add an interface on the container and assign an IP





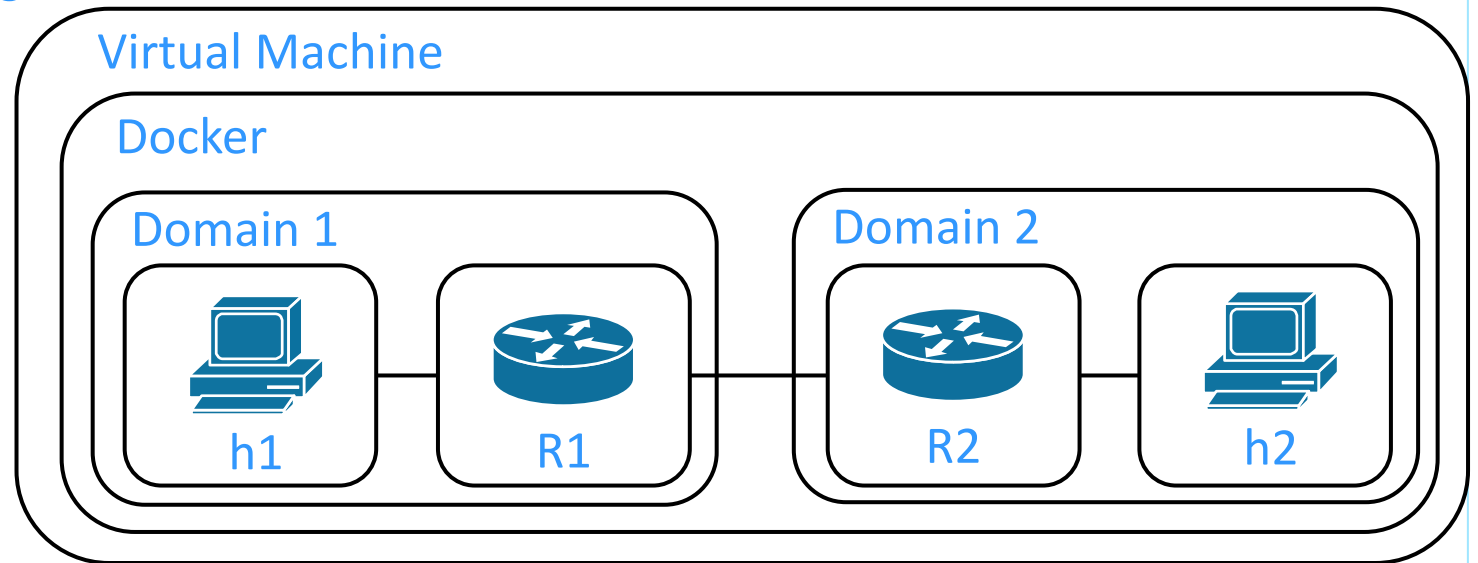
Outline

- Introduction
- Docker Installation
- Docker Usage
- **Example Scenario Setup**
- Introduction to Docker Compose
- Lab 3 Overview
- Submission & Scoring Criteria
- References



Steps to Setup Example Scenario

1. Prepare Docker images
2. Create containers
3. Setup container networks
 - Setup network for domains
 - Connect domains
4. Configure host gateways
 - Gateway of h1 = R1
 - Gateway of h2 = R2
5. Setup routers
6. Check routes





Step 1 – Prepare Docker images (1/3)

- Prepare docker image “host”
- Write Dockerfile
 - Create configuration file **host.Dockerfile**

```
$ vi host.Dockerfile
```

```
FROM ubuntu:22.04
```

```
RUN apt-get update -y \  
&& apt-get install -y net-tools \  
&& apt-get install -y iproute2 \  
&& apt-get install -y iputils-ping
```

```
CMD ["sleep","infinity"]
```



Step 1 – Prepare Docker images (2/3)

- Build docker image with Dockerfile

```
$ docker build -t host -f host.Dockerfile .
```

- -t: Name and optionally a tag in the “name:tag” format
- -f: Name of the Dockerfile (Default is 'PATH/Dockerfile')

- List image

```
$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
host	latest	97085d108578	About an hour ago	129MB
ubuntu	22.04	e4c58958181a	2 weeks ago	77.8MB



Step 1 – Prepare Docker images (3/3)

- Prepare Docker image quagga
- Pull image [opencord/quagga](#) from docker hub

```
$ docker pull opencord/quagga
```

- List image

```
$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
host	latest	97085d108578	About an hour ago	129MB
ubuntu	22.04	e4c58958181a	2 weeks ago	77.8MB
opencord/quagga	latest	2c638cd24154	6 years ago	457MB



Step 2 – Create Containers (1/2)

- Create a container with prepared image

```
$ docker run --privileged --cap-add NET_ADMIN \  
  --cap-add NET_BROADCAST -d -it \  
  --name <container name> <image>
```

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



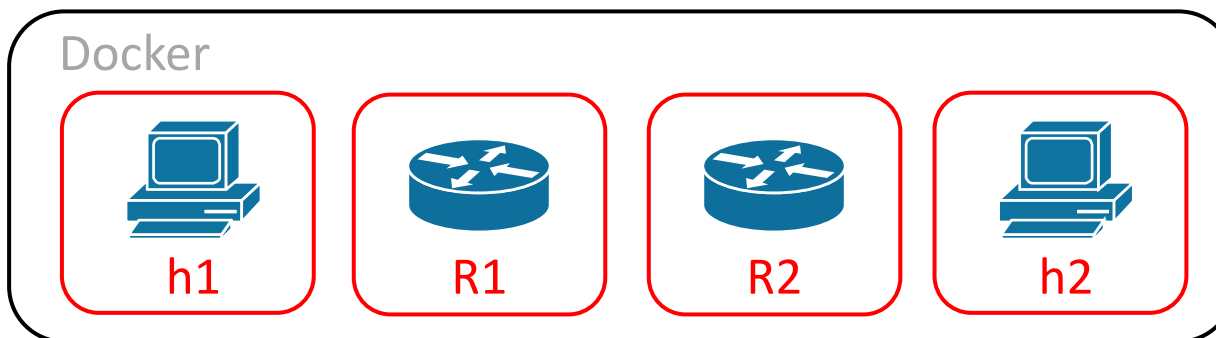
Step 2 – Create Containers (2/2)

- Create container for a host h1 (h2)

```
$ docker run --privileged --cap-add NET_ADMIN \  
  --cap-add NET_BROADCAST -d -it \  
  --name h1 host
```

- Create container for a virtual router R1 (R2)

```
$ docker run --privileged --cap-add NET_ADMIN \  
  --cap-add NET_BROADCAST -d -it \  
  --name R1 opencord/quagga
```





Step 3 – Setup Container Networks (1/3)

- Setup network for domains
- Create a bridge network R1h1br

```
$ docker network create R1h1br
```

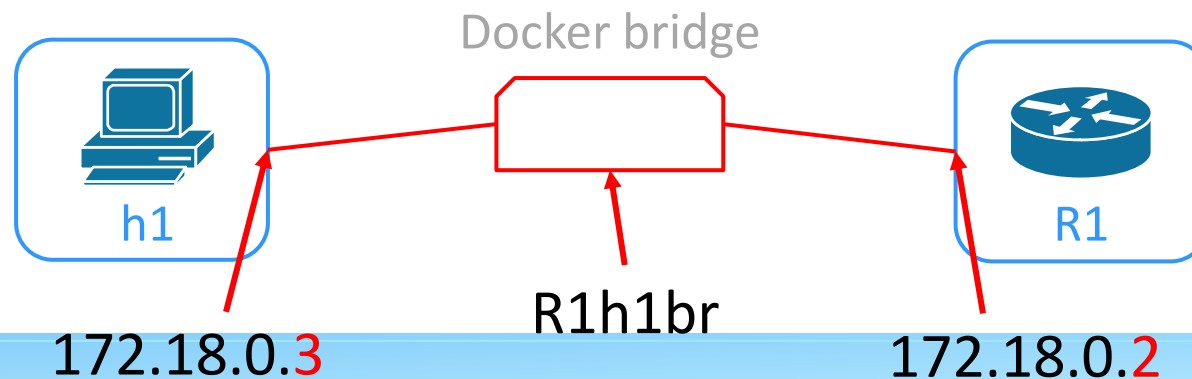
- R1h1br: Bridge name

- Connect containers h1 and R1 to bridge R1h1br

```
$ docker network connect R1h1br R1
```

```
$ docker network connect R1h1br h1
```

- Docker will assign IP to interfaces automatically





Step 3 – Setup Container Networks (2/3)

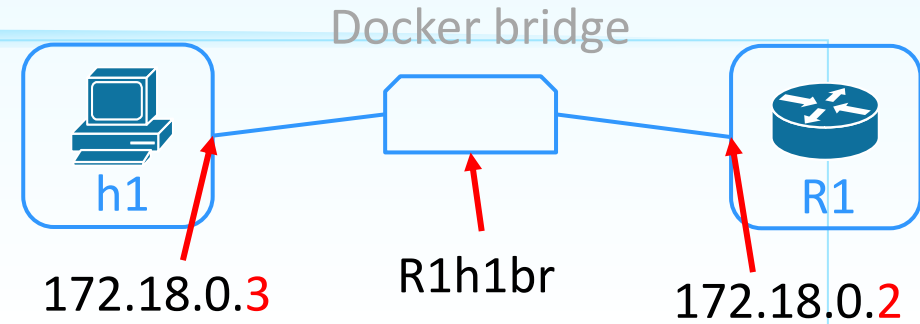
- Check the IP addresses of network interfaces

```
$ docker inspect h1 (R1)
```

Inspected result for h1

```
"Networks": {  
  "R1h1br": {  
    "IPAMConfig": {},  
    "Links": null,  
    "Aliases": [  
      "6e5f213a2112"  
    ],  
    "NetworkID": "39e76624b298ff26"  
    "EndpointID": "1f02cbd6d908505"  
    "Gateway": "172.18.0.1",  
    "IPAddress": "172.18.0.3",  
  },  
}
```

- Repeat network setup procedure for another domain
 - Create bridge network R2h2br
 - Connect containers R2 and h2



Inspected result for R1

```
"Networks": {  
  "R1h1br": {  
    "IPAMConfig": {},  
    "Links": null,  
    "Aliases": [  
      "6e3cc0bcd4dc"  
    ],  
    "NetworkID": "39e76624b298ff2"  
    "EndpointID": "580ccc41869ffc"  
    "Gateway": "172.18.0.1",  
    "IPAddress": "172.18.0.2",  
  },  
}
```



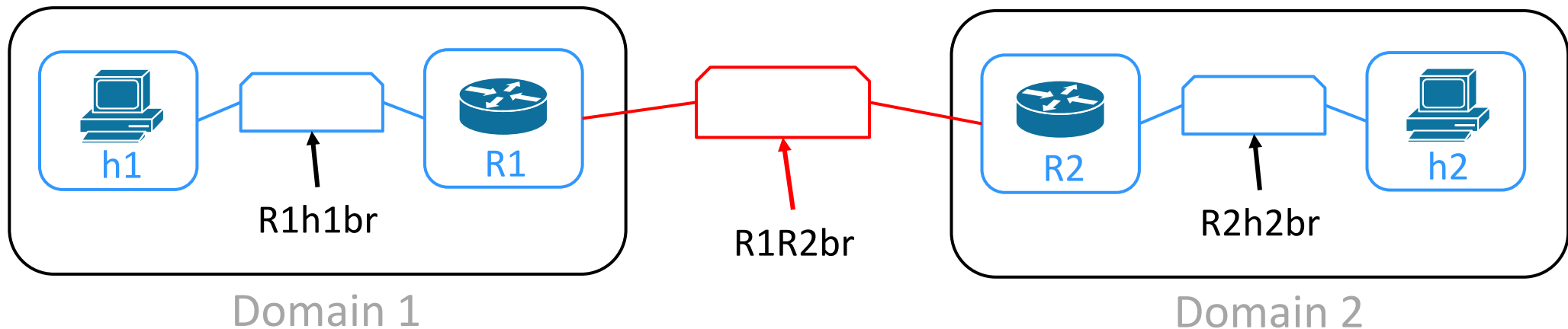
Step 3 – Setup Container Networks (3/3)

- Connect two domains
- Create inter domain bridge

```
$ docker network create R1R2br
```

- Connect containers R1 and R2 to bridge R1R2br

```
$ docker network connect R1R2br R1  
$ docker network connect R1R2br R2
```





Step 4 – Configure Host Gateways (1/1)

- Run bash on h1 (h2)

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

- Set R1 (R2) as default gateway of h1 (h2)

```
h1/# ip route del default
```

```
h1/# ip route add default via 172.18.0.2
```

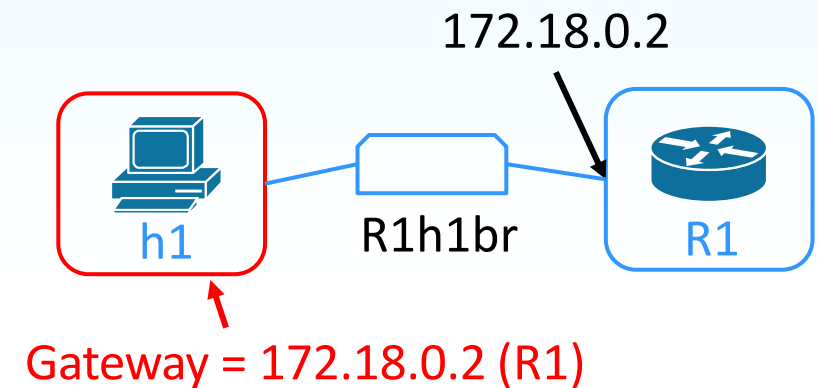
- Check route on h1 (h2)

```
h1/# route
```

h1's bash prompt

```
root@6e5f213a2112:/# route
Kernel IP routing table
Destination    Gateway         Genmask         Flags Metric Ref    Use Iface
default        R1.R1h1br      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
```

- Exit h1 bash via CTRL+D





Step 5 – Setup Routers (1/4)

- Run bash on R1 (R2)

```
$ docker exec -it R1 bash
```

- Enable IP forwarding on R1 (R2)

```
R1/# vi /etc/sysctl.conf
```

- Uncomment

“net.ipv4.ip_forward=1”

- Run sysctl to load configuration

```
R1/# sysctl -p
```

```
# Note: This may impact IPv6 TCP sessions too
#net.ipv4.tcp_syncookies=1

# Uncomment the next line to enable packet forwarding for IPv4
#net.ipv4.ip_forward=1

# Uncomment the next line to enable packet forwarding for IPv6
```



```
# Note: This may impact IPv6 TCP sessions too
#net.ipv4.tcp_syncookies=1

# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1
```



Step 5 – Setup Routers (2/4)

- Set hostname and password on Zebra on R1 (R2)

- Create Quagga configuration folder on R1 (R2)

```
R1/# mkdir /etc/quagga
```

- Edit configuration file zebra.conf on Quagga on R1 (R2)

```
R1/# vi /etc/quagga/zebra.conf
```

- Add router name and password in Zebra configuration file

```
hostname R1zebra (R2zebra)  
password vRouter  
log stdout
```

- Host name for identify the Zebra on R1 or R2 (for shell prompt)
- Password for user access verification



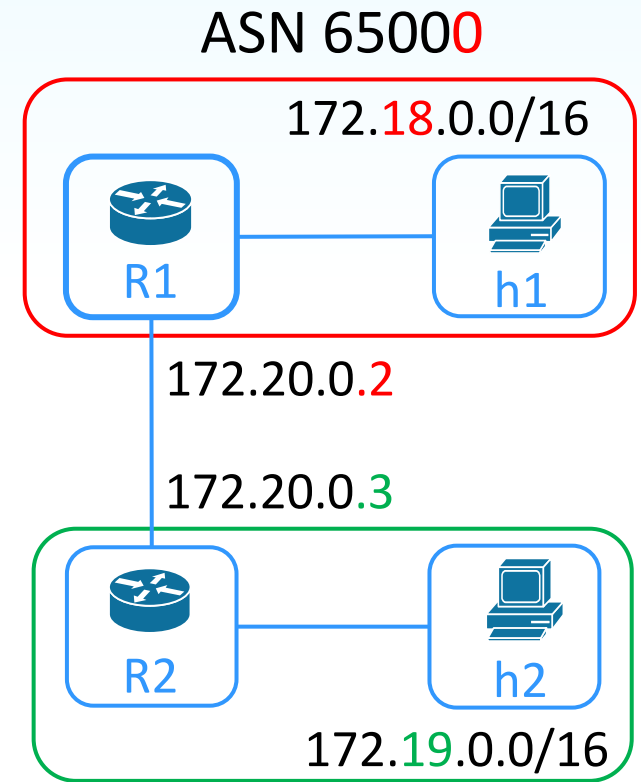
Step 5 – Setup Routers (3/4)

- Set BGP configuration of routers

- Edit configuration file **bgpd.conf** on Quagga on R1

```
R1/# vi /etc/quagga/bgpd.conf
```

```
! BGP configuration for R1
!
hostname R1bgp
password vRouter
!
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
```



ASN 65001

ASN 65000 —

ASN 65001 —



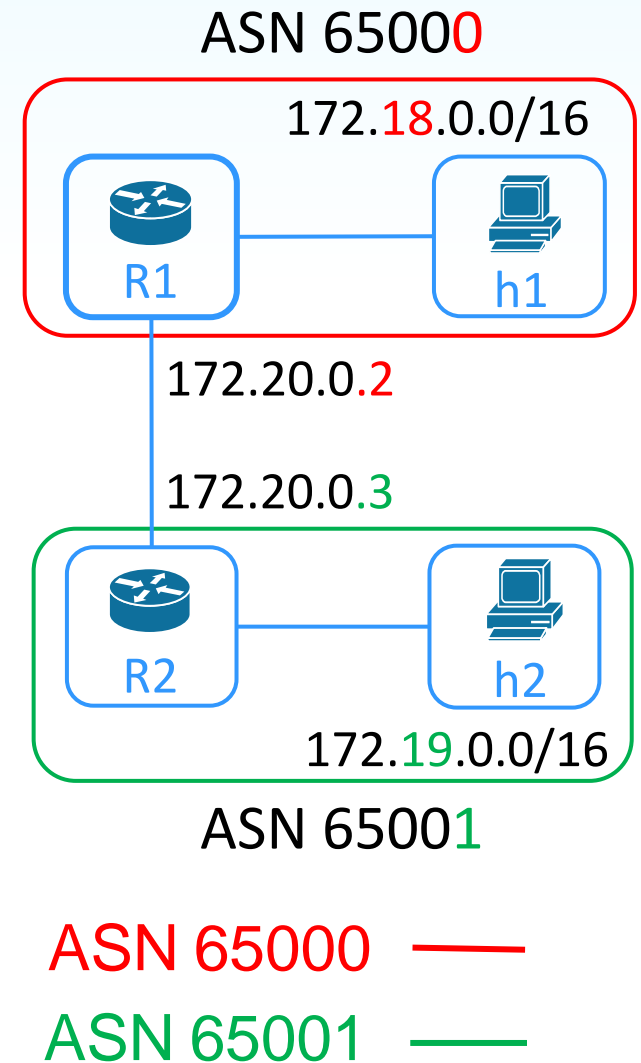
Step 5 – Setup Routers (4/4)

- Set BGP configuration of routers

- Edit configuration file **bgpd.conf** on Quagga on R2

```
R2/# vi /etc/quagga/bgpd.conf
```

```
! BGP configuration for R2
!
hostname R2bgp
password vRouter
!
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
```





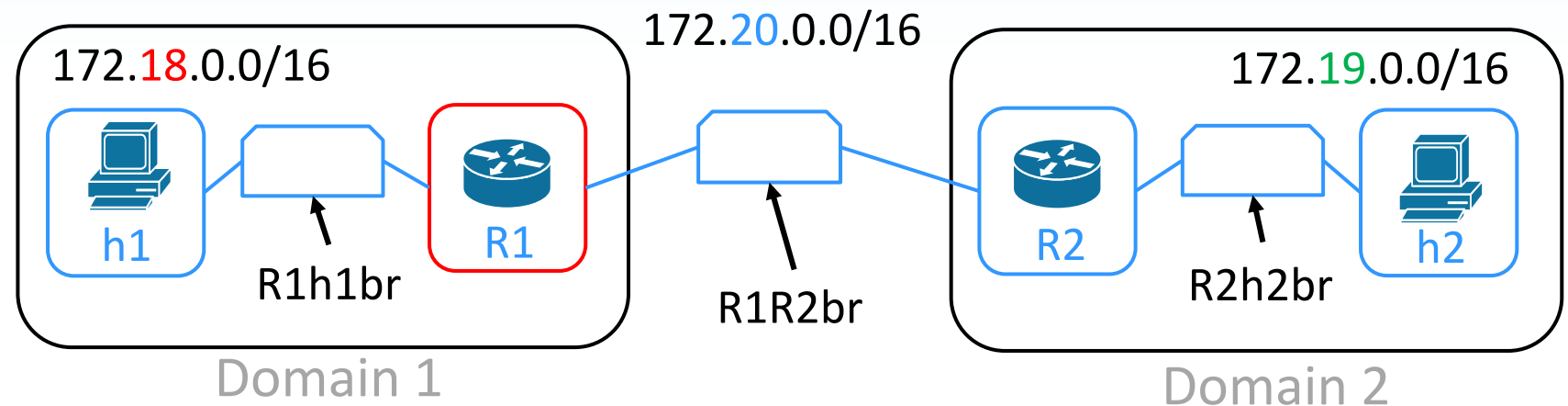
Step 6 – Check Routes (1/3)

- Restart Quagga on R1 (R2)

```
$ docker restart R1
```

- Check route on R1 (R2) bash

```
R1/# route
```



```
root@6e3cc0bcd4dc:/# route
Kernel IP routing table
Destination      Gateway          Genmask          Flags Metric Ref    Use Iface
default          172.20.0.1      0.0.0.0          UG    0     0        0 eth2
172.17.0.0       *               255.255.0.0      U     0     0        0 eth1
172.18.0.0       *               255.255.0.0      U     0     0        0 eth0
172.19.0.0       R2.R1R2br       255.255.0.0      UG    0     0        0 eth2
172.20.0.0       *               255.255.0.0      U     0     0        0 eth2
```



Step 6 – Check Routes (2/3)

- Telnet R1 zebra daemons (on port 2601)

```
R1/# telnet localhost 2601
```

– Password: vRouter

```
User Access Verification
```

```
Password:
```

```
R1zebra> 
```

- Show bgp route in R1zebra

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

```
R1zebra> 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, 00:04:02
```



Step 6 – Check Routes (3/3)

- Telnet R1 bgpd daemons (on port 2605)

```
R1/# telnet localhost 2605
```

– Password: vRouter

```
User Access Verification
```

```
Password:
```

```
R1bgp> 
```

- Show R1 bgp summary

```
R1bgp> show ip bgp summary
```

```
R1bgp> show ip bgp summary
```

```
BGP router identifier 172.20.0.2, local AS number 65000
```

```
RIB entries 3, using 336 bytes of memory
```

```
Peers 1, using 4568 bytes of memory
```

Neighbor	V	AS	MsgRcvd	MsgSent	TblVer	InQ	OutQ	Up/Down	State/PfxRcd
172.20.0.3	4	65001	185	187	0	0	0	00:09:07	1

```
Total number of neighbors 1
```



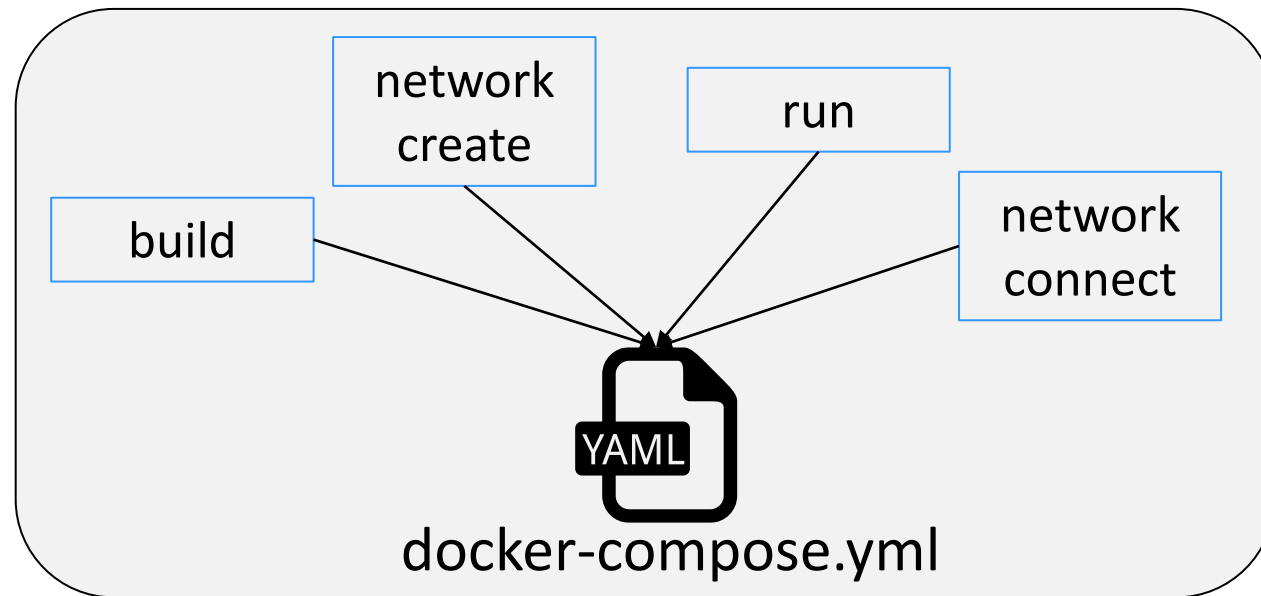
Outline

- Introduction
- Docker Installation
- Docker Usage
- Example Scenario Setup
- **Introduction to Docker Compose**
- Lab 3 Overview
- Submission & Scoring Criteria
- References



Introduction to Docker Compose

- A tool for **defining** and **running multi-container** Docker applications
- Use a **YAML file** to configure your application services
- Start all the services from your configuration with a **single command**
- The default name for Compose file is **docker-compose.yml**



■ Yaml template for example docker application

- ```
• services
 - h1
 - R1
• networks
 - R1h1br
```

docker-compose.yml

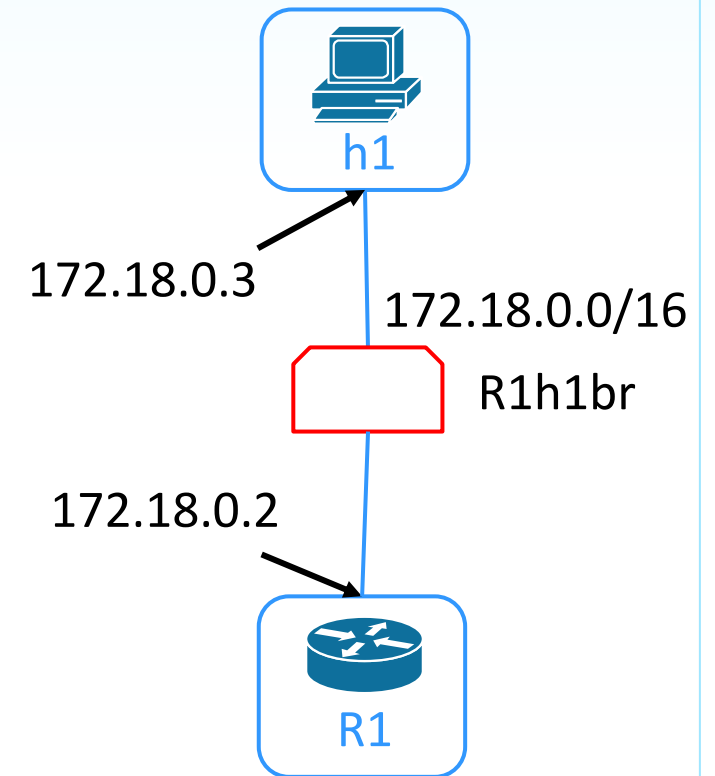


# Docker Compose Example (1/4)

## ● Create bridge R1h1br

network create

```
networks:
 R1h1br:
 driver: bridge
 ipam:
 config:
 - subnet: 172.18.0.0/16
```





# Docker Compose Example (2/4)

## ● Create container h1 build, run, network connect

services:

h1:

image: host

container\_name: h1

privileged: true

build:

context: .

dockerfile: host.Dockerfile

cap\_add:

- NET\_ADMIN

- NET\_BROADCAST

networks:

R1h1br:

ipv4\_address: 172.18.0.3

entrypoint: ["/bin/sh", "-c"]

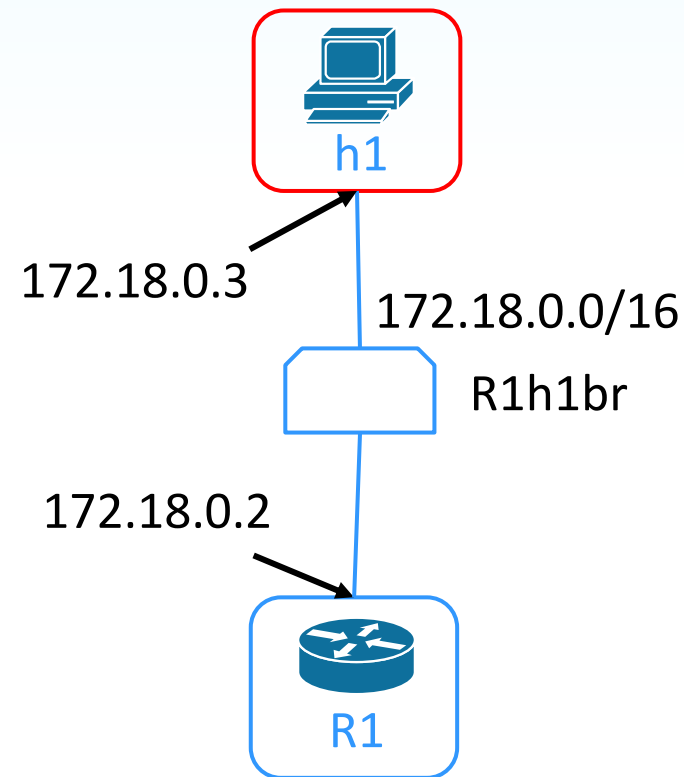
command:

- |

ip route del default

ip route add default via 172.18.0.2

sleep infinity





# Docker Compose Example (3/4)

## ● Create container R1 build, run, network connect

services:

R1:

image: opencord/quagga

container\_name: R1

privileged: true

sysctls:

- net.ipv4.ip\_forward=1

cap\_add:

- NET\_ADMIN
- NET\_BROADCAST

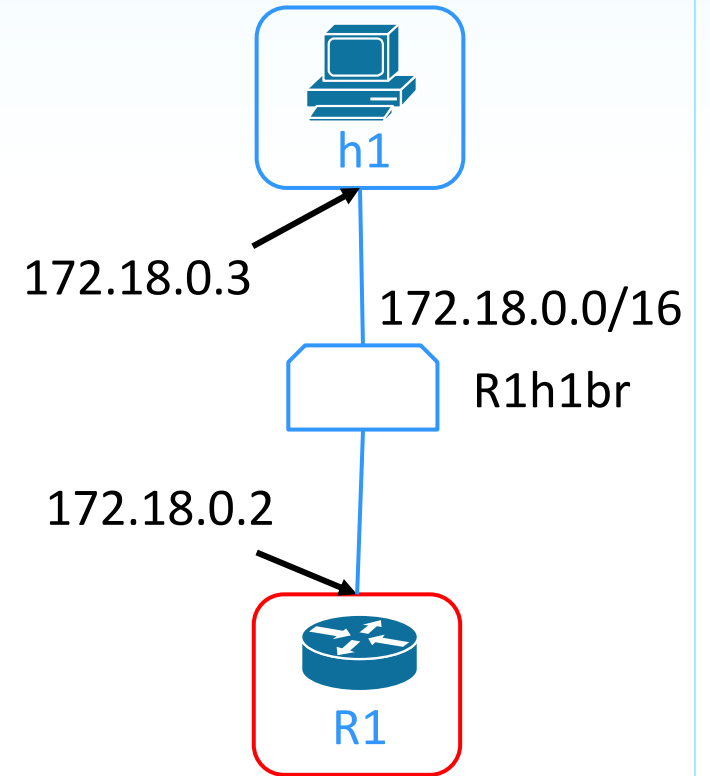
networks:

R1h1br:

ipv4\_address: 172.18.0.2

volumes:

- ./config/R1/zebra.conf:/etc/quagga/zebra.conf
- ./config/R1/bgpd.conf:/etc/quagga/bgpd.conf







# Docker Compose Example (4/4)

- At the root of the app project, write file docker-compose.yml

- Start the Docker application

```
$ docker compose up -d
```

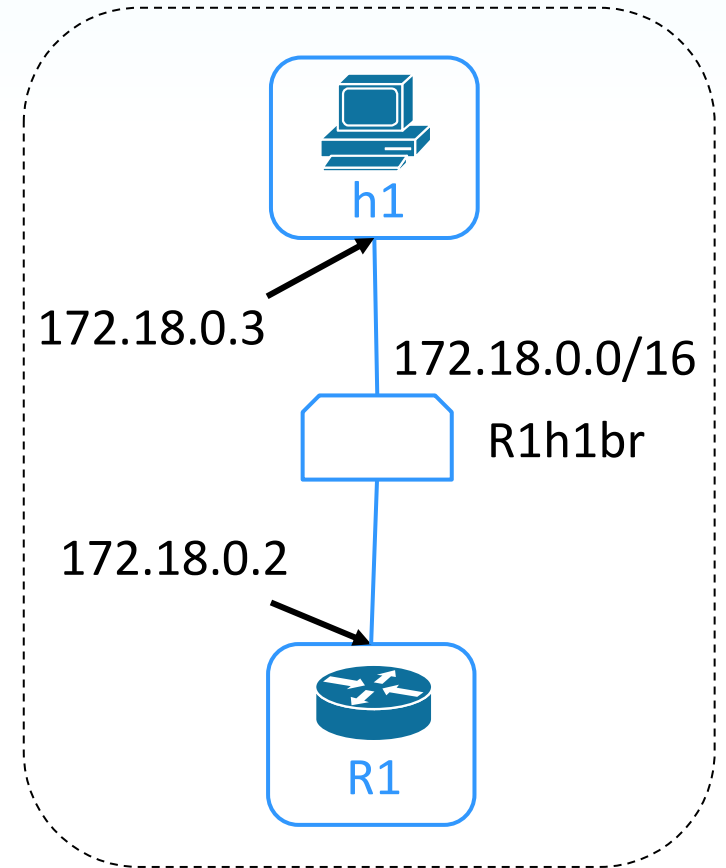
- List the running containers

```
$ docker ps
```

| CONTAINER ID | IMAGE           | NAMES |
|--------------|-----------------|-------|
| 58c33d3757ec | opencord/quagga | R1    |
| 2d364ce211a7 | host            | h1    |

- Stop and remove the Docker application

```
$ docker compose down
```



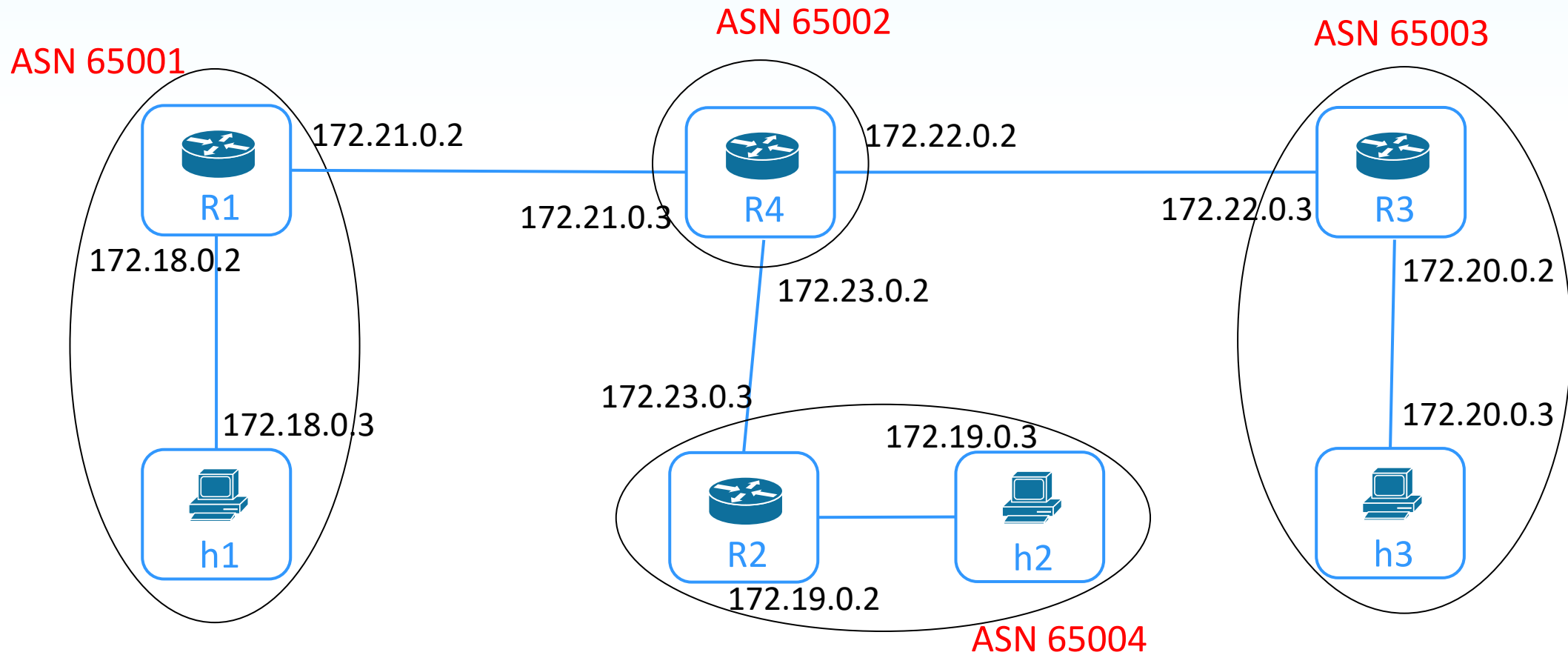


# Outline

- Introduction
- Docker Installation
- Docker Usage
- Example Scenario Setup
- Introduction to Docker Compose
- Lab 3 Overview
  - Overview & Workflow
  - How to Test your App
  - Supplements
- Submission & Scoring Criteria
- References



# Topology of Lab 3





# Outline

- Introduction
- Docker Installation
- Docker Usage
- Example Scenario Setup
- Introduction to Docker Compose
- **Lab 3 Overview**
  - Overview & Workflow
  - **How to Test your App**
  - Supplements
- Submission & Scoring Criteria
- References



# How to Test Your App (1/2)

- Check IP configuration

```
$ docker exec h1 ifconfig
```

```
eth0 Link encap:Ethernet HWaddr 02:42:ac:12:00:03
 inet addr:172.18.0.3 Bcast:172.18.255.255 Mask:255.255.0.0
 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
 RX packets:95 errors:0 dropped:0 overruns:0 frame:0
 TX packets:9 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 txqueuelen:0
 RX bytes:10725 (10.7 KB) TX bytes:714 (714.0 B)

lo Link encap:Local Loopback
 inet addr:127.0.0.1 Mask:255.0.0.0
 UP LOOPBACK RUNNING MTU:65536 Metric:1
```

- Check host communication

```
$ docker exec -it h1 ping 172.20.0.3 -c 3
```



## How to Test Your App (2/2)

- Run bash on R4

```
$ docker exec -it R4 bash
```

- Telnet R4 bgpd daemon (on port 2605)

```
R4/# telnet localhost 2605
```

- Show R4 bgp

```
R4bgp> show ip bgp
```

```
R4bgp> show ip bgp
BGP table version is 0, local router ID is 172.23.0.2
Status codes: s suppressed, d damped, h history, * valid, > best, = multipath,
 i internal, r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete

 Network Next Hop Metric LocPrf Weight Path
*> 172.18.0.0 172.21.0.2 0 0 65001 i
*> 172.19.0.0 172.23.0.3 0 0 65004 i
*> 172.20.0.0 172.22.0.3 0 0 65003 i

Total number of prefixes 3
```



# Outline

- Introduction
- Docker Installation
- Docker Usage
- Example Scenario Setup
- Introduction to Docker Compose
- **Lab 3 Overview**
  - Overview & Workflow
  - How to Test your App
  - **Supplements**
- Submission & Scoring Criteria
- References



# Supplements

- TA will provide the following
  - You can use Docker compose to create h1 and R1 containers based on these files

```
SDNFV_lab3/
├── config
│ └── R1
│ ├── bgpd.conf
│ └── zebra.conf
├── docker-compose.yml
└── host.Dockerfile

2 directories, 4 files
```





# Outline

- Introduction
- Docker Installation
- Docker Usage
- Example Scenario Setup
- Introduction to Docker Compose
- Lab 3 Overview
- Submission & Scoring Criteria
- References



# Scoring Criteria (1/2)

- Create Compose file
  - You **must** provide **Compose file**
  - The Compose file must be named **docker-compose.yml**
  - The Compose file should be **placed in the top-level** directory of the project
  - TA will use your Compose file to build your application
  - If you do not provide Compose file, you will **not earn any credit**
- **(35%)** Create containers
  - Use Docker to create **3 host** and **4 router** containers
  - You lose **5 credits** for each container that is not created



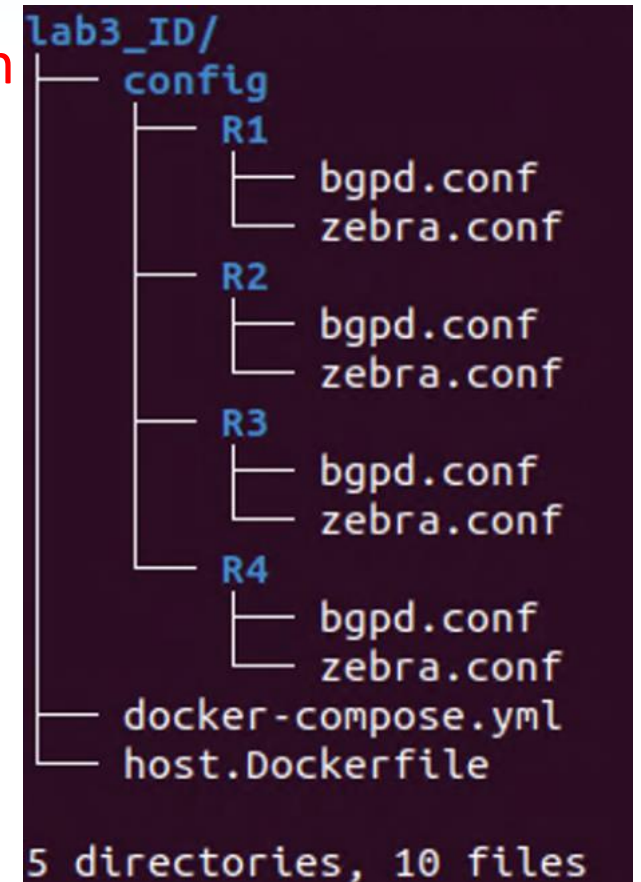
## Scoring Criteria (2/2)

- **(25%) Setup links**
  - Set the connections and IPs based on the topology on page 43
  - Use the Docker network to set a connection between each pair of container
  - Each link with a wrong setting will result in **4 points** deduction
- **(20%) Setup BGP configurations**
  - Set the AS number based on the topology on page 43
  - TA will check your settings in **bgpd.conf** for each router
  - Each router with a wrong setting will result in **5 points** deduction
- **(20%) Ping test**
  - **Every host** in the topology should be able to ping each other
  - Each host failing to reach some hosts will result in **7 points** deduction



# Submission Naming Convention

- Rename your directory as **lab3\_<student ID>**
- Compress the directory into a **zip** file named **lab3\_<student ID>.zip**
- Upload your zip file to [E3](#)
- Wrong file name or format will result in 10 points deduction
- 20% deduction for late submission in one week
  - Won't accept submissions over one week





# About help!

- For any lab problem, ask at the e3 forum
  - Ask at the e3 forum
  - TAs will help to clarify lab contents instead of giving answers!
  - Please describe your questions with sufficient context,
    - E.g., Environment setup, Input/Output, Screenshots, ...
- For personal problems, mail to [sdnta@win.cs.nctu.edu.tw](mailto:sdnta@win.cs.nctu.edu.tw)
  - You have particular problems so that you can't meet the deadline
  - You got a weird score with the lab
- No Fixed TA hour



# Outline

- Introduction
- Docker Installation
- Docker Usage
- Example Scenario Setup
- Introduction to Docker Compose
- Lab 3 Overview
- Submission & Scoring Criteria
- References



# References

- [Docker overview](#)
- [Docker commandline reference](#)
- [Docker compose reference](#)
- [Opencord/quagga Github](#)