



Project 6

Network Function Virtualization:
Software Router and Containerization

Deadline: 2022/12/07 (WED) 23:59



Outline

- Introduction
- Docker Installation
- Docker Usage
- Example Scenario Setup
- Introduction to Docker Compose
- Project 6 Overview
- Submission & Scoring Criteria
- References



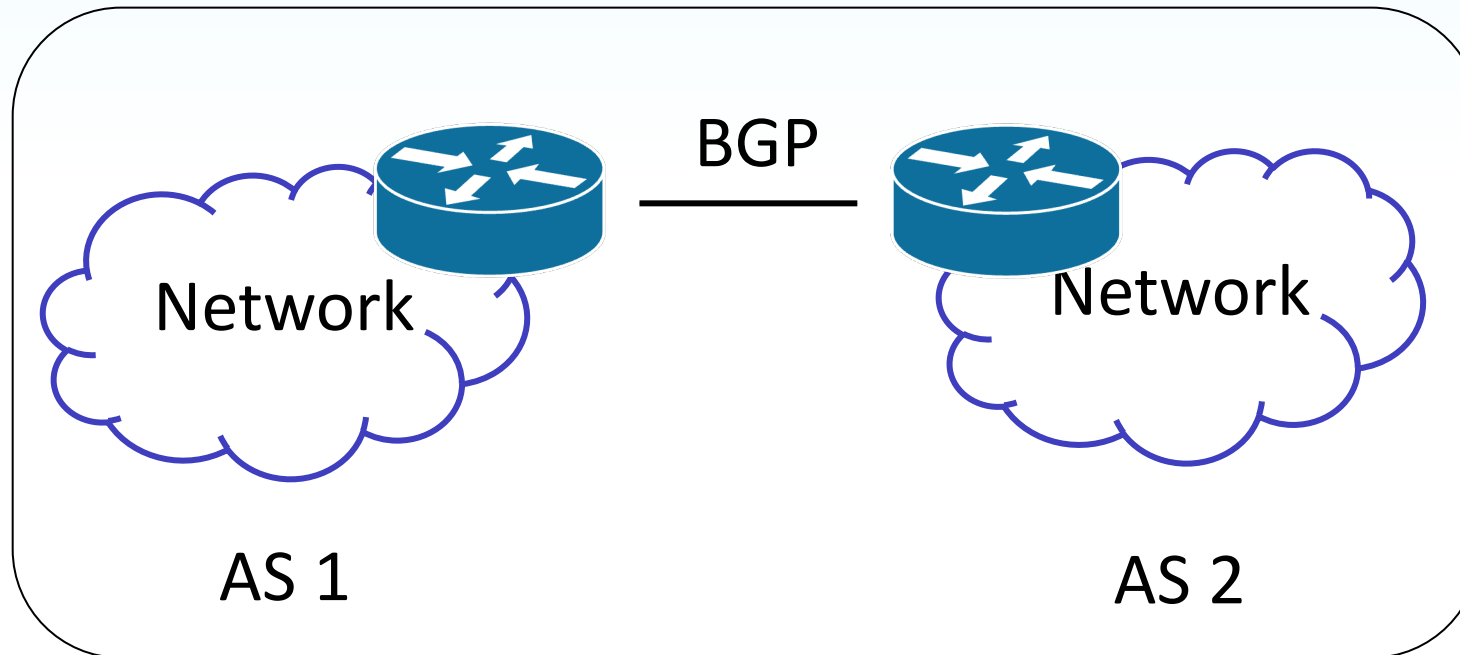
Outline

- Introduction
- Docker Installation
- Docker Usage
- Example Scenario Setup
- Introduction to Docker Compose
- Project 6 Overview
- Submission & Scoring Criteria
- References



Example Scenario

- Interconnection of two networks



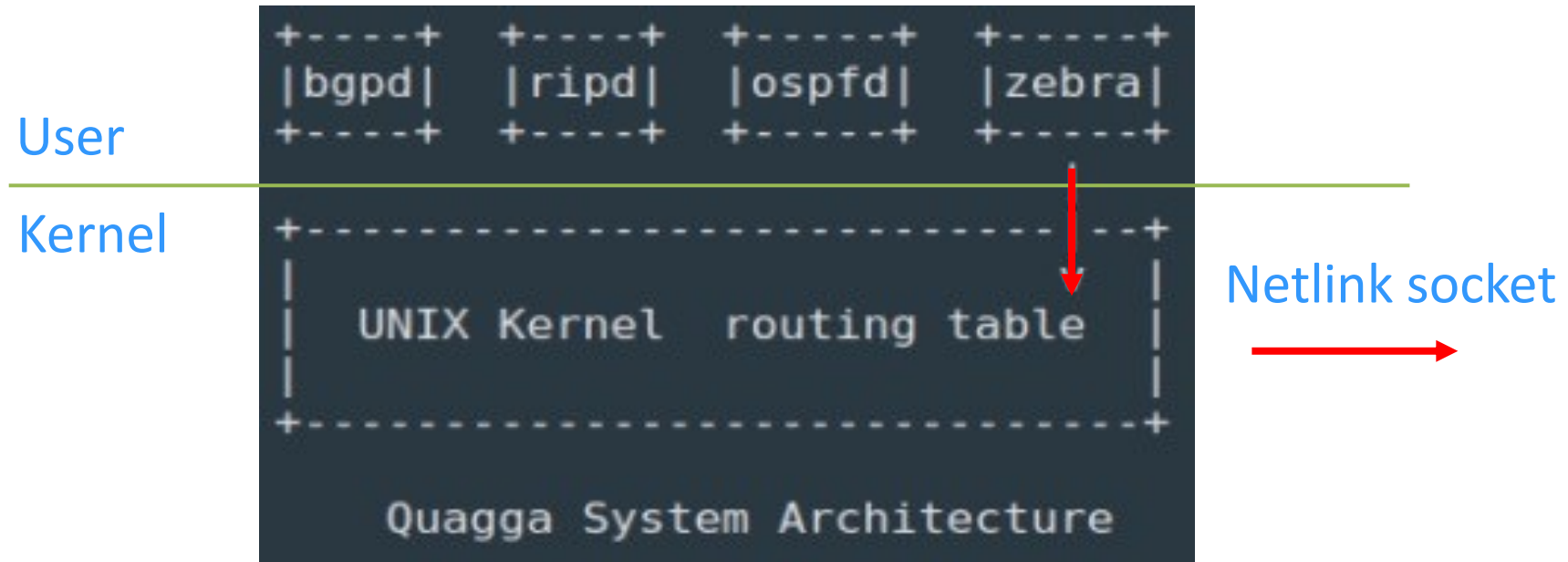
- **BGP**: Broder Gateway Protocol
- **AS**: Autonomous System



Introduction to Quagga

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

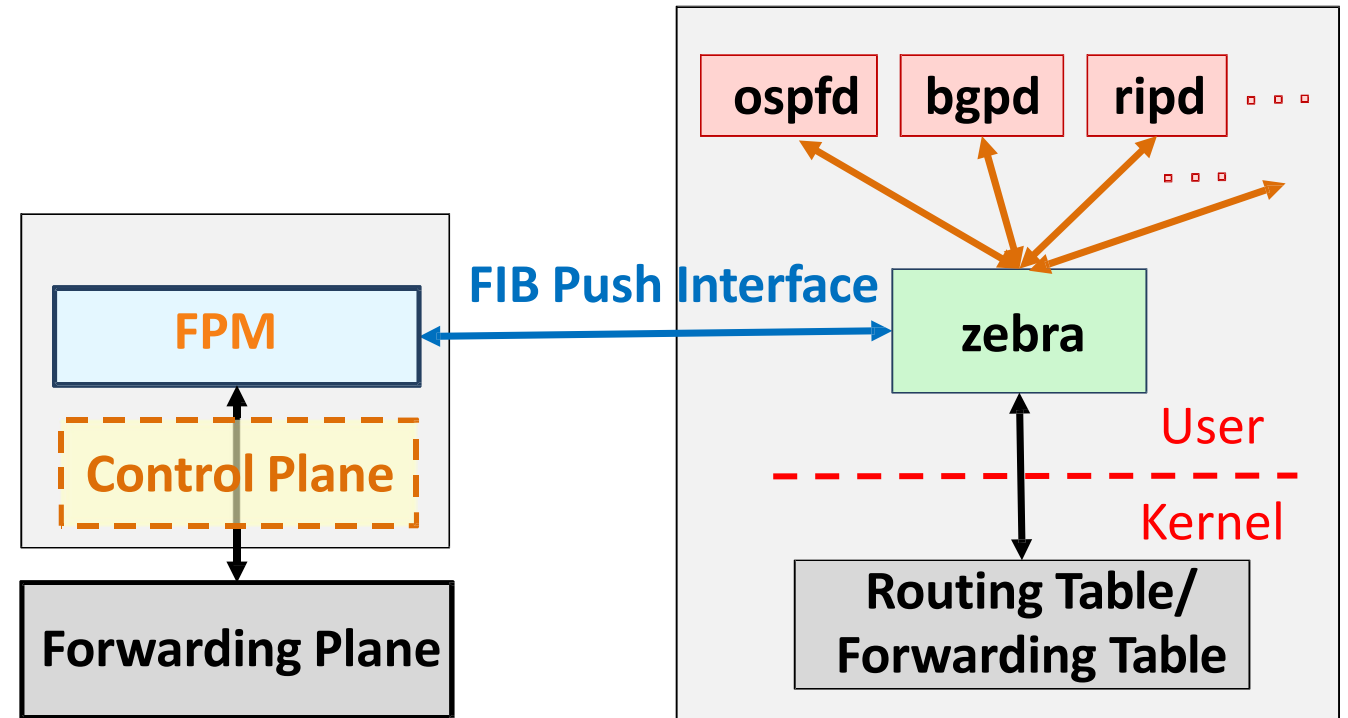
- Supports common routing protocols: BGP, OSPF, RIP, and IS-IS
- Consists of a **core daemon Zebra** and separate **routing protocol** daemons
- Routing Protocols (daemons) communicate their best routes to Zebra
- Zebra computes best routes and modifies **kernel routing table** through netlink





Introduction to FIB Pushing

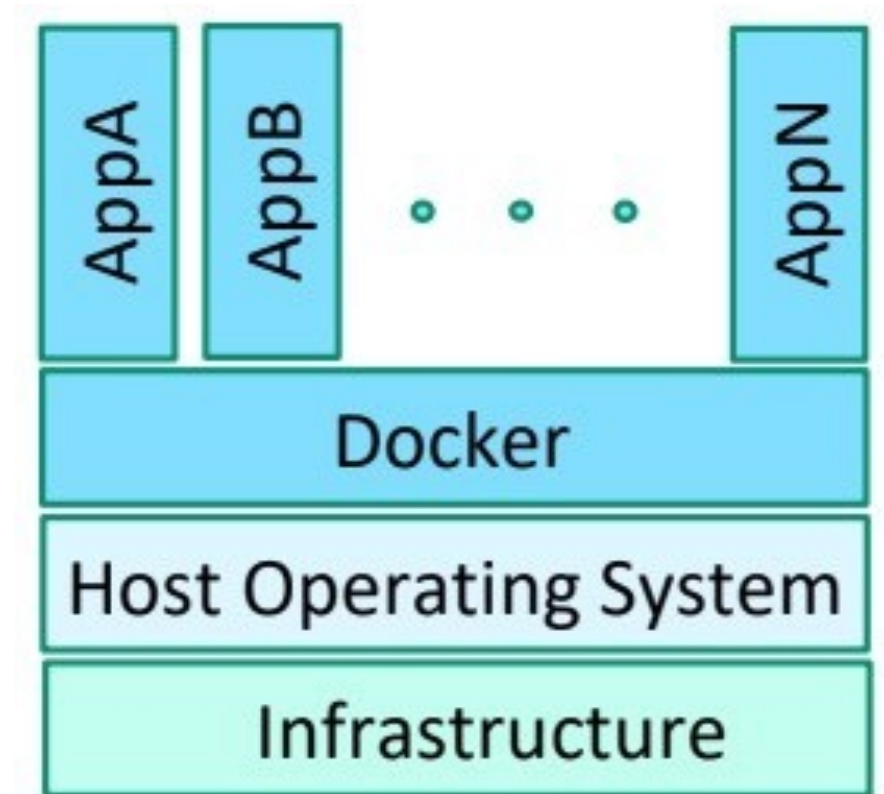
- Zebra supports a FIB Push Interface (FPI)
 - FPI allows an **external component** to **learn** the forwarding information
- Forwarding Plane Manager (FPM):
 - Receives FIB
 - Decode FIB into routes
 - Programs forwarding plane
- FIB Pushing:
 - FPM establishes a TCP connection with zebra
 - Zebra pushes 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
- Project 6 Overview
- Submission & Scoring Criteria
- References



Installation of Docker

- Update apt (confirm to install the latest package)

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

- Install curl for data transfer

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

- Retrieve Docker installation script and install Docker

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

- Manage Docker as a not-root user

```
bash$ sudo groupadd docker
```

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

```
bash$ newgrp docker
```



Outline

- Introduction
- Docker Installation
- **Docker Usage**
- Example Scenario Setup
- Introduction to Docker Compose
- Project 6 Overview
- Submission & Scoring Criteria
- References



Pull Image

- Usage

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

- E.g., Pull ubuntu:16.04 image from Docker hub registry

```
bash$ docker pull ubuntu:16.04
```

- List images

```
bash$ docker images
```

```
cu@SDN-NFV:~$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
ubuntu        16.04     b6f507652425  14 months ago 135MB
```



Docker run (1/2)

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

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

Create and Run a container Run a command in the new container

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

```
bash$ 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



Docker run (2/2)

- List containers

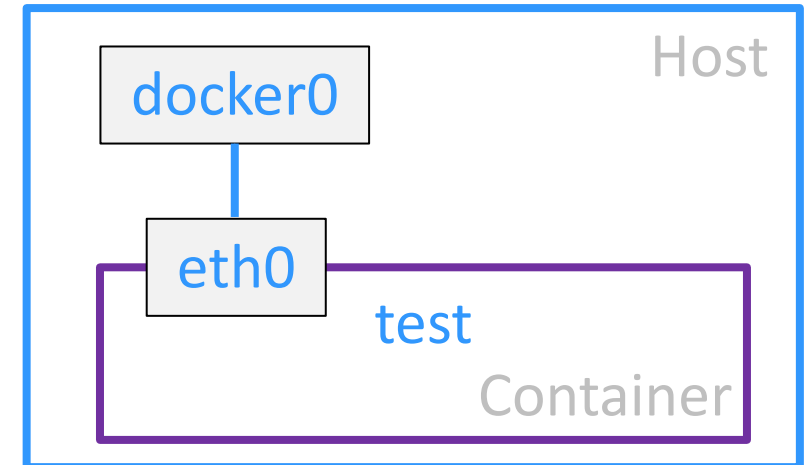
```
bash$ docker ps -a
```

- “--all”, “-a”: Show all containers

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

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
9eae738310dc	ubuntu:16.04	"/bin/bash"	8 seconds ago	Up 7 seconds		test

- Container is connected to docker0 bridge by default
- Default IP for docker0 bridge is 172.17.0.1/16





Docker exec

- Execute a command in a running container
- Usage

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

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

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

```
cu@SDN-NFV:~$ docker exec -it test bash
root@9eae738310dc:/#
```



Docker network - Create

- Create a bridge network
- Usage

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

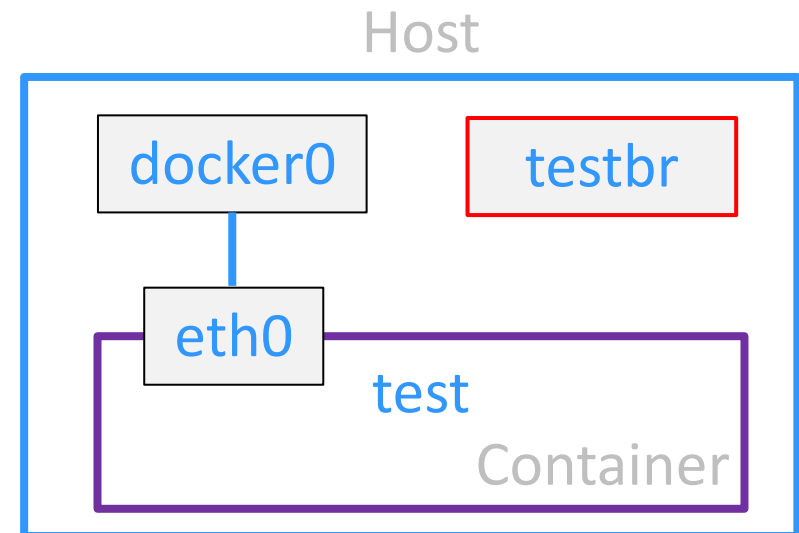
- E.g., Create a network bridge "testbr"

```
bash$ docker network create testbr
```

- List networks

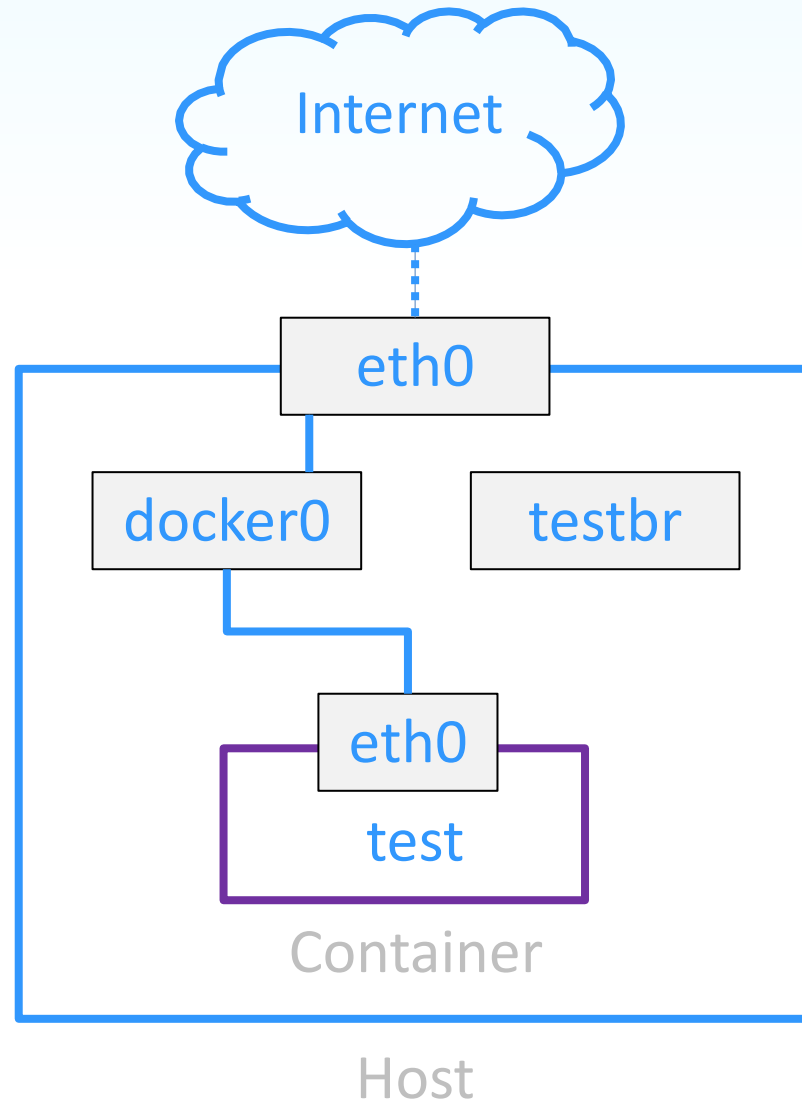
```
bash$ docker network ls
```

```
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

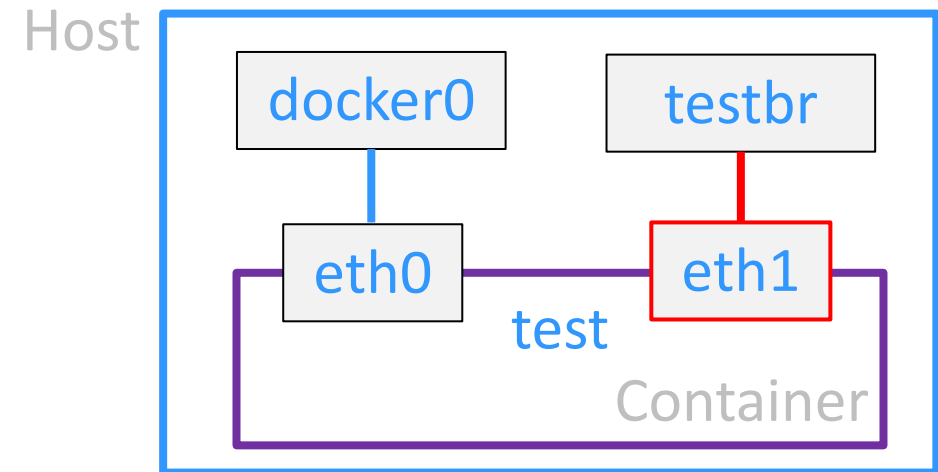
- Connect to a bridge network
- Usage

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

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

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

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





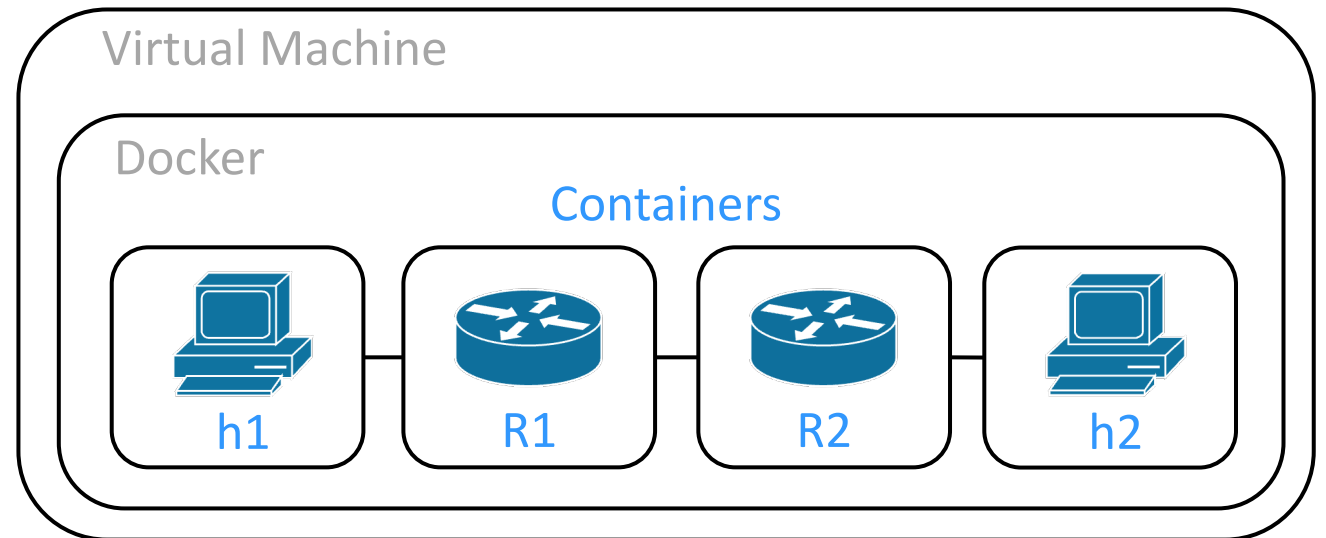
Outline

- Introduction
- Docker Installation
- Docker Usage
- Example Scenario Setup
- Introduction to Docker Compose
- Project 6 Overview
- Submission & Scoring Criteria
- References



Steps to Setup Example Scenario

1. Prepare Docker images
2. Create containers
3. Setup container networks
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**

```
bash$ vi host.Dockerfile
```

```
FROM ubuntu:16.04
```

```
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

```
bash$ 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

```
bash$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
host	latest	bfe912e02831	11 seconds ago	172MB
ubuntu	16.04	b6f507652425	14 months ago	135MB



Step 1 – Prepare Docker images (3/3)

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

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

- List image

```
bash$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
host	latest	bfe912e02831	2 minutes ago	172MB
ubuntu	16.04	b6f507652425	14 months ago	135MB
opencord/quagga	latest	2c638cd24154	5 years ago	457MB



Step 2 – Create Containers (1/2)

- Create a container with prepared image

```
bash$ 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



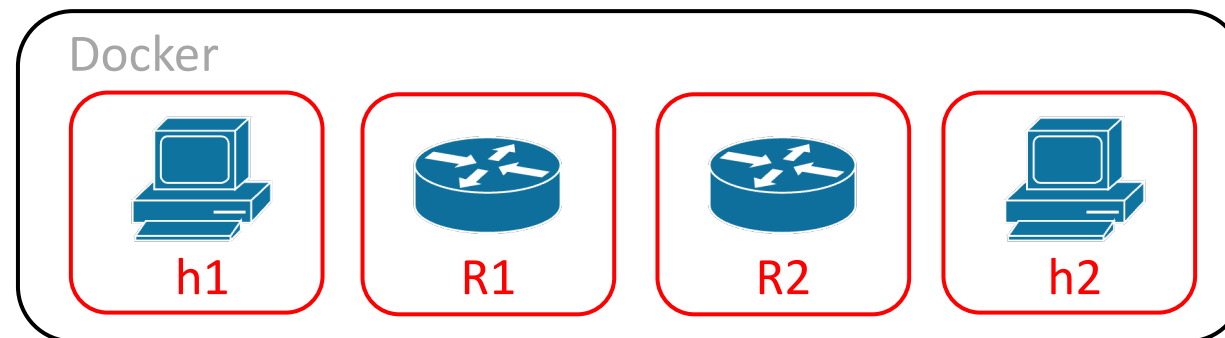
Step 2 – Create Containers (2/2)

- Create container for a host h1 (h2)

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

- Create container for a virtual router R1 (R2)

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





Step 3 – Setup Container Networks (1/3)

- Create a bridge network R1h1br

```
bash$ docker network create R1h1br
```

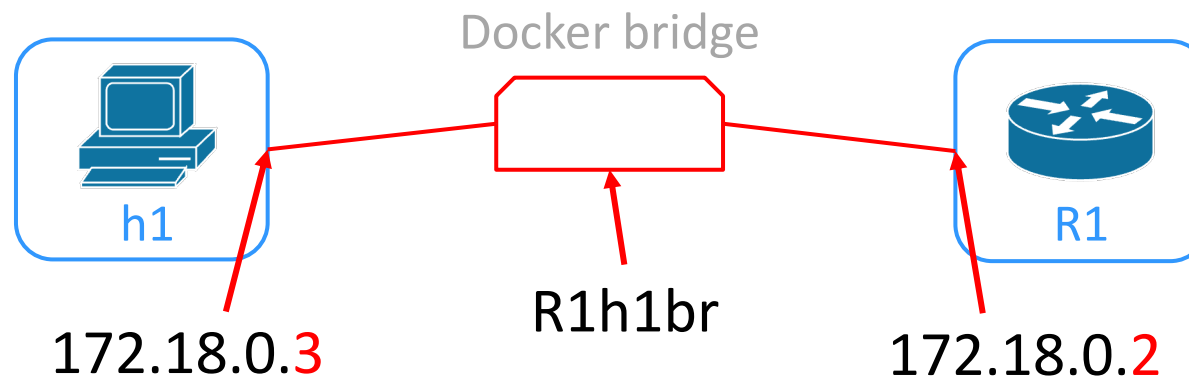
- R1h1br: Bridge name

- Connect containers h1 and R1 to bridge R1h1br

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

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

- Docker will assign IP to interfaces automatically





Step 3 – Setup Container Networks (2/3)

- Check the IP address of network interface

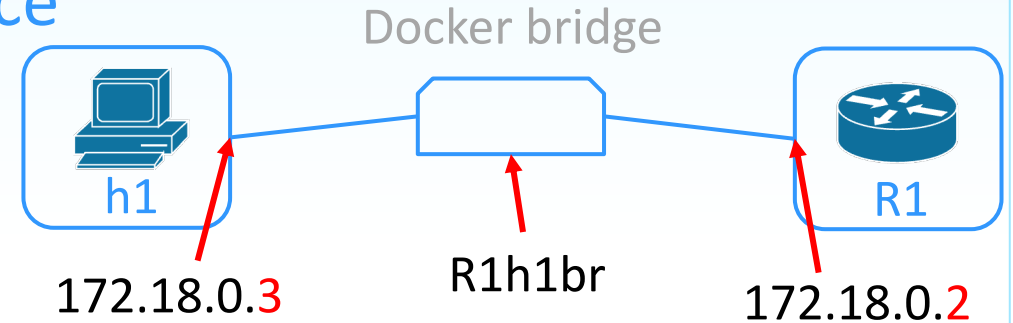
```
bash$ docker inspect h1 (R1)
```

■ h1

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

■ R1

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



- Repeat network setup procedure for each domain



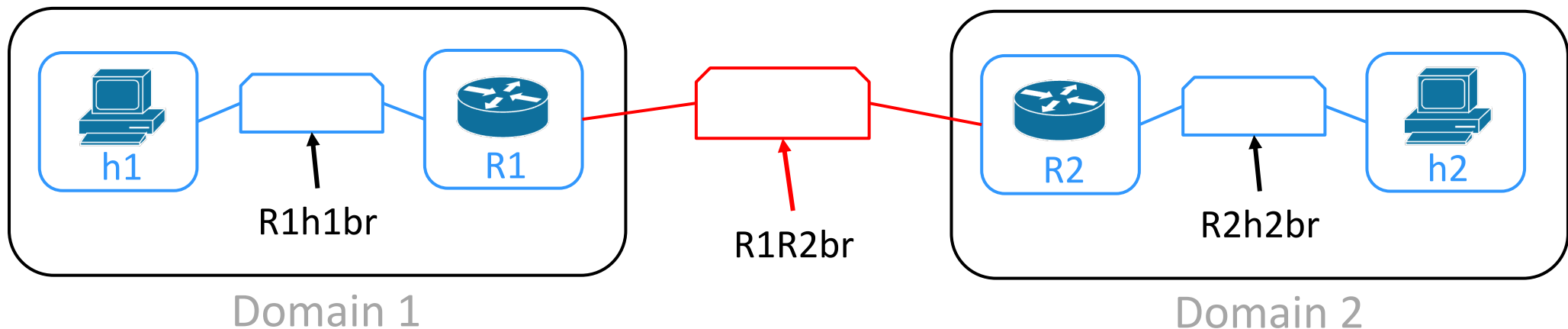
Step 3 – Setup Container Networks (3/3)

- Connect two domains
- Create inter domain bridge

```
bash$ docker network create R1R2br
```

- Connect containers R1 and R2 to bridge R1R2br

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





Step 4 – Configure Host Gateways (1/1)

- Run bash on h1 (h2)

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

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

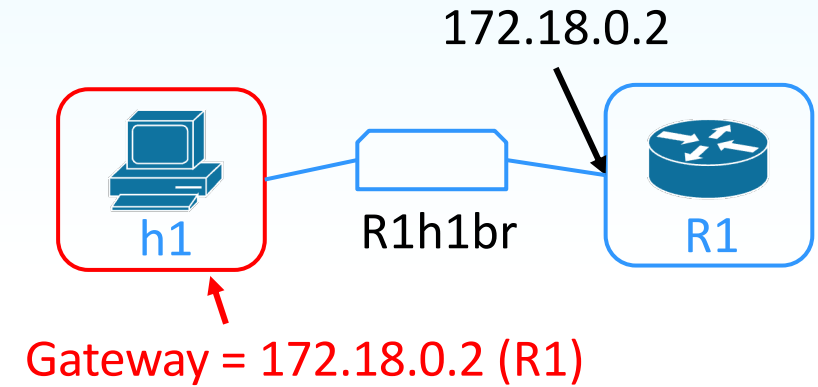
```
/# ip route del default
```

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

- Check route on h1 (h2)

```
/# route
```

```
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
```





Step 5 – Setup Routers (1/4)

- Run bash on R1 (R2)

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

- Enable IP forwarding on R1 (R2)

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

- Uncomment

“net.ipv4.ip_forward=1”

- Run sysctl to load configuration

```
/# 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)

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

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

```
/# 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



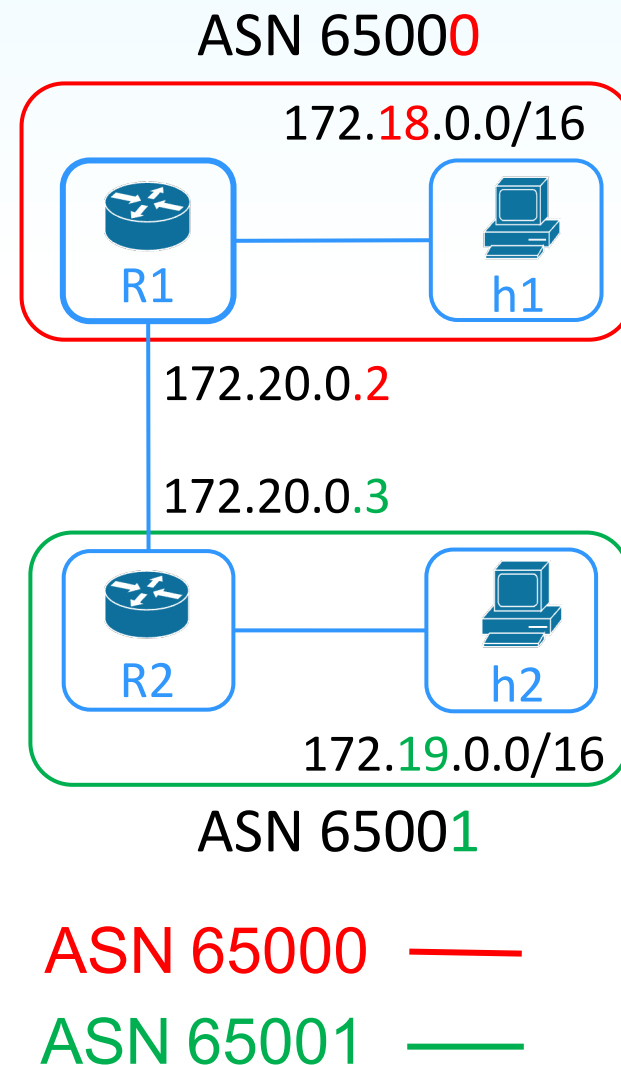
Step 5 – Setup Routers (3/4)

- Set BGP configuration of routers

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

```
#!/# 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
```





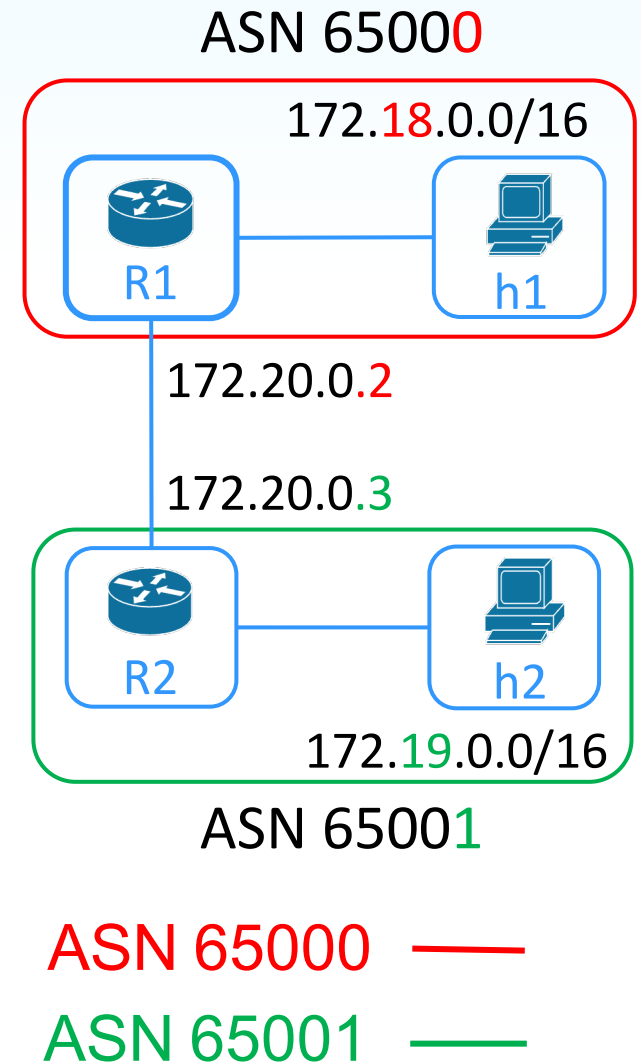
Step 5 – Setup Routers (4/4)

- Set BGP configuration of routers

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





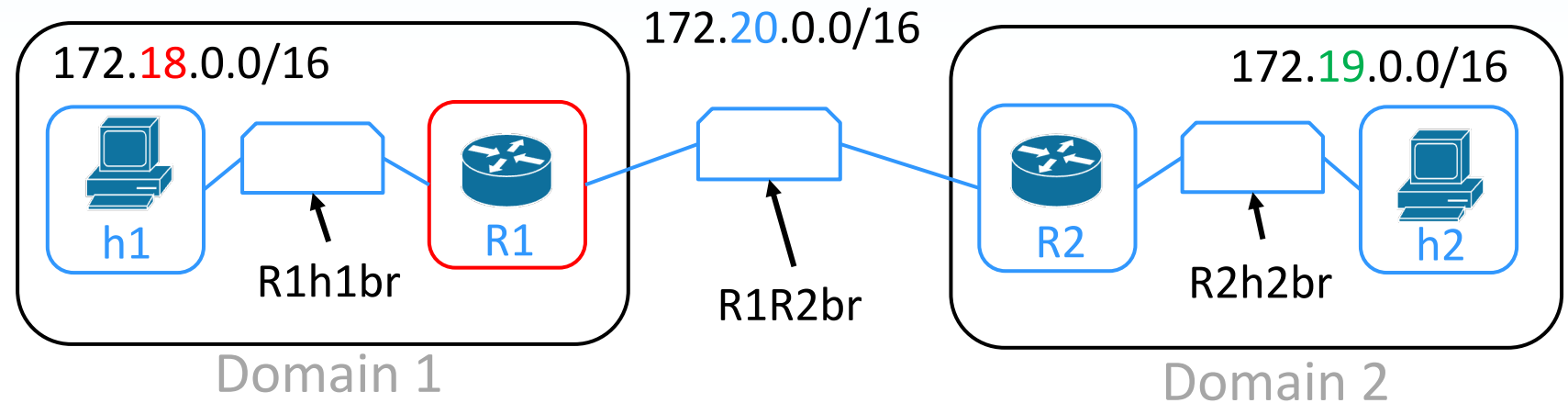
Step 6 – Check Routes (1/3)

- Restart Quagga on R1 (R2)

```
bash$ docker restart R1
```

- Check route on R1 (R2) bash

```
/# 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)

```
/# 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)

```
/# 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
```



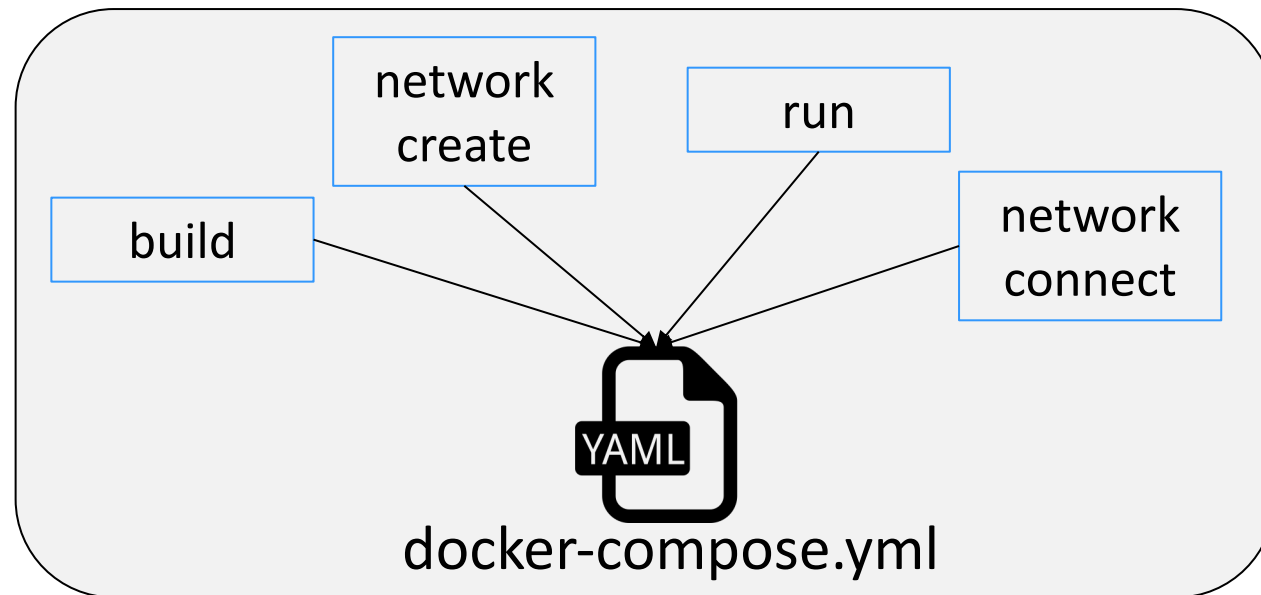
Outline

- Introduction
- Docker Installation
- Docker Usage
- Example Scenario Setup
- Introduction to Docker Compose
- Project 6 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

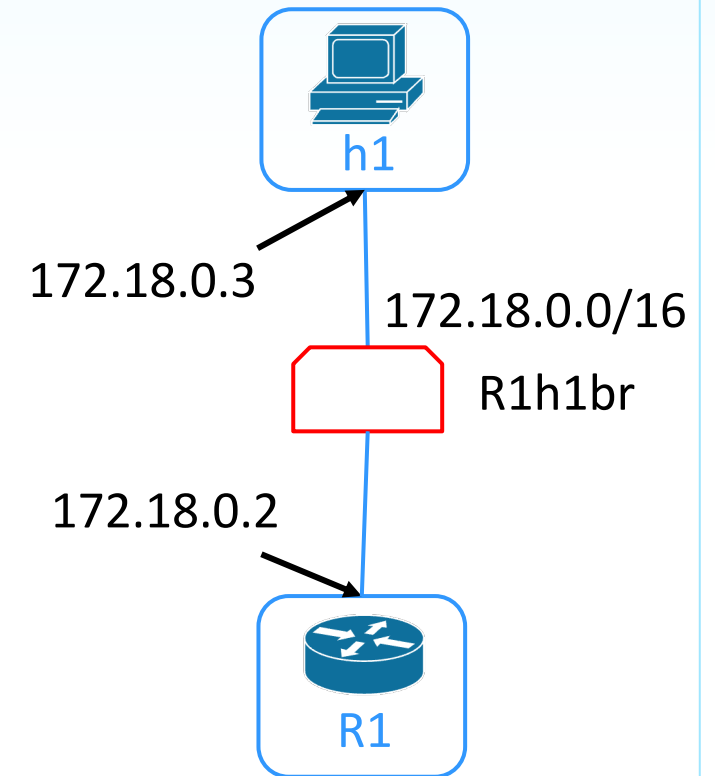
docker-compose.yml



Docker Compose Example (1/4)

- Create bridge R1h1br

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

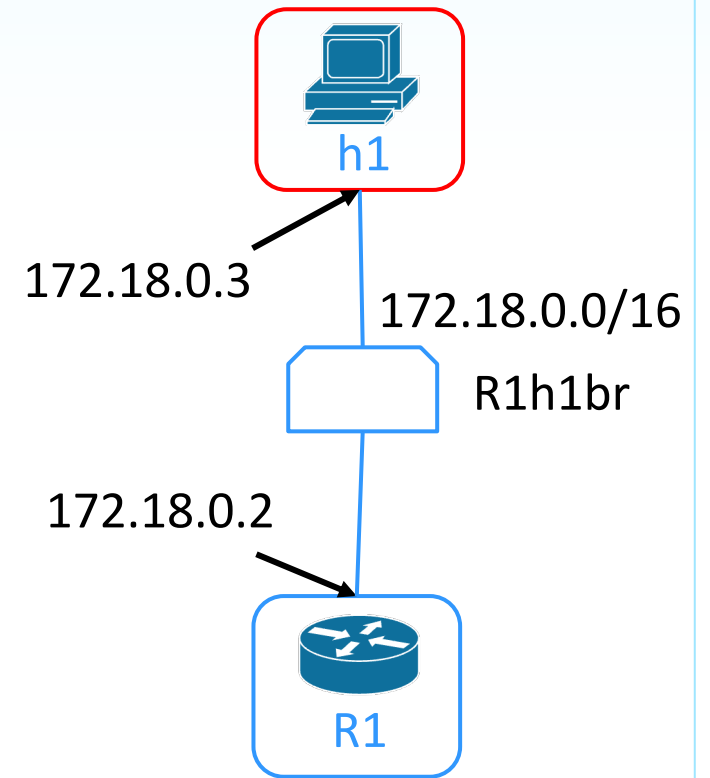




Docker Compose Example (2/4)

• Create container h1

```
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
```

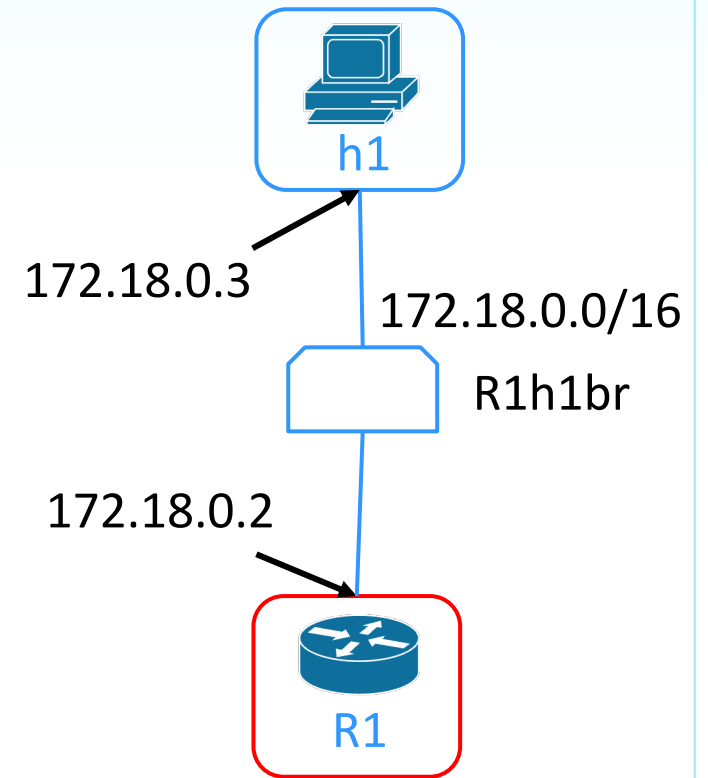




Docker Compose Example (3/4)

• Create container R1

```
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

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

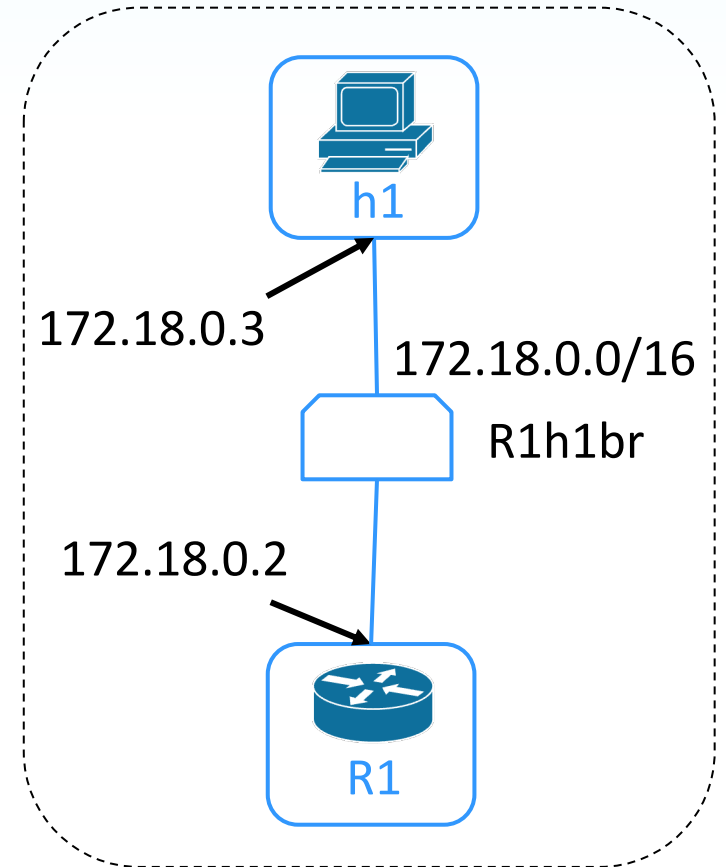
- List the running containers

```
bash$ docker ps
```

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

- Stop and remove the Docker application

```
bash$ docker compose down
```



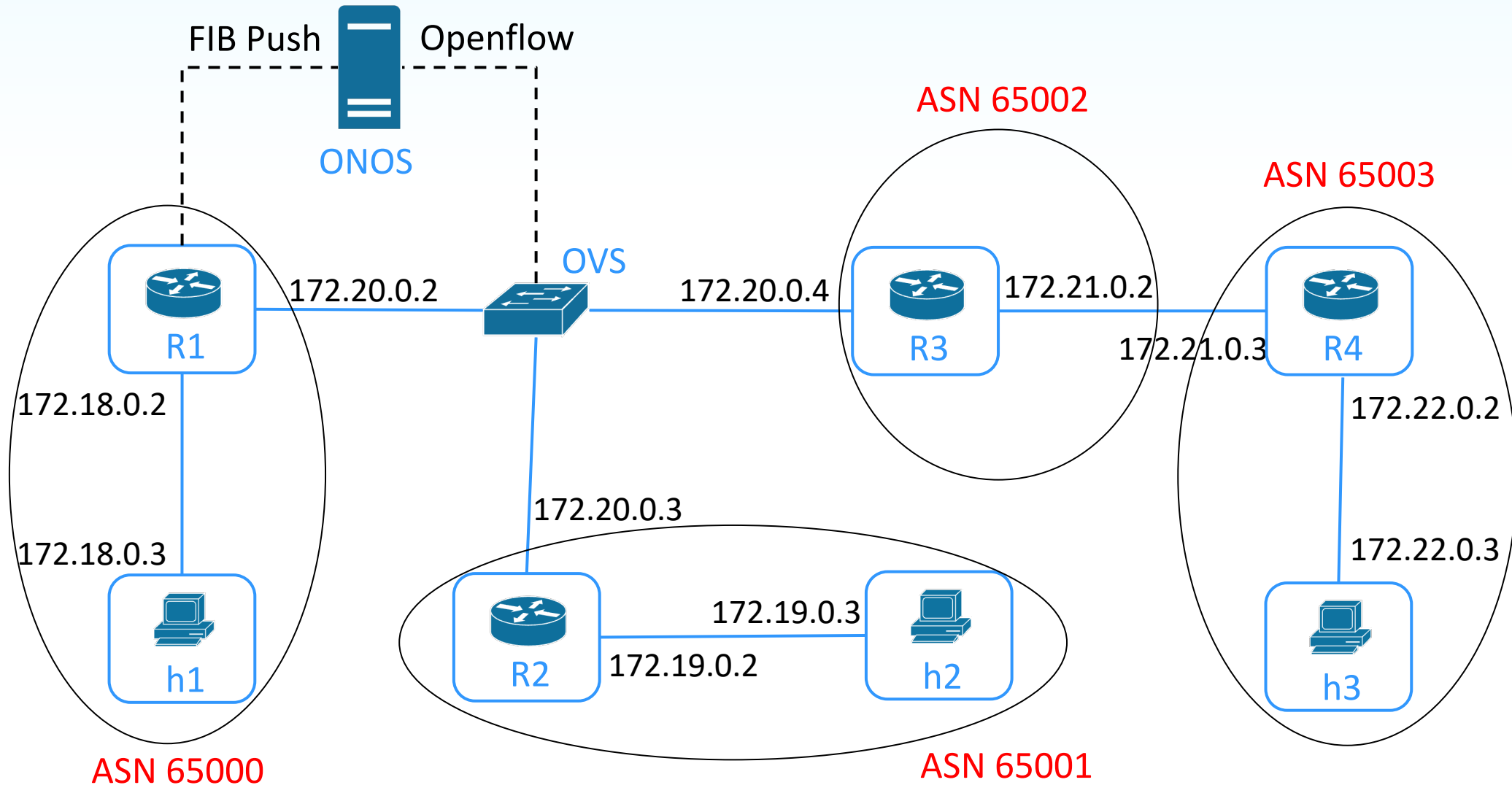


Outline

- Introduction
- Docker Installation
- Docker Usage
- Example Scenario Setup
- Introduction to Docker Compose
- **Project 6 Overview**
 - Overview & Workflow
 - How to Test your App
 - Supplements
- Submission & Scoring Criteria
- References



Topology of Project 6

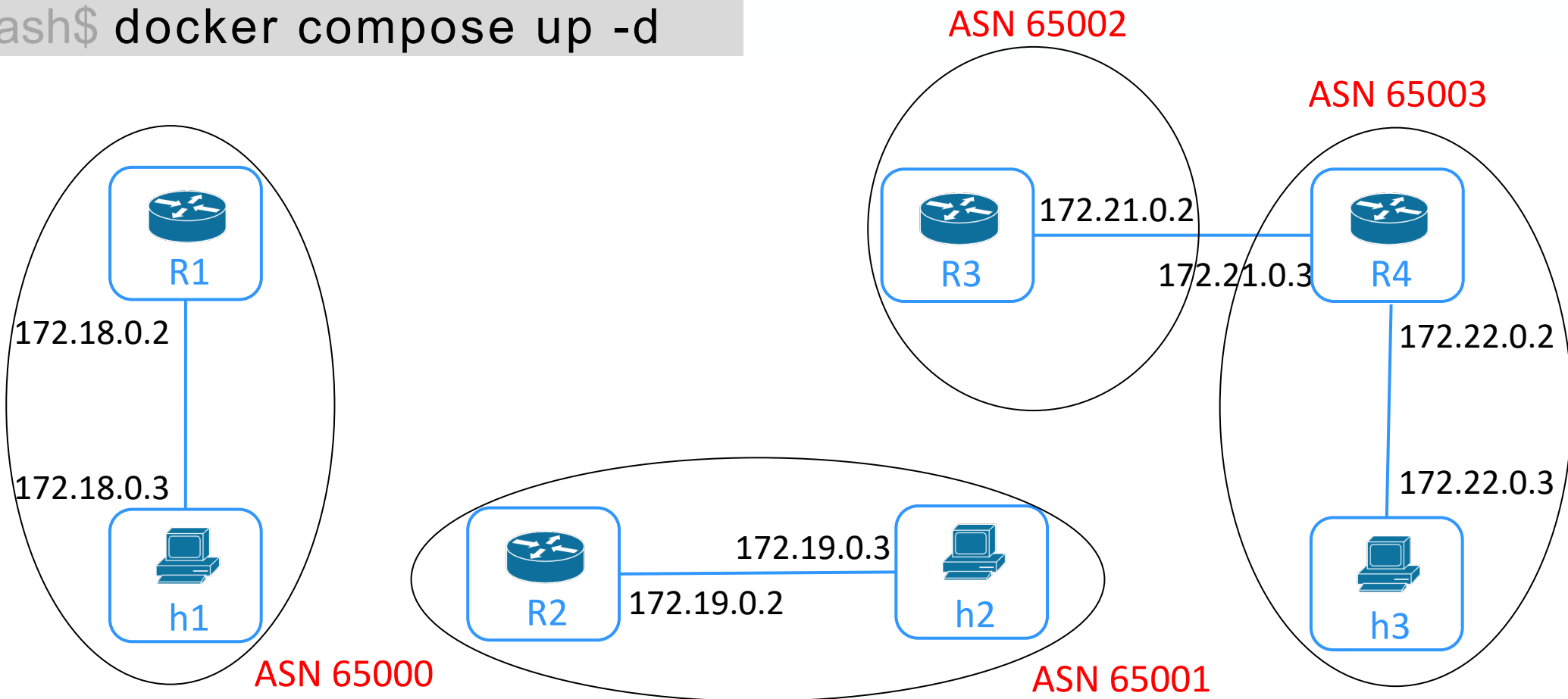




Step 1 – Create Containers

- Prepare docker-compose.yml for following network segments
- Create containers and networks with your docker-compose.yml

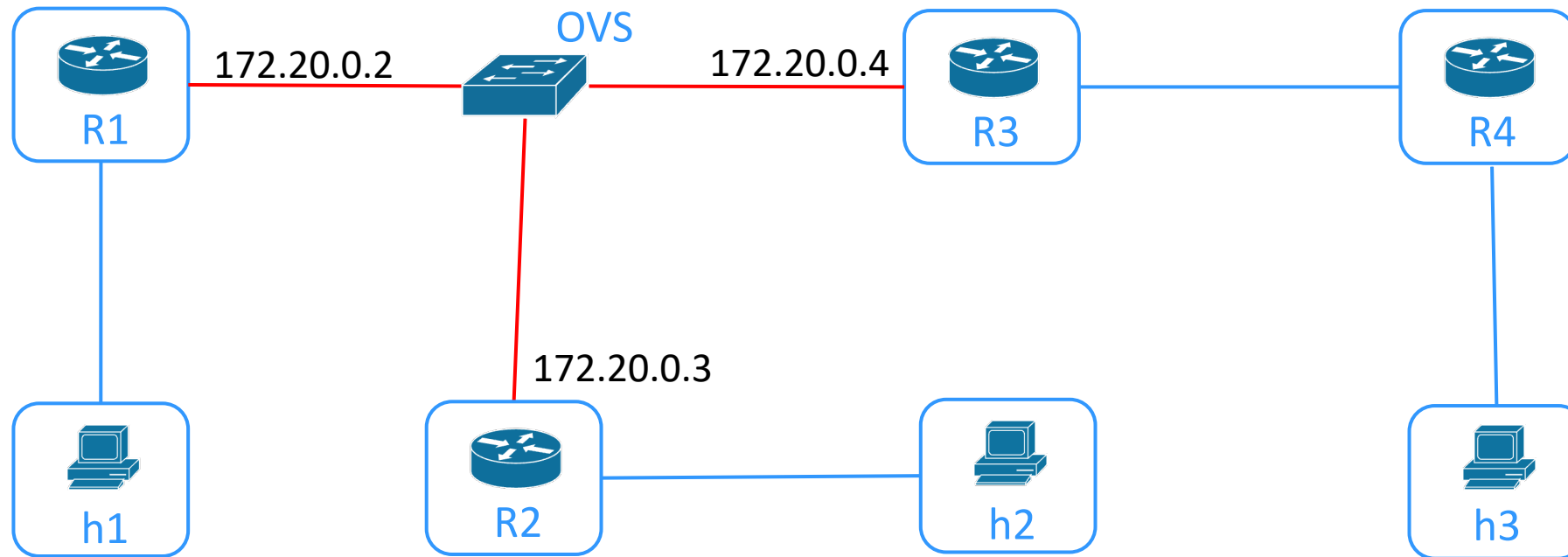
```
bash$ docker compose up -d
```





Step 2 – Create OVS Bridge

- Create an OVS bridge to connect to network segments
 - Use ovs-vsctl to create OVS bridge
 - Use ip or ovs-docker to set a connection between bridge and container



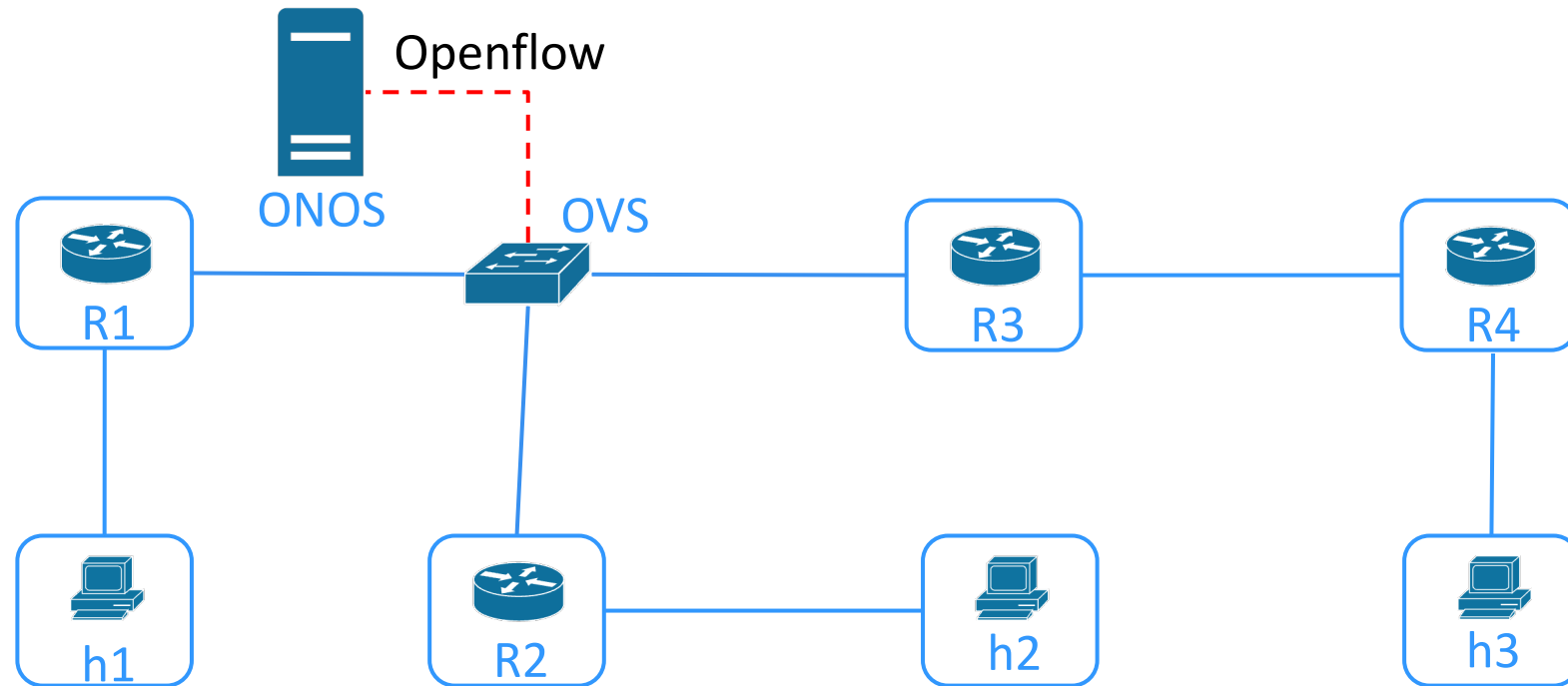


Step 3 – Enable ONOS fwd Application

- Enable ONOS reactive forwarding

```
onos@root > app activate fwd
```

- After activate fwd, the hosts can communicate with each other



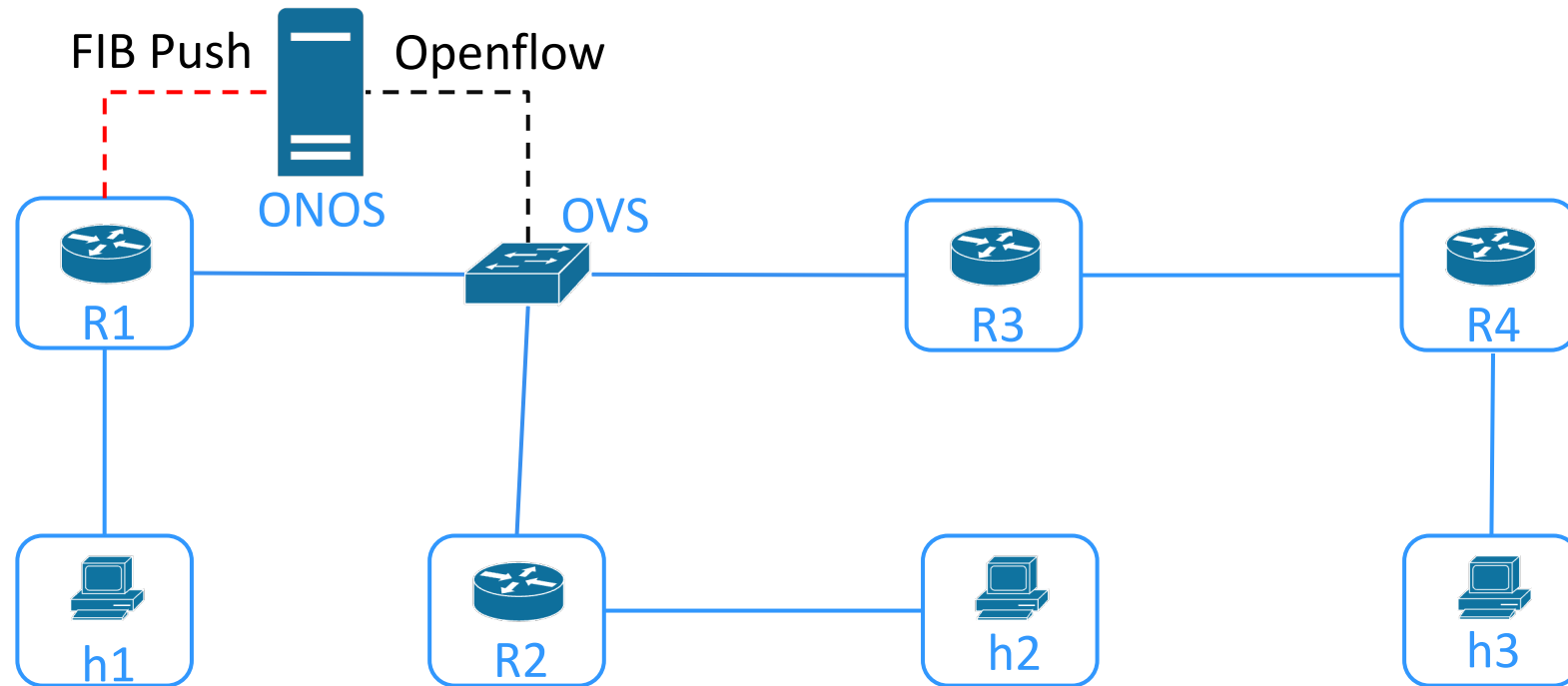


Step 4 – Enable FPI and FPM

- Set FIB Push configuration on R1
- Enable ONOS FPM manager

```
onos@root > app activate fpm
```

- Then, ONOS can receive FIB information from R1





Step 5 - Create Makefile

- Write a **Makefile** for your application which contains:
 - Container creation
 - OVS bridge creation
 - Link setup
- **Do not** activate any ONOS application
- Create the application with a single command

```
bash$ make
```

- Stop the application with a single command

```
bash$ make clean
```




Outline

- Introduction
- Docker Installation
- Docker Usage
- Example Scenario Setup
- Introduction to Docker Compose
- **Project 6 Overview**
 - Overview & Workflow
 - **How to Test your App**
 - Supplements
- Submission & Scoring Criteria
- References



How to Test Your App (1/3)

- Check IP configuration

```
bash$ 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
```

- Activate ONOS fwd application

```
onos@root > app activate fwd
```

- Check host communication

```
bash$ docker exec -it h1 ping 172.22.0.3 -c 3
```



How to Test Your App (2/3)

- Run bash on R1

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

- Telnet R1 bgpd daemon (on port 2605)

```
/# telnet localhost 2605
```

- Show R1 bgp

```
R1bgp> show ip bgp
```

```
BGP table version is 0, local router ID is 172.20.0.2
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      0.0.0.0             0         32768 i
* 172.19.0.0      172.20.0.3          0         0 65002 65001 i
*>                172.20.0.3          0         0 65001 i
* 172.20.0.0      172.20.0.4          0         0 65002 i
*                172.20.0.4          0         0 65001 65002 i
*> 172.21.0.0      172.20.0.4          0         0 65002 i
*                172.20.0.4          0         0 65001 65002 i
*> 172.22.0.0      172.20.0.4          0         0 65002 65003 i
*                172.20.0.4          0         0 65001 65002 65003 i
```



How to Test Your App (3/3)

- Activate ONOS FIB Push Manager (FPM)

```
onos@root > app activate fpm
```

- Show routing message from ONOS

```
onos@root > routes
```

```
cu@root > routes
```

B: Best route, R: Resolved route

Table: ipv4

B	R	Network	Next Hop	Source (Node)
		172.19.0.0/16	172.20.0.3	FPM (127.0.0.1)
		172.21.0.0/16	172.20.0.4	FPM (127.0.0.1)
		172.22.0.0/16	172.20.0.4	FPM (127.0.0.1)

Total: 3

Table: ipv6

B	R	Network	Next Hop	Source (Node)
---	---	---------	----------	---------------

Total: 0



Outline

- Introduction
- Docker Installation
- Docker Usage
- Example Scenario Setup
- Introduction to Docker Compose
- **Project 6 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_project6/  
├── config  
│   └── R1  
│       ├── bgpd.conf  
│       └── zebra.conf  
├── docker-compose.yml  
└── host.Dockerfile  
  
2 directories, 4 files
```

Note: **Not necessary** to use Docker compose to complete this project.



Outline

- Introduction
- Docker Installation
- Docker Usage
- Example Scenario Setup
- Introduction to Docker Compose
- Project 6 Overview
- Submission & Scoring Criteria
- References



Scoring Criteria (1/3)

- Create Makefile
 - You **must** provide **Makefile**, which creates an application with a **make** command
 - The Makefile should be **placed at the top layer** of the project directory
 - TA will use your Makefile to build your application
 - If you do not provide Makefile, you will **not earn any credit**
- **(20%)** Create containers
 - Use Docker to create **three** hosts and **four** routers containers
 - You lose **three credits** for each container that is not created
- **(10%)** Create an OVS bridge
 - You **must** use the ovs-vsctl command
 - You will **not earn any credit** if you do not conform to the above regulation



Scoring Criteria (2/3)

- **(20%) 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
 - Use ip or ovs-docker to set a connection between a bridge and a container
 - Each link with a wrong setting will result in **3 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
- **(15%) Ping test**
 - **Every host** in the topology should be able to ping each other
 - Each host failing to reach some hosts will result in **5 points** deduction



Scoring Criteria (3/3)

- (15%) Setup FIB Pushing
 - ONOS can receive FIB information from R1
- Reminders
 - Do not write any ONOS-related settings in your Makefile
 - We will activate fwd and fpm before testing your app

```
cu@root > apps -a -s
* 3 org.onosproject.route-service 2.7.0 Route Service Server
* 11 org.onosproject.drivers 2.7.0 Default Drivers
* 15 org.onosproject.fwd 2.7.0 Reactive Forwarding
* 21 org.onosproject.optical-model 2.7.0 Optical Network Model
* 23 org.onosproject.gui2 2.7.0 ONOS GUI2
* 35 org.onosproject.fpm 2.7.0 FIB Push Manager (FPM) Route Receiver
* 44 org.onosproject.openflow-base 2.7.0 OpenFlow Base Provider
* 45 org.onosproject.lldpprovider 2.7.0 LLDP Link Provider
* 46 org.onosproject.hostprovider 2.7.0 Host Location Provider
* 72 org.onosproject.openflow 2.7.0 OpenFlow Provider Suite
```



Submission Naming Convention

- Rename your directory as **project6_<student ID>**.
- Compress the directory into a **zip** file named **project6_<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

```
project6_ID/  
├── config  
│   ├── R1  
│   │   ├── bgpd.conf  
│   │   └── zebra.conf  
│   ├── R2  
│   │   ├── bgpd.conf  
│   │   └── zebra.conf  
│   ├── R3  
│   │   ├── bgpd.conf  
│   │   └── zebra.conf  
│   └── R4  
│       ├── bgpd.conf  
│       └── zebra.conf  
├── docker-compose.yml  
├── host.Dockerfile  
└── Makefile  
  
5 directories, 11 files
```



Demo

- TA will open a demo time-reserved table a week before the demo
- The dates will be chosen after the deadline
- Demo questions will appear at the start of the demo
- The score of the demo will occupy **40%** total score of this project
 - For example:
 - You earn 100% of the credits for submission
 - You earn 80% of the credits for the demo
 - then your total score for this project will be:
 - **$100 \times 60\% + 80 \times 40\% = 92.$**



About help!

- For any project problem, ask at the e3 forum
 - Ask at the e3 forum
 - TAs will help to clarify project 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
 - You have particular problems so that you can't meet the deadline
 - You got a weird score with the project
- No Fixed TA hour



Outline

- Introduction
- Docker Installation
- Docker Usage
- Example Scenario Setup
- Introduction to Docker Compose
- Project 6 Overview
- Submission & Scoring Criteria
- References



References

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