Objective 1

Section 1 - Container networking using default Docker bridge network

First create the Dockerfile that the guests will fun from:

FROM debian:latest

# Update apt-get and install necessary packages

RUN apt-get update && apt-get install -y \

    net-tools \

    iproute2 \

    tcpdump \

    iputils-ping \

    curl \

    wget \

    vim \

    dnsutils \

    traceroute \

 && rm -rf /var/lib/apt/lists/\*

Then, run the commands to pull the latest Debian image, create the guest image and then proceed to start both guest VMs, find out the IP address of either and initiate a ping:

docker pull debian

docker build -t guest .

docker run -dit --name guest\_1 guest bash

docker run -dit --name guest\_2 guest bash

docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' guest\_1

docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' guest\_2

docker exec -it guest\_1 bash

ping 172.17.0.3

A screen shot of a computer code

AI-generated content may be incorrect.

Section 2 - Container networking using Open vSwitch and SDN controller container

Step 1: Install OVS

sudo apt-get update

sudo apt-get -y install openvswitch-switch

Step 2: Create OVS bridge and assign IP:

sudo ovs-vsctl add-br ovs-br1

sudo ifconfig ovs-br1 173.16.1.1 netmask 255.255.255.0 up

Step 3: Run the Docker image for RYU

docker run -dit --name ryu\_cont --network=host osrg/ryu

Step 4: Add IPs to previously created images

sudo ovs-docker add-port ovs-br1 eth1 guest\_1 --ipaddress=173.16.1.2/24

sudo ovs-docker add-port ovs-br1 eth1 guest\_2 --ipaddress=173.16.1.3/24

Step 5: Run RYU Controller on container

docker exec -it ryu-controller ryu-manager ryu.app.simple\_switch\_13

A computer screen with text

AI-generated content may be incorrect.

Step 6: Check connectivity and configuration:

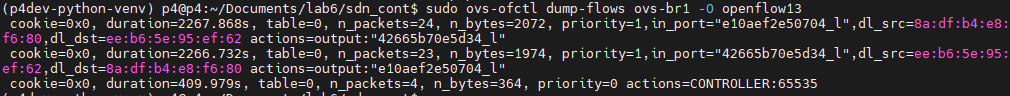
sudo ovs-vsctl show

sudo ovs-ofctl dump-flows ovs-br1 -O openflow13

sudo ovs-ofctl show ovs-br1 -O openflow13

A computer screen shot of a black screen

AI-generated content may be incorrect.



A screenshot of a computer

AI-generated content may be incorrect.

Ping connectivity:

A screen shot of a computer

AI-generated content may be incorrect.

Section 3 - Container networking using Open vSwitch and SDN controller running on host

Step 1: Install Ryu on the VM itself

pip install ryu

Step 2: Start the controller

ryu-manager ryu.app.simple\_switch\_13

A screenshot of a computer program

AI-generated content may be incorrect.

Step 3: Connect the controller to OVS

sudo ovs-vsctl set-controller ovs-br1 tcp:127.0.0.1:6633

Step 4: Verify connectivity

Refer to section 1 for how to set up a guest container and complete steps 1, 2 and 4 in section 2 to set up the OVS switch

A screenshot of a computer program

AI-generated content may be incorrect.

Objective 2

For objective 2, install VirtualBox and start 2 Ubuntu 22.04 images. Per the lab documentation, configure 2 host only adapters as well

Step 1: Install docker on host 1 and host 2 and create a docker bridge network for VxLAN

sudo apt update

sudo apt install -y docker.io

sudo docker network create --subnet 172.18.0.0/16 vxlan-net

Step 2: Create Dockerfiles on each machine to set up both guest images and start the images

Dockerfile:

FROM debian:latest

# Update apt-get and install necessary packages

RUN apt-get update && apt-get install -y \

    net-tools \

    iproute2 \

    tcpdump \

    iputils-ping \

    curl \

    wget \

    vim \

    dnsutils \

    traceroute \

 && rm -rf /var/lib/apt/lists/\*

Running the containers:

sudo docker run --name guest\_1 -d --net vxlan-net --ip 172.18.0.11 debian\_host sleep 3000

sudo docker run --name guest\_2 -d --net vxlan-net --ip 172.18.0.12 debian\_host sleep 3000

Step 3: Create VxLAN tunnel

Host1:

sudo ip link add vxlan-host type vxlan id 100 remote 192.168.56.3 dstport 4789 dev eth1

sudo ip link set vxlan-host up

sudo brctl addif br-af9b6a05e3be vxlan-host

Adapter and bridge:

A screen shot of a computer screen

AI-generated content may be incorrect.

Host2:

sudo ip link add vxlan-host type vxlan id 100 remote 192.168.56.2 dstport 4789 dev eth1

sudo ip link set vxlan-host up

sudo brctl addif br-ee467b029e92 vxlan-host

Adapter and bridge:

A screen shot of a computer code

AI-generated content may be incorrect.

Step 4: Test connectivity

Host2 to host1:

A screenshot of a computer screen

AI-generated content may be incorrect.

Host1 to host2:

A screenshot of a computer program

AI-generated content may be incorrect.

Packet capture from host1:

A computer screen shot of a computer

AI-generated content may be incorrect.