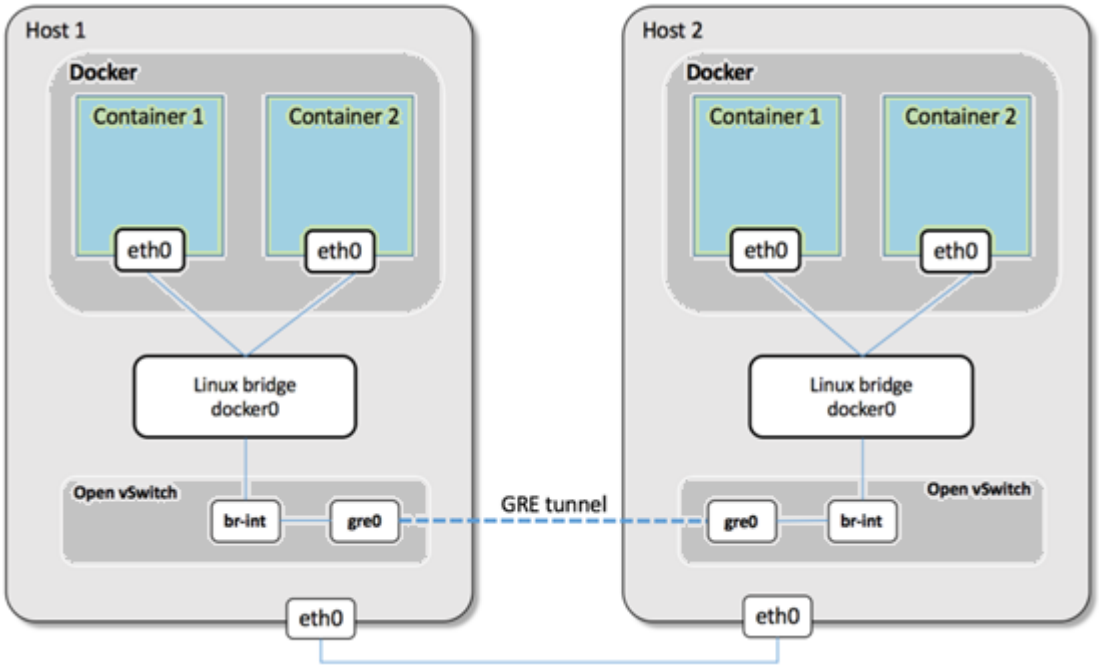
SDN in Docker - Project Report

Team Members:

Vivek Nagalapura Ravindra
Amith Kumar Matapady

Problem Statement:

To set-up a connection between two host each with docker container(s) using Open vSwitch and install OpenDaylight and its features.



Introduction:

Docker:

Docker is a set of platform as a service (PaaS) products that uses OS-level virtualization to deliver software in packages called containers. Containers are isolated from one another and bundle their own software, libraries and configuration files; they can communicate with each other through well-defined channels.

Open vSwitch:

Open vSwitch is an open-source implementation of a distributed virtual multilayer switch. br-int is the default name for the core bridge used on compute and network nodes.

Generic Routing Encapsulation:

Generic Routing Encapsulation (GRE) is a communication protocol used to establish a direct, point-to-point connection between network nodes.

OpenDaylight:

OpenDaylight (ODL) is a modular open platform for customizing and automating networks of any size and scale. In this project, we are going to launch OpenDaylight in Docker. We are using the docker version of OpenDaylight Carbon built on 26 May 2017.

Virtual Ethernet Device:

The virtual Ethernet devices (or veth devices) is a local ethernet tunnel. They can act as tunnels between network namespaces to create a bridge to a physical network device in another namespace, but can also be used as standalone network devices.

Linux Bridge:

Linux bridge is a layer 2 virtual device that on its own cannot receive or transmit anything unless you bind one or more real devices to it.

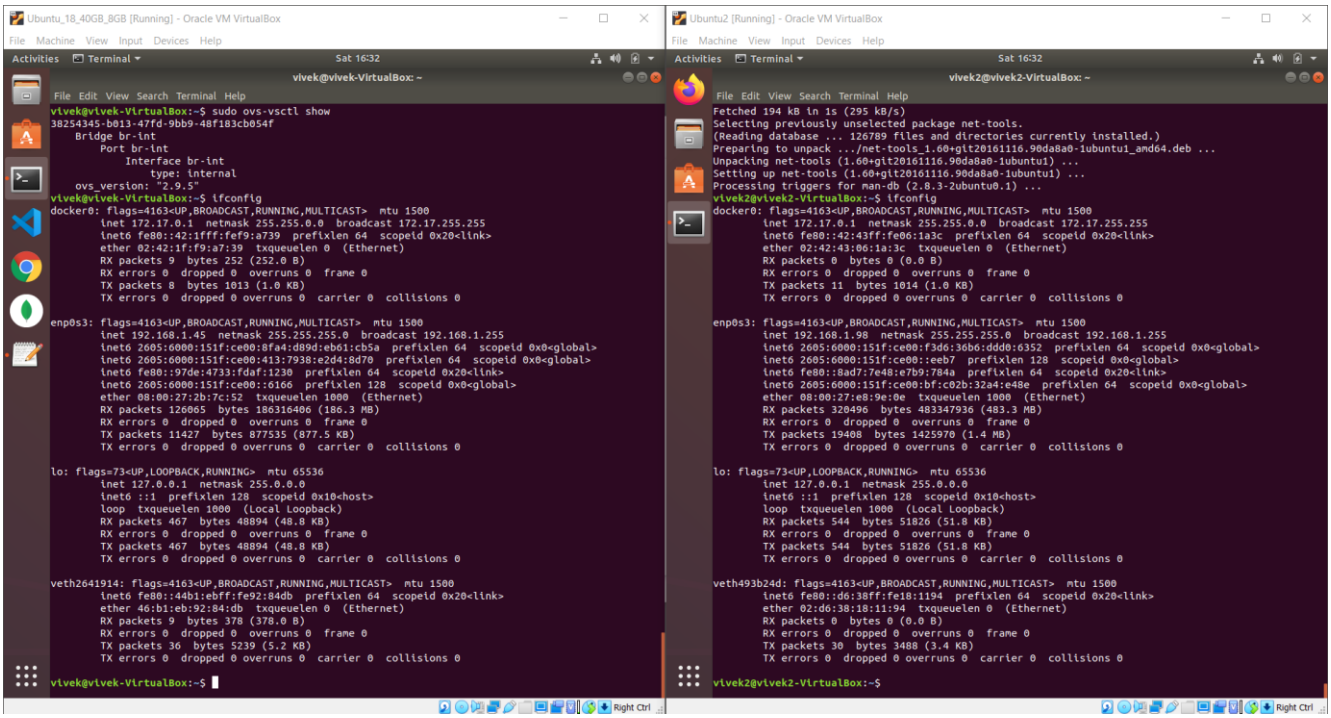
Wireshark:

Wireshark is a free and open-source packet analyzer. It is used for network troubleshooting, analysis, software and communications protocol development, and education.

Implementation:

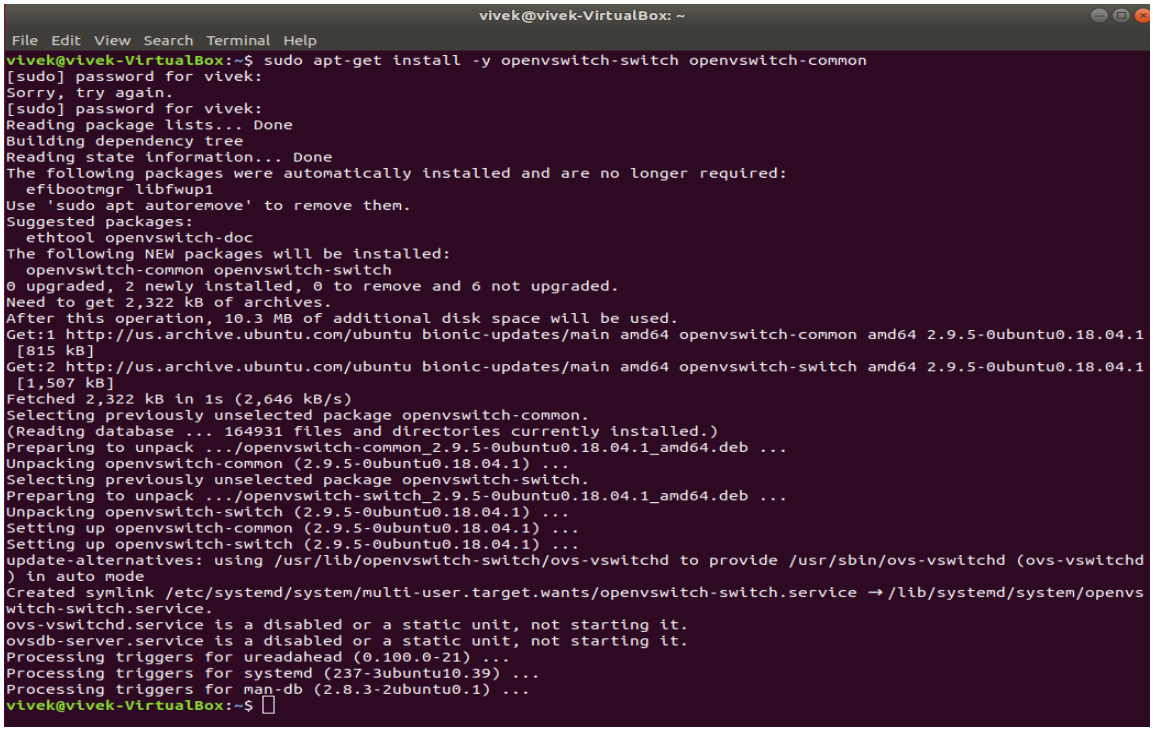
Installations:

We have instances of Ubuntu running on two Virtual Machines. These act as our host machines.

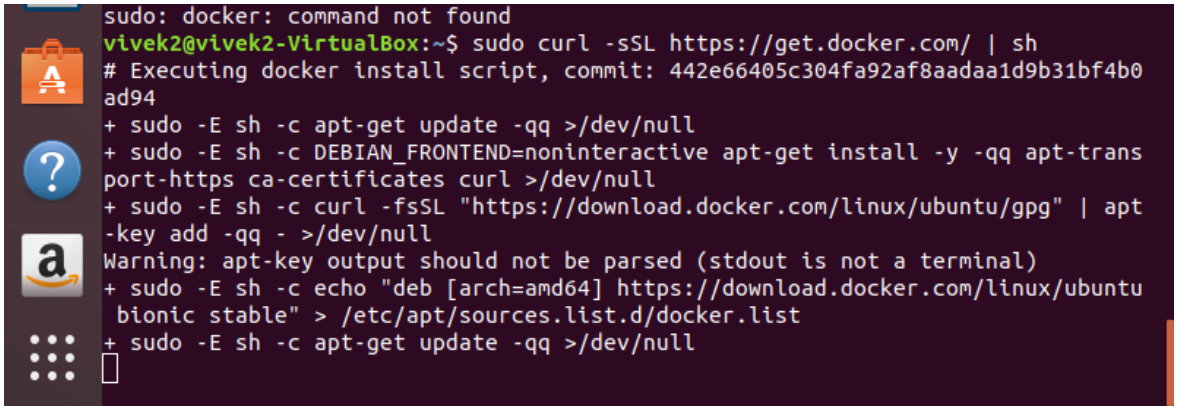


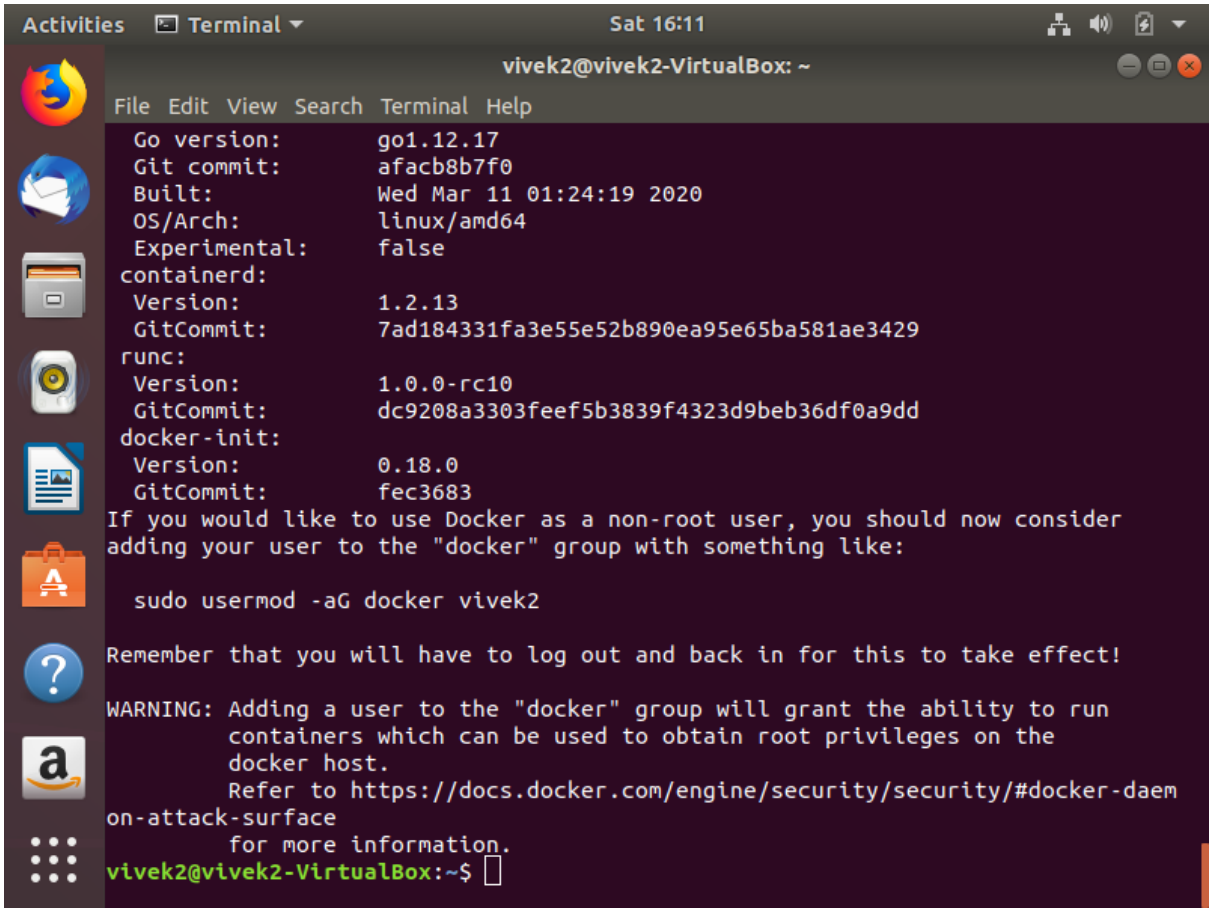
The below software were installed for this project. The Open vSwitch and Docker need to be installed on both the host machines. Rest of them need to be installed on at least one of them (all of these need to available on a single system).

1. Open VSwitch

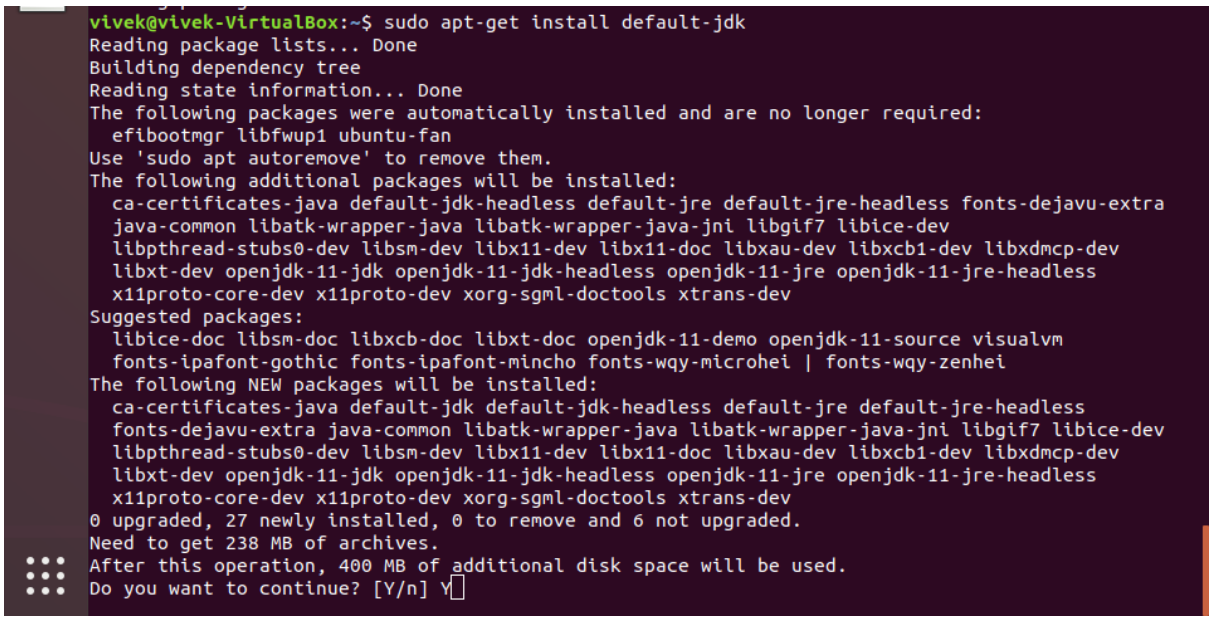


2. Docker

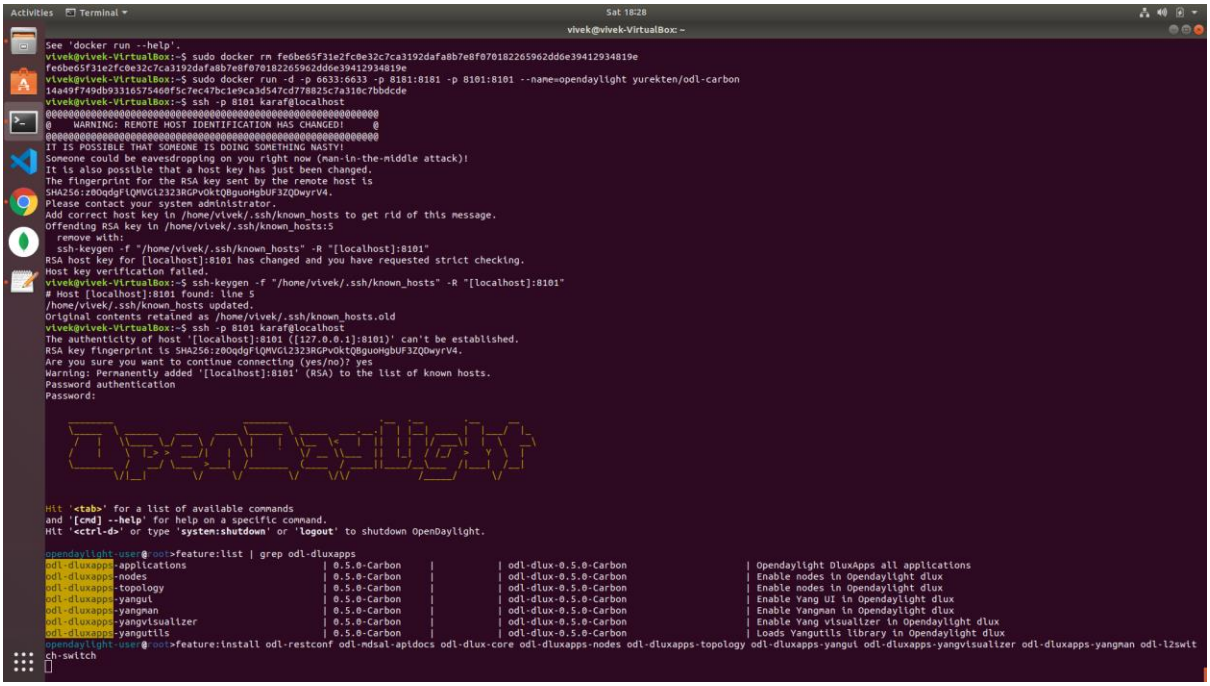




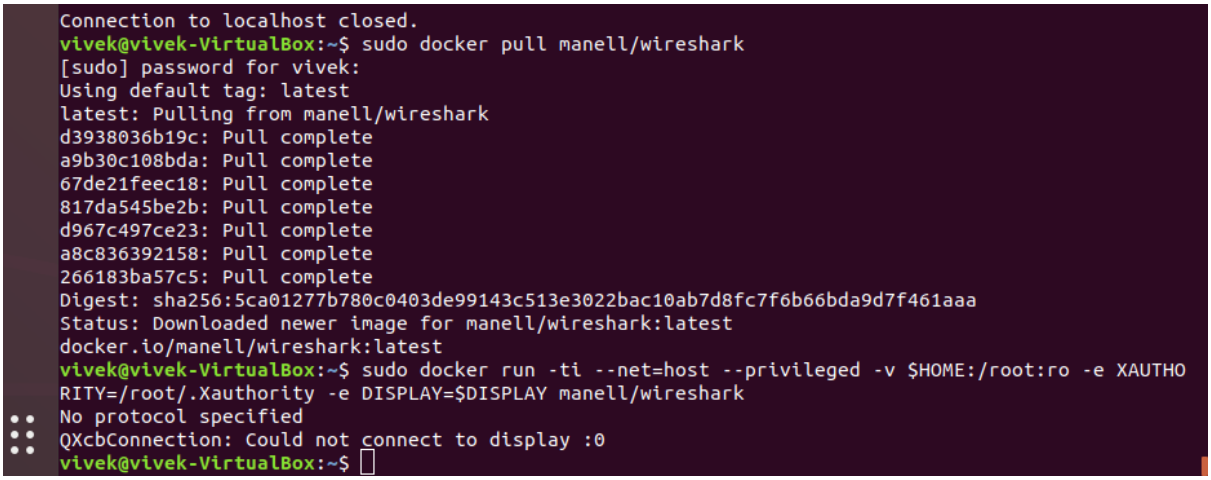
3. Java Development Kit(JDK) 11



4. OpenDaylight



5. Wireshark

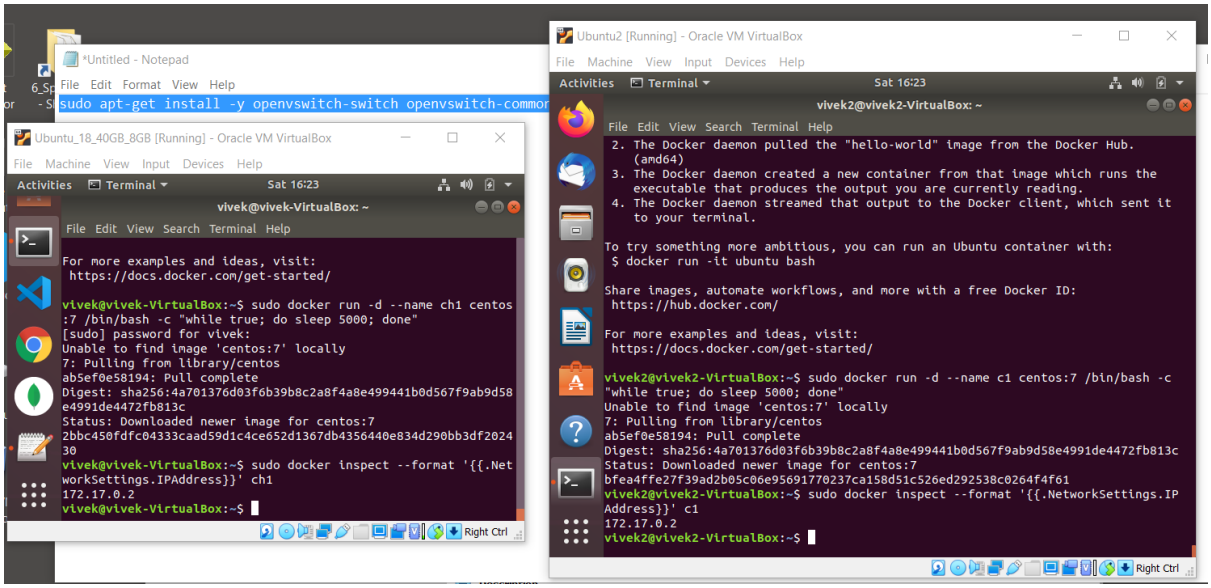


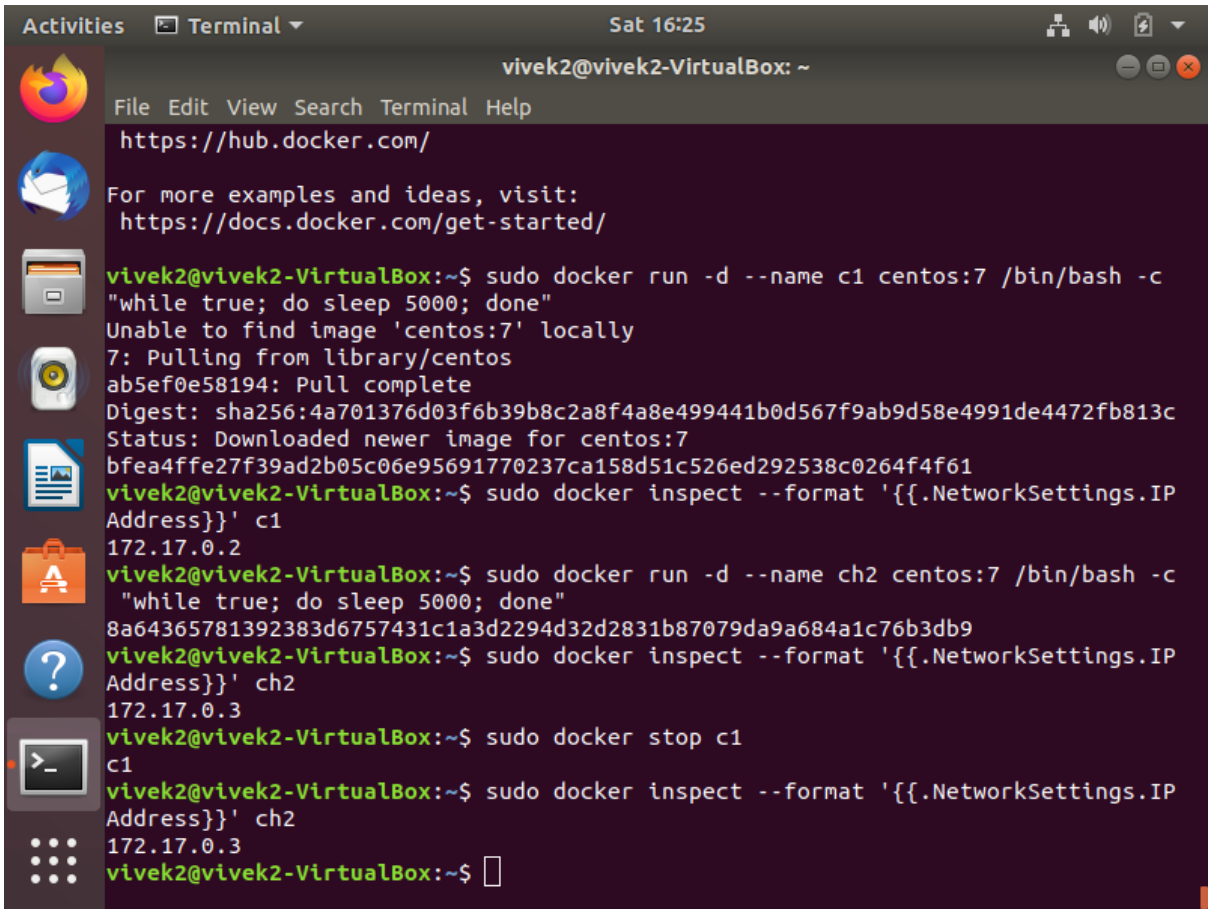
Connect docker using OVS bridge:

We connect the docker to one of the virtual ethernet device pair and the other device to the Open vSwitch bridge br-int (the core bridge).

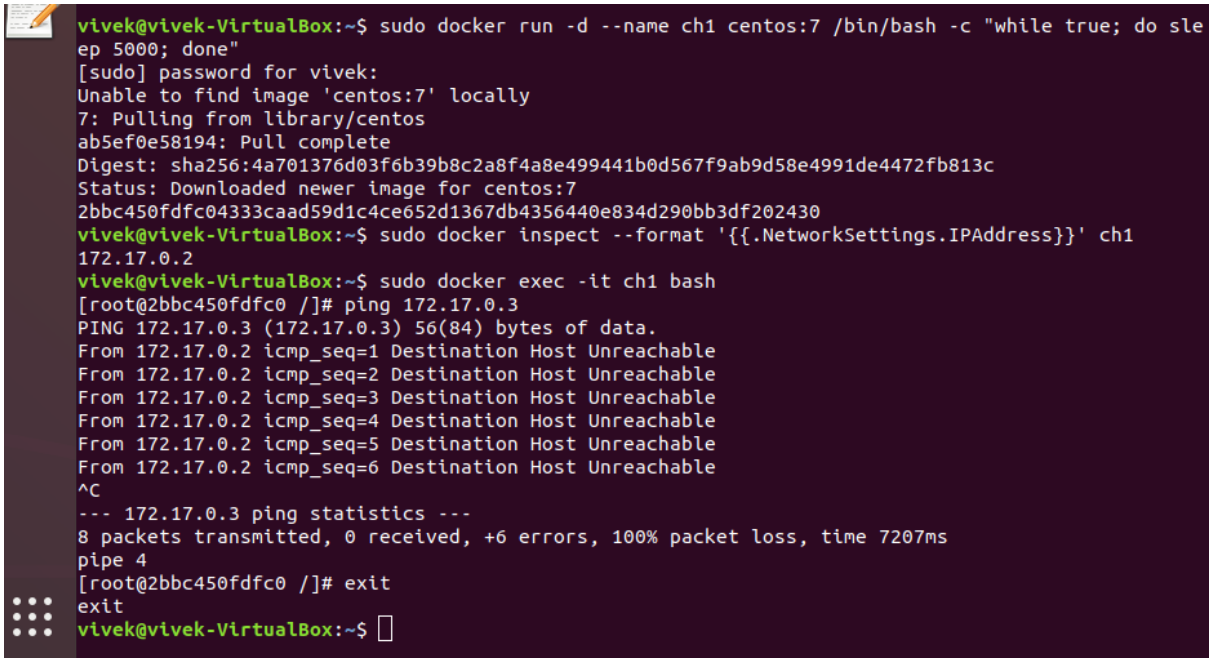
Steps:

1. Run the docker container on both the host machines and get the IP Addresses. We found that that both the containers have the same IP Address. So we ran another container and killed the first one on one of the host machines.

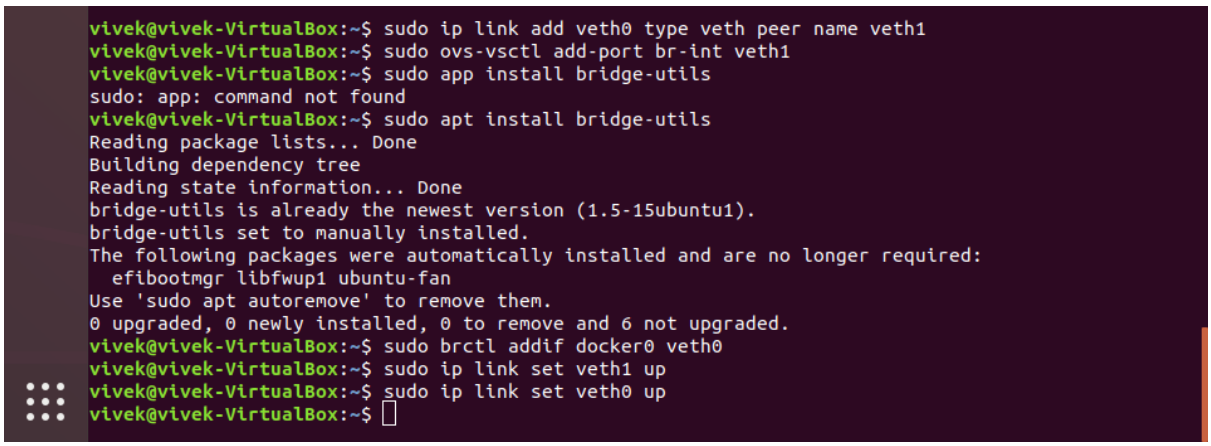

*Untitled - Notepad
File Edit Format View Help
sudo apt-get install -y openvswitch-switch openvswitch-commor
Ubuntu_18_40GB_8GB [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Sat 16:23
vivek@vivek-VirtualBox: ~
File Edit View Search Terminal Help
For more examples and ideas, visit:
https://docs.docker.com/get-started/
vivek@vivek-VirtualBox:~$ sudo docker run -d --name ch1 centos
:7 /bin/bash -c "while true; do sleep 5000; done"
[sudo] password for vivek:
Unable to find image 'centos:7' locally
7: Pulling from library/centos
ab5ef0e58194: Pull complete
Digest: sha256:4a701376d03f6b39b8c2a8f4a8e499441b0d567f9ab9d58
e4991de4472fb813c
Status: Downloaded newer image for centos:7
2bbc450fdfc04333caad59d1c4ce652d1367db4356440e834d290bb3df2024
30
vivek@vivek-VirtualBox:~$ sudo docker inspect --format '{{.Net
workSettings.IPAddress}}' ch1
172.17.0.2
vivek@vivek-VirtualBox:~$
Right Ctrl
Ubuntu2 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Sat 16:23
vivek2@vivek2-VirtualBox: ~
File Edit View Search Terminal Help
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
(amd64)
3. The Docker daemon created a new container from that image which runs the
executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
to your terminal.
To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash
Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/
For more examples and ideas, visit:
https://docs.docker.com/get-started/
vivek2@vivek2-VirtualBox:~$ sudo docker run -d --name c1 centos:7 /bin/bash -c
"while true; do sleep 5000; done"
Unable to find image 'centos:7' locally
7: Pulling from library/centos
ab5ef0e58194: Pull complete
Digest: sha256:4a701376d03f6b39b8c2a8f4a8e499441b0d567f9ab9d58e4991de4472fb813c
Status: Downloaded newer image for centos:7
bfea4ffe27f39ad2b05c06e95691770237ca158d51c526ed292538c0264f4f61
vivek2@vivek2-VirtualBox:~$ sudo docker inspect --format '{{.NetworkSettings.IP
Address}}' c1
172.17.0.2
vivek2@vivek2-VirtualBox:~$
Right Ctrl


Activities Terminal Sat 16:25
vivek2@vivek2-VirtualBox: ~
File Edit View Search Terminal Help
https://hub.docker.com/
For more examples and ideas, visit:
https://docs.docker.com/get-started/
vivek2@vivek2-VirtualBox:~$ sudo docker run -d --name c1 centos:7 /bin/bash -c
"while true; do sleep 5000; done"
Unable to find image 'centos:7' locally
7: Pulling from library/centos
ab5ef0e58194: Pull complete
Digest: sha256:4a701376d03f6b39b8c2a8f4a8e499441b0d567f9ab9d58e4991de4472fb813c
Status: Downloaded newer image for centos:7
bfea4ffe27f39ad2b05c06e95691770237ca158d51c526ed292538c0264f4f61
vivek2@vivek2-VirtualBox:~$ sudo docker inspect --format '{{.NetworkSettings.IP
Address}}' c1
172.17.0.2
vivek2@vivek2-VirtualBox:~$ sudo docker run -d --name ch2 centos:7 /bin/bash -c
"while true; do sleep 5000; done"
8a64365781392383d6757431c1a3d2294d32d2831b87079da9a684a1c76b3db9
vivek2@vivek2-VirtualBox:~$ sudo docker inspect --format '{{.NetworkSettings.IP
Address}}' ch2
172.17.0.3
vivek2@vivek2-VirtualBox:~$ sudo docker stop c1
c1
vivek2@vivek2-VirtualBox:~$ sudo docker inspect --format '{{.NetworkSettings.IP
Address}}' ch2
172.17.0.3
vivek2@vivek2-VirtualBox:~$

2. Then we tried to ping the other container (in the other host) from this container and found that the destination was unreachable.

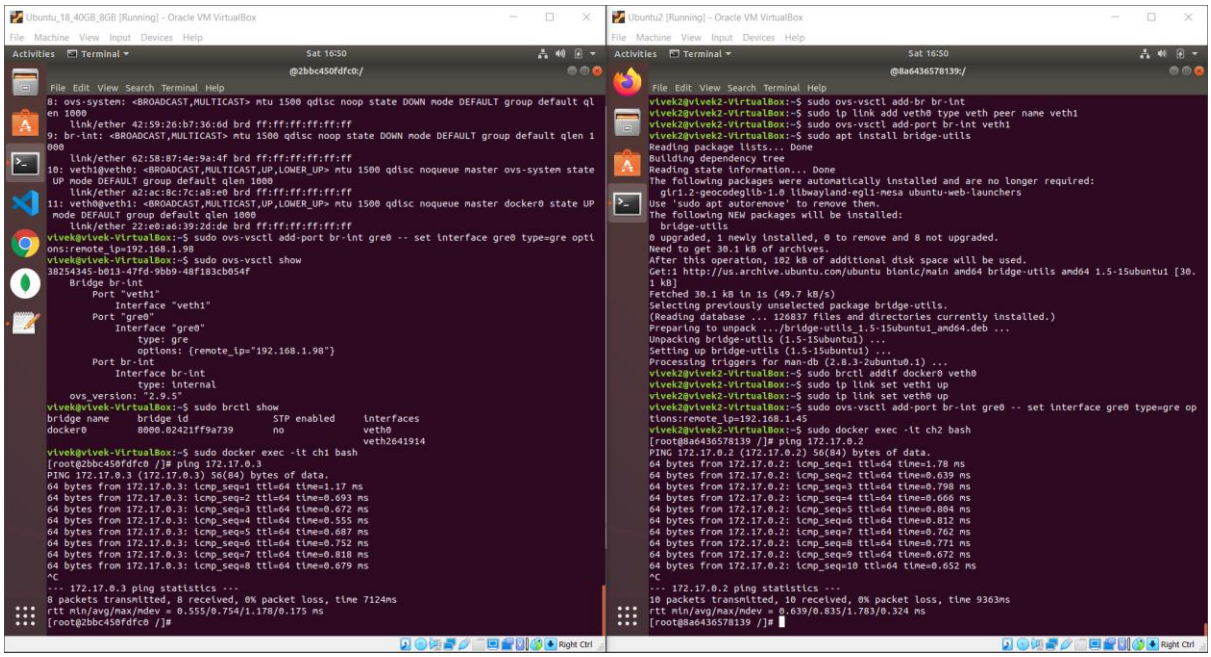


3. We started setting up the OVS bridge to establish the connection between the 2 containers on different hosts.



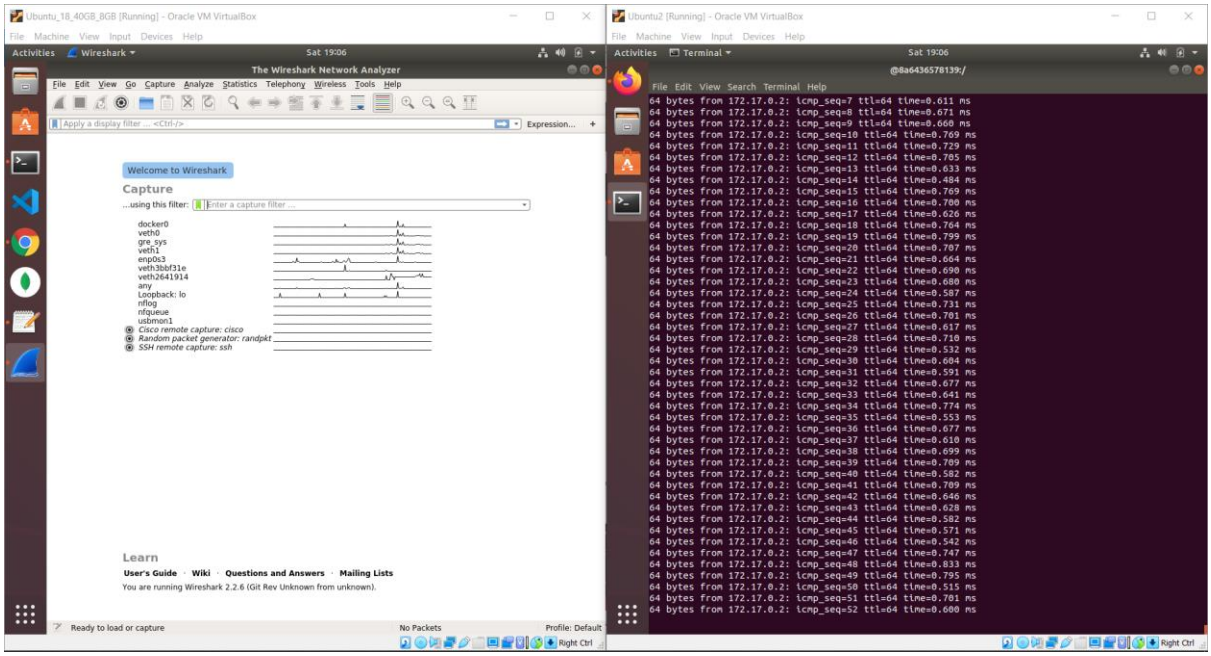
```
vivek@vivek-VirtualBox:~$ ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default ql
en 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT group
default qlen 1000
    link/ether 08:00:27:2b:7c:52 brd ff:ff:ff:ff:ff:ff
3: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT group
default
    link/ether 02:42:1f:f9:a7:39 brd ff:ff:ff:ff:ff:ff
7: veth2641914@if6: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker0 state
 UP mode DEFAULT group default
    link/ether 46:b1:eb:92:84:db brd ff:ff:ff:ff:ff:ff link-netnsid 0
8: ovs-system: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default ql
en 1000
    link/ether 42:59:26:b7:36:6d brd ff:ff:ff:ff:ff:ff
9: br-int: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1
000
    link/ether 62:58:87:4e:9a:4f brd ff:ff:ff:ff:ff:ff
10: veth1@veth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master ovs-system state
 UP mode DEFAULT group default qlen 1000
    link/ether a2:ac:8c:7c:a8:e0 brd ff:ff:ff:ff:ff:ff
11: veth0@veth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker0 state UP
 mode DEFAULT group default qlen 1000
    link/ether 22:e0:a6:39:2d:de brd ff:ff:ff:ff:ff:ff
vivek@vivek-VirtualBox:~$ sudo ovs-vsctl add-port br-int gre0 -- set interface gre0 type=gre opti
ons:remote_ip=192.168.1.98
vivek@vivek-VirtualBox:~$ sudo ovs-vsctl show
38254345-b013-47fd-9bb9-48f183cb054f
    Bridge br-int
        Port "veth1"
            Interface "veth1"
        Port "gre0"
            Interface "gre0"
                type: gre
                options: {remote_ip="192.168.1.98"}
        Port br-int
            Interface br-int
                type: internal
    ovs_version: "2.9.5"
vivek@vivek-VirtualBox:~$ sudo brctl show
bridge name     bridge id               STP enabled     interfaces
docker0         8000.02421ff9a739       no              veth0
                                                        veth2641914
vivek@vivek-VirtualBox:~$
```

4. After setting up the bridge, we were able to ping from one container to the other.

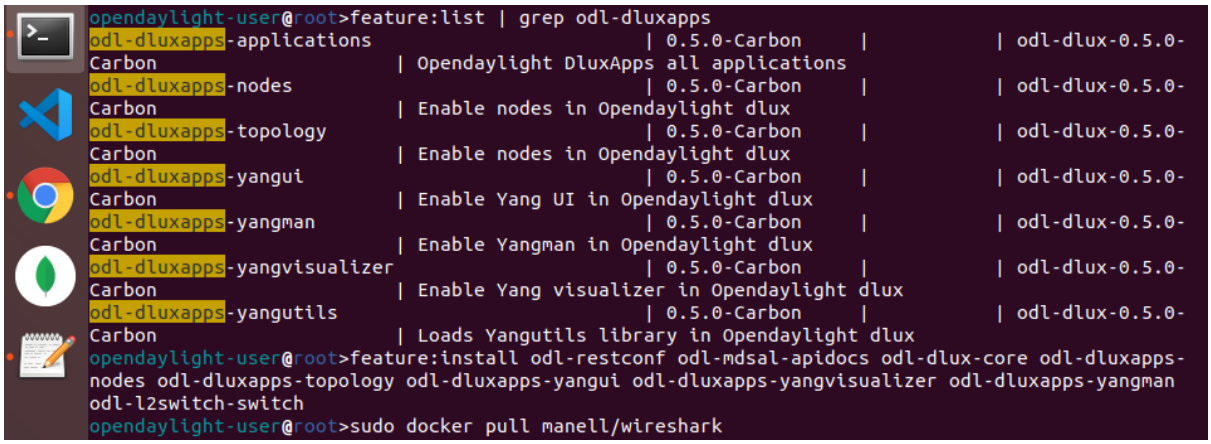


Wireshark:

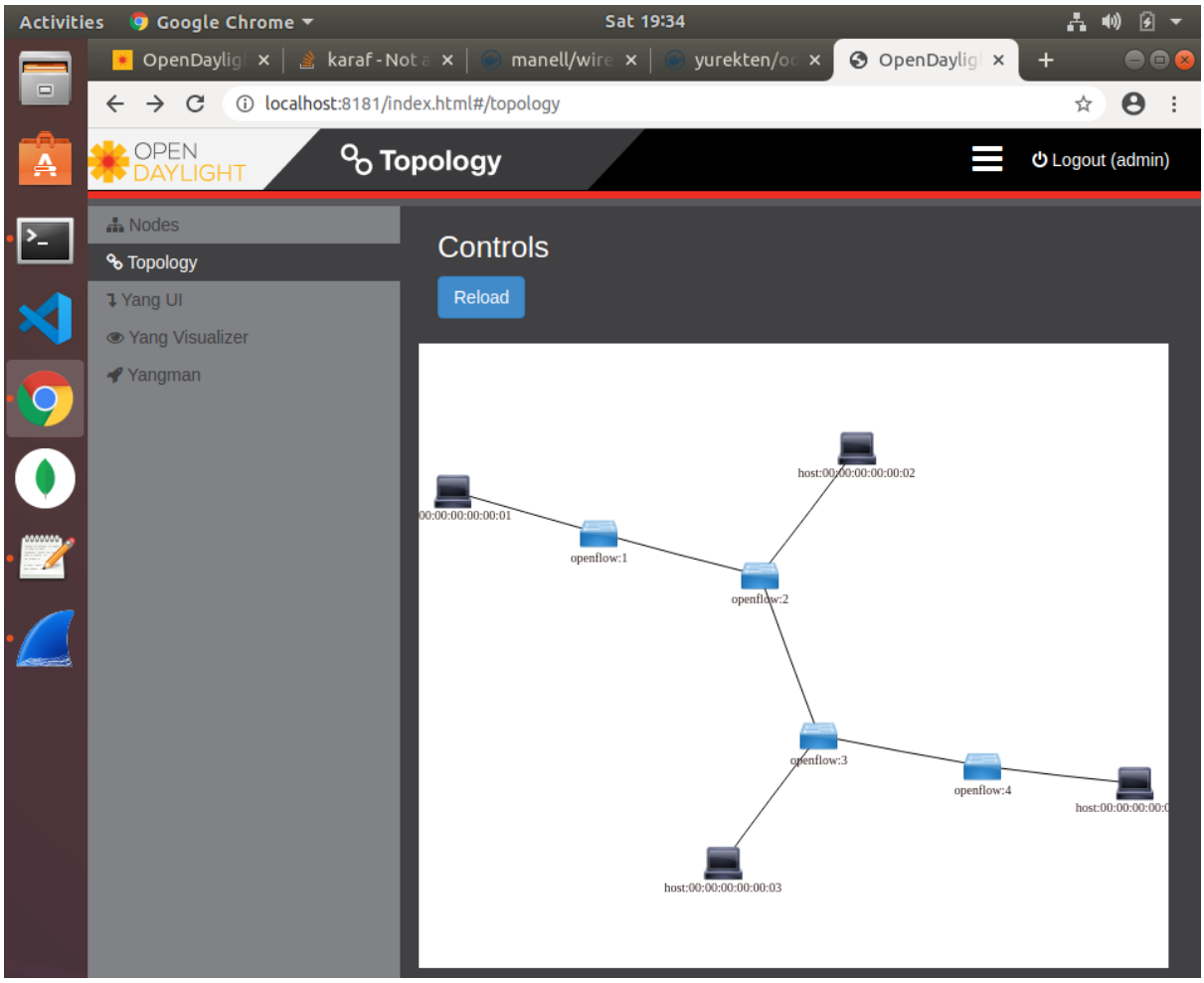
We pinged from one container to another and checked the traffic on Wireshark.

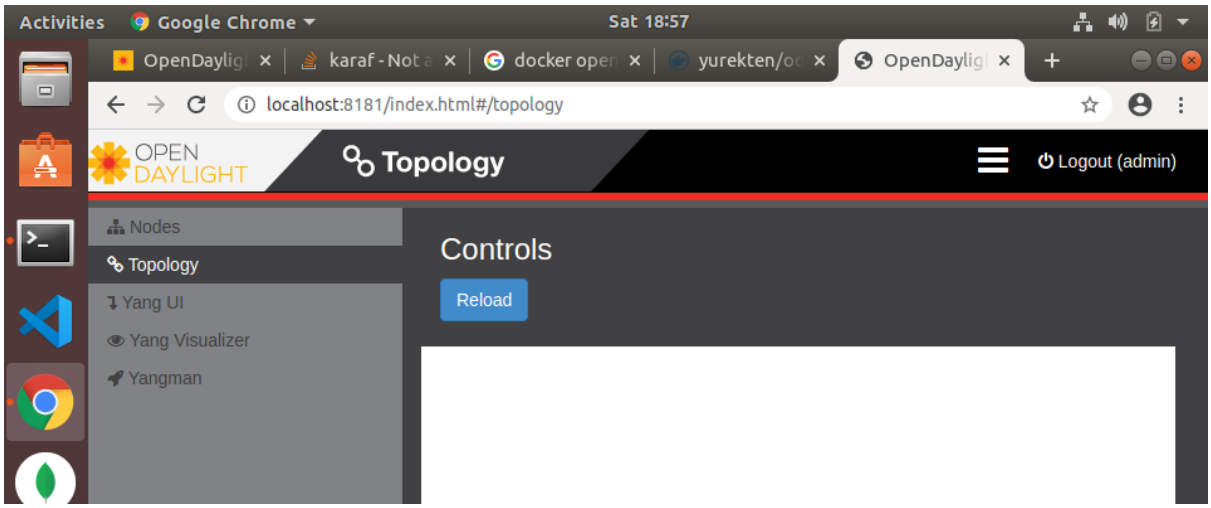


Open Daylight Feature Install:



We installed the OpenDaylight features and tested the mininet topology in this container.



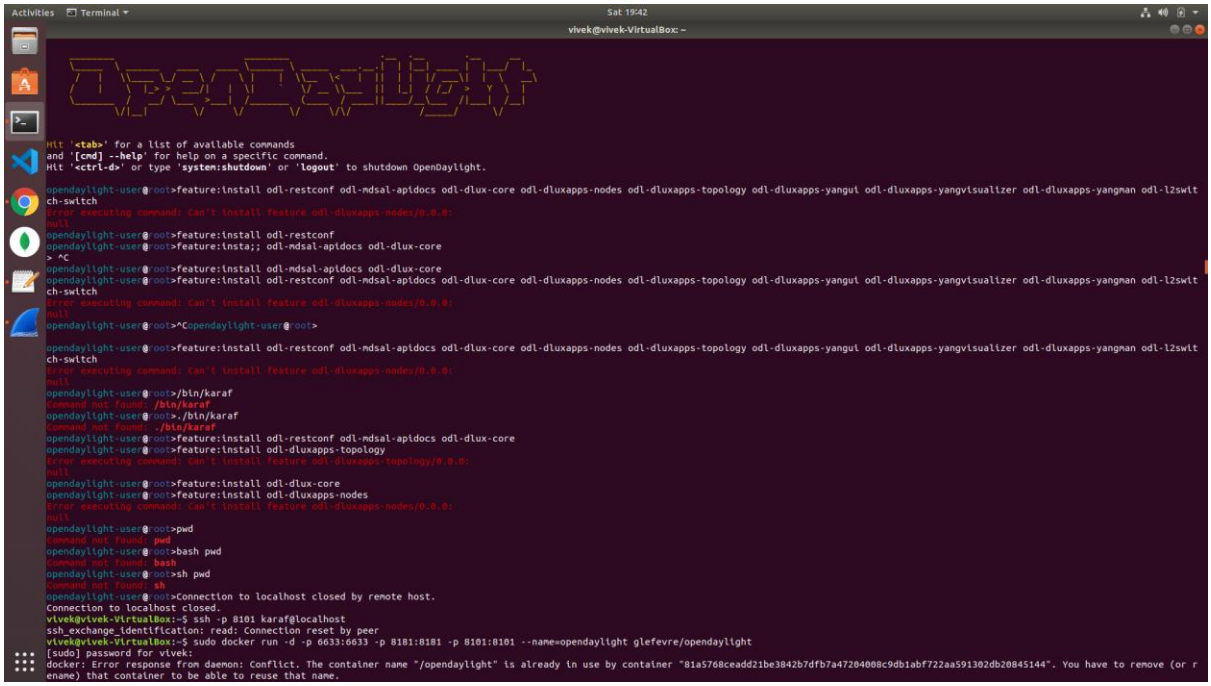


Challenges Faced

- Main hurdle was learning Docker and containerization of applications.
- Creating 2 heavy weight VMs on the same laptop machine was a challenge since we have limited processing power.
- Containers on different hosts had the same IP Addresses, so we had to respawn another to get a different IP on one of them.
- Issues with JAVA versions: Initially we had JDK 8 installed, but an upgrade changed it to JDK 11 but JAVA_HOME variable was still pointing to JDK 8 path which had become invalid.

Figuring out that the JDK had upgraded took us some time since the Open Daylight was failing to launch.

- Initially we had installed OpenDaylight Boron which did not have the dluxapps features. After figuring out the list of features that were available in Boron, we switched to Carbon. During this, we faced issues with docker images not stopping, we had to stop the images forcefully before we were able to switch to Carbon. Also when we re-started the container, SSH into the container didn't work as the host key had changed. So we had to generate and save the new SSH key.



Conclusion:

Docker is a easy and efficient way to build and deploy any application. We were successful in setting up a communication channel between 2 docker containers on different host machines using Open vSwitch. We were able to use a containerized OpenDaylight-Carbon to display the topology of a Mininet running on a separate VM.

References:

[1] https://en.wikipedia.org/wiki/Docker_(software)

[2] https://en.wikipedia.org/wiki/Open_vSwitch

[3] https://www.imperva.com/blog/what-is-gre-tunnel/

[4] https://hub.docker.com/r/yurekten/odl-carbon/

[5] https://goyalankit.com/blog/linux-bridge

[6] http://man7.org/linux/man-pages/man4/veth.4.html

[7] https://en.wikipedia.org/wiki/Wireshark

[8] https://app.box.com/s/t4yob7umn1m88b4p4vmbeu1scelee31e/folder/63985323299