Cloud/DevOps Curriculum Series for Online Learning

Lab 2 – Software Defined Networking on GCP with Cyber Defense app

Technical Contacts: Dr. Prasad Calyam (calyamp@missouri.edu); Dr. Songjie Wang (wangso@missouri.edu)

Release 1: May, 2021

Contents

1 Purpose of the Lab **3**

2 References to guide lab work **3**

3 Overview **3**

4 GCP VM instance Preparation **4**
4.1 Log into your Google GCP account console 4
4.2 Setup firewall policies for the networks in your project 5
4.3 Create VM instances for each of the node in the topology setup 5
4.4 Create VM instances for each of the node in the topology setup 7

5 Topology **7**
5.1 Once all the VM instances have been created and are up running, please note down the following information: 7
5.2 The virtual IPs and ports that we have configured for the topology 8

6 VM Instance Setup **10**
6.1 Controller node setup 10
6.2 General installation of Open vSwitch 15
6.3 Open vSwitch installation for the rest of the topology 16
6.4 Root Switch node setup 16
6.5 Slave Switch node setup 19
6.6 Server nodes setup 22
6.7 Attacker VM setup 22
6.8 QVM VM setup 23
6.9 User VM setup 24
6.10 Configure Dolus AdminUI and Pox SDN controller on the Controller VM 25
6.11 Setup default routing protocol in SDN 26
6.12 Stop default SDN routing 26

7 Congratulations! You have successfully implemented an virtual SDN overlay on top of the Google GCP network, and used the Pox controller to control the default network routing between the nodes in your topology. **27**

8 Homework Problems **27**

9 Lab Report **28**

1 Purpose of the Lab

In this lab, you will learn how to setup Software Defined Networking (SDN) capability using VXLAN protocol on a public cloud, Google GCP. You will learn the basics of how to create virtual network overlay on top of GCP layer3 network, and create a virtual network topology with a number of virtual machines nodes for Dolus, a cyber defense by pretense system, for the learning of a series of Dolus cyber defense labs.

2 References to guide lab work

Please use the links below to learn the related information for this lab.

- SDN – *https://en.wikipedia.org/wiki/Software-defined_networking/*
- Google GCP platform – *https://cloud.google.com/*
- Virtual Extensible LAN (VXLAN))– *https://en.wikipedia.org/wiki/Virtual_Extensible_LAN*
- Pox Controller – *https://github.com/noxrepo/pox*
- Open vSwitch – *https://www.openvswitch.org/*
- Frenetic Programming Interface – *http://frenetic-lang.org/*
- Frenetic Programming Guide – *https://github.com/frenetic-lang/manual/blob/master/programmers_guide/frenetic_programmers_guide.pdf/*
- Docker – *https://docs.docker.com*
- Docker Hub – *https://hub.docker.com/*

3 Overview

Software-defined networking (SDN) technology is an approach to network management that enables dynamic, programmatically efficient network configuration in order to improve network performance and monitoring, making it more like cloud computing than traditional network management. SDN was commonly associated with the OpenFlow protocol, which allows a controller to tell network switches where and how to send network packets, thus allowing network to be programmed independent of the individual switches, in other words, separating control plane from data plane, as illustrated in Figure 1. Conventionally, networks are hierarchical with tiers of switches arranged in a tree structure, which is static and not suitable for dynamic computing and storage needs of modern data centers and campuses environments. For the purpose of programmable networks, SDN has been implemented on traditional switching devices to allow controlling of network routing. However, with the vast growth and usage of cloud services in many enterprises and academic institutes, using traditional SDN is largely not feasible since cloud service providers will not allow users to manipulate their switch devices. Hence, Virtual Extensible LAN (VXLAN) protocol comes into play to address the network programmability problems associated with cloud platforms, as it provides network connectivity using tunneling to stretch Layer 2 connections over an underlying Layer 3 network, i.e. overlay networks that sit on top of the physical network, enabling the use of virtual networks.

In this lab, we will guide You to create an SDN virtual network with a Pox Controller node as the control plane, and create two Open vSwitches, a Root Switch and a Slave Switch, that are connected to the Controller via TCP/IP connection. And then you will create a number of host VM instances that are connected to the two vSwitches via VXLAN network overlay. These nodes will serve as the servers and clients of our example application, the Dolus application. Dolus application includes several cyber defense lab testbeds that will need a Server, an Attacker, a QVM machine, and a legitimate User node. In total, we will setup 7 VM instances in the SDN network to be used for each of the nodes we mentioned above. The topology setup for Dolus is shown in Figure 2.

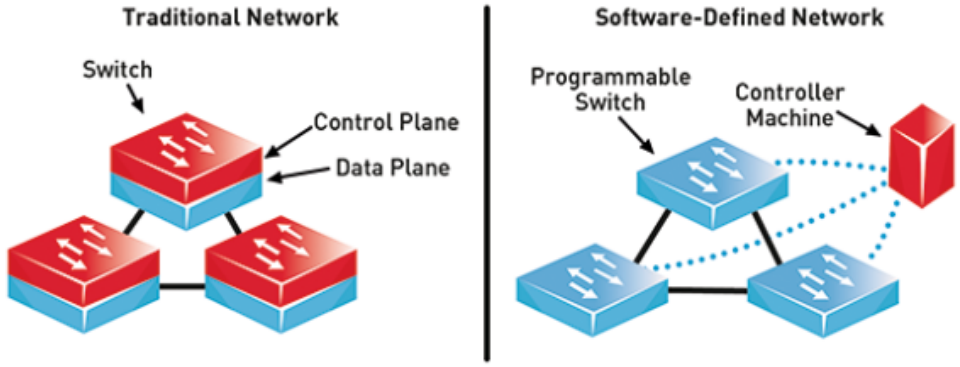


Figure 1: Decoupling of control plane from data plane in modern Software defined networks (SDN) as compared to traditional networks. (Image credit: *https://www.commsbusiness.co.uk/features/software-defined-networking-sdn-explained/*)

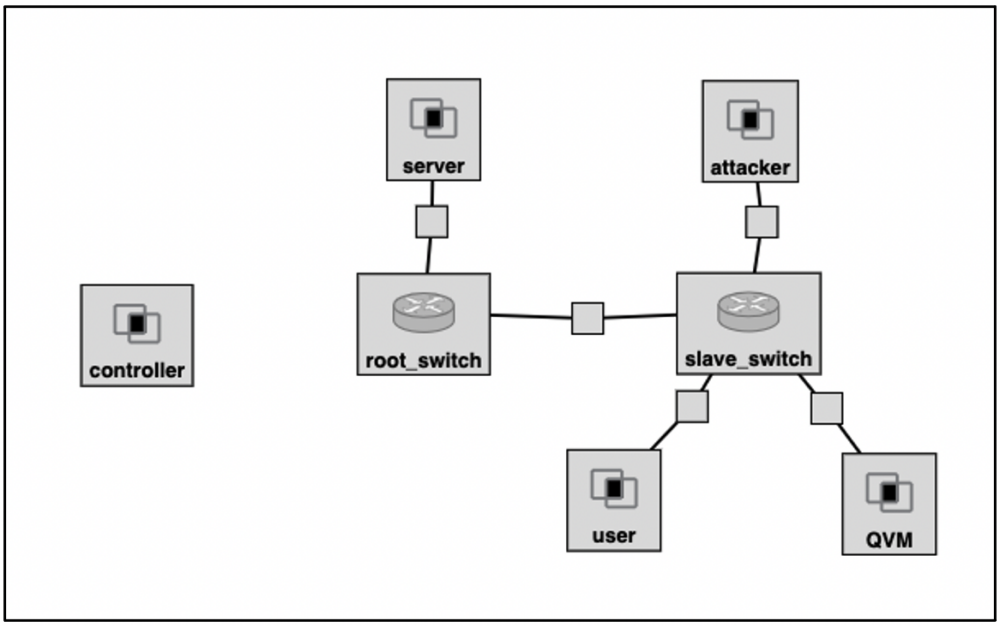


Figure 2: Topology setup of Dolus cyber defense by pretense system.

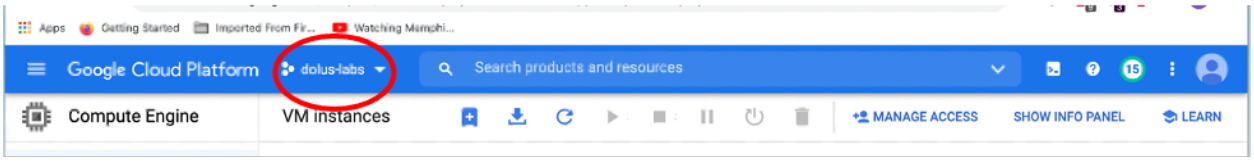
4 GCP VM instance Preparation

The first step of this lab to configure GCP cloud access. For this, you need to have

- A Google account
- A Google project
- (optional) Google SDK to allow command line access

4.1 Log into your Google GCP account console

- Navigate to your web console on GCP in the project:



4.2 Setup firewall policies for the networks in your project

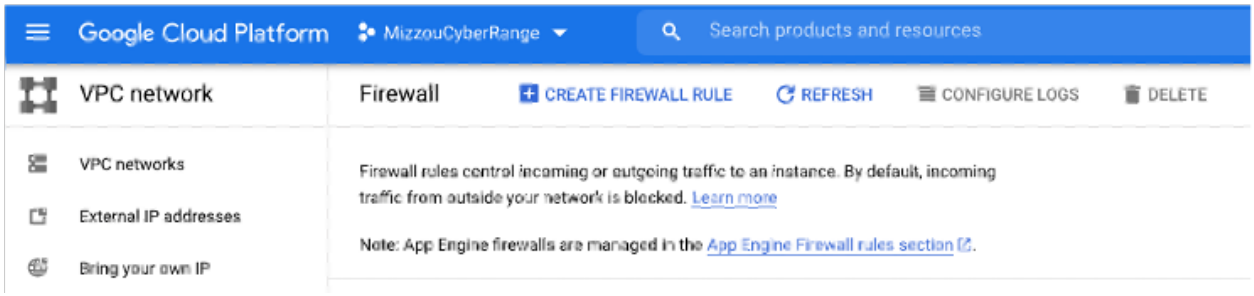
Warning

In this lab, it is essential that you configure your firewall policy in your VPC network. If you don't configure the policies correctly, your SDN network routing will not work!

Note

In this lab, we will use the "default" VPC and subnet in your GCP account.

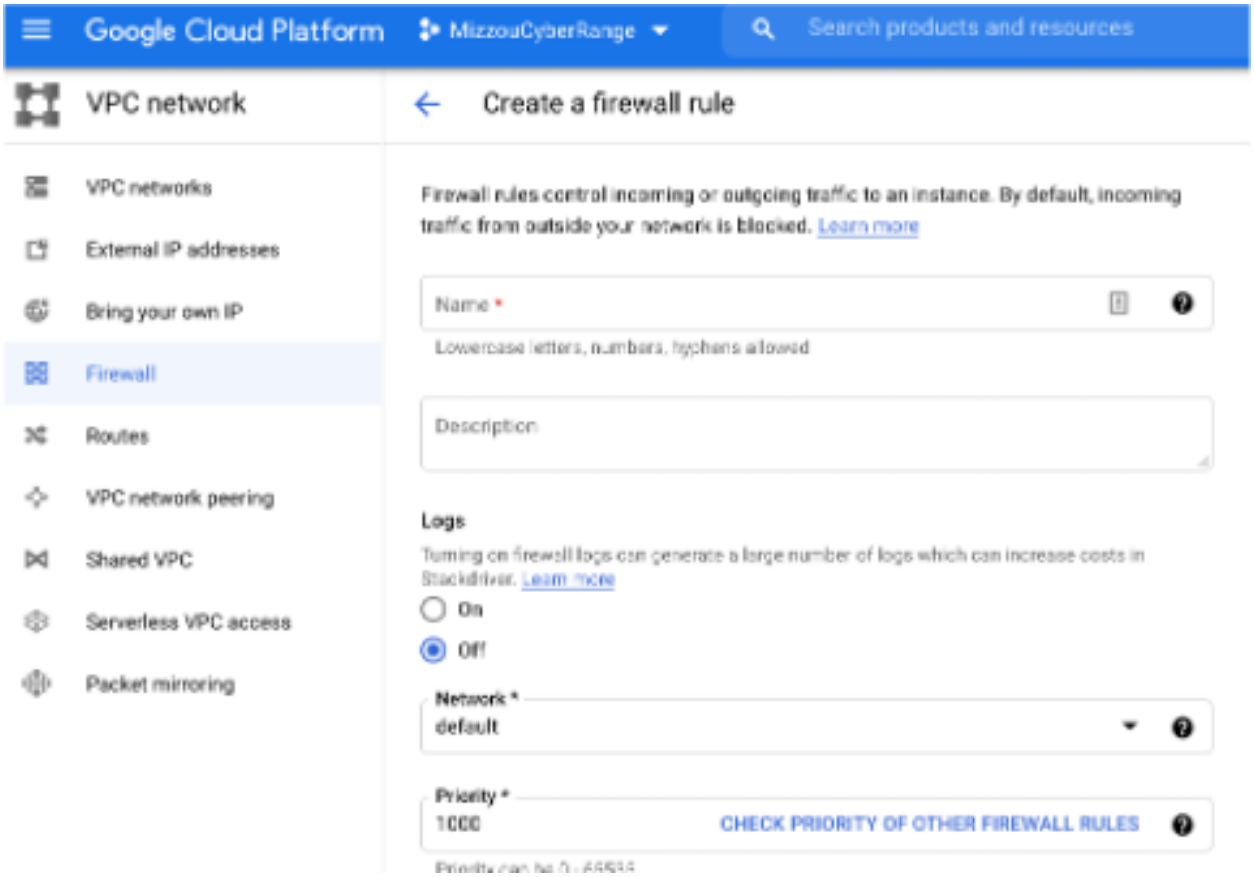
- Click the hamburger symbol, and select Networking $->$ Firewall.



- Click "Create Firewall Rule"
- Give the firewall rule a name, leave others as default, and fill out the Targets as indicated in the screen shot below and define source IP ranges as "0.0.0.0/0" to allow all IPs on the network to access the VPC network.
- Configure protocols and ports as indicated in the screen shot to open firewall ports required in the Dolus topology setup, i.e. ports 22, 80/8080, 3306/33060, 6633:
- Click "Create" to create the firewall policy.

4.3 Create VM instances for each of the node in the topology setup

After configuring VPC firewall policies, create the VM instances for Dolus topology setup.



Note

In this section, we guide you to create one of the VM instances in the topology, and you will use repeat the steps with similar settings to create all the other instances.

- Click the hamburger symbol, and select Compute Engine $->$ VM instances.
- From the console, click the symbol to create new VM instance.
- Give an appropriate name to the VM instance. We recommend you to use the following naming convention: "username-nodename", to easily remember what the instance is. For example, "wangso-controller" is a good name. Keep the default VM instance configuration as shown in the screenshot.

- In the "Boot disk", click Change. Leave the default Disk image as Debian GNU/Linux 10, and change Persistent disk size to 20GB. Click Select.
- Leave the Identity and API access to "Allow full access to all Cloud APIs", and check firewalls to allow both HTTP and HTTPs traffic.
- Keep default settings for Management, security, disks, networking, sole tenancy
- Click Create.

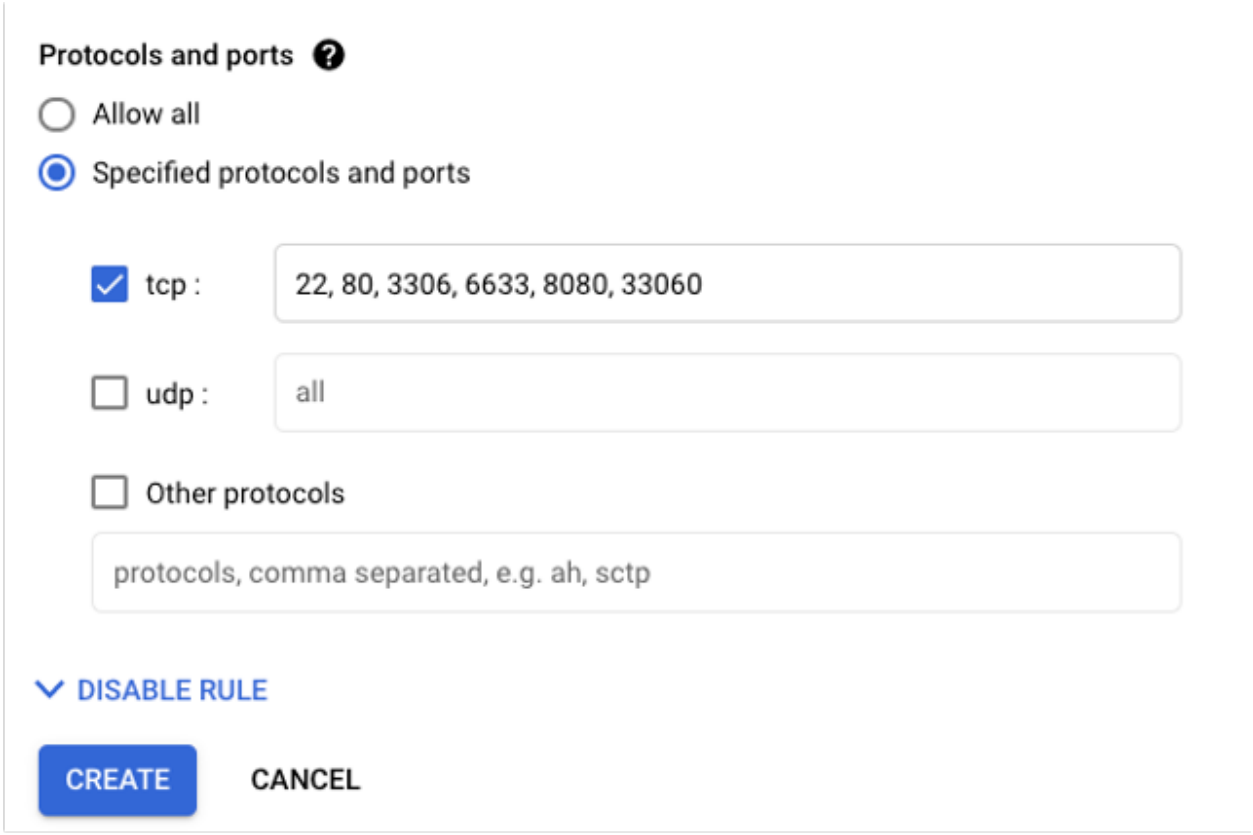
4.4 Create VM instances for each of the node in the topology setup

Repeat the above Section 4.3 to create all other VM nodes.

5 Topology

5.1 Once all the VM instances have been created and are up running, please note down the following information:

- Public IP (External IP) of the Controller node
- Private IP (Internal IP) of all the other nodes



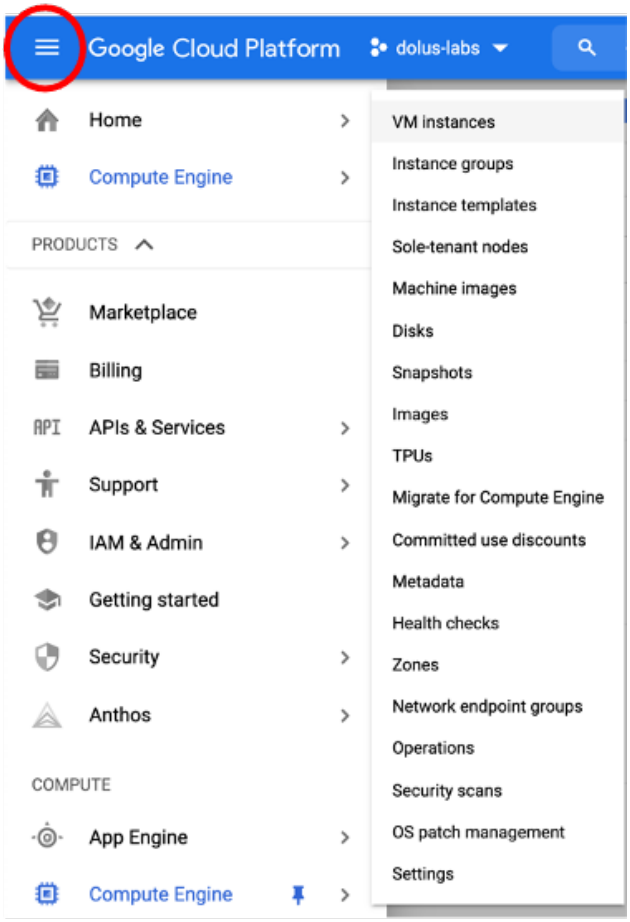
Node Name	Virtual IPs	Port # Root Switch	Port # on Slave Switch	UID
Root switch	100.0.0.100	/	/	/
Slave switch	100.0.0.101	/	/	/
Server	100.0.0.102	2	1	11
Attacker	100.0.0.103	1	2	1
QVM	100.0.0.104	1	3	2
User	100.0.0.105	1	4	3

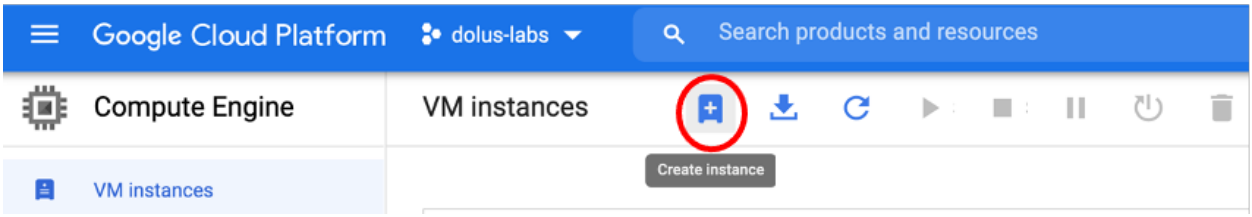
Table 1: Virtual IPs and ports that we have configured for the topology

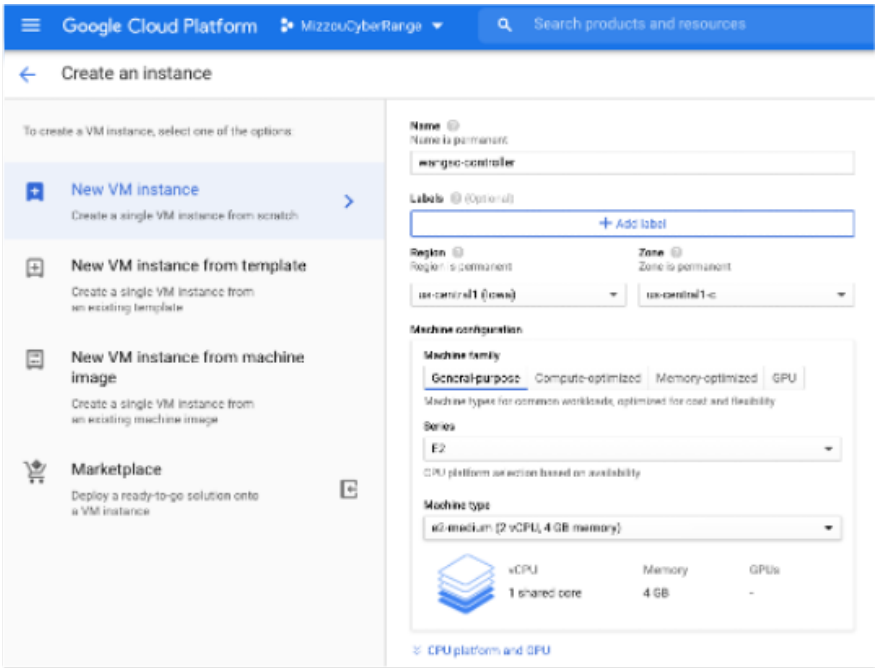
5.2 The virtual IPs and ports that we have configured for the topology

Note

Root and Slave switches do not have a port on each other, so they can not be reached using ping commands. In this way, we keep them safe by not allowing users to connect to them.


Google Cloud Platform
dolus-labs
Home
Compute Engine
PRODUCTS
Marketplace
Billing
APIs & Services
Support
IAM & Admin
Getting started
Security
Anthos
COMPUTE
App Engine
Compute Engine
VM instances
Instance groups
Instance templates
Sole-tenant nodes
Machine images
Disks
Snapshots
Images
TPUs
Migrate for Compute Engine
Committed use discounts
Metadata
Health checks
Zones
Network endpoint groups
Operations
Security scans
OS patch management
Settings





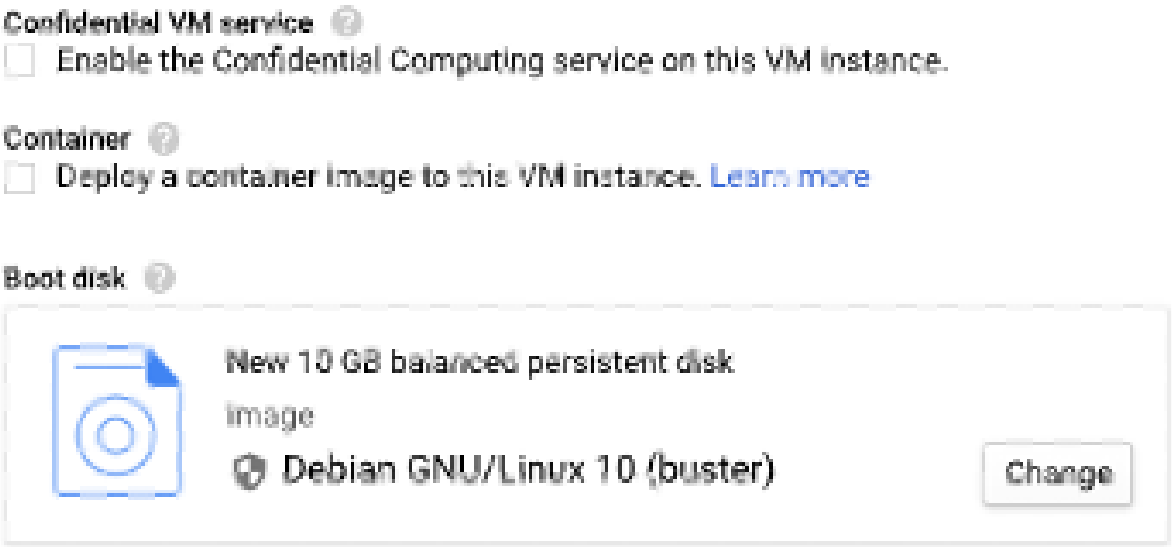
6 VM Instance Setup

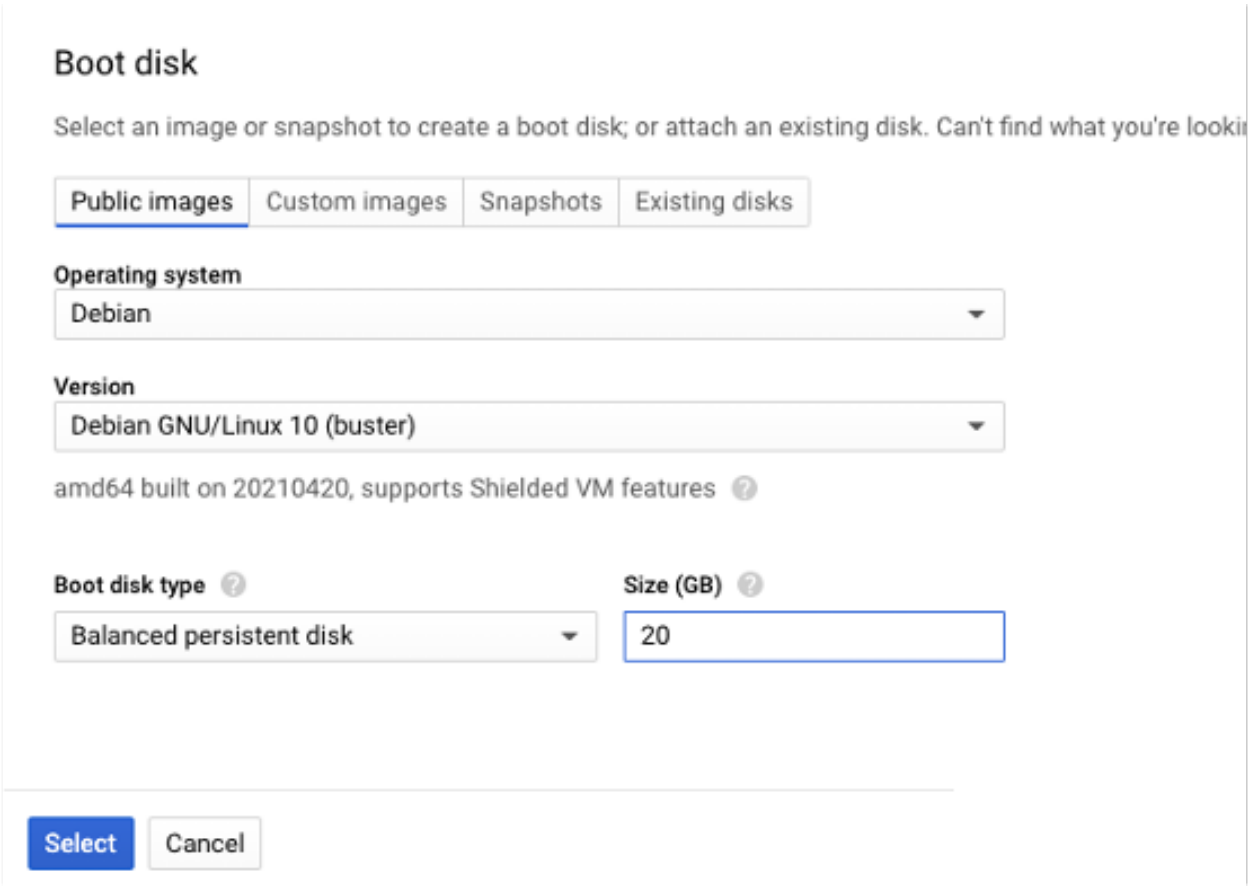
Once all the VM instances are all up and running, we are going to setup the SDN network connections.

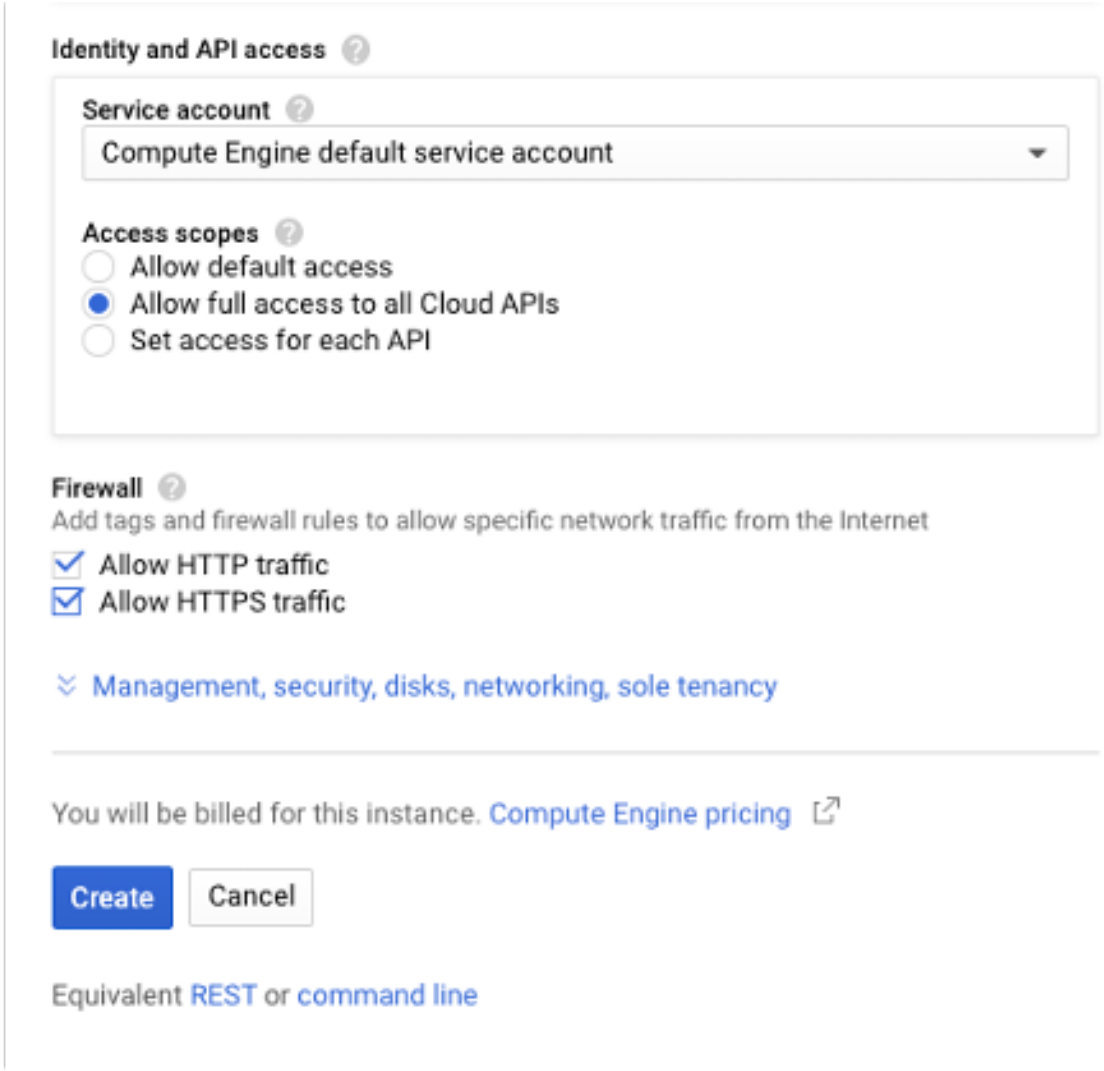
6.1 Controller node setup

On the Controller instance, we will install Dolus System that runs the Pox SDN controller. The controller is used to capture the network traffic and packet flow on the root-switch and slave-switch, calculate the suspiciousness scores, and detect an attack.

We have prepared the Controller node as a Docker container, and installed all the required software, libraries, and the Dolus web portal. However, you will need to configure the node to connect to the Root Switch and Slave Switch. This will need to be done after you correctly configure the two switches. To configure the Controller, please follow the steps below:


Confidential VM service
Enable the Confidential Computing service on this VM instance.
Container
Deploy a container image to this VM instance. Learn more
Boot disk
New 10 GB balanced persistent disk
image
Debian GNU/Linux 10 (buster)
Change


Boot disk
Select an image or snapshot to create a boot disk; or attach an existing disk. Can't find what you're looki
Public images
Custom images
Snapshots
Existing disks
Operating system
Debian
Version
Debian GNU/Linux 10 (buster)
amd64 built on 20210420, supports Shielded VM features
Boot disk type
Balanced persistent disk
Size (GB)
20
Select
Cancel



6.1.1 First, SSH login to the Controller node using your SSH private key.

6.1.2 After logging in, install Docker on it

Note

The containers that we will be running will take 10GB of space. Make sure your VM instance has enough space available, and then continue to setup the lab environment.

Follow the Docker installation guide and install Docker on the Controller node. Since we are using Debian Linux operating system, you need to use this link:

https://docs.docker.com/engine/install/debian/

Note

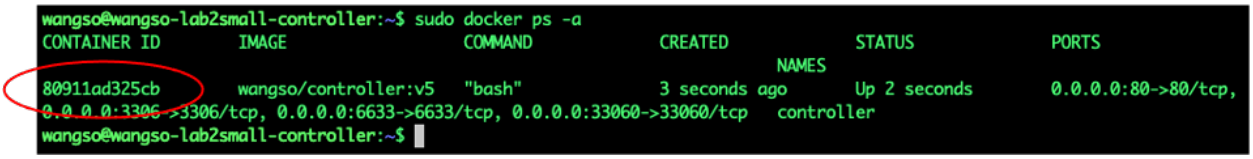
Docker also provides convenience scripts at for installing testing versions of Docker Engine quickly and non-interactively. You can use these scripts to quickly install Docker. Using these scripts is not recommended for production environments.

```
$ curl -fsSL https://get.docker.com -o get-docker.sh
$ sudo sh get-docker.sh
```

6.1.3 Once Docker is installed, download the Controller Docker image, and run a Controller container as follows:

```
$ sudo docker pull mizzouceri/dolus-controller:latest
$ sudo docker images
$ sudo docker run -dit -p 80:80 -p 6633:6633 -p 3306:3306 -p 33060:33060
--name controller mizzouceri/dolus-controller:latest
$ sudo docker ps -a
```

You should see the following output:



6.1.4 Login to the container:

```
$ sudo docker exec -it controller bash
```

6.1.5 Restart apache and MySql services:

```
$ service apache2 start
$ service mysql start
```

You should see the following output:

Note

Sometimes MySql fails to start. If it happens, run "service mysql restart" command

6.1.6 Inside the Container, run the following command to start Frenetic to track the DPID numbers of the Root Switch and Slave Switch:

```
$ ~/.opam/4.06.0/bin/frenetic http-controller --verbosity debug
```

```
root@5829f58c2e1f:/# service apache2 status
 * apache2 is running
root@5829f58c2e1f:/# service mysql status
 * /usr/bin/mysqladmin  Ver 8.42 Distrib 5.7.29, for Linux on x86_64
Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Server version          5.7.29-0ubuntu0.18.04.1
Protocol version        10
Connection              Localhost via UNIX socket
UNIX socket             /var/run/mysqld/mysqld.sock
Uptime:                 28 min 42 sec

Threads: 1  Questions: 432  Slow queries: 0  Opens: 248  Flush tables: 1  Open tables: 241  Queries per second avg: 0.250
root@5829f58c2e1f:/#
```

```
root@e52704823b4a:/# service mysql start
 * Starting MySQL database server mysqld                                                    [fail]
root@e52704823b4a:/# service mysql restart
 * Stopping MySQL database server mysqld                                                    [ OK ]
 * Starting MySQL database server mysqld                                                    [ OK ]
root@e52704823b4a:/#
```

You should see the following output (without any flow table shown):

Leave this Controller node terminal running, and proceed to Root Switch and Slave Switch configuration.

```
root@5829f58c2e1f:/var/www/public_html/Dolus_DDos/app/Python# ~/.opam/4.06.0/bin/frenetic http-controller --verbosity debug
 [INFO] Calling create!
 [INFO] Current uid: 0
 [INFO] Successfully launched OpenFlow controller with pid 978
 [INFO] Connecting to first OpenFlow server socket
 [INFO] Failed to open socket to OpenFlow server: (Unix.Unix_error "Connection refused" connect 127.0.0.1:8984)
 [INFO] Retrying in 1 second
 [INFO] Successfully connected to first OpenFlow server socket
 [INFO] Connecting to second OpenFlow server socket
 [INFO] Successfully connected to second OpenFlow server socket
```

Note

If you mess up with any setup steps and are not sure how to recover, here is a easy solution to start from fresh:

- If you are inside the Docker container, log out of the container by holding "ctrl" and hitting "p" then "q"
- From the Controller VM console, issue the following commands:

```
$ sudo docker ps -a
$ sudo docker stop container_id
$ sudo docker rm container_id
```

- Then go to step 5.1.2 to repeat Controller node setup

6.2 General installation of Open vSwitch

Other than the Controller node, all the other nodes in our Dolus topology need to be configured as SDN switches that use VXLAN to configure SDN connections among the nodes. To install Open vSwitch on a VM instance, we use the installation release that can be found from *https://www.openvswitch.org/download/*.

In this instruction, we use the Open vSwitch ver2.11.0. The bash scripts to install Open vSwitch on a Linux VM instance with Debian 10 Buster operating system is as follows. The script requires root user to run the installation process.

$ sudo su

$ cat ovs-install.sh

```
apt -y update && apt -y upgrade
apt -y install git automake autoconf gcc uml-utilities libtool
 build-essential pkg-config linux-headers-$(uname -r) iperf screen
 python3-pip wget

# install open vSwitch
mkdir /local && cd /local
wget http://openvswitch.org/releases/openvswitch-2.11.0.tar.gz
tar xvfz openvswitch-2.11.0.tar.gz
cd openvswitch-2.11.0/
./boot.sh
./configure --disable-ssl --disable-libcapng
make
make install
make modules_install
modprobe openvswitch

# create necessary directories for Open vSwitch
mkdir -p /usr/local/etc/openvswitch
mkdir -p /usr/local/var/run/openvswitch
mkdir -p /usr/local/var/log/openvswitch

# init database
```

```
ovsdb-tool create /usr/local/etc/openvswitch/conf.db
 /local/openvswitch-2.11.0/vswitchd/vswitch.ovsschema

# run ovs database
ovsdb-server -v --log-file --pidfile
 --remote=punix:/usr/local/var/run/openvswitch/db.sock >/var/log/ovsdb.log
 2>&1 &

# run ovs switch
ovs-vswitchd --pidfile >/var/log/vswitchd.log 2>&1 &
```

In this script, we first update the Linux OS to update software packages. We then will install several dependencies for running Open vSwitch. Then we download the OVS ver2.11.0 installation package, extract files, and compile and install the package. For vSwitch to store data and logs, we also need to create directories and files for database. At the end, we initialize database and start running the vSwitch.

6.3 Open vSwitch installation for the rest of the topology

We will go through building an actual SDN topology using Open vSwitch VXLAN building blocks to get hands-on experience with SDN and OpenFlow. In the topology, we specified to create two Open vSwitches that will be used in the Dolus application system, and other nodes including server, attacker, QVM, and user.

What is a switch? A network switch is a network device that forwards packets between its interfaces based on some logic given to it through its configuration. A switch can be a layer-2 or layer-3 switch, in which the switch is able to forward network packets to a local or remote network IP address. With the OpenFlow protocol, a switch can forward network packets based on the network policies defined in the control plane, e.g. on a controller machine. The defined policies mostly include the network frame's header information e.g. source/destination MAC addresses, IP packet's header fields, ICMP flags, UDP/TCP port number, and a lot more, e.g. as shown in Figure 3. The switch forwards flows and network flows can be defined based on any set of policies in the context of SDN.

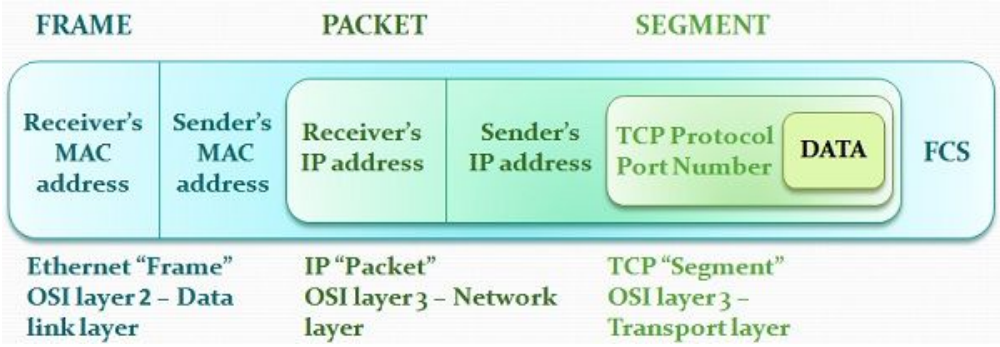


Figure 3: Network frame structure

6.4 Root Switch node setup

To configure the Root Switch, please follow the following steps:

6.4.1 SSH into the Root Switch using your method of choice.

6.4.2 Install Open vSwitch

Go to the /local directory, execute the following commands to create and start Open vSwitch:

```
$ sudo su
$ cd /local
$ nano ovs-install.sh
```

Paste the bash script from the above ovs-install.sh file, then

```
$ bash ./ovs-install.sh
$ exit
```

6.4.3 Run the following command to configure the Root Switch using the appriate IPs (make sure to use public IP for Controller):

Script to configure Root Switch (create the file and put in into the /local directory) :
$ root-switch_config.sh

```
#!/bin/bash
# $1 slave switch local IP
# $2 server1 local IP
# $3 Controller public IP

# set up bridges
sudo ovs-vsctl add-br br0

# setup VXLAN connections to slave switch,
sudo ovs-vsctl add-port br0 eth1 -- set interface eth1 type=vxlan
 options:remote_ip="$1" options:key=2001
sudo ovs-vsctl add-port br0 eth2 -- set interface eth2 type=vxlan
 options:remote_ip="$2" options:key=2002

# setup controller
sudo ovs-vsctl set-controller br0 tcp:"$3":6633

# start Virtual IP on the root switch
ifconfig br0 100.0.0.100 mtu 1400 up
```

To setup the Root Switch, run the following command, and replace the three arguments with the actual IPs:

```
$ sudo bash /local/rootSwitch_config.sh "slave_Local_IP" "server_Local_IP"
 "controller_Public_IP"
```

Example:

```
$ sudo bash /local/rootSwitch_config.sh 10.128.0.110 10.128.0.111
 35.225.251.103
```

You should the output as:

To check OVS setup, run the following command to check all the interfaces and Controller setup:

```
wangso@wangso-lab2-rootswitch:/local$ sudo bash ./rootSwitch_config.sh 10.128.0.34 10.128.0.35 35.188.174.107

Setting up briges...


#{BLUE}Setting up VXLAN connections ...


#{BLUE}Setting up controller...
```

```
$ sudo ovs-vsctl show
```

```
wangso@wangso-lab2small-rootswitch:/local$ sudo ovs-vsctl show
88a0b255-adfa-4fdf-9421-36a4703b3000
    Bridge "br0"
        Controller "tcp:35.225.251.103:6633"
            is_connected: true
        Port "eth1"
            Interface "eth1"
                type: vxlan
                options: {key="2001", remote_ip="10.128.0.110"}
        Port "eth2"
            Interface "eth2"
                type: vxlan
                options: {key="2002", remote_ip="10.128.0.111"}
        Port "br0"
            Interface "br0"
                type: internal
```

6.4.4 Checking for Switch ID on the Controller machine

Go back to the controller window where you have the Opam running, and check the output from Frenetic. You should see one entry in the flow table, which is the Root Switch info. Note down the long number which is the DPID of the Root Switch:

Note

If you mess up with switch configuration steps and are not sure how to recover, here is an easy solution to re-configure the node:
a) From the console, issue the following commands:

```
$ ovs-vsctl del-br br0
```

b) Then go back to step 5.4.2 to re-configure

6.5 Slave Switch node setup

6.5.1 SSH into the Slave Switch using your method of choice.

6.5.2 Intall Open vSwitch

Go to the /local directory, execute the following commands to create and start Open vSwitch:

```
$ sudo su
$ cd /local
$ nano ovs-install.sh
```

Paste the bash script from the above ovs-install.sh file, then

```
$ bash ./ovs-start.sh
$ exit
```

6.5.3 Run the following command to configure the Slave Switch using the appropriate IPs (make sure to use public IP for Controller):

Script to configure Slave Switch (create the file and put in into the /local directory) :
$ slave-switch_config.sh

```
# !/bin/bash
# $1 root switch local IP
# $2 attacker local IP
# $3 QVM public IP
# $4 user local IP
# $5 Controller public IP

# set up bridges
sudo ovs-vsctl add-br br0

# setup VXLAN connections to slave switch,
```

```
ovs-vsctl add-port br0 eth1 -- set interface eth1 type=vxlan
 options:remote_ip="$1" options:key=2001
ovs-vsctl add-port br0 eth2 -- set interface eth2 type=vxlan
 options:remote_ip="$2" options:key=2002
ovs-vsctl add-port br0 eth3 -- set interface eth3 type=vxlan
 options:remote_ip="$3" options:key=2003
ovs-vsctl add-port br0 eth4 -- set interface eth4 type=vxlan
 options:remote_ip="$4" options:key=2004

# setup controller
sudo ovs-vsctl set-controller br0 tcp:"$5":6633

# start Virtual IP on the slave switch
ifconfig br0 100.0.0.101 mtu 1400 up
```

To setup the Slave Switch, run the following command, and replace the three arguments with the actual IPs:

```
$ sudo bash /local/slaveSwitch_config.sh "rootswitch_Local_IP"
 "attacker_Local_IP" "QVM_Local_IP" "user_Local_IP" "controller_Public_IP"
```

Example:

```
$ sudo bash /local/rootSwitch_config.sh 10.128.0.109 10.128.0.112
 10.128.0.113 10.128.0.114 35.225.251.103
```

To check OVS setup, run the following command to check all the interfaces and Controller setup:

```
$ sudo ovs-vsctl show
```

6.5.4 Checking for Slave Switch ID on the Controller machine

Go back to the Controller node where you have the Opam running, and check the output from Frenetic. You should see two entries in the flow table, one of which is the Root Switch and the other is Slave Switch. You should already have the Root Switch DPID, now note down the other number, which is the Slave Switch DPID.

Note

If you mess up with switch configuration steps and are not sure how to recover, here is an easy solution to re-configure the node:
a) From the console, issue the following commands:

```
  $ ovs-vsctl del-br br0
```

b) Then go back to step 5.5.2 to re-configure

```
wangso@wangso-lab2small-slaveswitch:/local$ sudo ovs-vsctl show
37778cd0-a91b-4f65-b5f2-57e0b8bfb1ca
    Bridge "br0"
        Controller "tcp:35.225.251.103:6633"
            is_connected: true
        Port "eth4"
            Interface "eth4"
                type: vxlan
                options: {key="2004", remote_ip="10.128.0.114"}
        Port "eth1"
            Interface "eth1"
                type: vxlan
                options: {key="2001", remote_ip="10.128.0.109"}
        Port "eth3"
            Interface "eth3"
                type: vxlan
                options: {key="2003", remote_ip="10.128.0.113"}
        Port "eth2"
            Interface "eth2"
                type: vxlan
                options: {key="2002", remote_ip="10.128.0.112"}
        Port "br0"
            Interface "br0"
                type: internal
```

```
root@80911ad325cb:/# ~/.opam/4.06.0/bin/frenetic http-controller --verbosity debug
 [INFO] Calling create!
 [INFO] Current uid: 0
 [INFO] Successfully launched OpenFlow controller with pid 1263
 [INFO] Connecting to first OpenFlow server socket
 [INFO] Failed to open socket to OpenFlow server: (Unix.Unix_error "Connection refused" connect 127.0.0.1:8984)
 [INFO] Retrying in 1 second
 [INFO] Failed to open socket to OpenFlow server: (Unix.Unix_error "Connection refused" connect 127.0.0.1:8984)
 [INFO] Retrying in 1 second
 [INFO] Successfully connected to first OpenFlow server socket
 [INFO] Connecting to second OpenFlow server socket
 [INFO] Successfully connected to second OpenFlow server socket
[DEBUG] Setting up flow table
+----------------------------------+
| 192259385492800 | Pattern | Action |
|----------------------------------|
|                          |       |
+----------------------------------+

[DEBUG] Setting up flow table
+----------------------------------+
| 50933832370506 | Pattern | Action |
|----------------------------------|
|                          |       |
+----------------------------------+
```

6.6 Server nodes setup

6.6.1 SSH into the Server VM using your SSH key.

6.6.2 Install Open vSwitch

Go to the /local directory, execute the following commands to create and start Open vSwitch:

```
$ sudo su
$ cd /local
$ nano ovs-install.sh
```

Paste the bash script from the above ovs-install.sh file, then

```
$ bash ./ovs-start.sh
$ exit
```

6.6.3 Run the following command to configure the Server using the appropriate IPs:

Script to configure Server (create the file and put in into the /local directory):
$ server_config.sh

```
# !/bin/bash
# $1 root switch local IP

# set up bridges
sudo ovs-vsctl add-br br0

# setup VXLAN connections to slave switch,
sudo ovs-vsctl add-port br0 eth1 -- set interface eth1 type=vxlan
 options:remote_ip="$1" options:key=2002

# start Virtual IP on the slave switch
ifconfig br0 100.0.0.102 mtu 1400 up
```

To setup the Server, run the following command, and replace the argument with the Root Switch local IP:

```
$ sudo bash /local/server_config.sh "rootswitch_Local_IP"
```

Example:

```
$ sudo bash /local/slaveSwitch_config.sh 10.128.0.110
```

To check OVS setup, run the following command to check all the interfaces and Controller setup:

```
$ sudo ovs-vsctl show
```

6.7 Attacker VM setup

6.7.1 SSH into the Attacker VM using your SSH key.

6.7.2 Install Open vSwitch

Go to the /local directory, execute the following commands to create and start Open vSwitch:

```
$ sudo su
$ cd /local
$ nano ovs-install.sh
```

Paste the bash script from the above ovs-install.sh file, then

```
$ bash ./ovs-start.sh
$ exit
```

6.7.3 Run the following command to configure the Attacker using the appropriate IPs:

Script to configure Server (create the file and put in into the /local directory):
$ attacker_config.sh

```
# !/bin/bash
# $1 slave switch local IP

# set up bridges
sudo ovs-vsctl add-br br0

# setup VXLAN connections to slave switch,
sudo ovs-vsctl add-port br0 eth1 -- set interface eth1 type=vxlan
 options:remote_ip="$1" options:key=2002

# start Virtual IP on the slave switch
ifconfig br0 100.0.0.103 mtu 1400 up
```

To setup the Attacker, run the following command, and replace the argument with the Slave Switch local IP:

```
$ sudo bash /local/attacker_config.sh "slaveswitch_Local_IP"
```

Example:

```
$ sudo bash /local/attacker_config.sh 10.128.0.109
```

To check OVS setup, run the following command to check all the interfaces setup:

```
$ sudo ovs-vsctl show
```

6.8 QVM VM setup

6.8.1 SSH into the QVM VM using your SSH key.

6.8.2 Install Open vSwitch

Go to the /local directory, execute the following commands to create and start Open vSwitch:

```
$ sudo su
$ cd /local
$ nano ovs-install.sh
```

Paste the bash script from the above ovs-install.sh file, then

```
$ bash ./ovs-start.sh
$ exit
```

6.8.3 Run the following command to configure the QVM using the appropriate IPs:

Script to configure QVM (create the file and put in into the /local directory):
$ qvm_config.sh

```
# !/bin/bash
# $1 slave switch local IP

# set up bridges
sudo ovs-vsctl add-br br0

# setup VXLAN connections to slave switch,
sudo ovs-vsctl add-port br0 eth1 -- set interface eth1 type=vxlan
 options:remote_ip="$1" options:key=2003

# start Virtual IP on the slave switch
ifconfig br0 100.0.0.104 mtu 1400 up
```

To setup the QVM, run the following command, and replace the argument with the Slave Switch local IP:

```
$ sudo bash /local/qvm_config.sh "slaveswitch_Local_IP"
```

Example:

```
$ sudo bash /local/qvm_config.sh 10.128.0.109
```

To check OVS setup, run the following command to check all the interfaces setup:

```
$ sudo ovs-vsctl show
```

6.9 User VM setup

6.9.1 SSH into the User VM using your SSH key.

6.9.2 Install Open vSwitch

Go to the /local directory, execute the following commands to create and start Open vSwitch:

```
$ sudo su
$ cd /local
$ nano ovs-install.sh
```

Paste the bash script from the above ovs-install.sh file, then

```
$ bash ./ovs-start.sh
$ exit
```

6.9.3 Run the following command to configure the User using the appropriate IPs:

Script to configure Server (create the file and put in into the /local directory):
$ User_config.sh

```
# !/bin/bash
# $1 slave switch local IP
```

```
# set up bridges
sudo ovs-vsctl add-br br0

# setup VXLAN connections to slave switch,
sudo ovs-vsctl add-port br0 eth1 -- set interface eth1 type=vxlan
 options:remote_ip="$1" options:key=2004

# start Virtual IP on the User
ifconfig br0 100.0.0.105 mtu 1400 up
```

To setup the User, run the following command, and replace the argument with the Slave Switch local IP:

```
$ sudo bash /local/user_config.sh "slaveSwitch_Local_IP"
```

Example:

```
$ sudo bash /local/user_config.sh 10.128.0.109
```

To check OVS setup, run the following command to check all the interfaces setup:

```
$ sudo ovs-vsctl show
```

6.10 Configure Dolus AdminUI and Pox SDN controller on the Controller VM

Dolus AdminUI monitors the cloud network, keeps track of network components and network traffic, policies generated by Frenetic, and manages SDN network routing. In this part, we will install Dolus system on the Controller machine.

6.10.1 Open another terminal, login to the Controller VM using your SSH key.

6.10.2 Update database with the network topology and SDN network policy data

Inside the container, run the database_update.sh script, with the Root Switch and Slave Switch DPID as arguments. This step update the Dolus database with the DPIDs from your switches.

```
$ cd /var/www/public_html/Dolus_DDos/app/Python
$ bash ./database_update.sh rootDPID slaveDPID
```

Example:

```
$ bash ./database_update.sh 192259385492800 50933832370506
```

You should see the following output:

```
root@e52704823b4a:/var/www/public_html/Dolus_DDos/app/Python# bash ./database_update.sh 37767887839559 138561012800590
The AdminUI webpage has been configured properly!

The AdminUI installation has been completed!
You can now complete the rest of the lab!
root@e52704823b4a:/var/www/public_html/Dolus_DDos/app/Python#
```

6.11 Setup default routing protocol in SDN

6.11.1 Keep the first Controller terminal running the Frenetic/Opam.

6.11.2 In the second Controller terminal where you are logged in the container, go to "/var/www/public_html/ Dolus_DDos/app/Python/" directory, and edit the "openNetwork-ADAPTS.py" file:

Update the "root_switch = " and "slave_switch_1 = " with your own DPIDs of both switch (make sure the DPIDs match to the correct switches!).

Warning

Pay attention to the IP addresses in the IP4DstEq() and ports in the SetPort(). The IP addresses are OpenFlow virtual IP addresses, and the ports are the interfaces shown in the bridge br0 in the switch configuration steps.

6.11.3 After updating the openNetwork-ADAPTS.py file, run the following command from the same directory to initiate the default network routing (Your other Controller terminal must be running the Frenetic now, otherwise you will see error message):

```
$ python openNetwork-ADAPTS.py
```

The output in this terminal will look like this:

```
root@5829f58c2e1f:/var/www/public_html/Dolus_DDos/app/Python# python openNetwork-ADAPTS.py
No client_id specified. Using b30376807e7e45f2bdf56a61f677555e
Starting the tornado event loop (does not return).
```

From your other Controller terminal, you should see the default policy table in the output of Frenetic like below (only showing a part of the table):

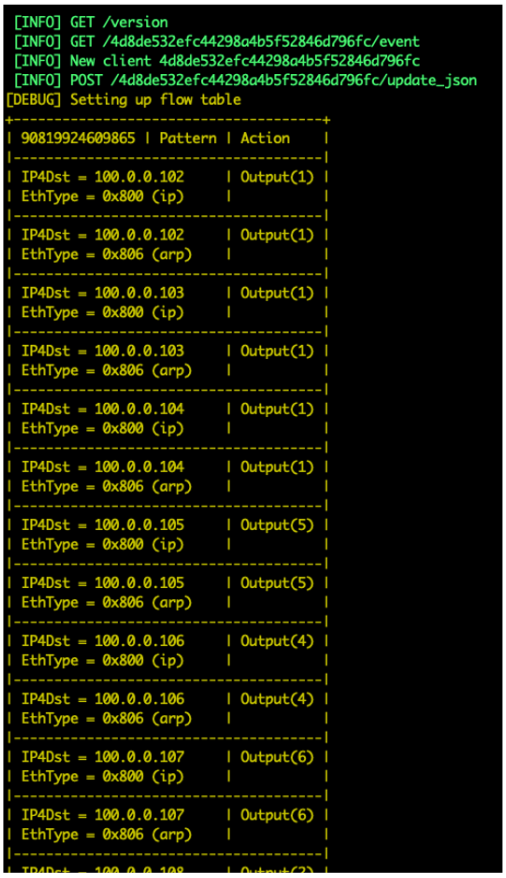
6.11.4 Now that the file has been executed, the bridge has been created to allow the servers connected to the Root Switch to be able to communicate with the servers connected to the slave-switch. Log into User connected to the Slave Switch and ping the Server connected to the Root Switch.

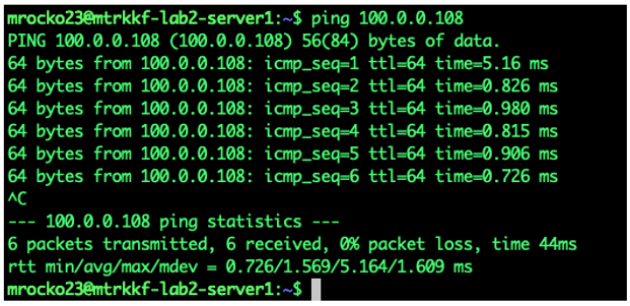
```
$ ping ip_addr
```

Replace ip_addr with the virtual IP address of the device you want to ping, which in the case of the Server is 100.0.0.102. Also try to ping from any node to any other node in the topology (except the Controller). Each ping should succeed, showing that network traffic can be sent via the connection between the two switches. An example ping output will look like this:

6.12 Stop default SDN routing

On your controller, stop the SDN controller by terminating the running openNetwork-ADAPTS.py script. Repeat the above ping commands and see the differences.





7 Congratulations! You have successfully implemented an virtual SDN overlay on top of the Google GCP network, and used the Pox controller to control the default network routing between the nodes in your topology.

8 Homework Problems

1) Problem 1
2) Problem 2

3) Problem 3
4) Problem 4

9 Lab Report

1) Submit the required screen shots in the lab steps
2) Short assay answers to the homework problems