

SDN_ADC_MM SETUP GUIDE:

TOOLS USED FOR INSTALLATION:

Git

A code management application used for pulling and storing code in a safe and reliable manner. It is used in this project to get the code needed for installation.

Screen

An application included on most Linux flavours, it allows the user to detach a process and reattach at a later point without losing progress or logs. It is used in this project to simplify the interactive deployment of network virtualization application Mininet.

Docker

Used to virtualize, containerize and manage the various applications in this project. It greatly reduces resource overhead and allows for more efficient operation and deployment.

TOOLS USED WITHIN PROJECT APPLICATION:

FLOODLIGHT SDN CONTROLLER

An opensource SDN Controller known for its developer-friendly API and many features. In this project it is used as the primary logical control point for the network.

MININET

A virtualization application that simulates SDN-enabled networks for development and testing.

TELEGRAF

The first step in the project's data pipeline. A data collector with many options for inputs and outputs. In this project it is using the 'exec' plugin and running python scripts that generate JSON data from the Floodlight API. From there it uses the 'Kafka' output plugin to forward that data to the Kafka application.

KAFKA

A message queueing and replication application. Used in this project to temporarily store and forward data from Telegraf to Logstash.

LOGSTASH

First element of the data pipeline's ELK (Elasticsearch, Logstash, Kibana) Stack. It pulls the data from Kafka before manipulating and indexing it to Elasticsearch.

ELASTICSEARCH

Primary datastore of the project. Used to index and store data received from Logstash then allow its retrieval for visualisations by Kibana and Grafana.

KIBANA

Data visualization tool provided by the ELK stack, also functions as Elasticsearch's GUI.

GRAFANA

Primary visualization tool used in this project. An opensource application with high amounts of versatility and adaptability. It is used in this project for data analysis, visualization and verification.

INTRODUCTION:

The solution presented in this setup guide was developed exclusively with the Ubuntu OS in mind, as a result, it is best to use either a physical or virtual machine running Ubuntu Server or Ubuntu Desktop.

All code used within this guide is available at: https://github.com/Faux212/SDN_ADC_MM

NOTE:

- *(The tools used within this deployment are available for a wide number of operating systems and could easily be adapted and installed on them – However, due to project time constraints, a focus was placed on a single OS).*
- *(Knowledge of Virtual Machines and how to create, boot and manage them is required for this installation. There are tools available to assist in virtualization – some of the popular solutions are free and available through VMware and Virtualbox).*
- *(If using the server edition of the OS, knowledge of port forwarding is required in place of accessing browsers within the VM).*

ENVIRONMENT:

To effectively run and maintain the software suite in this project, your machine should have the following specifications:

- 2 CPU Cores
- 2GB RAM
- 25GB (or more) available storage

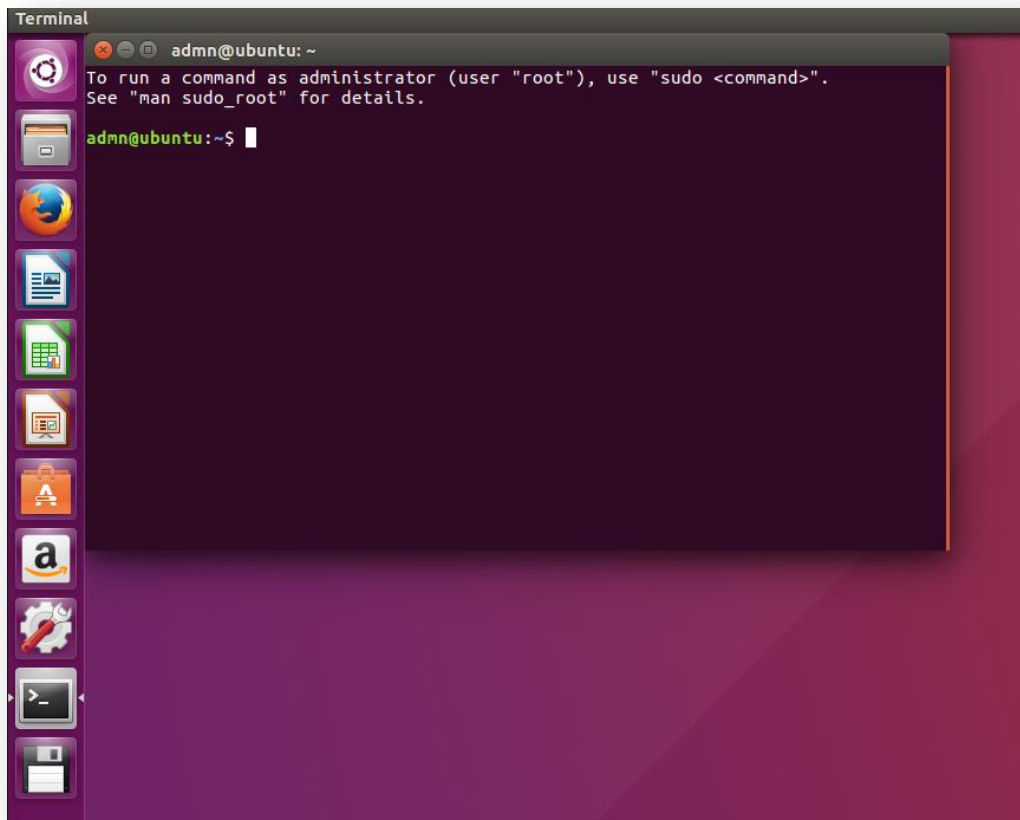
(If suite is to be implemented in live environments, it is best to consider a larger allotment of resources to store and analyse data.)

BOOT UP THE HOST:

Boot up your blank or existing Ubuntu server or VM.

UPDATE UBUNTU:

Use the key combination '**Ctrl+Alt+T**' to open a new terminal in ubuntu.



It is necessary to ensure the OS used is updated to the current version to eliminate risk of incompatibility or instability.

Input the following commands to ensure an up-to-date install of Ubuntu:

sudo apt-get update -y

sudo apt-get upgrade -y

```
adm@ubuntu: ~  
Setting up libreoffice-writer (1:5.1.6-rc2-0ubuntu1-xenial6) ...  
Setting up libreoffice-draw (1:5.1.6-rc2-0ubuntu1-xenial6) ...  
Setting up libreoffice-impress (1:5.1.6-rc2-0ubuntu1-xenial6) ...  
Setting up libreoffice-ogltrans (1:5.1.6-rc2-0ubuntu1-xenial6) ...  
Setting up libreoffice-pdfimport (1:5.1.6-rc2-0ubuntu1-xenial6) ...  
Setting up python3-uno (1:5.1.6-rc2-0ubuntu1-xenial6) ...  
Setting up libreoffice-math (1:5.1.6-rc2-0ubuntu1-xenial6) ...  
Setting up libreoffice-avmedia-backend-gstreamer (1:5.1.6-rc2-0ubuntu1-xenial6) ...  
Setting up ubuntu-release-upgrader-gtk (1:16.04.26) ...  
Setting up update-manager (1:16.04.15) ...  
Setting up update-notifier (3.168.10) ...  
Setting up libcamel-1.2-54:amd64 (3.18.5-1ubuntu1.1) ...  
Setting up libedataserver-1.2-21:amd64 (3.18.5-1ubuntu1.1) ...  
Setting up libebook-1.2-10:amd64 (3.18.5-1ubuntu1.1) ...  
Setting up evolution-data-server-online-accounts (3.18.5-1ubuntu1.1) ...  
Setting up libebook-contacts-1.2-2:amd64 (3.18.5-1ubuntu1.1) ...  
Setting up libedata-book-1.2-25:amd64 (3.18.5-1ubuntu1.1) ...  
Setting up libebook-1.2-16:amd64 (3.18.5-1ubuntu1.1) ...  
Setting up libecal-1.2-19:amd64 (3.18.5-1ubuntu1.1) ...  
Setting up libedata-cal-1.2-28:amd64 (3.18.5-1ubuntu1.1) ...  
Setting up evolution-data-server (3.18.5-1ubuntu1.1) ...  
Setting up bluez-obexd (5.37-0ubuntu5.1) ...  
Setting up libedataserverui-1.2-1:amd64 (3.18.5-1ubuntu1.1) ...  
Setting up ubuntu-desktop (1.361.2) ...  
Processing triggers for libgtk-3-0:amd64 (3.18.9-1ubuntu3.3) ...  
Processing triggers for libc-bin (2.23-0ubuntu10) ...  
Processing triggers for initramfs-tools (0.122ubuntu8.14) ...  
update-initramfs: Generating /boot/initrd.img-4.10.0-28-generic  
Processing triggers for ca-certificates (20170717-16.04.2) ...  
Updating certificates in /etc/ssl/certs...  
17 added, 42 removed; done.  
Running hooks in /etc/ca-certificates/update.d...  
done.  
Processing triggers for resolvconf (1.78ubuntu6) ...  
Processing triggers for systemd (229-4ubuntu21.15) ...  
Processing triggers for ureadahead (0.100.0-19) ...  
adm@ubuntu:~$
```

After the terminal arrives back at the prompt (as seen below), you can continue to the next step of install.

INSTALL GIT:

To access the project code, it is necessary to install and use the application Git. Git is a code management solution that allows code to be tracked and stored efficiently, especially during software development.

To install git, enter the following command into the terminal:

sudo apt-get install git -y

Once this has finished and you have returned to the prompt, move on to the next step.

CLONE CODE REPOSITORY:

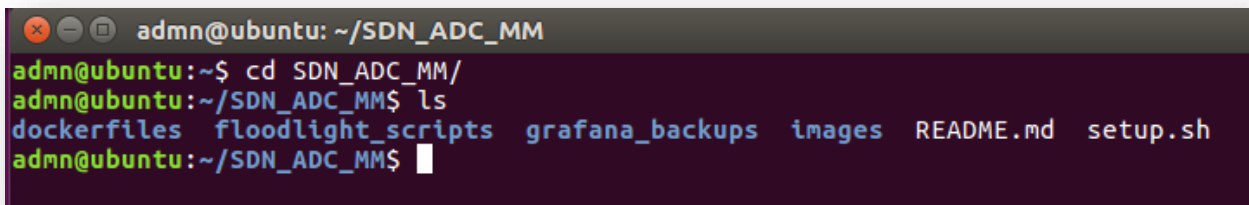
You are now going to clone the Git repository housing the project code. To do this, enter the following command into the terminal:

git clone https://github.com/faux212/SDN_ADC_MM

This may take a minute or two depending upon your internet connection. Once it has finished and you have again arrived at the prompt, change into the new code directory 'SDN_ADC_MM' using the following command:

cd SDN_ADC_MM/

To view the contents of the directory use the 'ls' command.

A terminal window screenshot with a dark background. The prompt is 'adm@ubuntu: ~/SDN_ADC_MM'. The user enters 'cd SDN_ADC_MM/' and the prompt changes to 'adm@ubuntu: ~/SDN_ADC_MM\$'. Then the user enters 'ls' and the output is displayed: 'dockerfiles floodlight_scripts grafana_backups images README.md setup.sh'.

```
adm@ubuntu: ~/SDN_ADC_MM
adm@ubuntu:~$ cd SDN_ADC_MM/
adm@ubuntu:~/SDN_ADC_MM$ ls
dockerfiles  floodlight_scripts  grafana_backups  images  README.md  setup.sh
adm@ubuntu:~/SDN_ADC_MM$
```

RUN SETUP SCRIPT:

In this step we will run the major setup script included in the project. This script takes **approximately 10 minutes to complete** depending upon internet speed and machine resources. To start it, use the following command:

sudo bash setup.sh

The script will download, setup and install:

- Java
- Python
- Docker
- Docker management network
- Containerised Applications (*Descriptions available above*):
 - o Portainer
 - o Floodlight Controller
 - o Telegraf
 - o Kafka (Three brokers and a zookeeper)
 - o Logstash
 - o Elasticsearch
 - o Kibana
 - o Grafana

It will also enable statistic collection on the Floodlight Controller.

NOTE: (This script will be very verbose in the terminal window as it displays the logs for all running containers it builds).

The terminal will display many errors when the script completes, indicating a JSON message cannot be parsed. This is because a virtual network has not been created yet.

```
2019-02-13T09:30:00Z D! [outputs.kafka] buffer fullness: 0 / 10000 metrics.
2019-02-13T09:30:00Z E! [inputs.exec]: Error in plugin: unable to parse out as JSON, json: cannot unmarshal array into Go value of type map[string]interface {}
2019-02-13T09:30:00Z E! [inputs.exec]: Error in plugin: unable to parse out as JSON, json: cannot unmarshal array into Go value of type map[string]interface {}
2019-02-13T09:30:00Z E! [inputs.exec]: Error in plugin: unable to parse out as JSON, json: cannot unmarshal array into Go value of type map[string]interface {}
2019-02-13T09:30:00Z E! [inputs.exec]: Error in plugin: unable to parse out as JSON, invalid character '}' looking for beginning of value
```

Verify the containers are up and running by opening a new terminal and using the command:

sudo docker ps

(This will list all the running Docker containers)

```
adm@ubuntu:~$ sudo docker ps
[sudo] password for adm:
CONTAINER ID        IMAGE               COMMAND             NAMES                  CREATED             STATUS
PORTS
4be35bff6390       telegraf_collector "/entrypoint.sh tele..." 3 minutes ago       Up 3 minutes
8092/tcp, 8094/tcp, 8092/udp, 8125/tcp, 8125/udp telegraf_collector
c517edfc537f       grafana_doc        "/run.sh"          grafana                6 minutes ago       Up 5 minutes
0.0.0.0:3000->3000/tcp
84c05fbf1447       kibana_doc         "/bin/bash /usr/loca..." 6 minutes ago       Up 6 minutes
0.0.0.0:5601->5601/tcp kibana
7de8f5b6023b       elasticsearch_doc  "/usr/local/bin/dock..." 6 minutes ago       Up 6 minutes
9200/tcp, 9300/tcp elasticsearch
21eb3ed2832b       logstash_doc       "/usr/local/bin/dock..." 6 minutes ago       Up 6 minutes
5044/tcp, 9600/tcp logstash
1d478a3f95c3       kafka_doc_9092     "/kafka/start.sh"    11 minutes ago      Up 11 minutes
9092/tcp kafka_9092_local
09f55998da38       zookeeper         "/docker-entrypoint..." 11 minutes ago      Up 11 minutes
2181/tcp, 2888/tcp, 3888/tcp zookeeper
b94684626ce5       kafka_doc_9091     "/kafka/start.sh"    11 minutes ago      Up 11 minutes
9091/tcp kafka_9091_local
3c0e02e7b90b       kafka_doc_9090     "/kafka/start.sh"    11 minutes ago      Up 11 minutes
9090/tcp kafka_9090_local
7c0035628825       portainer/portainer "/portainer"         Portainer_GUI          14 minutes ago      Up 4 minutes
0.0.0.0:9000->9000/tcp
24900d2f3b4e       glefevre/floodlight "/bin/sh -c 'java -j..." 14 minutes ago      Up 14 minutes
6653/tcp, 8080/tcp Floodlight_Controller
adm@ubuntu:~$
```

The output should be similar to the following image:

Now most of our infrastructure is functional, we can deploy a virtual network for it to monitor and configure.

RUN MININET:

This step is only intended for education and development purposes, all real-world deployments should point existing network equipment at the virtual SDN controller.

Mininet is an application that builds a virtual network for the software suite to interact with. It is an interactive application that we will use throughout the lab manual after setup completion. As such, it is important to **open a new terminal window** and use the 'screen' application.

Screen is recommended for use as it will allow detachment of the container and optional reattachment later. Documentation is available at: <https://www.gnu.org/software/screen/manual/screen.html>

Open a new instance of screen using the command:

screen

Press **enter** to proceed past the splash screen.

Copy and paste the following into the new terminal:

sudo docker pull iwaseyusuke/mininet && sudo docker run -it --privileged -e DISPLAY-- --net SDNet_Docker --ip 172.18.0.3 --name Mininet_Container -v /tmp/.X11-unix:/tmp/.X11-unix -v /lib/modules:/lib/modules iwaseyusuke/mininet

This command will download the docker image and run a new, **interactive** instance of the Mininet.

After this has completed a **root** prompt that is attached to the container will appear:

```
adm@ubuntu:~$ sudo docker pull iwaseyusuke/mininet && sudo docker run -it --privileged -e DISPLAY-- --net SDNet_Docker --ip 172.18.0.3 --name Mininet_Container -v /tmp/.X11-unix:/tmp/.X11-unix -v /lib/modules iwaseyusuke/mininet
Using default tag: latest
latest: Pulling from iwaseyusuke/mininet
Digest: sha256:d7c172273cb8dc9cb534dc8f5b3806a10c927315ef5fbb20b5f2c7bea9144d68
Status: Image is up to date for iwaseyusuke/mininet:latest
* Inserting openvswitch module
* /etc/openvswitch/conf.db does not exist
* Creating empty database /etc/openvswitch/conf.db
* Starting ovsdb-server
* Configuring Open vSwitch system IDs
* Starting ovs-vswitchd
* Enabling remote OVSDB managers
root@eac85c1e7e11:~#
```

Here we can specify the requirements of the virtual network we want Mininet to build. For this implementation, we will use a tree topology with 7 switches and 8 end-hosts. We are also connecting it to our SDN controller at 172.18.0.2.

Use the command:

mn --controller=remote,ip=172.18.0.2 --topo=tree,3,2

```
root@eac85c1e7e11:~# mn --controller=remote,ip=172.18.0.2 --topo=tree,3,2
*** Error setting resource limits. Mininet's performance may be affected.
*** Creating network
*** Adding controller
Connecting to remote controller at 172.18.0.2:6653
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7
*** Adding links:
(s1, s2) (s1, s5) (s2, s3) (s2, s4) (s3, h1) (s3, h2) (s4, h3) (s4, h4) (s5, s6) (s5, s7) (s6, h5) (s6, h6) (s7, h7) (s7, h8)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8
*** Starting controller
c0
*** Starting 7 switches
s1 s2 s3 s4 s5 s6 s7 ...
*** Starting CLI:
mininet> █
```

After the network has initialized, a new 'mininet>' prompt will appear. From this prompt it is possible to issue instructions to network devices and hosts. More documentation is available at:

<https://github.com/mininet/mininet/wiki/Documentation>

We now have a fully integration virtual network and software suite to monitor and configure it. The next step revolves around the potential of automatic configuration.

RUN AUTO-CONFIGURATION SCRIPT:

Open another terminal window. At this stage you will have 3 windows open.

Navigate to the folder: SDN_ADC_MM/floodlight_scripts and run the 'ls' command

```
adm@ubuntu:~$ cd SDN_ADC_MM/floodlight_scripts/
adm@ubuntu:~/SDN_ADC_MM/floodlight_scripts$ ls
get_net_stats.py      tele_get_net_stats.py  test_api.py
set_flow.py           tele_get_switch_data.py
tele_get_device_data.py tele_get_switch_flows.py
```


Listed in this directory are a number of python scripts that interact with the Floodlight Controller's API. Most are used by the monitoring system to gather data, but the one that performs automatic configuration is **set_flow.py**.

This script will automatically detect best paths to all end-points from each switch and dynamically set the relevant flows on each switch for each device. The flow will dictate onto which port should the switch forward data destined for the end-point's MAC address.

To run the python script simply enter the command:

python set_flow.py

```
adm@ubuntu:~/SDN_ADC_MM/floodlight_scripts$ python set_flow.py
```

If all the infrastructure and Mininet have been built correctly, a number of 'OK' messages should print to screen.

```
SETTING 8 FLOWS ON SWITCH 00:00:00:00:00:00:03. THEY ARE:
Flow_0: Port 1 --> 6e:40:7a:e0:0e:02
(200, 'OK', '{"status" : "Entry pushed"}')
Flow_1: Port 2 --> 1e:65:26:00:39:f1
(200, 'OK', '{"status" : "Entry pushed"}')
Flow_2: Port 3 --> 9e:2f:87:4a:be:56
(200, 'OK', '{"status" : "Entry pushed"}')
Flow_3: Port 3 --> c6:be:5f:bb:bf:ac
(200, 'OK', '{"status" : "Entry pushed"}')
Flow_4: Port 3 --> de:74:09:ca:46:6a
(200, 'OK', '{"status" : "Entry pushed"}')
Flow_5: Port 3 --> ae:69:49:45:d7:23
(200, 'OK', '{"status" : "Entry pushed"}')
Flow_6: Port 3 --> 46:54:66:eb:59:20
(200, 'OK', '{"status" : "Entry pushed"}')
Flow_7: Port 3 --> 82:ae:73:78:73:d3
(200, 'OK', '{"status" : "Entry pushed"}')
```

Now flows for the entire network have been set for each switch. This script can be set up to run autonomously for further effective automation.

ACCESSING APPLICATION WEBPAGES:

Many of the containerized applications have web-interfaces that are extremely useful to view and verify data as well as overall network health. Listed in the table below are various addresses to access the respective application's interfaces.

Simply open your web-browser and place the web address below into the address field at the top of the screen.

<i>Application</i>	<i>Web Address</i>
<i>Grafana</i>	localhost:3000
<i>Portainer</i>	localhost:9000
<i>Kibana</i>	localhost:5601
<i>Floodlight Controller</i>	172.18.0.2:8080/ui/pages/index.html

MOVE ON TO LAB MATERIALS:

Now that the entire application has been setup and verified as running smoothly, please continue reading the second document provided "SDN_ADC_MM Lab Manual" to explore the functionality and manageability of the software suite you just built.