SDN_ADC_MM LAB MANUAL:

(Complete this lab after using the Lab Setup Guide provided.)

LAB PURPOSE:

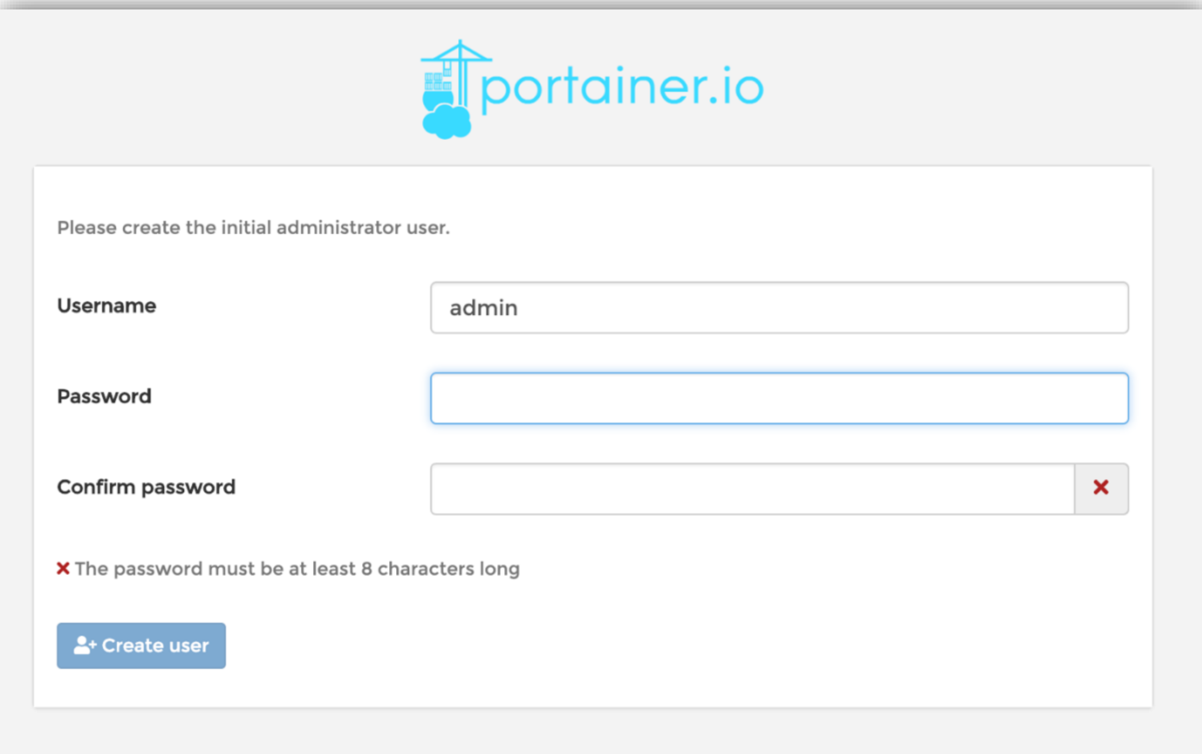
To educate, illustrate and demonstrate on:

- Virtualisation (Containers and Mininet)
- Software Defined Networking (SDN)
- Basic UNIX operation
- Python scripting
- Data Analysis/Collection/Visualisation

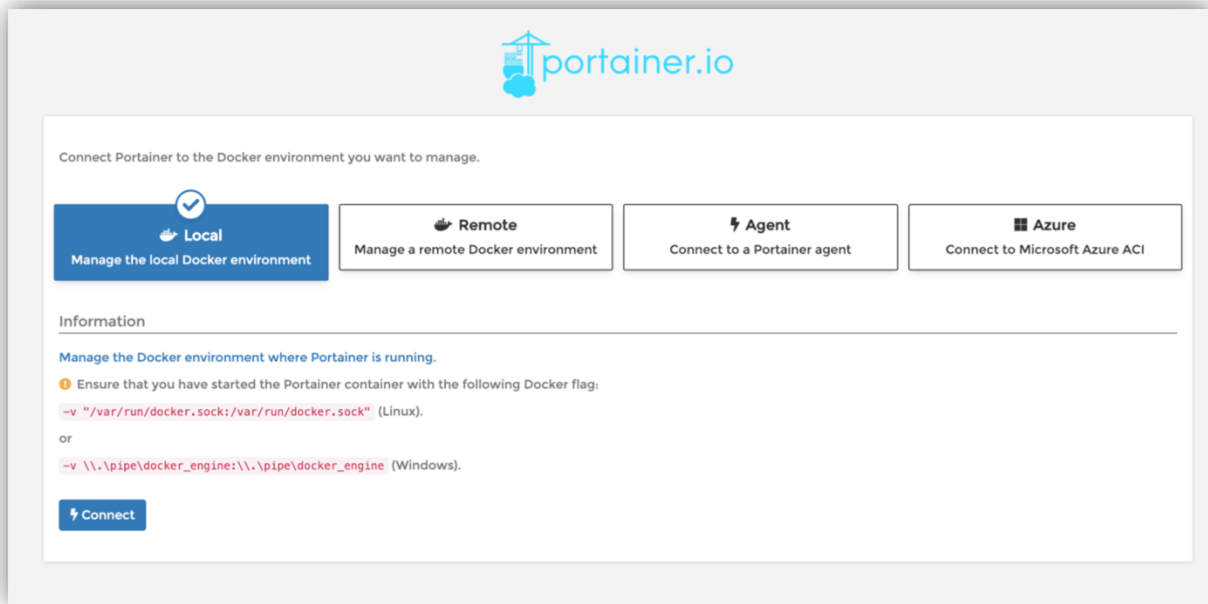
STEP 1: ACCESS PORTAINER

To verify setup went as intended, access LOCALHOST:9000 from a browser in your Virtual Machine.

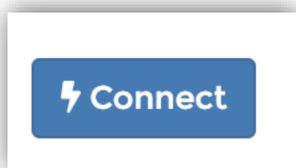
We'll setup some basic authentication for portainer, leave the username as "**admin**" and set the password to "**admin123**".



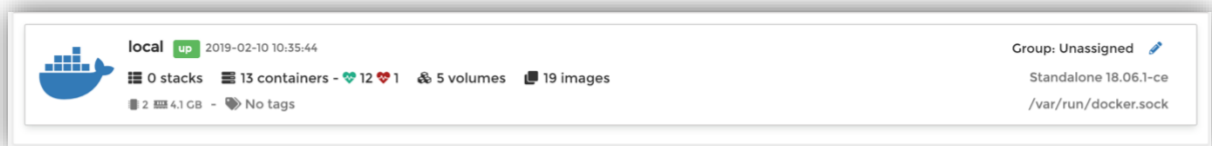
After creating the user, we are now going to select the **local** instance of Docker for Portainer to manage:



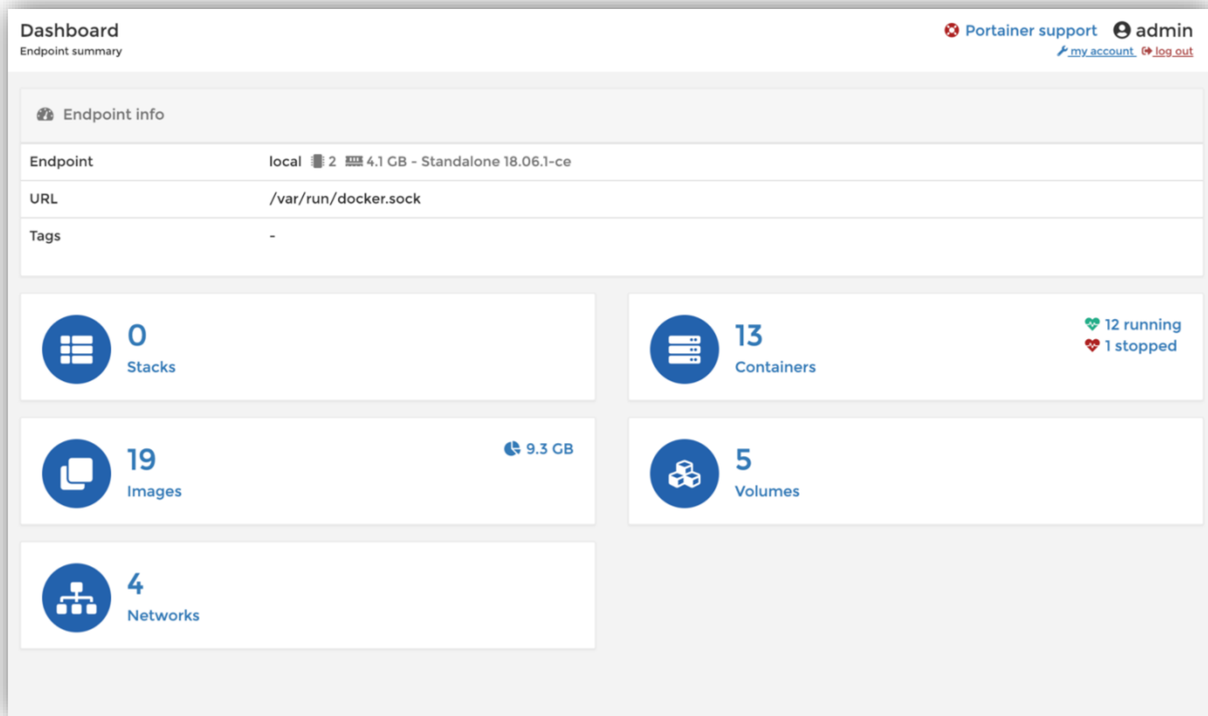
After selecting **Local**, click the **connect** button.



Then select the local endpoint container stack:



The details of your stack should match the following:



STEP 2: VERIFY DATA FLOW

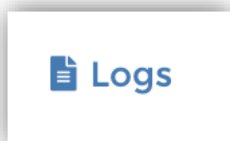
We will now check to see if monitoring data is successfully passing through the data pipeline. We will be checking the logs of application Logstash, as it contains a comprehensive log of the data passing through it.

Click on Containers:

From here you will see a list of the running containers and their overviews.

Click "logstash" from the list of containers.

Click on "Logs":

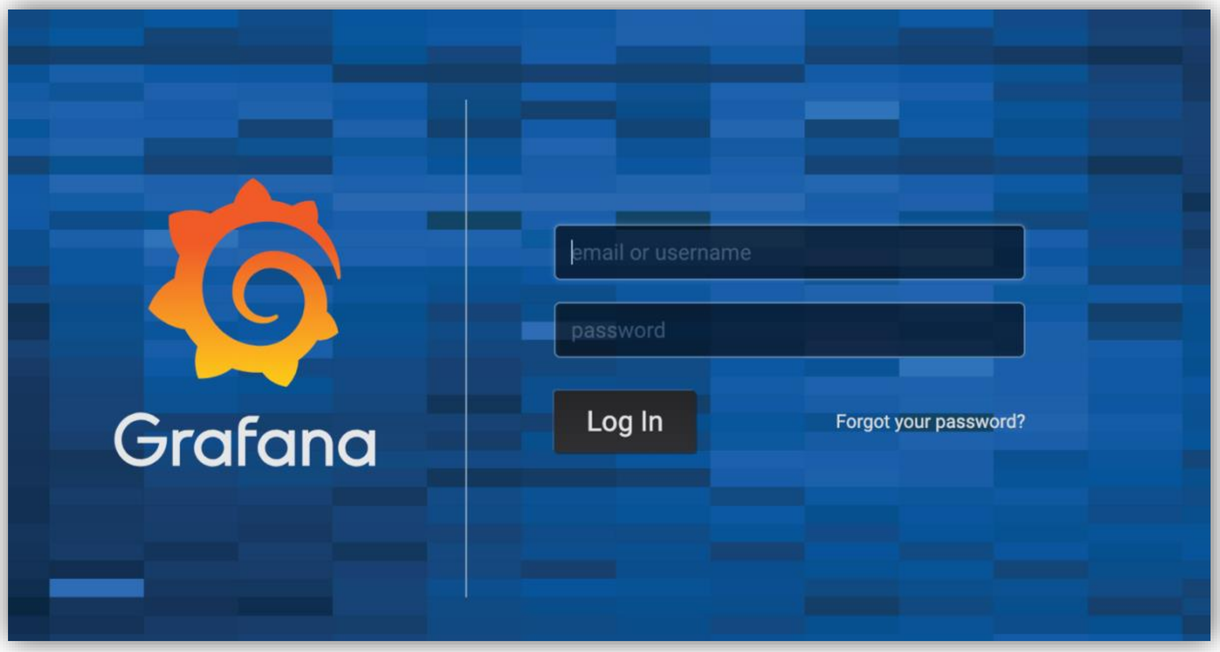


The window at the bottom of the page should display a live feed of JSON data being processed by Logstash and sent to Elasticsearch for storing:


```
           "tags" => {
         "openFlowVersion" => "OF_14",
             "inetAddress" => "172.18.0.3:35834",
              "switchDPID" => "00:00:00:00:00:00:00:03"
       }
   }
   {
         "@version" => "1",
           "fields" => {
         "link-speed-bits-per-second" => 10000000,
                               "port" => 2,
                 "bits-per-second-tx" => 126,
                 "bits-per-second-rx" => 0
       },
       "@timestamp" => 2019-02-09T23:42:20.091Z,
             "name" => "exec",
        "timestamp" => 1549755735,
             "tags" => {
              "dpid" => "00:00:00:00:00:00:00:06",
           "updated" => "Sat Feb 09 23:42:14 GMT 2019"
       }
   }
```

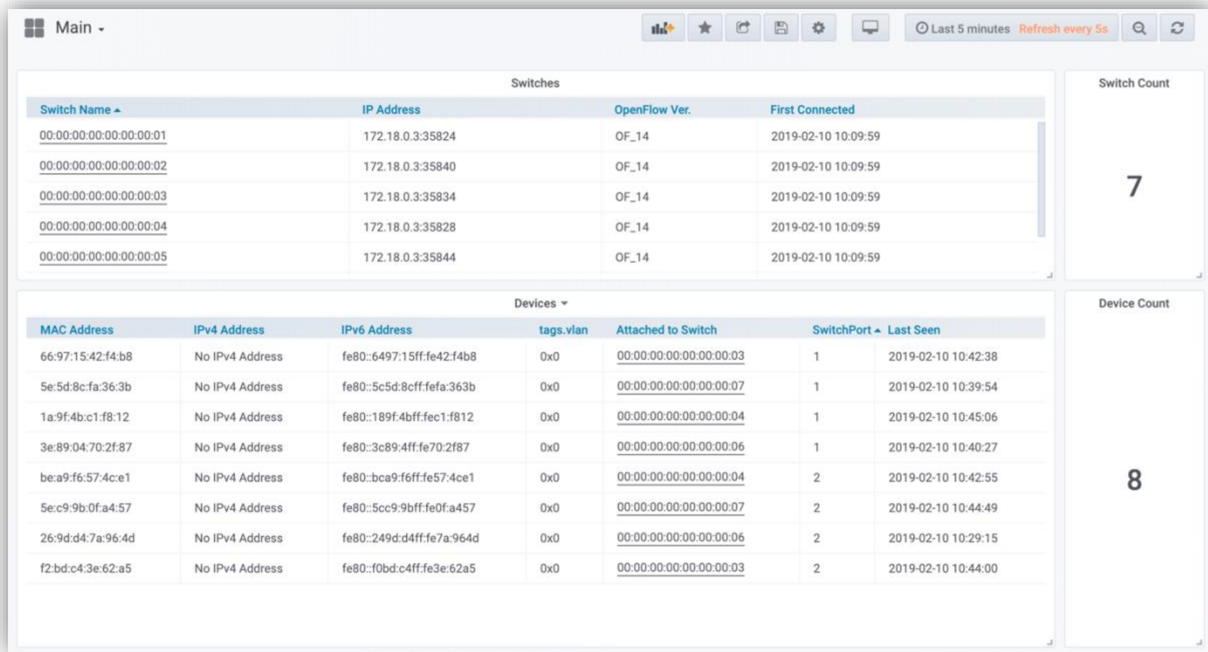

STEP 3: ACCESS GRAFANA

To reach the Grafana application, access localhost:3000 in your browser:



Authentication details are: User:Admin, Password:ouf44t

After this, you will be redirected to the main dashboard as displayed below.



Switches

Switch Name	IP Address	OpenFlow Ver.	First Connected
00:00:00:00:00:00:00:01	172.18.0.3:35824	OF_14	2019-02-10 10:09:59
00:00:00:00:00:00:00:02	172.18.0.3:35840	OF_14	2019-02-10 10:09:59
00:00:00:00:00:00:00:03	172.18.0.3:35834	OF_14	2019-02-10 10:09:59
00:00:00:00:00:00:00:04	172.18.0.3:35828	OF_14	2019-02-10 10:09:59
00:00:00:00:00:00:00:05	172.18.0.3:35844	OF_14	2019-02-10 10:09:59

Switch Count

7

Devices

MAC Address	IPv4 Address	IPv6 Address	tags.vlan	Attached to Switch	SwitchPort	Last Seen
66:97:15:42:f4:b8	No IPv4 Address	fe80::6497:15ff:fe42:f4b8	0x0	00:00:00:00:00:00:00:03	1	2019-02-10 10:42:38
5e:5d:8c:fa:36:3b	No IPv4 Address	fe80::5c5d:8cff:fefa:363b	0x0	00:00:00:00:00:00:00:07	1	2019-02-10 10:39:54
1a:9f:4b:c1:f8:12	No IPv4 Address	fe80::189f:4bff:fec1:f812	0x0	00:00:00:00:00:00:00:04	1	2019-02-10 10:45:06
3e:89:04:70:2f:87	No IPv4 Address	fe80::3c89:4ff:fe70:2f87	0x0	00:00:00:00:00:00:00:06	1	2019-02-10 10:40:27
be:a9:f6:57:4c:e1	No IPv4 Address	fe80::bca9:f6ff:fe57:4ce1	0x0	00:00:00:00:00:00:00:04	2	2019-02-10 10:42:55
5e:c9:9b:0f:a4:57	No IPv4 Address	fe80::5cc9:9bff:fe0f:a457	0x0	00:00:00:00:00:00:00:07	2	2019-02-10 10:44:49
26:9d:d4:7a:96:4d	No IPv4 Address	fe80::249d:d4ff:fe7a:964d	0x0	00:00:00:00:00:00:00:06	2	2019-02-10 10:29:15
f2:bd:c4:3e:62:a5	No IPv4 Address	fe80::f0bd:c4ff:fe3e:62a5	0x0	00:00:00:00:00:00:00:03	2	2019-02-10 10:44:00

Device Count

8

This page displays all the virtual network devices we setup using Mininet in the setup guide. There should be 7 Switches and 8 Devices total.

If you scroll down on this dashboard, you will see an empty table named "Flow Table", we are going to use an automation script to set these flows dynamically.

STEP 4: SETUP SWITCH FLOWS

```
admn@admn-virtual-machine:~/git$ cd SDN_ADC_MM/floodlight_scripts/
```

In your VM, open a new terminal and navigate to the floodlight_scripts directory within the git repository:

From there, run the command "python set_flow.py" to automatically set best path flows for all the switches to all devices.

```
admn@admn-virtual-machine:~/git/SDN_ADC_MM/floodlight_scripts$ python set_flow.py
```

The screen should display the following output:

```
#### Switch 00:00:00:00:00:00:00:02 ####
{'Latency': 2, 'Direction': 'bidirectional', 'Link_Class': 'Switch-Switch', 'Dest_SW': '00:00:00:00:00:00:00:03', 'Type': 'internal', 'Dest_Port': 3}
{'Latency': 2, 'Direction': 'bidirectional', 'Link_Class': 'Switch-Switch', 'Dest_SW': '00:00:00:00:00:00:00:04', 'Type': 'internal', 'Dest_Port': 3}
{'Latency': 1, 'Direction': 'bidirectional', 'Link_Class': 'Switch-Switch', 'Dest_SW': '00:00:00:00:00:00:00:01', 'Type': 'internal', 'Dest_Port': 1}
[NOTICE]:     Finding Path for end host: 3e:89:04:70:2f:87. (On Switch: 00:00:00:00:00:00:00:06 Port Number: 1)
[NOTICE]:     Finding Path for end host: 1a:9f:4b:c1:f8:12. (On Switch: 00:00:00:00:00:00:00:04 Port Number: 1)
[NOTICE]:    Setting flow for device (1a:9f:4b:c1:f8:12) for this switch (00:00:00:00:00:00:00:02) on Port 2
[NOTICE]:     Finding Path for end host: 5e:5d:8c:fa:36:3b. (On Switch: 00:00:00:00:00:00:00:07 Port Number: 1)
[NOTICE]:     Finding Path for end host: be:a9:f6:57:4c:e1. (On Switch: 00:00:00:00:00:00:00:04 Port Number: 2)
[NOTICE]:    Setting flow for device (be:a9:f6:57:4c:e1) for this switch (00:00:00:00:00:00:00:02) on Port 2
[NOTICE]:     Finding Path for end host: 5e:c9:9b:0f:a4:57. (On Switch: 00:00:00:00:00:00:00:07 Port Number: 2)
[NOTICE]:     Finding Path for end host: 66:97:15:42:f4:b8. (On Switch: 00:00:00:00:00:00:00:03 Port Number: 1)
[NOTICE]:    Setting flow for device (66:97:15:42:f4:b8) for this switch (00:00:00:00:00:00:00:02) on Port 1
[NOTICE]:     Finding Path for end host: 26:9d:d4:7a:96:4d. (On Switch: 00:00:00:00:00:00:00:06 Port Number: 2)
[NOTICE]:     Finding Path for end host: f2:bd:c4:3e:62:a5. (On Switch: 00:00:00:00:00:00:00:03 Port Number: 2)
[NOTICE]:    Setting flow for device (f2:bd:c4:3e:62:a5) for this switch (00:00:00:00:00:00:00:02) on Port 1

 SETTING 8 FLOWS ON SWITCH 00:00:00:00:00:00:00:02. THEY ARE:
Flow_0: Port 3 --> 3e:89:04:70:2f:87
(200, 'OK', '{"status" : "Entry pushed"}')
Flow_1: Port 2 --> 1a:9f:4b:c1:f8:12
(200, 'OK', '{"status" : "Entry pushed"}')
Flow_2: Port 3 --> 5e:5d:8c:fa:36:3b
(200, 'OK', '{"status" : "Entry pushed"}')
Flow_3: Port 2 --> be:a9:f6:57:4c:e1
(200, 'OK', '{"status" : "Entry pushed"}')
Flow_4: Port 3 --> 5e:c9:9b:0f:a4:57
(200, 'OK', '{"status" : "Entry pushed"}')
Flow_5: Port 1 --> 66:97:15:42:f4:b8
(200, 'OK', '{"status" : "Entry pushed"}')
Flow_6: Port 3 --> 26:9d:d4:7a:96:4d
(200, 'OK', '{"status" : "Entry pushed"}')
Flow_7: Port 1 --> f2:bd:c4:3e:62:a5
(200, 'OK', '{"status" : "Entry pushed"}')

admn@admn-virtual-machine:~/git/SDN_ADC_MM/floodlight_scripts$
```

STEP 5: CHECK RESULTS IN GRAFANA

Log back into Grafana and check the Flow Table panel. After automatically pushing the flows with the python script, the SDN Controller acknowledges the new flows for each switch.

There should be enough flows for each device on all switches (A total of 8). If there are some missing, attempt to refresh the page and it should fix the problem.

Panel Title

Flow Name	Destination MAC	Instructions	Priority	Flag(s)
Flow_00:00:00:00:00:00:00:03_0	9e:ff:0f:dd:d9:a8	output=1	32768	1
Flow_00:00:00:00:00:00:00:03_1	82:4d:8d:48:3a:43	output=2	32768	1
Flow_00:00:00:00:00:00:00:03_2	9e:ea:0b:2a:23:cc	output=3	32768	1
Flow_00:00:00:00:00:00:00:03_3	e2:07:87:b7:15:28	output=3	32768	1
Flow_00:00:00:00:00:00:00:03_4	7a:b0:19:cb:9a:85	output=3	32768	1
Flow_00:00:00:00:00:00:00:03_5	e6:79:15:b1:46:39	output=3	32768	1
Flow_00:00:00:00:00:00:00:03_6	4a:d8:81:d3:e8:b8	output=3	32768	1
Flow_00:00:00:00:00:00:00:03_7	e2:05:05:d3:62:bf	output=3	32768	1

Attached Devices

MAC Address	IPv4 Address	IPv6 Address	tags.vlan	Attached to Switch	SwitchPort	Last Seen
9e:ff:0f:dd:d9:a8	No IPv4 Address	fe80::9cff:fff:fedd:d9a8	0x0	00:00:00:00:00:00:00:03	1	2019-02-04 18:10:22
82:4d:8d:48:3a:43	No IPv4 Address	fe80::804d:8dff:fe48:3a43	0x0	00:00:00:00:00:00:00:03	2	2019-02-04 18:14:44

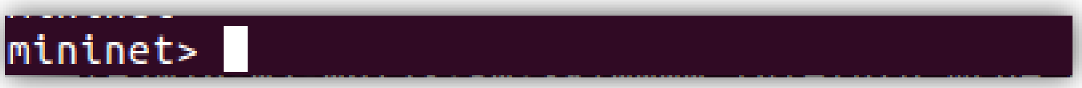
For the next step we will see how live traffic can be manipulated with Mininet and represented live in Grafana.

STEP 6: USING MININET

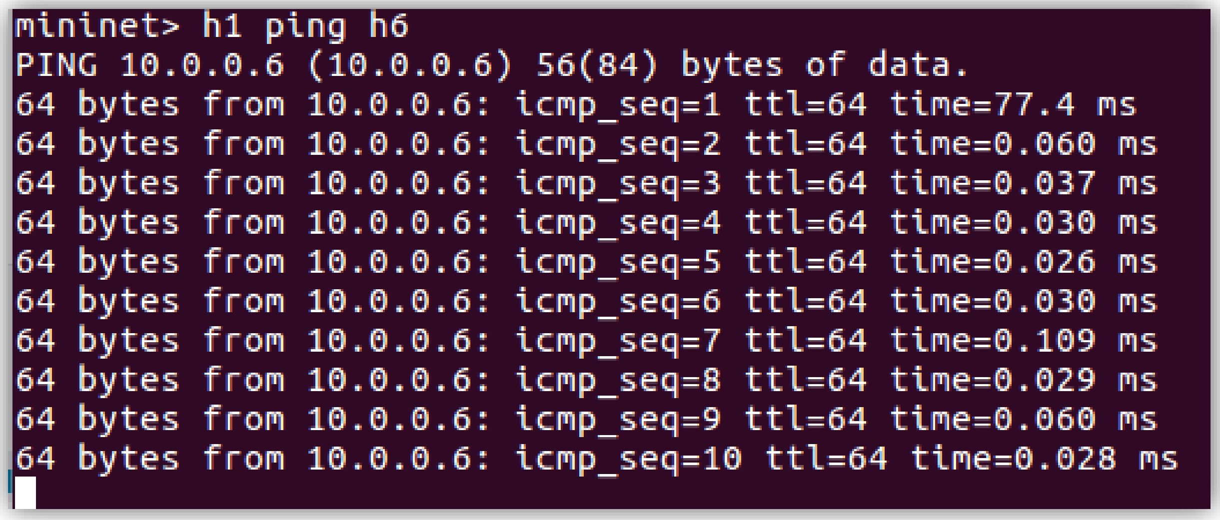
From the terminal used to startup the Mininet container we will input some commands to simulate live traffic from one host to another. There are far more robust methods of doing this, but for our purposes, using ping is just fine.

At the 'mininet>' prompt, input:

h1 ping h6



```
mininet>
```

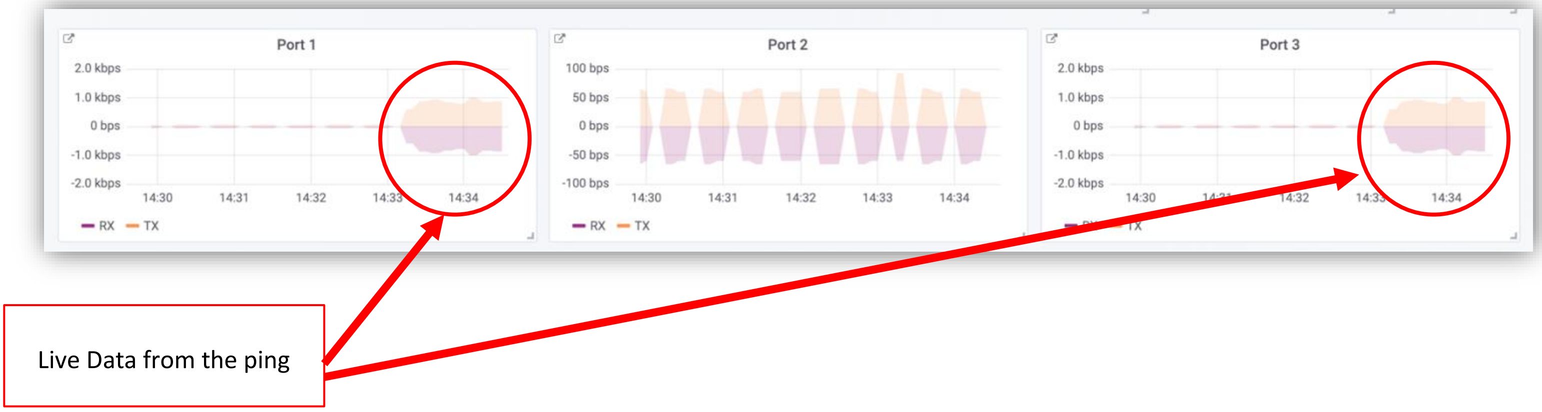


```
mininet> h1 ping h6
PING 10.0.0.6 (10.0.0.6) 56(84) bytes of data.
64 bytes from 10.0.0.6: icmp_seq=1 ttl=64 time=77.4 ms
64 bytes from 10.0.0.6: icmp_seq=2 ttl=64 time=0.060 ms
64 bytes from 10.0.0.6: icmp_seq=3 ttl=64 time=0.037 ms
64 bytes from 10.0.0.6: icmp_seq=4 ttl=64 time=0.030 ms
64 bytes from 10.0.0.6: icmp_seq=5 ttl=64 time=0.026 ms
64 bytes from 10.0.0.6: icmp_seq=6 ttl=64 time=0.030 ms
64 bytes from 10.0.0.6: icmp_seq=7 ttl=64 time=0.109 ms
64 bytes from 10.0.0.6: icmp_seq=8 ttl=64 time=0.029 ms
64 bytes from 10.0.0.6: icmp_seq=9 ttl=64 time=0.060 ms
64 bytes from 10.0.0.6: icmp_seq=10 ttl=64 time=0.028 ms
```

Then, after a about 10 seconds hit Ctrl+c to end the ping.

STEP 7: WATCH LIVE TRAFFIC

If we now return to Grafana and look at the different switch sections of the Main dashboard, we can see the live data travelling along the flows we set previously. This is shown due to higher data transfer rates on respective switchports.



Try to use the graphs to determine which Switch and Port Host 1 and Host 6 are on respectively.

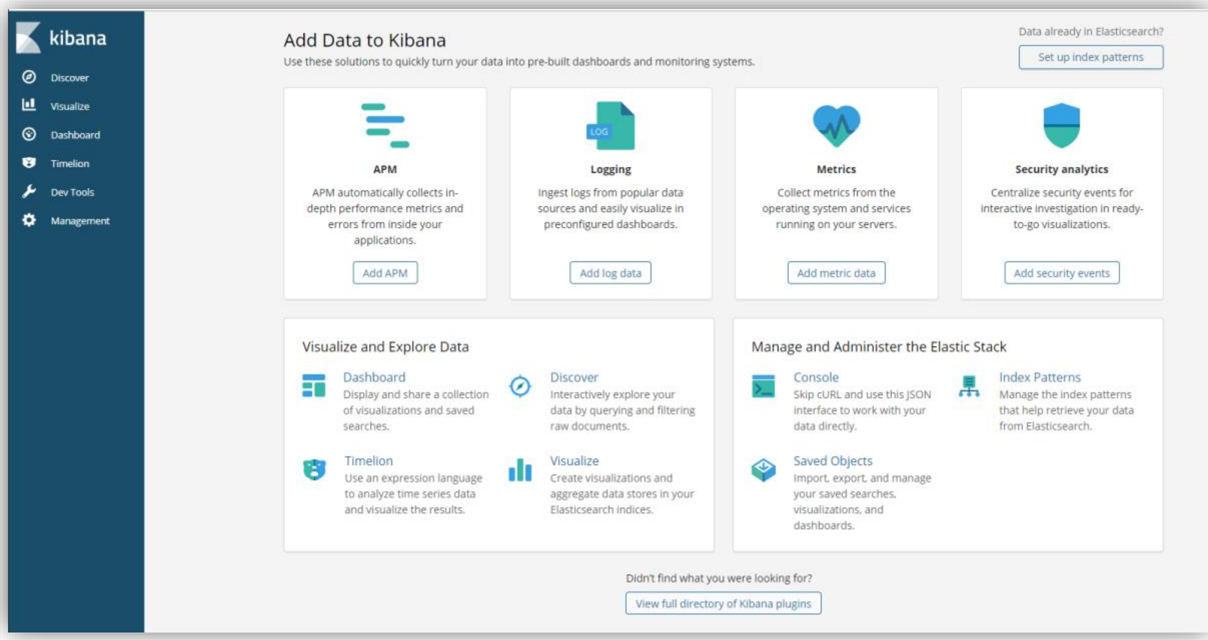
ADDITIONAL NOTES/TROUBLESHOOTING

If there is an issue present wherein there is no data travelling through the pipeline, it may be necessary to check the logs of all relevant Docker containers through Portainer (Telegraf, Kafka, Logstash, Elasticsearch, Grafana).

If data has been misinterpreted or has changed JSON headers, it may be necessary to check the data actually being ingested to Elasticsearch – this can be done using its visual counterpart Kibana.

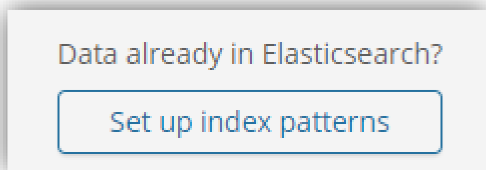
OPTIONAL: VERIFYING STORED DATA THROUGH KIBANA

Access the Kibana webpage at localhost:5601 in your VM's browser. You will be greeted with the following page:

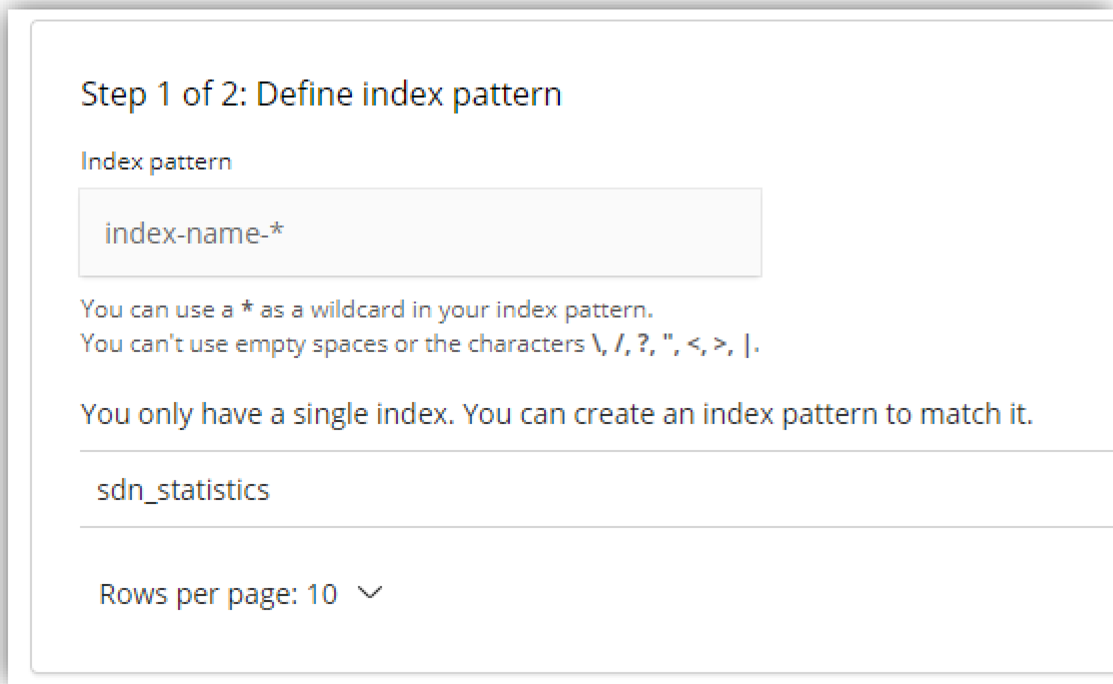


This is Kibana's landing page, from here you can access all these features available to Elasticsearch.

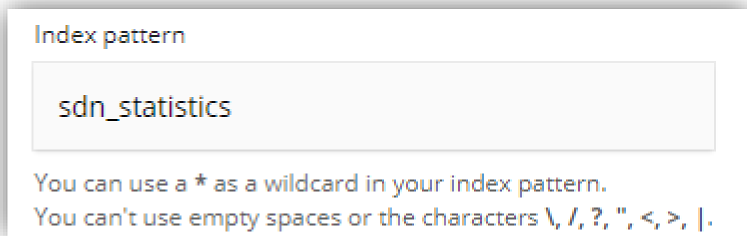
We want to check on the data being ingested and stored, so to do this, we need to define our index. Click the "Set up index patterns" button to continue.



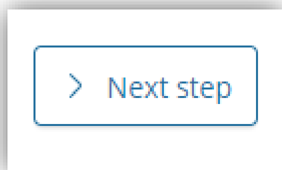
On the following page, you will need to define the index we are going search Elasticsearch for. A list of existing indexes are provided beneath the entry field.



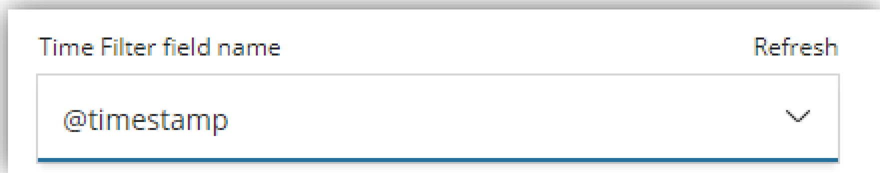
Type ‘sdn_statistics’ into the empty field. (This index is assigned to the data when it is processed by Logstash).



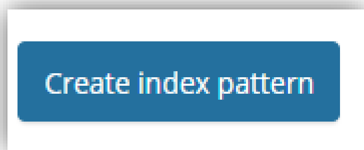
Click ‘Next step’.



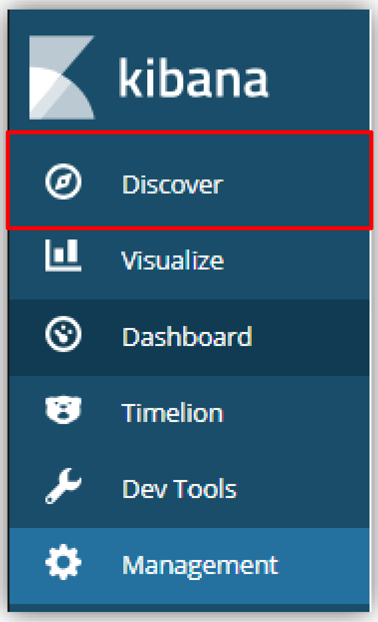
Choose a time field – for our data we use a field named '@timestamp'. (This is the time the data was processed by Logstash).



Click 'Create index pattern'.

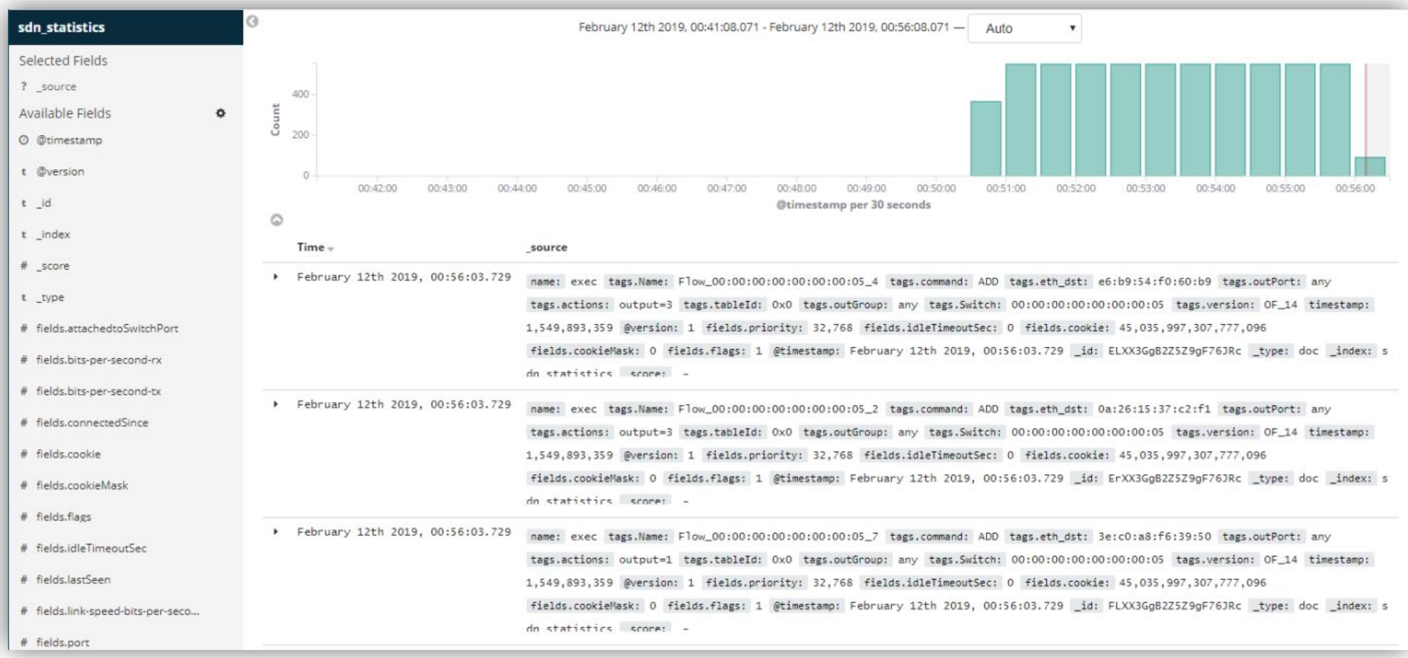


On the left navigation column, press the 'Discover' button.

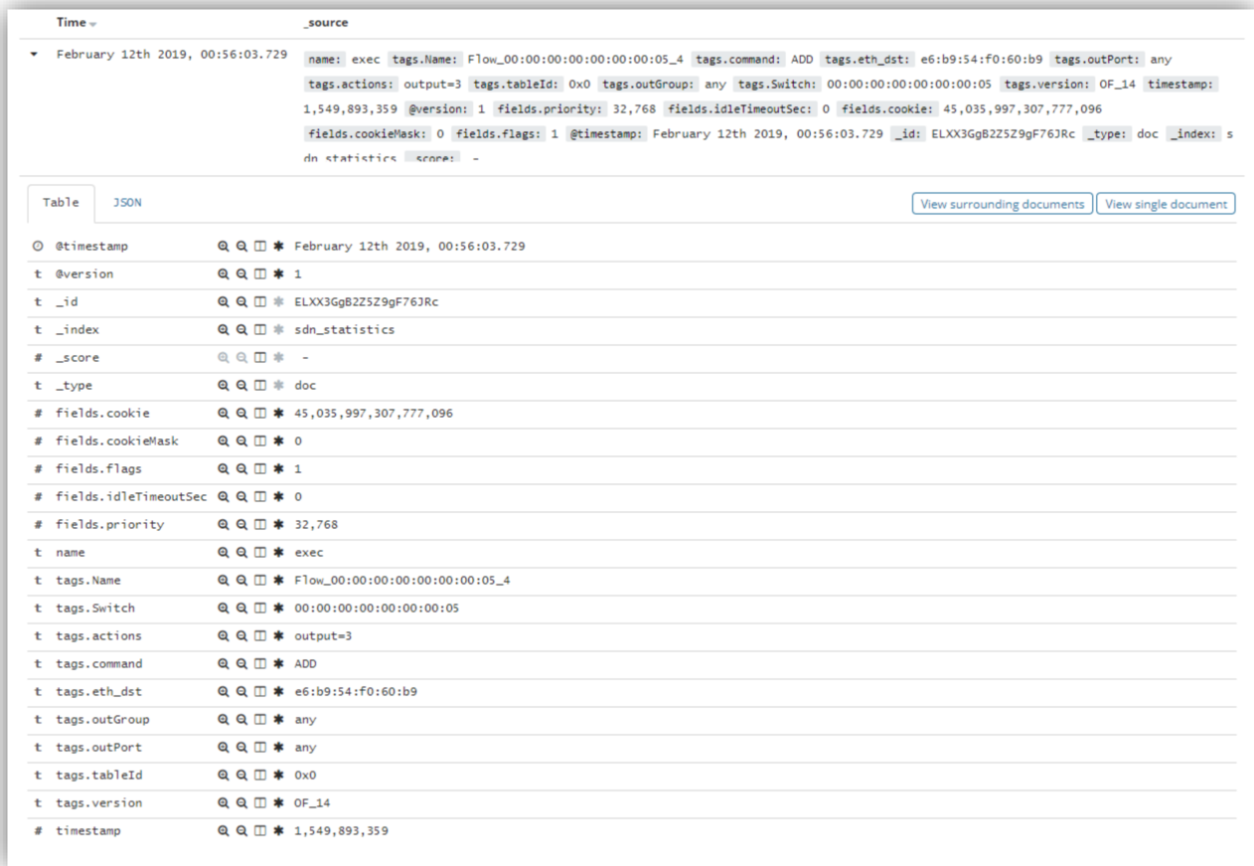


You will arrive at the below screen.

This screen offers many details on the live data currently being manipulated and stored in our ELK stack. The bar graph at the top illustrates the rate and amount of data stored over time, the titles to the left are the JSON headers, and the messages at the bottom represent individual JSON messages.



Take a deeper look at one of the recent messages to understand how the system is storing and then visualising data over time.



If all is well with the data storage system, this data should be present on this page. If not, there is an issue in Telegraf, Kafka or Logstash as they precede Elasticsearch in the pipeline. To investigate this, use the skills you have learnt during this laboratory to check the logs of these containerised applications and determine the problem.

CONCLUSION

You now have had exposure to a cloud-native application that runs a virtual network, virtual SDN controller and data analysis applications. Some things you were exposed to include:

- Virtualisation
- Container Management (Portainer/Docker)
- Specific Applications:
  - Mininet
  - Grafana
  - ELK Stack (Elasticsearch, Logstash and Kibana)
- Basic UNIX operations
- Python and Bash scripting
- Automatic Network Deployment
- SDN and general network management concepts

Feel free to research any of the above applications or terms for further information.

The code used in this Lab is available on GitHub. (https://github.com/Faux212/SDN_ADC_MM)