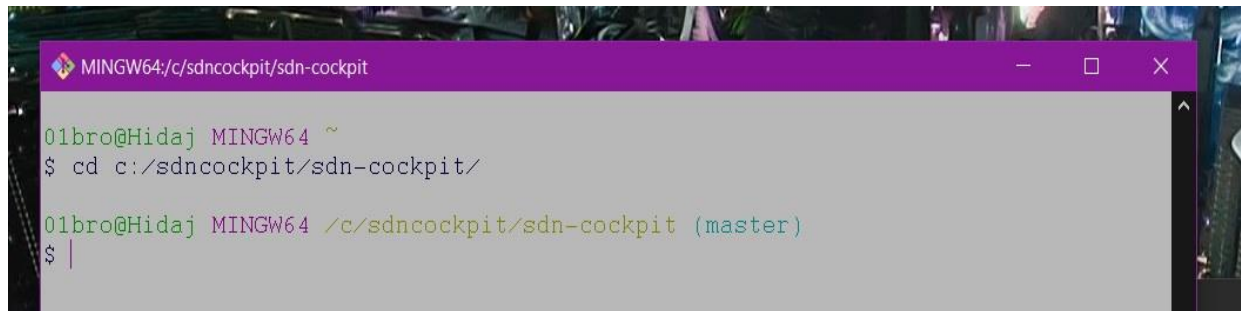


A Guide to Launching an SDN-Cockpit Application

This guide will assume the installation and setup of the environment as depicted in 1_SDN-CockpitInstallManual.pdf was followed.

Step 1: Open Git Bash and change the directory as depicted in figure 1.

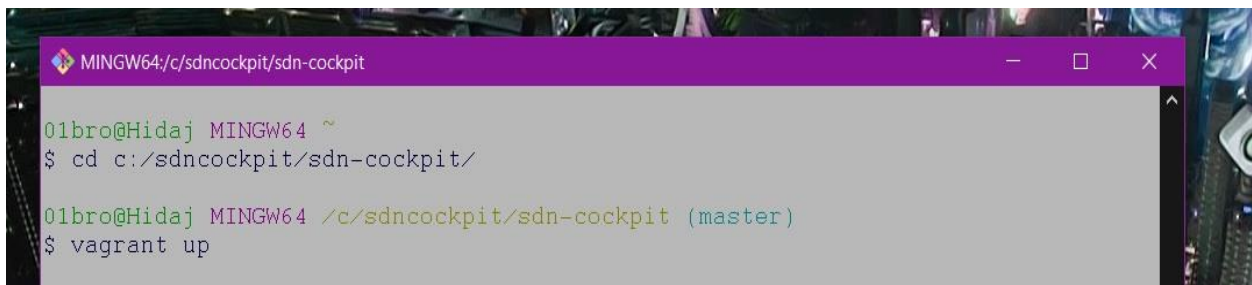


```
MINGW64/c/sdncockpit/sdn-cockpit
01bro@Hidaj MINGW64 ~
$ cd c:/sdncockpit/sdn-cockpit/

01bro@Hidaj MINGW64 /c/sdncockpit/sdn-cockpit (master)
$ |
```

Figure 1. Depicts git bash terminal with the directory change instigated.

Step 2: Utilise the command `vagrant up` to instantiate a virtual machine (vm).

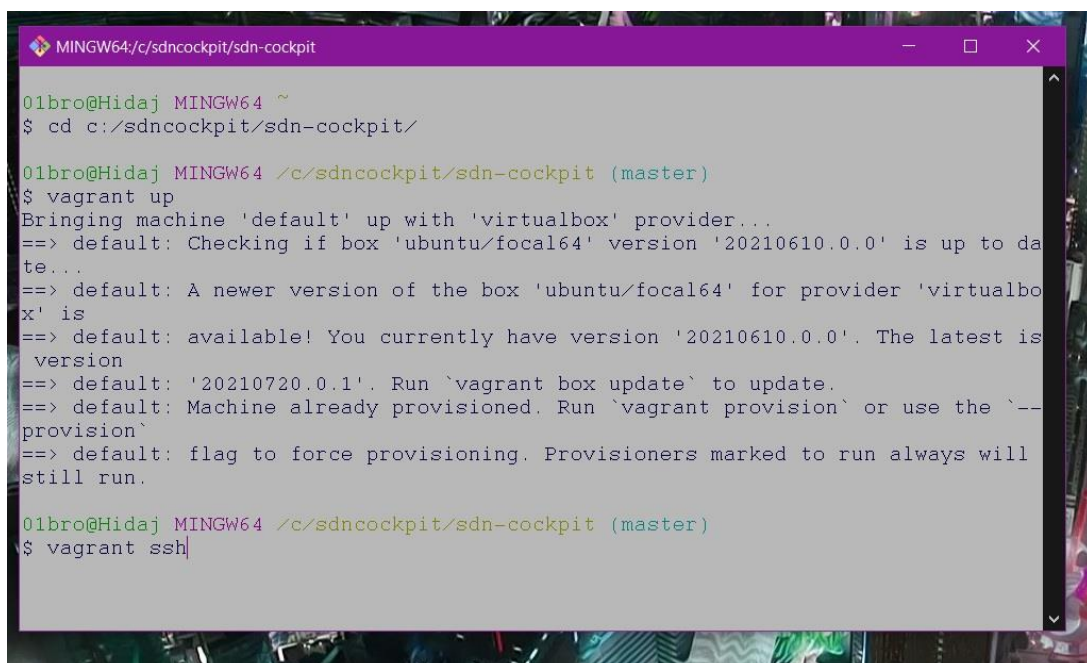


```
MINGW64/c/sdncockpit/sdn-cockpit
01bro@Hidaj MINGW64 ~
$ cd c:/sdncockpit/sdn-cockpit/

01bro@Hidaj MINGW64 /c/sdncockpit/sdn-cockpit (master)
$ vagrant up
```

Figure 2. Depicts the `vagrant up` command about to be run.

Step 3: When `vagrant up` has finished instantiating the virtual environment use the `vagrant ssh` command to connect to the vm.



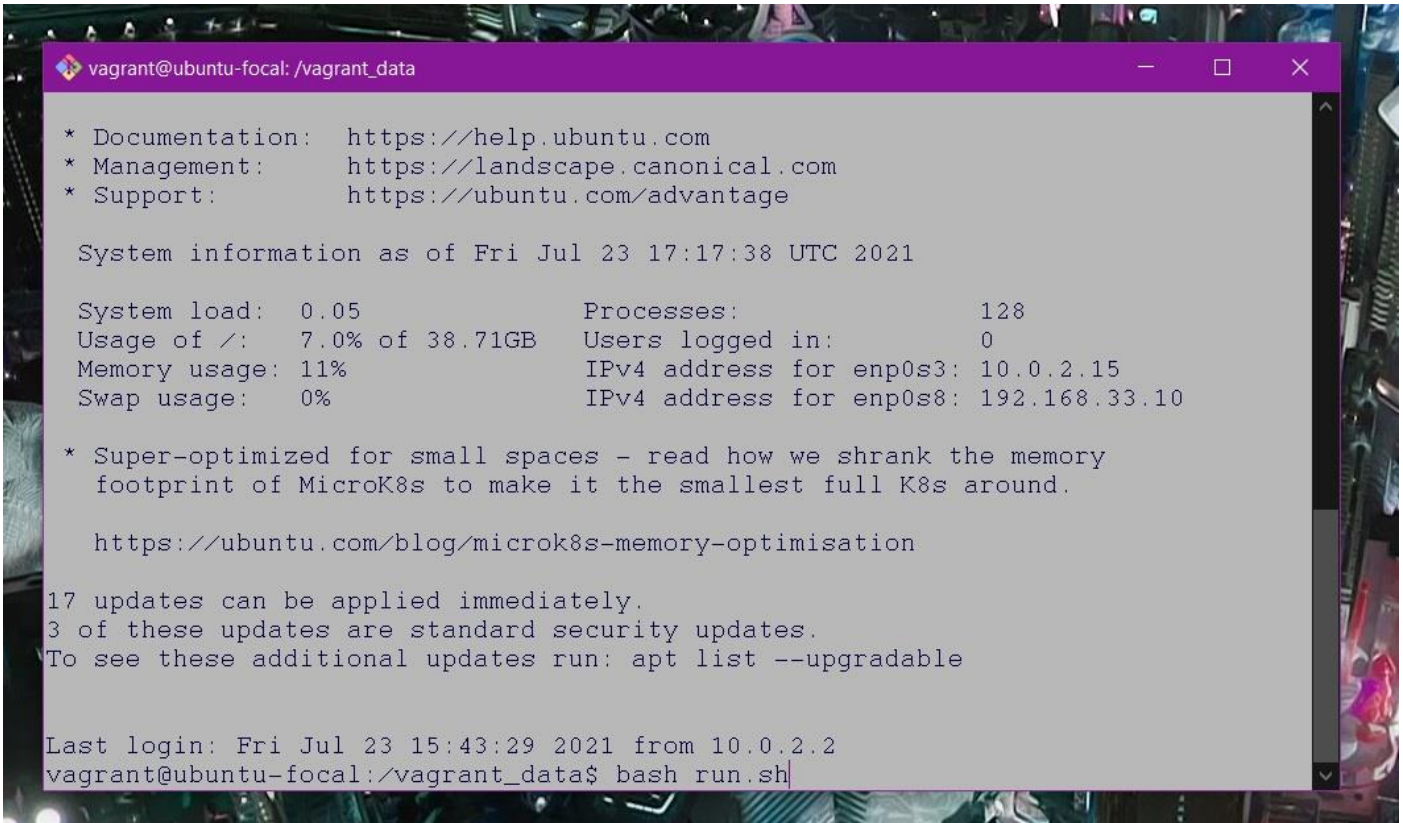
```
MINGW64/c/sdncockpit/sdn-cockpit
01bro@Hidaj MINGW64 ~
$ cd c:/sdncockpit/sdn-cockpit/

01bro@Hidaj MINGW64 /c/sdncockpit/sdn-cockpit (master)
$ vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Checking if box 'ubuntu/focal64' version '20210610.0.0' is up to date...
==> default: A newer version of the box 'ubuntu/focal64' for provider 'virtualbox' is
available! You currently have version '20210610.0.0'. The latest is
version
==> default: '20210720.0.1'. Run `vagrant box update` to update.
==> default: Machine already provisioned. Run `vagrant provision` or use the `--
provision`
==> default: flag to force provisioning. Provisioners marked to run always will
still run.

01bro@Hidaj MINGW64 /c/sdncockpit/sdn-cockpit (master)
$ vagrant ssh
```

Figure 3. Depicts completed vm instantiation and `vagrant ssh` command execution.

Step 4: Once a connection to the vm has been made use the command `bash run.sh` to launch the SDN-Cockpit environment.



```
vagrant@ubuntu-focal: /vagrant_data

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:        https://ubuntu.com/advantage

System information as of Fri Jul 23 17:17:38 UTC 2021

System load:  0.05          Processes:           128
Usage of /:   7.0% of 38.71GB Users logged in:       0
Memory usage: 11%          IPv4 address for enp0s3: 10.0.2.15
Swap usage:   0%           IPv4 address for enp0s8: 192.168.33.10

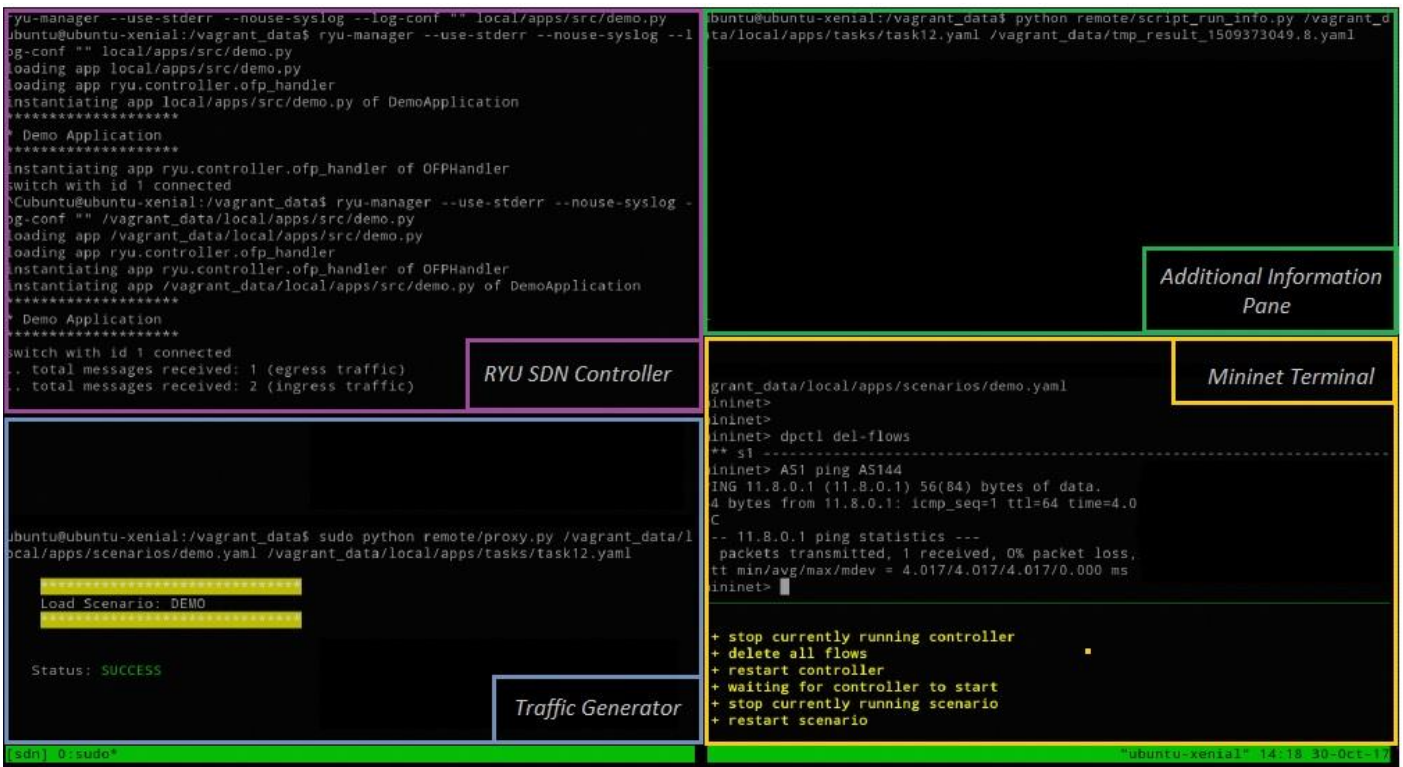
* Super-optimized for small spaces - read how we shrank the memory
  footprint of MicroK8s to make it the smallest full K8s around.

https://ubuntu.com/blog/microk8s-memory-optimisation

17 updates can be applied immediately.
3 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Last login: Fri Jul 23 15:43:29 2021 from 10.0.2.2
vagrant@ubuntu-focal: /vagrant_data$ bash run.sh
```

Figure 4. Depicts the completion of the `vagrant ssh` command and the command to launch SDN-Cockpit.



<pre>ryu-manager --use-stderr --no-use-syslog --log-conf "" local/apps/src/demo.py ubuntu@ubuntu-xenial:/vagrant_data\$ ryu-manager --use-stderr --no-use-syslog --l og-conf "" local/apps/src/demo.py loading app local/apps/src/demo.py loading app ryu.controller.ofp_handler instantiating app local/apps/src/demo.py of DemoApplication ***** * Demo Application ***** instantiating app ryu.controller.ofp_handler of OFPHandler switch with id 1 connected ubuntu@ubuntu-xenial:/vagrant_data\$ ryu-manager --use-stderr --no-use-syslog - log-conf "" /vagrant_data/local/apps/src/demo.py loading app /vagrant_data/local/apps/src/demo.py loading app ryu.controller.ofp_handler instantiating app ryu.controller.ofp_handler of OFPHandler instantiating app /vagrant_data/local/apps/src/demo.py of DemoApplication ***** * Demo Application ***** switch with id 1 connected .. total messages received: 1 (egress traffic) .. total messages received: 2 (ingress traffic)</pre> <p>RYU SDN Controller</p>	<pre>ubuntu@ubuntu-xenial:/vagrant_data\$ python remote/script_run_info.py /vagrant_d ta/local/apps/tasks/task12.yaml /vagrant_data/tmp_result_1509373049.8.yaml</pre>
<pre>ubuntu@ubuntu-xenial:/vagrant_data\$ sudo python remote/proxy.py /vagrant_data/l ocal/apps/scenarios/demo.yaml /vagrant_data/local/apps/tasks/task12.yaml Load Scenario: DEMO Status: SUCCESS</pre> <p>Traffic Generator</p>	<p>Mininet Terminal</p> <pre>grant_data/local/apps/scenarios/demo.yaml mininet> mininet> mininet> dpctl del-flows ** s1 mininet> A51 ping A5144 ING 11.8.0.1 (11.8.0.1) 56(84) bytes of data. 4 bytes from 11.8.0.1: icmp_seq=1 ttl=64 time=4.0 C -- 11.8.0.1 ping statistics --- packets transmitted, 1 received, 0% packet loss, tt min/avg/max/mdev = 4.017/4.017/4.017/0.000 ms mininet> + stop currently running controller + delete all flows + restart controller + waiting for controller to start + stop currently running scenario + restart scenario</pre>

Figure 4. Depicts the SDN-Cockpit environment and its constituent components have been highlighted. By default SDN Cockpit loads a demo configuration.

Step 5: Replace the Demo.py application in the RYU SDN controller pane with unconfigured.py to view the unmodified application, my solution is named configured.py.

```

vagrant@ubuntu-focal: /vagrant_data
loading app local/apps/src/configured.py
loading app ryu.controller.ofp_handler
instantiating app local/apps/src/configured.py of SecureGateway
instantiating app ryu.controller.ofp_handler of OFFHandler
switch with id 1 connected

/// Switch connected. ID: 1
/// [Switch 1]: PACKET-IN (#1) on port: 3
Cvagrant@ubuntu-focal:/vagrant_data$ python3 -m py_compile local/apps/src/configured.py
loading app local/apps/src/configured.py
loading app ryu.controller.ofp_handler
instantiating app local/apps/src/configured.py of SecureGateway
instantiating app ryu.controller.ofp_handler of OFFHandler
switch with id 1 connected

/// Switch connected. ID: 1
/// [Switch 1]: PACKET-IN (#1) on port: 4
/// [Switch 1]: PACKET-IN (#2) on port: 3
/// [Switch 1]: PACKET-IN (#3) on port: 1
/// [Switch 1]: PACKET-IN (#4) on port: 2

vagrant@ubuntu-focal:/vagrant_data$ sudo python3 remote/proxy.py /vagrant_data/local/apps/scenarios/demo.yaml

***** ERROR *****
Scenario file not found: /vagrant_data/local/apps/scenarios/demo.yaml
Make sure to specify an existing scenario!

CTraceback (most recent call last):
  File "remote/proxy.py", line 786, in <module>
    time.sleep(1)
KeyboardInterrupt

vagrant@ubuntu-focal:/vagrant_data$ sudo python3 remote/proxy.py /vagrant_data/local/apps/scenarios/firewall.yaml

Task: Secure Medical IOT Gateway Topology:
Scenario: firewall
Description:
Implement the following rules:
1. Patient Network (L1) may exchange packets with Doctor Network (H1) 2. Patient Records (R1) may exchange packets with Doctor Network (H1) 3. Noone from these three networks (L1,R1,H1) may send packets to Public Network Access (M1). 4. Patient Network (L1) and Patient Records (R1) may not exchange packets.
Result: SUCCESS

***** ERROR *****
Scenario file /vagrant_data/local/apps/scenarios/demo.yaml invalid (could not be parsed): [Errno 2] No such file or directory: '/vagrant_data/local/apps/scenarios/demo.yaml'

vagrant@ubuntu-focal:/vagrant_data$ sudo python3 remote/script_run_mininet.py /vagrant_data/local/apps/scenarios/firewall.yaml
Unable to contact the remote controller at 127.0.0.1:6633
mininet> exit
vagrant@ubuntu-focal:/vagrant_data$ sudo python3 remote/script_run_mininet.py /vagrant_data/local/apps/scenarios/firewall.yaml
Unable to contact the remote controller at 127.0.0.1:6633
mininet>
vagrant@ubuntu-focal:/vagrant_data$ python3 remote/script_run_watch.py

```

Figure 5. Depicts the configured firewall application being loaded into the controller.

Step 6: Load the firewall.yaml file from the tasks folder in the additional information pane by replacing the None with /vagrant_data/local/apps/tasks/firewall.yaml.

```

vagrant@ubuntu-focal: /vagrant_data
File "/usr/lib/python3.8/py_compile.py", line 215, in <module>
    sys.exit(main())
File "/usr/lib/python3.8/py_compile.py", line 207, in main
    compile(filename, doraise=True)
File "/usr/lib/python3.8/py_compile.py", line 142, in compile
    source_bytes = loader.get_data(file)
File "<frozen importlib._bootstrap_external>", line 1037, in get_data
FileNotFoundError: [Errno 2] No such file or directory: '/local/apps/src/demo.py'

vagrant@ubuntu-focal:/vagrant_data$ ^C
vagrant@ubuntu-focal:/vagrant_data$ python3 -m py_compile local/apps/src/configured.py && ryu-manager --use-stderr --nouse-syslog --log-conf "" local/apps/src/configured.py
loading app local/apps/src/configured.py
loading app ryu.controller.ofp_handler
instantiating app local/apps/src/configured.py of SecureGateway
instantiating app ryu.controller.ofp_handler of OFFHandler
switch with id 1 connected

/// Switch connected. ID: 1
/// [Switch 1]: PACKET-IN (#1) on port: 3
Cvagrant@ubuntu-focal:/vagrant_data$ python3 -m py_compile local/apps/src/configured.py
loading app local/apps/src/configured.py
loading app ryu.controller.ofp_handler
instantiating app local/apps/src/configured.py of SecureGateway
instantiating app ryu.controller.ofp_handler of OFFHandler
switch with id 1 connected

/// Switch connected. ID: 1
/// [Switch 1]: PACKET-IN (#1) on port: 3
Cvagrant@ubuntu-focal:/vagrant_data$ python3 -m py_compile local/apps/src/configured.py
loading app local/apps/src/configured.py
loading app ryu.controller.ofp_handler
instantiating app local/apps/src/configured.py of SecureGateway
instantiating app ryu.controller.ofp_handler of OFFHandler
switch with id 1 connected

/// Switch connected. ID: 1
/// [Switch 1]: PACKET-IN (#1) on port: 4
/// [Switch 1]: PACKET-IN (#2) on port: 3
/// [Switch 1]: PACKET-IN (#3) on port: 1
/// [Switch 1]: PACKET-IN (#4) on port: 2

vagrant@ubuntu-focal:/vagrant_data$ sudo python3 remote/proxy.py /vagrant_data/local/apps/scenarios/demo.yaml

***** ERROR *****
Scenario file not found: /vagrant_data/local/apps/scenarios/demo.yaml
Make sure to specify an existing scenario!

CTraceback (most recent call last):
  File "remote/proxy.py", line 786, in <module>
    time.sleep(1)
KeyboardInterrupt

vagrant@ubuntu-focal:/vagrant_data$ sudo python3 remote/proxy.py /vagrant_data/local/apps/scenarios/firewall.yaml

vagrant@ubuntu-focal:/vagrant_data$ python3 remote/script_run_info.py BootDummY

***** ERROR *****
There is currently no task selected

Select an application (or create a new one) and add the
[tm task=taskname] makro to start task. Please contact your
supervisor if there are any problems!

CTraceback (most recent call last):
  File "remote/script_run_info.py", line 54, in <module>
    time.sleep(10) # don't return
KeyboardInterrupt

vagrant@ubuntu-focal:/vagrant_data$ python3 remote/script_run_info.py /vagrant_data/local/apps/tasks/firewall.yaml /vagrant_data/tmp_result_1627058395.8042104.yaml

vagrant@ubuntu-focal:/vagrant_data$ sudo python3 remote/script_run_mininet.py /vagrant_data/local/apps/scenarios/demo.yaml

***** ERROR *****
Scenario file /vagrant_data/local/apps/scenarios/demo.yaml invalid (could not be parsed): [Errno 2] No such file or directory: '/vagrant_data/local/apps/scenarios/demo.yaml'

vagrant@ubuntu-focal:/vagrant_data$ sudo python3 remote/script_run_mininet.py /vagrant_data/local/apps/scenarios/firewall.yaml
Unable to contact the remote controller at 127.0.0.1:6633
mininet> exit
vagrant@ubuntu-focal:/vagrant_data$ sudo python3 remote/script_run_mininet.py /vagrant_data/local/apps/scenarios/firewall.yaml
Unable to contact the remote controller at 127.0.0.1:6633
mininet>
vagrant@ubuntu-focal:/vagrant_data$ python3 remote/script_run_watch.py

```

Figure 6. The firewall task file being loaded into the additional information pane.

Hint: Sometimes the `vagrant_data/tmp_result_1627058395.80421.yaml` does not sync correctly. If so change the temporary file in the pane to point to the temporary result file on your system.

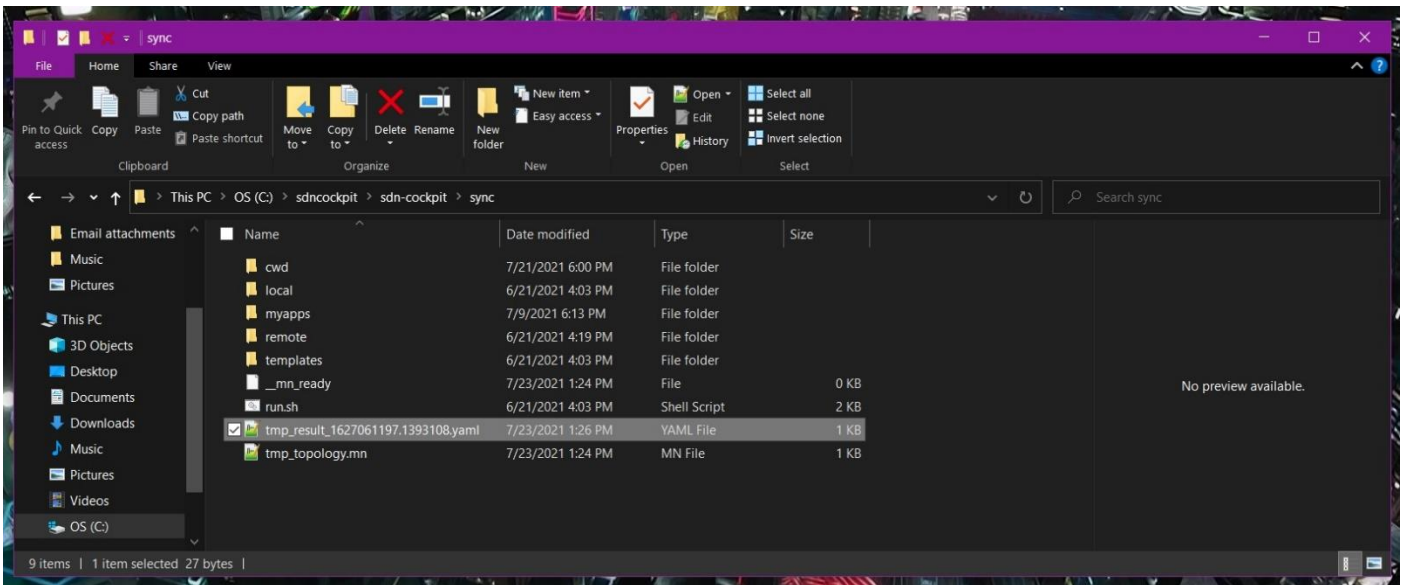


Figure 7. Shows the location of the temporary results file, if there is an incompatibility change the location in the additional information pane.

Step 7: Enter the command `exit` in the mininet pane then replace the `demo.yaml` file with `firewall.yaml` from the scenario folder to configure the network topology depicted in the additional information pane.

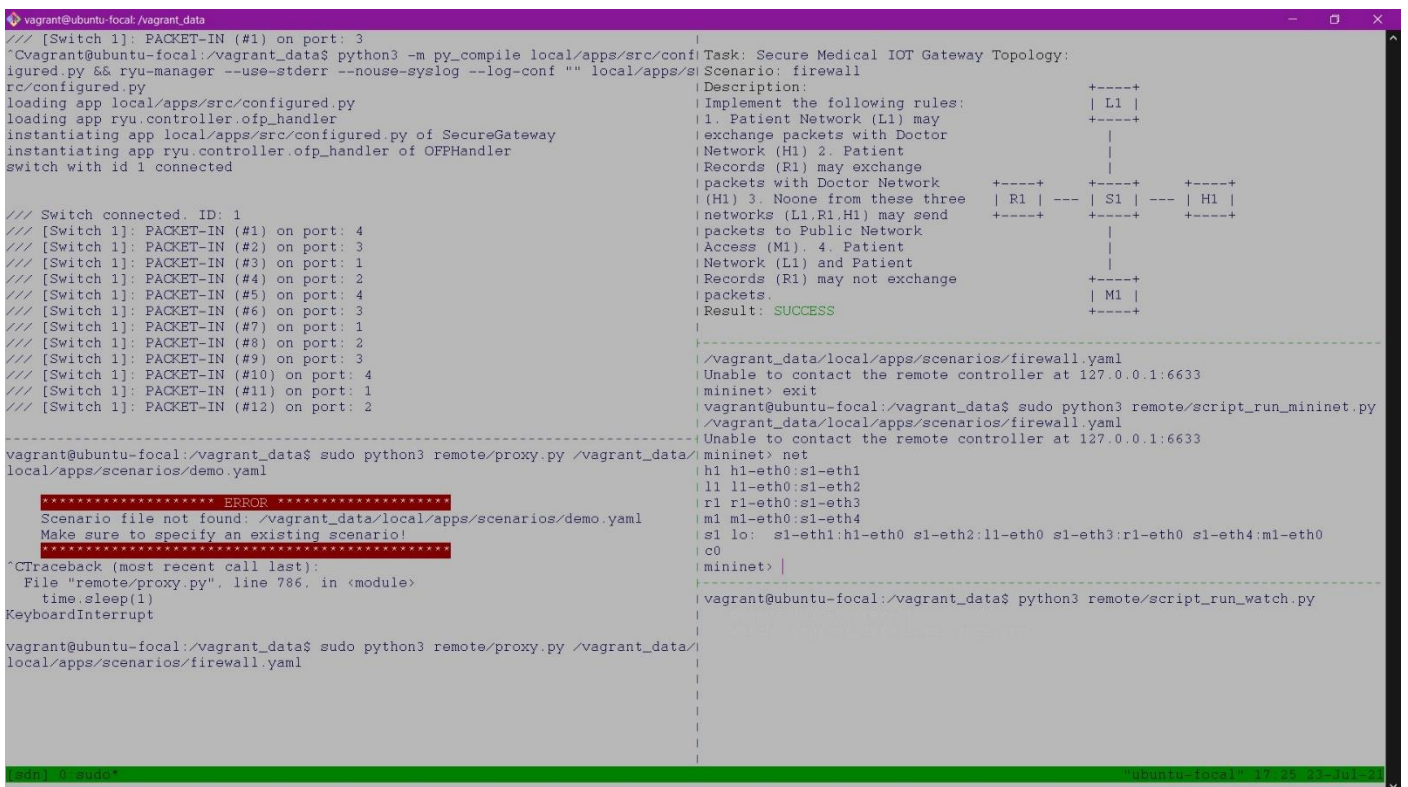


Figure 8. Depicts the successfully loaded task and network topology in the mininet pane.

Step 8: In the traffic generator pane replace the scenarios/demo.yaml file with the scenarios/firewall.yaml file to generate traffic in the configured network topology.

```

vagrant@ubuntu-focal: /vagrant_data$
Task: Secure Medical IOT Gateway Topology:
Scenario: firewall
Description:
Implement the following rules:
1. Patient Network (L1) may
exchange packets with Doctor
Network (H1) 2. Patient
Records (R1) may exchange
packets with Doctor Network
(H1) 3. Noone from these three
networks (L1,R1,H1) may send
packets to Public Network
Access (M1). 4. Patient
Network (L1) and Patient
Records (R1) may not exchange
packets.
Result: SUCCESS
+-----+ +-----+ +-----+
| L1 |
+-----+
| R1 | --- | S1 | --- | H1 |
+-----+ +-----+ +-----+
| M1 |
+-----+

/vagrant_data/local/apps/scenarios/firewall.yaml
Unable to contact the remote controller at 127.0.0.1:6633
mininet: exit
vagrant@ubuntu-focal: /vagrant_data$ sudo python3 remote/script_run_mininet.py
/vagrant_data/local/apps/scenarios/firewall.yaml
Unable to contact the remote controller at 127.0.0.1:6633
mininet: net
h1 h1-eth0:s1-eth1
l1 l1-eth0:s1-eth2
r1 r1-eth0:s1-eth3
m1 m1-eth0:s1-eth4
s1 lo: s1-eth1:h1-eth0 s1-eth2:l1-eth0 s1-eth3:r1-eth0 s1-eth4:m1-eth0
c0
mininet)
vagrant@ubuntu-focal: /vagrant_data$ python3 remote/script_run_watch.py

***** ERROR *****
Scenario file not found: /vagrant_data/local/apps/scenarios/demo.yaml
Make sure to specify an existing scenario!
*****
CTraceback (most recent call last):
File "remote/proxy.py", line 786, in <module>
time.sleep(1)
KeyboardInterrupt

vagrant@ubuntu-focal: /vagrant_data$ sudo python3 remote/proxy.py /vagrant_data/
local/apps/scenarios/firewall.yaml

*****
Load Scenario: FIREWALL
*****

```

Figure 9. Depicts the firewall scenario traffic generator loaded to test the functionality of the configured controller.

Step 9: The traffic generated will then be filtered, if successful it will be indicated in the additional information panes result field and traffic generator pane status field as a ‘SUCCESS’ otherwise the result will be listed as a ‘FAILURE’ in the additional information pane and the traffic generator will describe the number of packets expected through the switch to each network.

```

vagrant@ubuntu-focal: /vagrant_data$
Task: Secure Medical IOT Gateway Topology:
Scenario: firewall
Description:
Implement the following rules:
1. Patient Network (L1) may
exchange packets with Doctor
Network (H1) 2. Patient
Records (R1) may exchange
packets with Doctor Network
(H1) 3. Noone from these three
networks (L1,R1,H1) may send
packets to Public Network
Access (M1). 4. Patient
Network (L1) and Patient
Records (R1) may not exchange
packets.
Result: SUCCESS
+-----+ +-----+ +-----+
| L1 |
+-----+
| R1 | --- | S1 | --- | H1 |
+-----+ +-----+ +-----+
| M1 |
+-----+

/vagrant_data/local/apps/scenarios/firewall.yaml
Unable to contact the remote controller at 127.0.0.1:6633
mininet: exit
vagrant@ubuntu-focal: /vagrant_data$ sudo python3 remote/script_run_mininet.py
/vagrant_data/local/apps/scenarios/firewall.yaml
Unable to contact the remote controller at 127.0.0.1:6633
mininet: net
h1 h1-eth0:s1-eth1
l1 l1-eth0:s1-eth2
r1 r1-eth0:s1-eth3
m1 m1-eth0:s1-eth4
s1 lo: s1-eth1:h1-eth0 s1-eth2:l1-eth0 s1-eth3:r1-eth0 s1-eth4:m1-eth0
c0
mininet)
vagrant@ubuntu-focal: /vagrant_data$ python3 remote/script_run_watch.py

*****
Load Scenario: FIREWALL
*****

Status: SUCCESS

```

Figure 10. Depicts the configured firewall application correctly routing packages to their respective networks.

Step 10: To exit the SDN Cockpit environment input ‘ctrl+b d’ and vagrant halt to close the connection to the vm.