SDN Fundamentals & Techniques

Chapter 5 - Demo 1 - Using ONOS RESTful API interface to manage hosts, devices, applications, and settings

Jing Yan (yanj3, jing.yan@aalto.fi) Mar 9th, 2020

1 Task 1

Unlike other demonstrations, in this particular demo, the ONOS applications are not yet activated and you are not allowed to start them manually. Thus, as an initial task, you are asked to start the required ONOS applications using a python-based approach. In your solution, you may use the default ONOS credentials, i.e., username: onos, password: rocks, to access ONOS RESTful API. The list of applications to be started can be found in the following file “list_applications” at http://www.mosaic-lab.org/SDN_Demos.aspx under “Chapter 5 Demos- ONOS-based orchestration system Demo1”

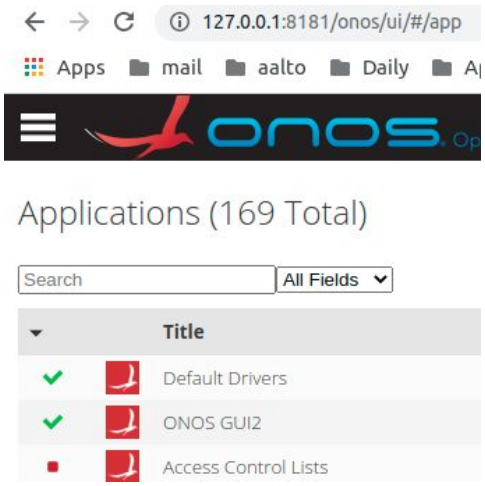
Code

```
#!/usr/bin/env python3
import requests, json
from requests.auth import HTTPBasicAuth

if __name__ == "__main__":
   # Activate "Host Location Provider", "Host Mobility", "LLDP Link Provider",
"OpenFlow Agent", "OpenFlow Base Provider" Application, "OpenFlow Provider Suite",
"Optical Application", "Proxy ARP/NDP", "Reactive Forwarding"
   app_list = ["org.onosproject.hostprovider", "org.onosproject.mobility",
"org.onosproject.lldpprovider", "org.onosproject.ofagent",
"org.onosproject.openflow-base", "org.onosproject.openflow",
"org.onosproject.roadm", "org.onosproject.proxyarp", "org.onosproject.fwd"]
   for app in app_list:
       res = requests.post("http://100.109.0.1:8181/onos/v1/applications/%s/active"
% app, auth = HTTPBasicAuth('onos', 'rocks'))
```

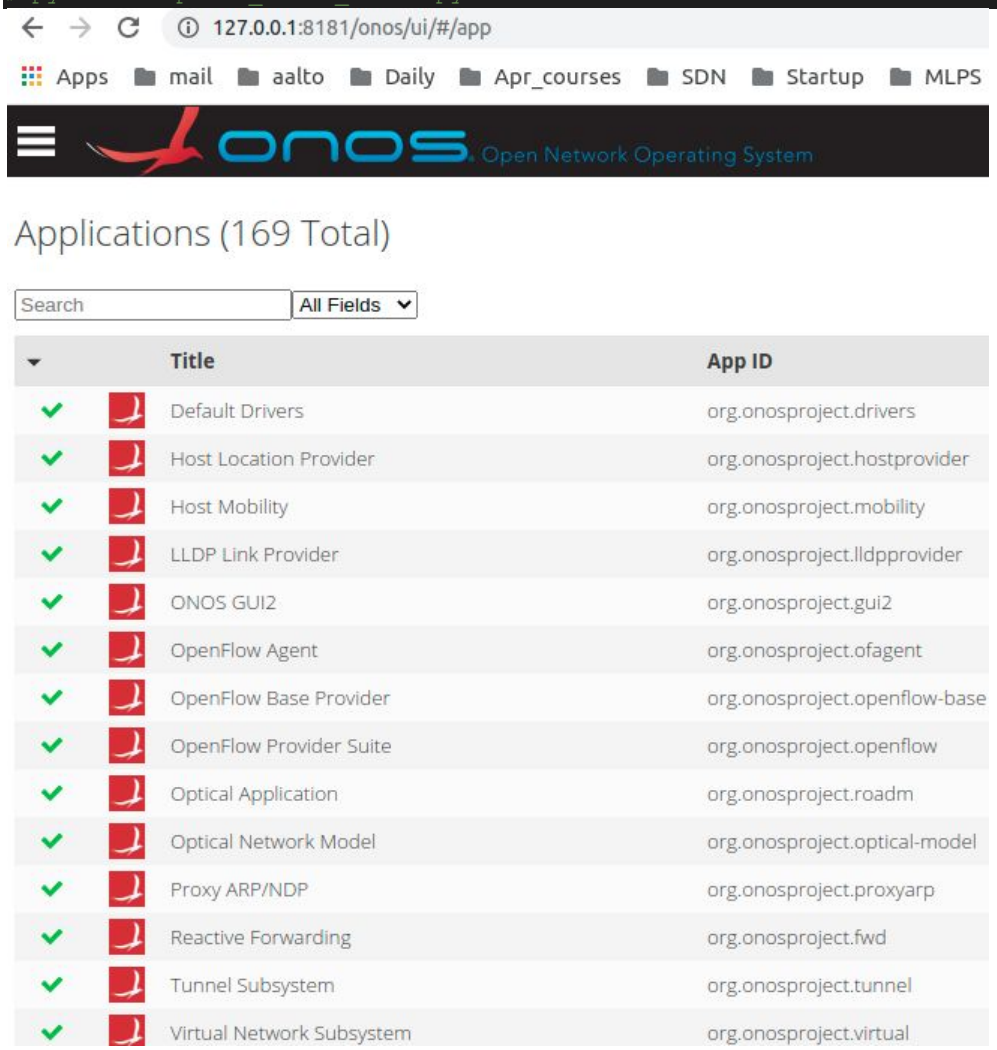
Result

- Before running the python script



- After running the python script

```
# python chapter5_demo1_task1.py
```



2 Task 2

The following task is provided with a pre-defined topology, i.e., use script file “task.sh” at http://www.mosaic-lab.org/SDN_Demos.aspx under “Chapter 5 Demos- ONOS-based orchestration system Demo1”, in which students are asked to:

2.1 Create a python program to list all available devices by their IDs

Code

```
#!/usr/bin/env python3
import requests, json
from requests.auth import HTTPBasicAuth

if __name__ == "__main__":
   # Lists all infrastructure devices
   res1 = requests.get("http://100.109.0.1:8181/onos/v1/devices", auth =
HTTPBasicAuth('onos', 'rocks'))
   devices_list = res1.json()["devices"]
   for index, value in enumerate(devices_list):
       # Lists details of a specific infrastructure device based on deviceid
       res2 = requests.get("http://100.109.0.1:8181/onos/v1/devices/%s" %
value["id"], auth = HTTPBasicAuth('onos', 'rocks'))
       print("-----Present the info for device with id is %s as follows: %s" %
(value["id"], res2.json()))
```

Result

```
# python chapter5_demo1_task2_2.1.py
-----Present the info for device with id is of:00006a5ff8be3c41 as follows: {'id':
'of:00006a5ff8be3c41', 'type': 'SWITCH', 'available': True, 'role': 'MASTER',
'mfr': 'Nicira, Inc.', 'hw': 'Open vSwitch', 'sw': '2.9.8', 'serial': 'None',
'driver': 'ovs', 'chassisId': '6a5ff8be3c41', 'lastUpdate': '1615381487217',
'humanReadableLastUpdate': 'connected 8m37s ago', 'annotations': {'channelId':
'100.109.0.2:54316', 'managementAddress': '100.109.0.2', 'protocol': 'OF_13'}}
-----Present the info for device with id is of:0000069a5175a24d as follows: {'id':
'of:0000069a5175a24d', 'type': 'SWITCH', 'available': True, 'role': 'MASTER',
'mfr': 'Nicira, Inc.', 'hw': 'Open vSwitch', 'sw': '2.9.8', 'serial': 'None',
'driver': 'ovs', 'chassisId': '69a5175a24d', 'lastUpdate': '1615381487204',
'humanReadableLastUpdate': 'connected 8m37s ago', 'annotations': {'channelId':
'100.109.0.2:54314', 'managementAddress': '100.109.0.2', 'protocol': 'OF_13'}}
-----Present the info for device with id is of:000032c182e2cc4d as follows: {'id':
'of:000032c182e2cc4d', 'type': 'SWITCH', 'available': True, 'role': 'MASTER',
'mfr': 'Nicira, Inc.', 'hw': 'Open vSwitch', 'sw': '2.9.8', 'serial': 'None',
'driver': 'ovs', 'chassisId': '32c182e2cc4d', 'lastUpdate': '1615381487205',
'humanReadableLastUpdate': 'connected 8m37s ago', 'annotations': {'channelId':
'100.109.0.2:54312', 'managementAddress': '100.109.0.2', 'protocol': 'OF_13'}}
-----Present the info for device with id is of:000056f927a4cd4c as follows: {'id':
'of:000056f927a4cd4c', 'type': 'SWITCH', 'available': True, 'role': 'MASTER',
'mfr': 'Nicira, Inc.', 'hw': 'Open vSwitch', 'sw': '2.9.8', 'serial': 'None',
'driver': 'ovs', 'chassisId': '56f927a4cd4c', 'lastUpdate': '1615381487205',
'humanReadableLastUpdate': 'connected 8m37s ago', 'annotations': {'channelId':
'100.109.0.2:54310', 'managementAddress': '100.109.0.2', 'protocol': 'OF_13'}}
-----Present the info for device with id is of:00000a32409edf45 as follows: {'id':
'of:00000a32409edf45', 'type': 'SWITCH', 'available': True, 'role': 'MASTER',
'mfr': 'Nicira, Inc.', 'hw': 'Open vSwitch', 'sw': '2.9.8', 'serial': 'None',
'driver': 'ovs', 'chassisId': 'a32409edf45', 'lastUpdate': '1615381487204',
'humanReadableLastUpdate': 'connected 8m37s ago', 'annotations': {'channelId':
'100.109.0.2:54308', 'managementAddress': '100.109.0.2', 'protocol': 'OF_13'}}
```

2.2 Create a python program to get the IP management address and the OpenFlow version used by a given device in the pre-defined architecture

Code

```
#!/usr/bin/env python3
import requests, json
from requests.auth import HTTPBasicAuth

if __name__ == "__main__":
   # Lists all infrastructure devices
   res1 = requests.get("http://100.109.0.1:8181/onos/v1/devices", auth =
HTTPBasicAuth('onos', 'rocks'))
   devices_list = res1.json()["devices"]
   for index, value in enumerate(devices_list):
       # Lists details of a specific infrastructure device based on deviceid
       res2 = requests.get("http://100.109.0.1:8181/onos/v1/devices/%s" %
value["id"], auth = HTTPBasicAuth('onos', 'rocks'))
       print("-----Present the ip:port info for device with id is %s as follows:
%s" % (value["id"], res2.json()["annotations"]["channelId"]))
       print("-----Present the openflow version info for device with id is %s as
follows: %s" % (value["id"], res2.json()["annotations"]["protocol"]))
```

Result

```
# python chapter5_demo1_task2_2.2.py
-----Present the ip:port info for device with id is of:00006a5ff8be3c41 as follows:
100.109.0.2:54316
-----Present the openflow version info for device with id is of:00006a5ff8be3c41 as
follows: OF_13
-----Present the ip:port info for device with id is of:0000069a5175a24d as follows:
100.109.0.2:54314
-----Present the openflow version info for device with id is of:0000069a5175a24d as
follows: OF_13
-----Present the ip:port info for device with id is of:000032c182e2cc4d as follows:
100.109.0.2:54312
-----Present the openflow version info for device with id is of:000032c182e2cc4d as
follows: OF_13
-----Present the ip:port info for device with id is of:000056f927a4cd4c as follows:
100.109.0.2:54310
-----Present the openflow version info for device with id is of:000056f927a4cd4c as
follows: OF_13
-----Present the ip:port info for device with id is of:00000a32409edf45 as follows:
100.109.0.2:54308
-----Present the openflow version info for device with id is of:00000a32409edf45 as
follows: OF_13
```

2.3 Create a python program using the same device id, i.e., used in the previous question, to get the currently active MAC addresses and the Port names

I did not see any MAC info or Port info for the OVS devices.

3 Task 3

Considering the same pre-defined topology used in Task 1, students are asked to:

3.1 Create a python program to list all available hosts by their id, MAC address, and IP address

Code

```
#!/usr/bin/env python3
import requests, json
from requests.auth import HTTPBasicAuth
if __name__ == "__main__":
   # Lists all hosts
   res1 = requests.get("http://100.109.0.1:8181/onos/v1/hosts", auth =
HTTPBasicAuth('onos', 'rocks'))
   hosts_list = res1.json()["hosts"]
   for index, value in enumerate(hosts_list):
       # Lists details of the hosts based on id
       res2 = requests.get("http://100.109.0.1:8181/onos/v1/hosts/%s" %
value["id"], auth = HTTPBasicAuth('onos', 'rocks'))
       print("The %s device ip is %s and mac is %s" % (res2.json()["id"],
res2.json()["ipAddresses"][0], res2.json()["mac"]))
```

Result

```
# python chapter5_demo1_task3_3.1.py
The 72:69:FD:5B:24:0C/None device ip is 10.0.0.129 and mac is 72:69:FD:5B:24:0C
The 2A:7C:E9:34:50:5E/None device ip is 10.0.0.131 and mac is 2A:7C:E9:34:50:5E
The 16:61:35:9C:65:D5/None device ip is 10.0.0.133 and mac is 16:61:35:9C:65:D5
The AE:D8:D6:FF:CB:21/None device ip is 10.0.0.130 and mac is AE:D8:D6:FF:CB:21
The 7A:D1:53:C6:52:EE/None device ip is 10.0.0.2 and mac is 7A:D1:53:C6:52:EE
The 2E:C0:69:82:FF:22/None device ip is 10.0.0.4 and mac is 2E:C0:69:82:FF:22
The 96:35:A8:8B:08:14/None device ip is 10.0.0.132 and mac is 96:35:A8:8B:08:14
The 96:42:A5:37:22:E6/None device ip is 10.0.0.6 and mac is 96:42:A5:37:22:E6
The 46:D9:26:EE:5C:8D/None device ip is 10.0.0.3 and mac is 46:D9:26:EE:5C:8D
The E6:97:E8:7B:E3:F6/None device ip is 10.0.0.5 and mac is E6:97:E8:7B:E3:F6
```

3.2 Create a python program to get the device id and the port used by the host having “10.0.0.130" as an IP address in the pre-defined architecture

Code

```
#!/usr/bin/env python3
import requests, json
from requests.auth import HTTPBasicAuth
if __name__ == "__main__":
   # Lists all hosts
   res1 = requests.get("http://100.109.0.1:8181/onos/v1/hosts", auth =
HTTPBasicAuth('onos', 'rocks'))
   hosts_list = res1.json()["hosts"]
   for index, value in enumerate(hosts_list):
       # Lists details of the hosts based on id
       res2 = requests.get("http://100.109.0.1:8181/onos/v1/hosts/%s" %
value["id"], auth = HTTPBasicAuth('onos', 'rocks'))
       # Print the id and port info for host with 10.0.0.130
```

```
        if res2.json()["ipAddresses"][0] == "10.0.0.130":
            print("The id is %s and the port is %s for host with '10.0.0.130'" %
(res2.json()["id"], res2.json()["locations"][0]["port"]))
```

Result

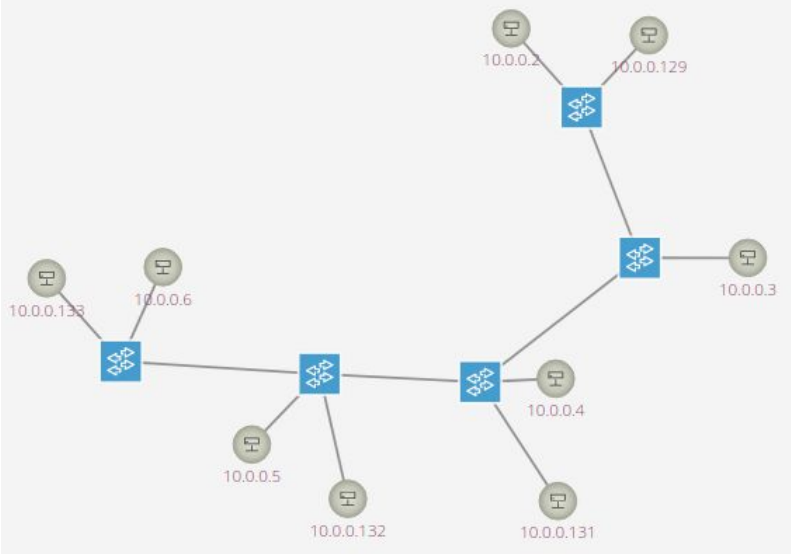
```
# python chapter5_demo1_task3_3.2.py
The id is AE:D8:D6:FF:CB:21/None and the port is 6 for host with '10.0.0.130'
```

3.3 Create a python program using the same host id, i.e., used in the previous question, to remove the host from the pre-defined architecture

Code

```
#!/usr/bin/env python3
import requests, json
from requests.auth import HTTPBasicAuth
if __name__ == "__main__":
    # Lists all hosts
    res1 = requests.get("http://100.109.0.1:8181/onos/v1/hosts", auth =
HTTPBasicAuth('onos', 'rocks'))
    hosts_list = res1.json()["hosts"]
    for index, value in enumerate(hosts_list):
        # Lists details of the hosts based on id
        res2 = requests.get("http://100.109.0.1:8181/onos/v1/hosts/%s" %
value["id"], auth = HTTPBasicAuth('onos', 'rocks'))
        # Print the id and port info for host with 10.0.0.130
        if res2.json()["ipAddresses"][0] == "10.0.0.130":
            id = res2.json()["id"]
            # Delete the host based on id info
            res3 = requests.delete("http://100.109.0.1:8181/onos/v1/hosts/%s" % id,
auth = HTTPBasicAuth('onos', 'rocks'))
```

Result



3.4 Ping the removed host, what do you observe?

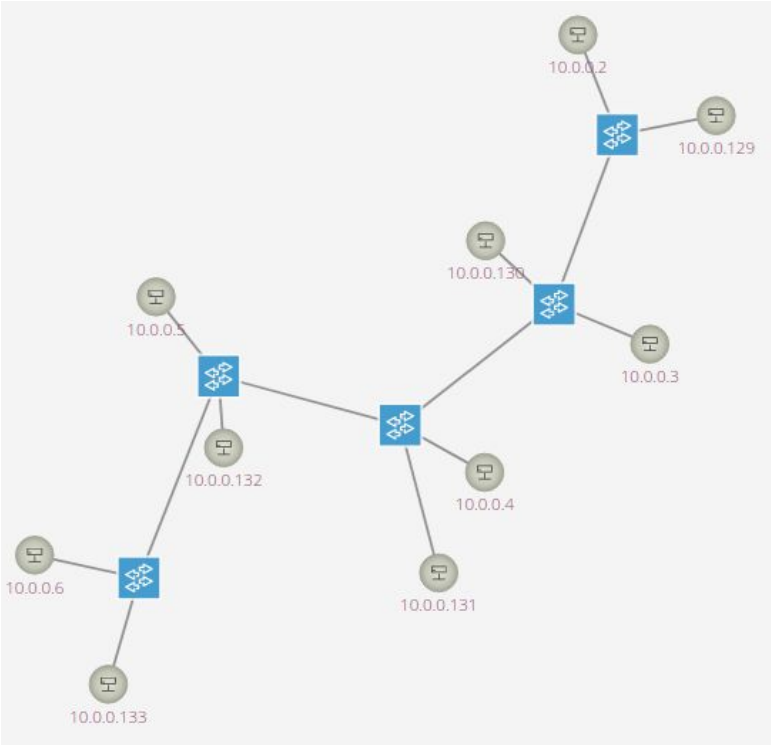
Ping Cmd

```
# ip netns exec red1 ping -c1 10.0.0.130
PING 10.0.0.130 (10.0.0.130) 56(84) bytes of data.
64 bytes from 10.0.0.130: icmp_seq=1 ttl=64 time=0.101 ms

--- 10.0.0.130 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.101/0.101/0.101/0.000 ms
```

Result

The removed host in the previous question appears again.



4 Task4

Considering the same pre-defined topology of Task 1, students are asked to:

4.1 Create a python program to list all ACTIVE links in the pre-defined topology, the output should be a table containing device id source, port source, device id destination, port destination.

Code

```
#!/usr/bin/env python3
import requests, json
```

```
from requests.auth import HTTPBasicAuth
if __name__ == "__main__":
   # Lists all links
   res1 = requests.get("http://100.109.0.1:8181/onos/v1/links", auth =
HTTPBasicAuth('onos', 'rocks'))
   links_list = res1.json()["links"]
   print("src_device_id \t src_port \t dst_device_id \t dst_port")
   for index, value in enumerate(links_list):
       # Generate the table
       print("%s \t %s \t %s \t %s" % (value["src"]["device"],
value["src"]["port"], value["dst"]["device"], value["dst"]["port"]))
```

Result

```
# python chapter5_demo1_task4_4.1.py
src_device_id    src_port    dst_device_id   dst_port
of:000056f927a4cd4c      3   of:000032c182e2cc4d     3
of:000032c182e2cc4d      2   of:0000069a5175a24d     2
of:00006a5ff8be3c41      1   of:0000069a5175a24d     1
of:000056f927a4cd4c      4   of:00000a32409edf45     4
of:00000a32409edf45      4   of:000056f927a4cd4c     4
of:000032c182e2cc4d      3   of:000056f927a4cd4c     3
of:0000069a5175a24d      2   of:000032c182e2cc4d     2
of:0000069a5175a24d      1   of:00006a5ff8be3c41     1
```

4.2 Create a python program to list all the flows applied to a device of your choice, the output may show the flow-id, the application id, the device id, and the instructions.

Code

```
#!/usr/bin/env python3
import requests, json
from requests.auth import HTTPBasicAuth
if __name__ == "__main__":
   # Lists all flows
   res1 = requests.get("http://100.109.0.1:8181/onos/v1/flows", auth =
HTTPBasicAuth('onos', 'rocks'))
   flows_list = res1.json()["flows"]
   print("flow_id \t appId \t deviceId \t instructions")
   for index, value in enumerate(flows_list):
       # Generate the table
       print("%s \t %s \t %s \t %s" % (value["id"], value["appId"],
value["deviceId"], value["treatment"]["instructions"]))
```

Result

```
# python chapter5_demo1_task4_4.2.py
flow_id      appId   deviceId    instructions
281475937560871      org.onosproject.core    of:00006a5ff8be3c41     [{'type':
'OUTPUT', 'port': 'CONTROLLER'}]
281476070693067      org.onosproject.core    of:00006a5ff8be3c41     [{'type':
'OUTPUT', 'port': 'CONTROLLER'}]
281475744851921      org.onosproject.core    of:00006a5ff8be3c41     [{'type':
'OUTPUT', 'port': 'CONTROLLER'}]
281476490248436      org.onosproject.core    of:00006a5ff8be3c41     [{'type':
'OUTPUT', 'port': 'CONTROLLER'}]
281476724739200      org.onosproject.core    of:0000069a5175a24d     [{'type':
'OUTPUT', 'port': 'CONTROLLER'}]
281476384375467      org.onosproject.core    of:0000069a5175a24d     [{'type':
'OUTPUT', 'port': 'CONTROLLER'}]
```

```
281477560749653      org.onosproject.core    of:0000069a5175a24d     [{'type':
'OUTPUT', 'port': 'CONTROLLER'}]
281478413654880      org.onosproject.core    of:0000069a5175a24d     [{'type':
'OUTPUT', 'port': 'CONTROLLER'}]
281476078956930      org.onosproject.core    of:000032c182e2cc4d     [{'type':
'OUTPUT', 'port': 'CONTROLLER'}]
281478434469917      org.onosproject.core    of:000032c182e2cc4d     [{'type':
'OUTPUT', 'port': 'CONTROLLER'}]
281475336638134      org.onosproject.core    of:000032c182e2cc4d     [{'type':
'OUTPUT', 'port': 'CONTROLLER'}]
281476165196600      org.onosproject.core    of:000032c182e2cc4d     [{'type':
'OUTPUT', 'port': 'CONTROLLER'}]
281478276221576      org.onosproject.core    of:000056f927a4cd4c     [{'type':
'OUTPUT', 'port': 'CONTROLLER'}]
281476861117707      org.onosproject.core    of:000056f927a4cd4c     [{'type':
'OUTPUT', 'port': 'CONTROLLER'}]
281478489784750      org.onosproject.core    of:000056f927a4cd4c     [{'type':
'OUTPUT', 'port': 'CONTROLLER'}]
281476376779355      org.onosproject.core    of:000056f927a4cd4c     [{'type':
'OUTPUT', 'port': 'CONTROLLER'}]
281476086571297      org.onosproject.core    of:00000a32409edf45     [{'type':
'OUTPUT', 'port': 'CONTROLLER'}]
281477444935778      org.onosproject.core    of:00000a32409edf45     [{'type':
'OUTPUT', 'port': 'CONTROLLER'}]
281475092248806      org.onosproject.core    of:00000a32409edf45     [{'type':
'OUTPUT', 'port': 'CONTROLLER'}]
281475715196917      org.onosproject.core    of:00000a32409edf45     [{'type':
'OUTPUT', 'port': 'CONTROLLER'}]
```

4.3 Create a python program to list all intents.

Code

```
#!/usr/bin/env python3
import requests, json
from requests.auth import HTTPBasicAuth
if __name__ == "__main__":
   # Lists all intents
   res1 = requests.get("http://100.109.0.1:8181/onos/v1/intents", auth =
HTTPBasicAuth('onos', 'rocks'))
   intents_list = res1.json()
   print(intents_list)
```

Result

```
# python chapter5_demo1_task4_4.3.py
{'intents': [{'type': 'HostToHostIntent', 'id': '0x1', 'key': '0x1', 'appId':
'org.onosproject.cli', 'resources': ['00:00:00:00:00:01/None',
'00:00:00:00:00:10/None'], 'state': 'FAILED'}, {'type': 'HostToHostIntent', 'id':
'0x0', 'key': '0x0', 'appId': 'org.onosproject.cli', 'resources':
['00:00:00:00:00:01/None', '00:00:00:00:00:10/None'], 'state': 'FAILED'}]}
```