

Laboratory Design Activity

Network Programmability using NETCONF

Intended Learning Outcomes:

- Configure OSPF routing protocol with NETCONF using Python ncclient in Cisco IOS-XE device.
- Configure an IPV4 ACL for NETCONF sessions.
- Configure IP Route with NETCONF using Python ncclient in Cisco IOS-XE device.
- Use pyATS to test your network

Resources:

- Virtual Box
- DEVASC-LABVM virtual machine
- CSR1000v virtual machine

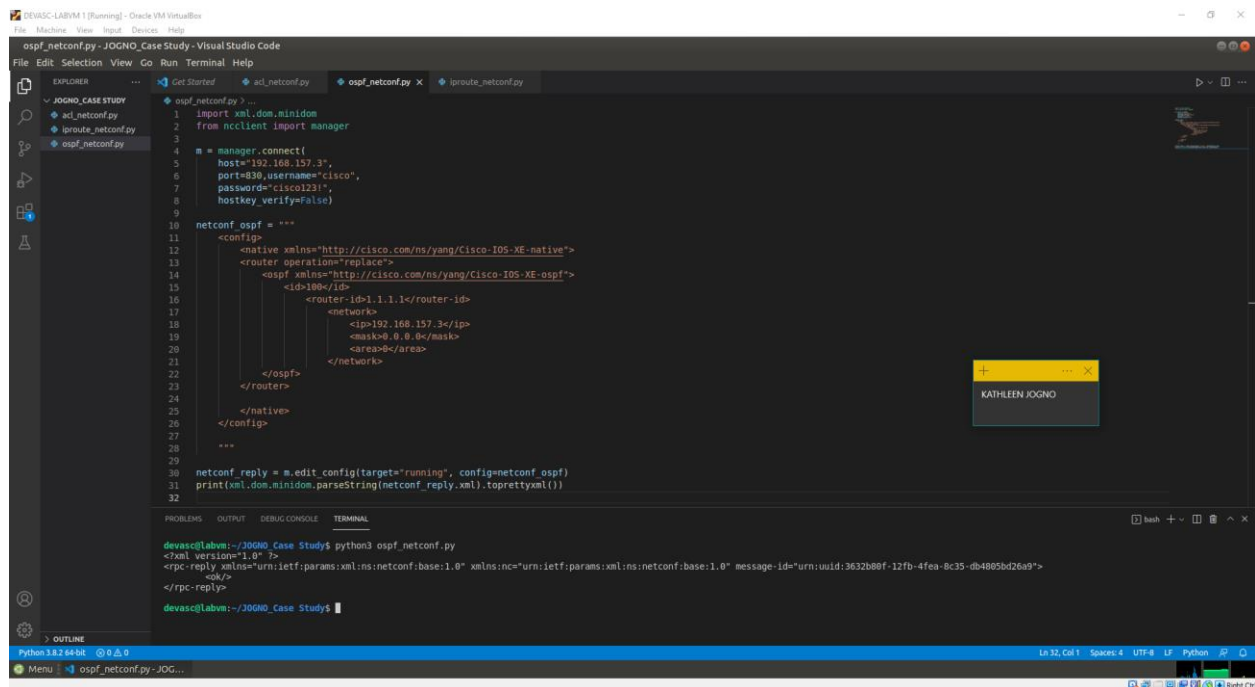
Submitted by:

Kathleen Q. Jogno

CPE41S2

Network Programmability using NETCONF

1. Start DEVASC-LABVM virtual machine
2. Configure an OSPF routing protocol with process id of 100 using NETCONF.



The screenshot shows the Visual Studio Code editor with the file `ospf_netconf.py` open. The script uses the `ncclient` library to connect to a CSR1000v virtual machine and configure OSPF via NETCONF. The configuration includes setting the native XML namespace, the operation to replace the configuration, and the specific OSPF configuration for router-id 1.1.1.1, including network 192.168.157.3/24 and area 0. The terminal output shows the successful execution of the script, including the XML configuration being sent to the device.

```
ospf_netconf.py - JOGNO_Case Study - Visual Studio Code
File Edit Selection View Go Run Terminal Help

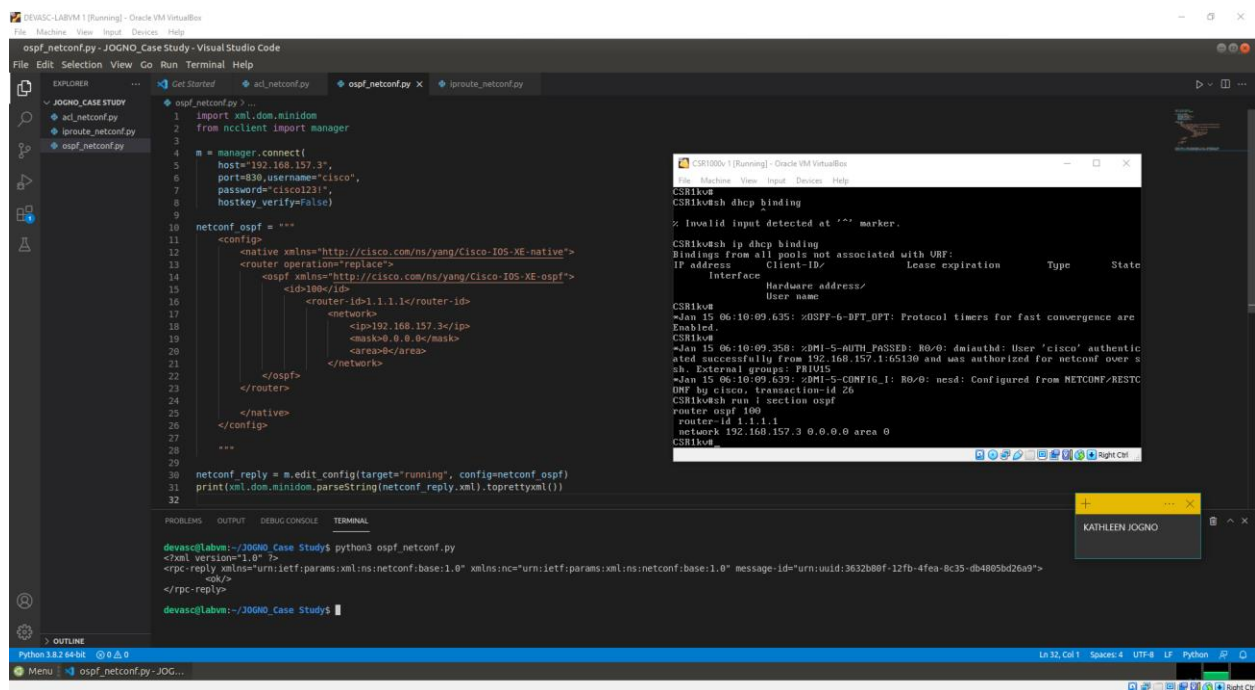
EXPLORER
  JOGNO_Case Study
    acl_netconf.py
    iproute_netconf.py
    ospf_netconf.py

Get Started
  ospf_netconf.py
1 import xml.dom.minidom
2 from ncclient import manager
3
4 m = manager.connect(
5     host='192.168.157.3',
6     port=830, username='cisco',
7     password='cisco123!',
8     hostkey_verify=False)
9
10 netconf_ospf = """
11 <config>
12   <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
13     <router operation="replace">
14       <ospf xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-ospf">
15         <id=100/>
16         <router-id=1.1.1.1/>
17         <network>
18           <ip>192.168.157.3/</ip>
19           <mask>0.0.0.0/</mask>
20           <area>0/</area>
21         </network>
22       </ospf>
23     </router>
24   </native>
25 </config>
26 """
27
28 netconf_reply = m.edit_config(target="running", config=netconf_ospf)
29 print(xml.dom.minidom.parseString(netconf_reply.xml).toprettyxml())
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
devasc@labvm:~/JOGNO_Case Study$ python3 ospf_netconf.py
<?xml version="1.0"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="urn:uuid:3632b0ff-12fb-4fea-bc35-db4085bd26a9">
  <ok/>
</rpc-reply>

devasc@labvm:~/JOGNO_Case Study$
```

3. Run the file and check the output in the CSR1000v virtual machine.



This screenshot shows the same Visual Studio Code environment as the previous one, but with an additional window titled "CSR1000v" open. This window displays the output of the configuration on the CSR1000v virtual machine. The output shows the successful configuration of OSPF process 100, including the router-id 1.1.1.1 and the network 192.168.157.3/24 in area 0. The terminal in the main window also shows the successful execution of the script.

```
ospf_netconf.py - JOGNO_Case Study - Visual Studio Code
File Edit Selection View Go Run Terminal Help

EXPLORER
  JOGNO_Case Study
    acl_netconf.py
    iproute_netconf.py
    ospf_netconf.py

Get Started
  ospf_netconf.py
1 import xml.dom.minidom
2 from ncclient import manager
3
4 m = manager.connect(
5     host='192.168.157.3',
6     port=830, username='cisco',
7     password='cisco123!',
8     hostkey_verify=False)
9
10 netconf_ospf = """
11 <config>
12   <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
13     <router operation="replace">
14       <ospf xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-ospf">
15         <id=100/>
16         <router-id=1.1.1.1/>
17         <network>
18           <ip>192.168.157.3/</ip>
19           <mask>0.0.0.0/</mask>
20           <area>0/</area>
21         </network>
22       </ospf>
23     </router>
24   </native>
25 </config>
26 """
27
28 netconf_reply = m.edit_config(target="running", config=netconf_ospf)
29 print(xml.dom.minidom.parseString(netconf_reply.xml).toprettyxml())
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
devasc@labvm:~/JOGNO_Case Study$ python3 ospf_netconf.py
<?xml version="1.0"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="urn:uuid:3632b0ff-12fb-4fea-bc35-db4085bd26a9">
  <ok/>
</rpc-reply>

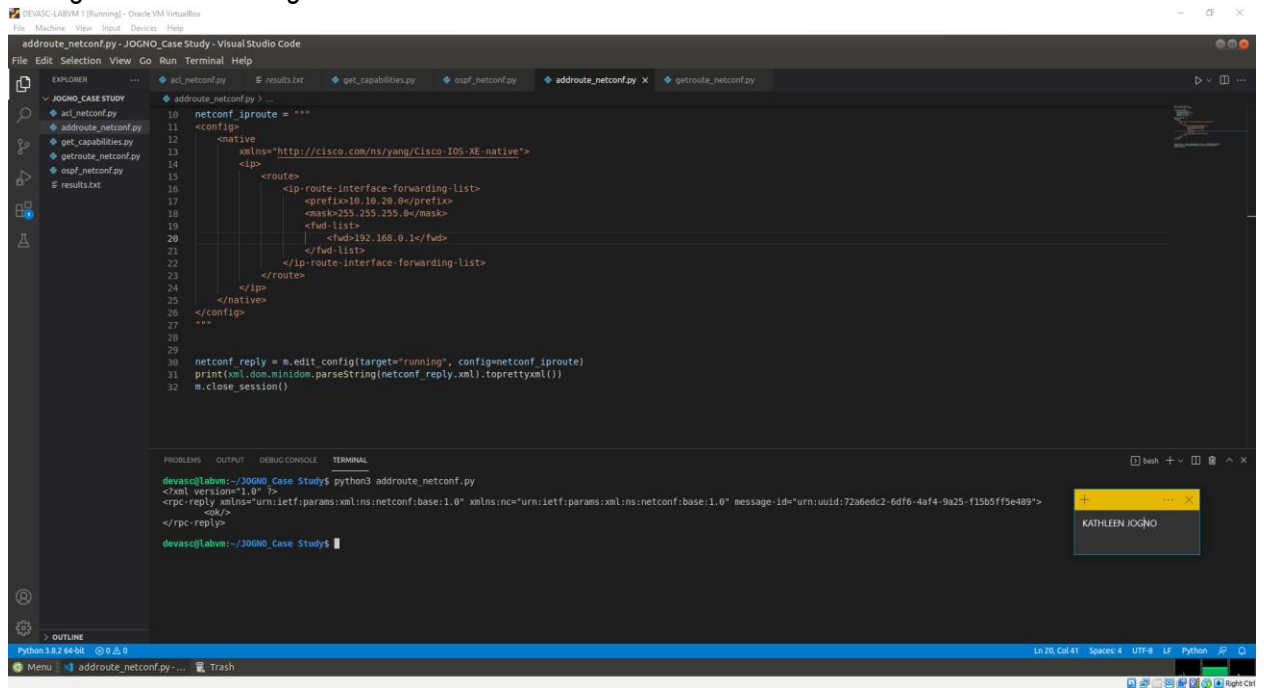
devasc@labvm:~/JOGNO_Case Study$
```

```
CSR1000v
CSR100v
CSR100vsh dhcp binding
% Invalid input detected at '^' marker.
CSR100vsh ip dhcp binding
Bindings from all pools not associated with VRF:
IP address Client-ID Lease expiration Type State
Interface Hardware address/ User name
CSR100v
Jan 15 06:10:09.635: %OSPF-6-BFT_OPT: Protocol timers for fast convergence are
Enabled.
CSR100v
Jan 15 06:10:09.350: %DMI-5-AUTH PASSED: R0/0: dmlauthd: User 'cisco' authentic
ated successfully from 192.168.157.1:65130 and was authorized for netconf over s
sh. External groups: PRIV15
Jan 15 06:10:09.639: %DMI-5-CONF16_1: R0/0: mesd: Configured from NETCONF/RESTC
ONF by cisco, transaction id 26
CSR100vsh run i section ospf
router ospf 100
router-id 1.1.1.1
network 192.168.157.3 0.0.0.0 area 0
CSR100v
```

4. Configure an IPV4 ACL for NETCONF sessions.

[illegible]

5. Configure IP Route using NETCONF.



```
addroute_netconf.py: JOGNO_Case Study: Visual Studio Code
File Edit Selection View Go Run Terminal Help

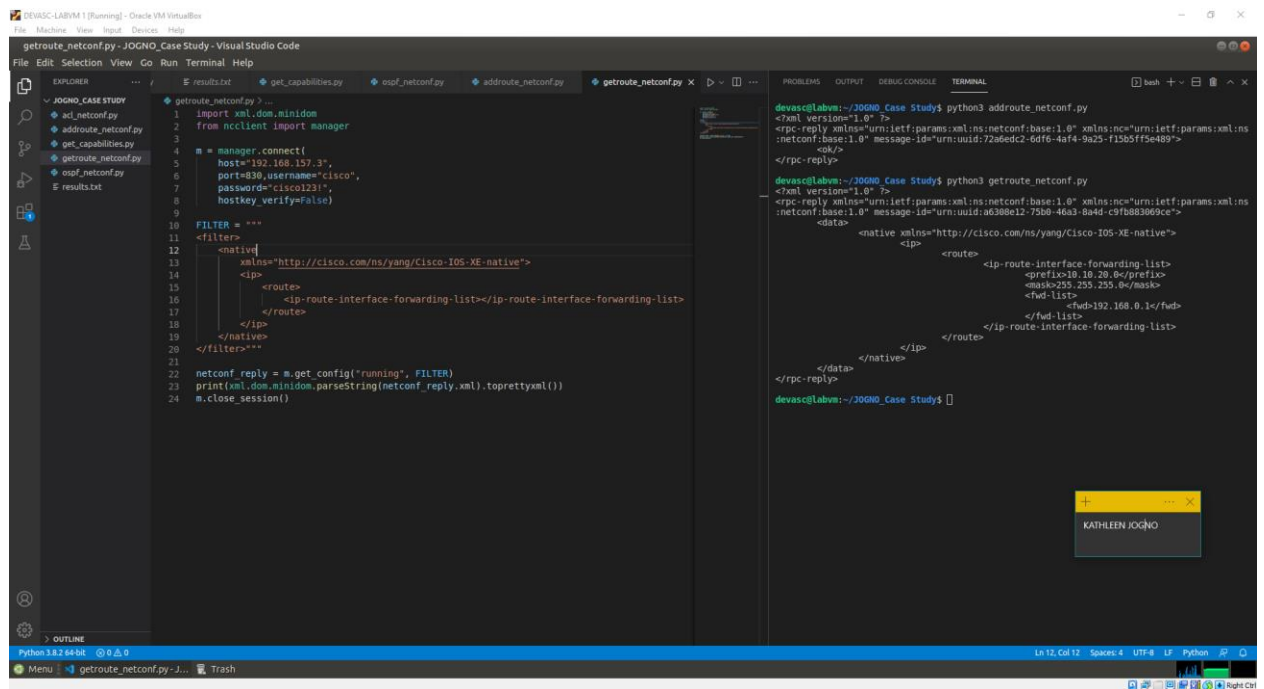
EXPLORER
  JOGNO_CASE STUDY
    addroute_netconf.py
    getroute_netconf.py
    get_capabilities.py
    ospf_netconf.py
    results.txt

addroute_netconf.py
10 netconf_iproute = """
11 <config>
12   <native
13     xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
14     <ip>
15       <route>
16         <ip-route-interface-forwarding-list>
17           <prefix>10.10.20.0/</prefix>
18           <mask>255.255.0/</mask>
19           <fwd-list>
20             <fwd>192.168.0.1/</fwd>
21           </fwd-list>
22         </ip-route-interface-forwarding-list>
23       </route>
24     </ip>
25   </native>
26 </config>
27 """
28
29 netconf_reply = m.edit_config(target="running", config=netconf_iproute)
30 print(xml.dom.minidom.parseString(netconf_reply.xml).toprettyxml())
31 m.close_session()
32
```

```
devasc@labvm:~/JOGNO_Case Study$ python3 addroute_netconf.py
<?xml version="1.0" ?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="urn:uuid:72a6edc2-6df6-4af4-9a25-f15b5ff5e489">
  <ok/>
</rpc-reply>

devasc@labvm:~/JOGNO_Case Study$
```

6. Run the file and check the output in the CSR1000v virtual machine.



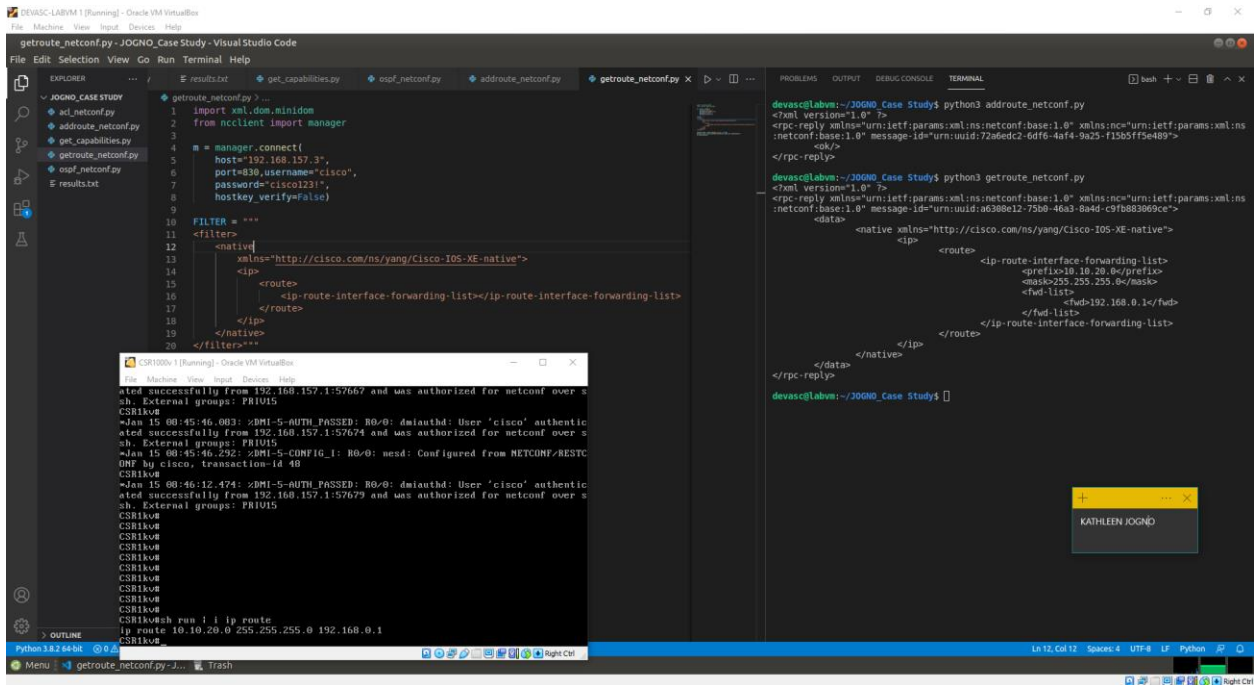
```
getroute_netconf.py: JOGNO_Case Study: Visual Studio Code
File Edit Selection View Go Run Terminal Help

EXPLORER
  JOGNO_CASE STUDY
    addroute_netconf.py
    getroute_netconf.py
    get_capabilities.py
    ospf_netconf.py
    results.txt

getroute_netconf.py
1 import xml.dom.minidom
2 from ncclient import manager
3
4 m = manager.connect(
5   host="192.168.157.3",
6   port=830, username="cisco",
7   password="cisco123!",
8   hostkey_verify=False)
9
10 FILTER = """
11 <filter>
12   <native
13     xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
14     <ip>
15       <route>
16         <ip-route-interface-forwarding-list></ip-route-interface-forwarding-list>
17       </route>
18     </ip>
19   </native>
20 </filter>"""
21
22 netconf_reply = m.get_config("running", FILTER)
23 print(xml.dom.minidom.parseString(netconf_reply.xml).toprettyxml())
24 m.close_session()
25
```

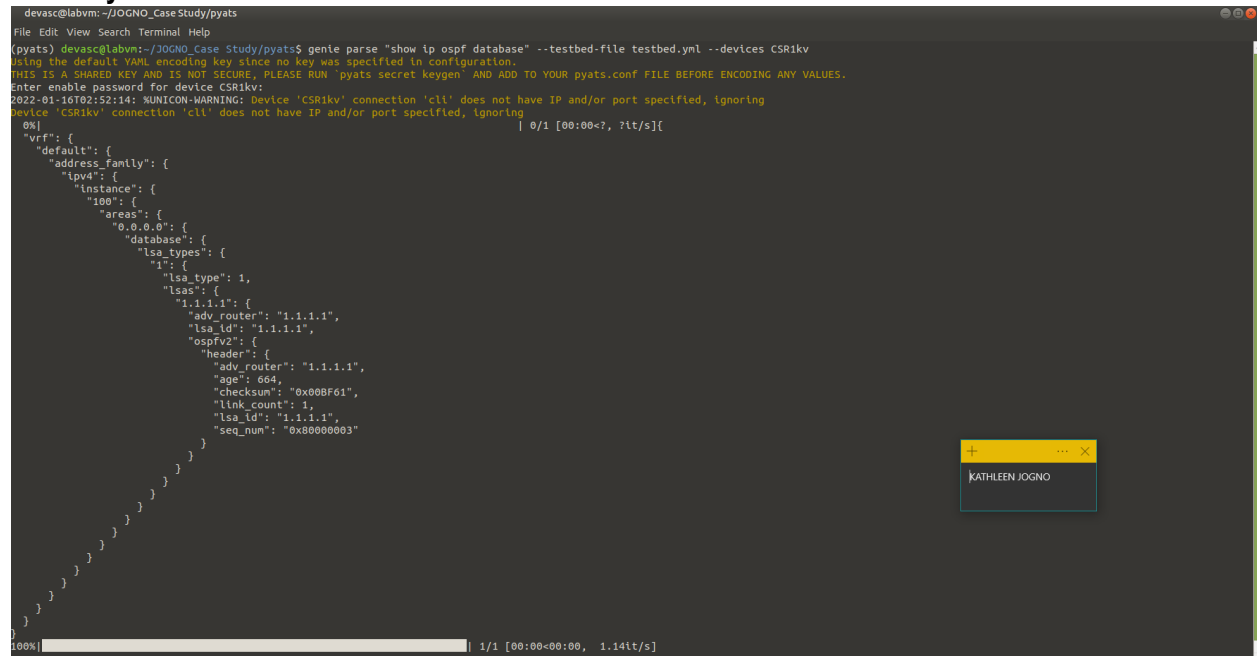
```
devasc@labvm:~/JOGNO_Case Study$ python3 getroute_netconf.py
<?xml version="1.0" ?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="urn:uuid:a638be12-75b0-46a3-8aed-c9fb838696ce">
  <data>
    <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
      <ip>
        <route>
          <ip-route-interface-forwarding-list>
            <prefix>10.10.20.0/</prefix>
            <mask>255.255.0/</mask>
            <fwd-list>
              <fwd>192.168.0.1/</fwd>
            </fwd-list>
          </ip-route-interface-forwarding-list>
        </route>
      </ip>
    </native>
  </data>
</rpc-reply>

devasc@labvm:~/JOGNO_Case Study$
```



Use PYATS to test Network

1. Verify OSPF



2. Verify ACL

```
devasc@labvm:~/JOGNO_CaseStudy/pyats$
File Edit View Search Terminal Help
devasc@labvm:~/JOGNO_CaseStudy/pyats$
(pyats) devasc@labvm:~/JOGNO_CaseStudy/pyats$ genie parse "show ip access-lists" --testbed-file testbed.yml --devices CSRikv
Using the default YAML encoding key since no key was specified in configuration.
THIS IS A SHARED KEY AND IS NOT SECURE. PLEASE RUN 'pyats secret keygen' AND ADD TO YOUR pyats.conf FILE BEFORE ENCODING ANY VALUES.
Enter enable password for device CSRikv:
2022-01-16T02:57:37: NUNICON-WARNING: Device 'CSRikv' connection 'cli' does not have IP and/or port specified, ignoring
Device 'CSRikv' connection 'cli' does not have IP and/or port specified, ignoring
0/1 [00:00:~7, 71t/s]
{
  "acl_permit": {
    "aces": {
      "10": {
        "actions": {
          "forwarding": "permit"
        },
        "matches": {
          "13": {
            "ipv4": {
              "protocol": "ip4",
              "source_network": {
                "192.168.255.0 0.0.0.255": {
                  "source_network": "192.168.255.0 0.0.0.255"
                }
              }
            }
          }
        },
        "name": "10"
      },
      "20": {
        "actions": {
          "forwarding": "deny"
        },
        "matches": {
          "13": {
            "ipv4": {
              "protocol": "ip4",
              "source_network": {
                "any": {
                  "source_network": "any"
                }
              }
            }
          }
        },
        "name": "20"
      }
    },
    "acl_type": "standard",
    "name": "acl_permit",
    "type": "ipv4-acl-type"
  }
}
100% | 1/1 [00:00:00:00, 1.261t/s]
(pyats) devasc@labvm:~/JOGNO_CaseStudy/pyats$
```

3. Verify IP Route

```
devasc@labvm:~/JOGNO_CaseStudy/pyats$
File Edit View Search Terminal Help
devasc@labvm:~/JOGNO_CaseStudy/pyats$
(pyats) devasc@labvm:~/JOGNO_CaseStudy/pyats$ genie parse "show ip route" --testbed-file testbed.yml --devices CSRikv
Using the default YAML encoding key since no key was specified in configuration.
THIS IS A SHARED KEY AND IS NOT SECURE. PLEASE RUN 'pyats secret keygen' AND ADD TO YOUR pyats.conf FILE BEFORE ENCODING ANY VALUES.
Enter enable password for device CSRikv:
2022-01-16T03:00:22: NUNICON-WARNING: Device 'CSRikv' connection 'cli' does not have IP and/or port specified, ignoring
Device 'CSRikv' connection 'cli' does not have IP and/or port specified, ignoring
0/1 [00:00:~7, 71t/s]
{
  "verf": {
    "default": {
      "address_family": {
        "ipv4": {
          "routes": {
            "192.168.157.0/24": {
              "active": true,
              "next_hop": {
                "outgoing_interface": {
                  "GigabitEthernet1": {
                    "outgoing_interface": "GigabitEthernet1"
                  }
                }
              },
              "route": "192.168.157.0/24",
              "source_protocol": "connected",
              "source_protocol_codes": "C"
            },
            "192.168.157.3/32": {
              "active": true,
              "next_hop": {
                "outgoing_interface": {
                  "GigabitEthernet1": {
                    "outgoing_interface": "GigabitEthernet1"
                  }
                }
              },
              "route": "192.168.157.3/32",
              "source_protocol": "local",
              "source_protocol_codes": "L"
            }
          }
        }
      }
    }
  }
}
100% | 1/1 [00:00:00:00, 1.241t/s]
(pyats) devasc@labvm:~/JOGNO_CaseStudy/pyats$
```