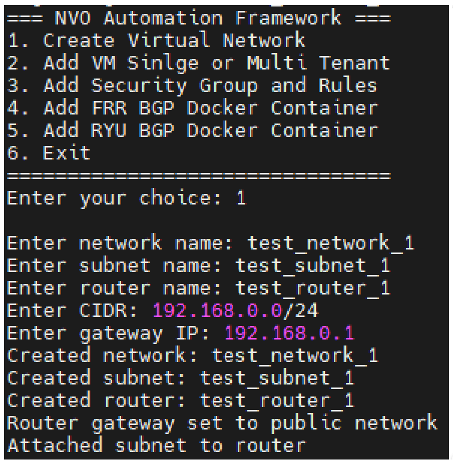
Lab9 Walkthrough

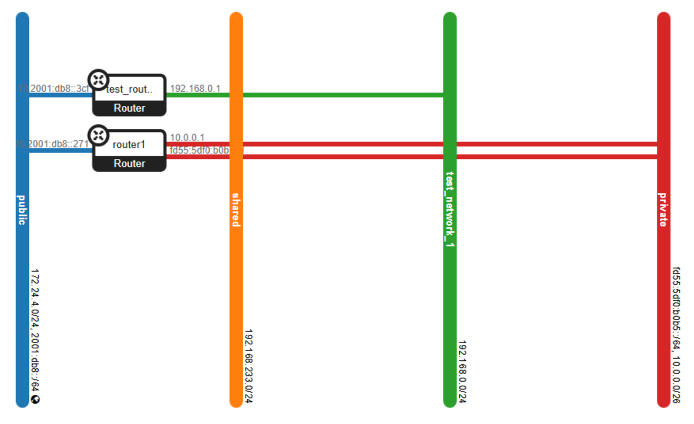
Virtual Network Automation:

Located in the virtual_network_automate.py, there is functionality to create virtual networks that have connection to the public network

Example:



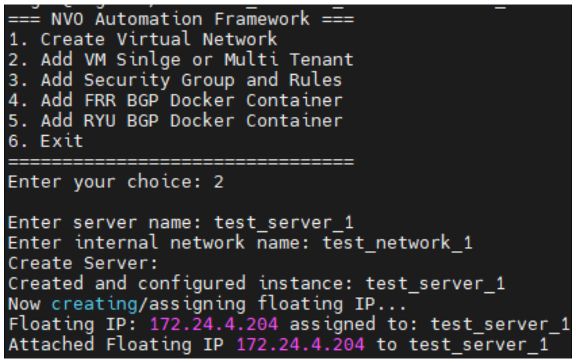
```
=== NVO Automation Framework ===
1. Create Virtual Network
2. Add VM Sinlge or Multi Tenant
3. Add Security Group and Rules
4. Add FRR BGP Docker Container
5. Add RYU BGP Docker Container
6. Exit
================================
Enter your choice: 1

Enter network name: test_network_1
Enter subnet name: test_subnet_1
Enter router name: test_router_1
Enter CIDR: 192.168.0.0/24
Enter gateway IP: 192.168.0.1
Created network: test_network_1
Created subnet: test_subnet_1
Created router: test_router_1
Router gateway set to public network
Attached subnet to router
```



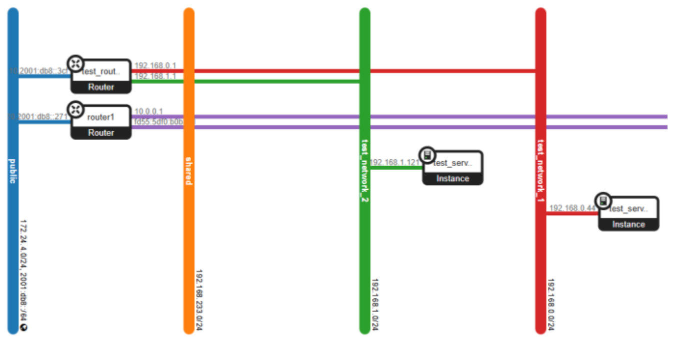
VM Automation:

Located in the vm_automate.py, there is functionality to create virtual machines that are connected to a given internal network and is automatically given a floating ip.

Example:



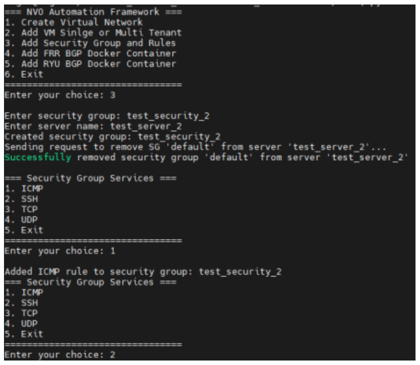
```
=== NVO Automation Framework ===
1. Create Virtual Network
2. Add VM Sinlge or Multi Tenant
3. Add Security Group and Rules
4. Add FRR BGP Docker Container
5. Add RYU BGP Docker Container
6. Exit
================================
Enter your choice: 2

Enter server name: test_server_1
Enter internal network name: test_network_1
Create Server:
Created and configured instance: test_server_1
Now creating/assigning floating IP...
Floating IP: 172.24.4.204 assigned to: test_server_1
Attached Floating IP 172.24.4.204 to test_server_1
```



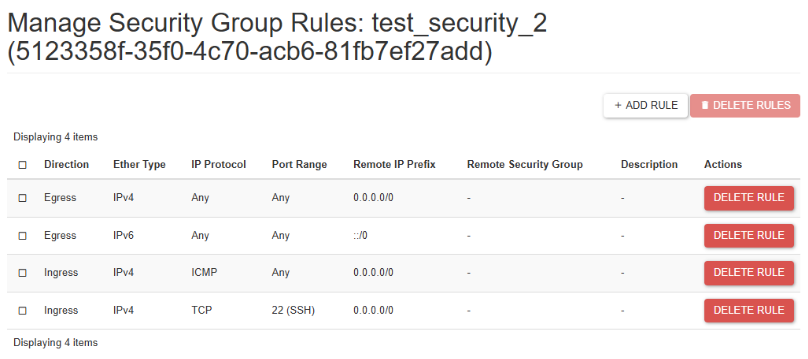
Security Group Automation:

Located in the security_automate.py, there is functionality to create a security group for a specific virtual machine. I then have added functionality for adding certain services like ICMP, TCP, UDP, etc.

Example:



```
=== NVO Automation Framework ===
1. Create Virtual Network
2. Add VM Sinlge or Multi Tenant
3. Add Security Group and Rules
4. Add FRR BGP Docker Container
5. Add RYU BGP Docker Container
6. Exit
================================
Enter your choice: 3

Enter security group: test_security_2
Enter server name: test_server_2
Created security group: test_security_2
Sending request to remove SG 'default' from server 'test_server_2'...
Successfully removed security group 'default' from server 'test_server_2'

=== Security Group Services ===
1. ICMP
2. SSH
3. TCP
4. UDP
5. Exit
================================
Enter your choice: 1

Added ICMP rule to security group: test_security_2
=== Security Group Services ===
1. ICMP
2. SSH
3. TCP
4. UDP
5. Exit
================================
Enter your choice: 2
```



Manage Security Group Rules: test_security_2 (5123358f-35f0-4c70-acb6-81fb7ef27add)

Displaying 4 items

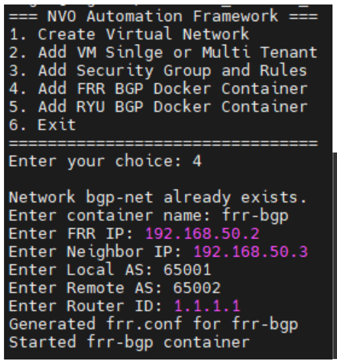
☐	Direction	Ether Type	IP Protocol	Port Range	Remote IP Prefix	Remote Security Group	Description	Actions
☐	Egress	IPv4	Any	Any	0.0.0.0/0	-	-	DELETE RULE
☐	Egress	IPv6	Any	Any	::/0	-	-	DELETE RULE
☐	Ingress	IPv4	ICMP	Any	0.0.0.0/0	-	-	DELETE RULE
☐	Ingress	IPv4	TCP	22 (SSH)	0.0.0.0/0	-	-	DELETE RULE

Displaying 4 items

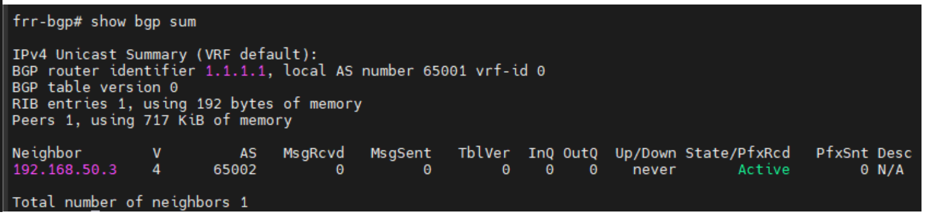
FRR BGP Docker Automation:

Located in the frr_automate.py, there is functionality to create a docker container with FRR and adds it to a docker bridge network. It then applies a simple eBGP peering configuration with the SDN BGP Docker container:

Example:



```
=== NVO Automation Framework ===
1. Create Virtual Network
2. Add VM Sinlge or Multi Tenant
3. Add Security Group and Rules
4. Add FRR BGP Docker Container
5. Add RYU BGP Docker Container
6. Exit
================================
Enter your choice: 4

Network bgp-net already exists.
Enter container name: frr-bgp
Enter FRR IP: 192.168.50.2
Enter Neighbor IP: 192.168.50.3
Enter Local AS: 65001
Enter Remote AS: 65002
Enter Router ID: 1.1.1.1
Generated frr.conf for frr-bgp
Started frr-bgp container
```

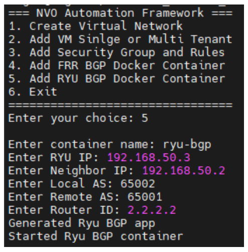


```
frr-bgp# show bgp sum

IPv4 Unicast Summary (VRF default):
BGP router identifier 1.1.1.1, local AS number 65001 vrf-id 0
BGP table version 0
RIB entries 1, using 192 bytes of memory
Peers 1, using 717 KiB of memory

Neighbor        V         AS   MsgRcvd   MsgSent   TblVer  InQ OutQ  Up/Down State/PfxRcd   PfxSnt Desc
192.168.50.3    4      65002         0         0        0    0    0    never       Active        0 N/A

Total number of neighbors 1
```

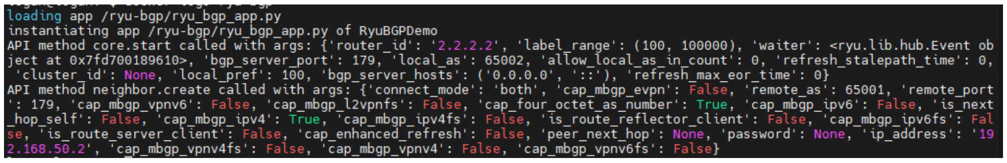
RYU BGP Docker Automation:

Located in the ryu_automate.py, there is functionality to create a docker container with Ryu controller and adds it to a docker bridge network. It then applies a simple eBGP peering configuration with the FRR BGP docker container.

Example:



```
=== NVO Automation Framework ===
1. Create Virtual Network
2. Add VM Sinlge or Multi Tenant
3. Add Security Group and Rules
4. Add FRR BGP Docker Container
5. Add RYU BGP Docker Container
6. Exit
================================
Enter your choice: 5

Enter container name: ryu-bgp
Enter RYU IP: 192.168.50.3
Enter Neighbor IP: 192.168.50.2
Enter Local AS: 65002
Enter Remote AS: 65001
Enter Router ID: 2.2.2.2
Generated Ryu BGP app
Started Ryu BGP container
```



```
loading app /ryu-bgp/ryu_bgp_app.py
instantiating app /ryu-bgp/ryu_bgp_app.py of RyuBGPDemo
API method core.start called with args: {'router_id': '2.2.2.2', 'label_range': (100, 100000), 'waiter': <ryu.lib.hub.Event ob
ject at 0x7fd700189610>, 'bgp_server_port': 179, 'local_as': 65002, 'allow_local_as_in_count': 0, 'refresh_stalepath_time': 0,
 'cluster_id': None, 'local_pref': 100, 'bgp_server_hosts': ('0.0.0.0', '::'), 'refresh_max_eor_time': 0}
API method neighbor.create called with args: {'connect_mode': 'both', 'cap_mbgp_evpn': False, 'remote_as': 65001, 'remote_port
': 179, 'cap_mbgp_vpnv6': False, 'cap_mbgp_l2vpnfs': False, 'cap_four_octet_as_number': True, 'cap_mbgp_ipv6': False, 'is_next
_hop_self': False, 'cap_mbgp_ipv4': True, 'cap_mbgp_ipv4fs': False, 'is_route_reflector_client': False, 'cap_mbgp_ipv6fs': Fal
se, 'is_route_server_client': False, 'cap_enhanced_refresh': False, 'peer_next_hop': None, 'password': None, 'ip_address': '19
2.168.50.2', 'cap_mbgp_vpnv4fs': False, 'cap_mbgp_vpnv4': False, 'cap_mbgp_vpnv6fs': False}
```