CSCI 5380 - Network Virtualization and Orchestration

Lab 9
Automate VM, VN, Docker, and BGP path

University of Colorado Boulder
Department of Computer Science
Network Engineering

Professor Levi Perigo, Ph.D.

Summary:

In this lab, you will use what you have learned in previous labs and automate the processes into a single application.

Required technologies:

- BGP
- Hypervisor/Orchestrator (such as OpenStack)
- Containers
- SDN Controller
- Hardware server
- Service-chain

Objectives: Virtualized Network Automation

Create an application that meets the following functionality (each objective must be a separate Python module in your code i.e. your main .py file should import the different modules you write):

1) Automate the creation of multiple virtual networks (VNs) within the hypervisor and their connection to the public network.

Creating networks:

```
Virtual Networks menu
1. Create
2. Delete
3. List
4. Main menu
Select an option (1-4) or type 'exit': 1

Enter the number of virtual networks to create: 1
Enter name for network 1: ash-net-1
Enter CIDR for ash-net-1 (e.g., 192.168.1.0/24): 10.10.10.0/24

************************************************************
Created network ash-net-1 with subnet 10.10.10.0/24
************************************************************
```

Delete network:

```
Select the network you want to delete:
1. ash-net-1
2. vn1
3. public
4. vn2
5. network_20
6. shared
7. network_10
8. vn2
9. vn1
10. private
Select a network to delete (1-10): 1

Are you sure you want to delete network 'ash-net-1'? (yes/no): yes

************************************************************
Deleted network: ash-net-1
************************************************************
```

List networks:

```
Virtual Networks menu
1. Create
2. Delete
3. List
4. Main menu
Select an option (1-4) or type 'exit': 3

************************************************************
List of Networks:
1. ash-net-1
2. vn1
3. public
4. vn2
5. network_20
6. shared
7. network_10
8. vn2
9. vn1
10. private
************************************************************
```

Create Router:

```
Virtual Router menu
1. Create
2. Delete
3. Connect
4. List
5. Main menu
Select an option (1-4) or type 'exit': 1

Enter the number of virtual routers to create: 1
Enter name for router 1: ash-router-1

************************************************************
Created router ash-router-1
************************************************************

Do you want to connect this router to the gateway first? (yes/no): yes

************************************************************
Connected router ash-router-1 to the gateway public
************************************************************

Enter the number of subnets to connect this router to: 1

Available Subnets:
1. vn1-subnet (4536550f-4bf1-4b16-9f6a-57dbfcc39822)
2. private-subnet (48c6d1e6-d190-4948-965c-24930d8fb1a5)
3. shared-subnet (5ad8ec15-9a06-4bbd-af8b-a6f819cf027d)
4. vn2-subnet (5c565027-6bbb-499f-933d-4acd7b596b10)
5. subnet_10 (74e0bd4c-2887-4272-8f88-25736647f13c)
6. ipv6-private-subnet (9466aa9d-afad-400e-9fd5-64f1e51bdee6)
7. subnet_20 (b865e5cd-d4ac-4667-a9a1-665e06bc8b60)
8. vn2-subnet (ca7d6462-1165-4577-b88f-41e805811b44)
9. public-subnet (ce3a2015-8cd4-4f1d-af03-c3e2f9b36f2f)
10. ash-net-1-subnet (f272523f-febb-4ef4-aca8-b403ac04857c)
11. ipv6-public-subnet (fc96c986-dc36-49b7-8ec8-7cdcc635391c)
12. vn1-subnet (fea3a8dc-5327-4c61-80f1-de7912ecd5fe)
Select subnet 1 to connect (1-12): 10

Connecting router 5a084b30-8113-405b-b1e4-3048eb94170b to subnet f272523f-febb-4ef4-aca8-b403ac04857c...

************************************************************
Connected subnet ash-net-1-subnet to router ash-router-1
************************************************************
```

Delete Router:

```
Select the router you want to delete:
1. vn2-router
2. router1
3. vn2-router
4. vn1-router
5. shared_router
6. ash-router-1
7. vn1-router
Select a router to delete (1-7): 6

Are you sure you want to delete router 'ash-router-1'? (yes/no): yes

Deleting router: ash-router-1...
Removing gateway from router ash-router-1...

*********************************************************
Gateway removed from ash-router-1
*********************************************************
Detached subnet cdfc769c-c193-4953-a676-7c3c36d3964c from router ash-router-1

Skipping subnet ce3a2015-8cd4-4f1d-af03-c3e2f9b36f2f, it is already removed while clearing the gateway

*********************************************************
Deleted router: ash-router-1
*********************************************************
```

List Router:

```
Virtual Router menu
1. Create
2. Delete
3. Connect
4. List
5. Main menu
Select an option (1-4) or type 'exit': 4

*********************************************************
List of Routers:
1. vn2-router
2. router1
3. vn2-router
4. vn1-router
5. shared_router
6. vn1-router
*********************************************************
```

2) Automate the creation of multiple VMs within the hypervisor-

   a) Both single tenant (same VN) and multi-tenant (different VNs).

Creating servers:

```
Virtual Server menu
1. Create
2. Delete
3. List
4. Main menu
Select an option (1-4) or type 'exit': 1

Enter the number of virtual servers to create: 1
Enter name for server 1: ash-vm-1

Available Images:
1. cirros-0.6.3-x86_64-disk (2a81014a-2663-4d07-8a18-f4f340d8467d)
Select the image for server 1 (1-1): 1

Available Flavors:
1. m1.tiny (1)
2. m1.small (2)
3. m1.medium (3)
4. m1.large (4)
5. m1.nano (42)
6. m1.xlarge (5)
7. m1.micro (84)
8. cirros256 (c1)
9. ds512M (d1)
10. ds1G (d2)
11. ds2G (d3)
12. ds4G (d4)
Select the flavor for server 1 (1-12): 2

Available Networks:
1. vn1 (45778fdf-f280-470a-89a7-f9736eeb4327)
2. public (4c549ae7-3095-44b1-910f-5461392e9e83)
3. vn2 (559ba611-0d4b-43dd-bfba-fe144bdb79c9)
4. network_20 (6712be43-a193-4f0a-aa2b-57a79efdcb04)
5. shared (7b340f69-5110-4413-8cc5-229dbe131122)
6. ash-net-1 (7b558ac6-80a8-4046-948b-9ee6aa494a6b)
7. network_10 (9b8fb01a-3749-468f-9586-dde622e2c842)
8. vn2 (9d916897-69bb-469c-ac9a-2cd6cb066071)
9. vn1 (9eb34eff-c595-43de-9caa-154bdfe9a6d6)
10. private (ed60a3a5-7bd8-4fb6-aef2-782a1110b281)
Select the network for server 1 (1-10): 6

**********************************************************
Created server ash-vm-1
**********************************************************



Server ash-vm-1 is in BUILD state. Waiting...
Server ash-vm-1 is in BUILD state. Waiting...
Server ash-vm-1 is now ACTIVE.

Assign a floating IP? (yes/no): yes
```

```
**********************************************************
Assigned Floating IP: 172.24.4.236 to server ash-vm-1
**********************************************************

Allow all traffic to the internet? (yes/no): yes

**********************************************************
Created and assigned security group: ash-vm-1-security-group to server ash-vm-1
**********************************************************
```

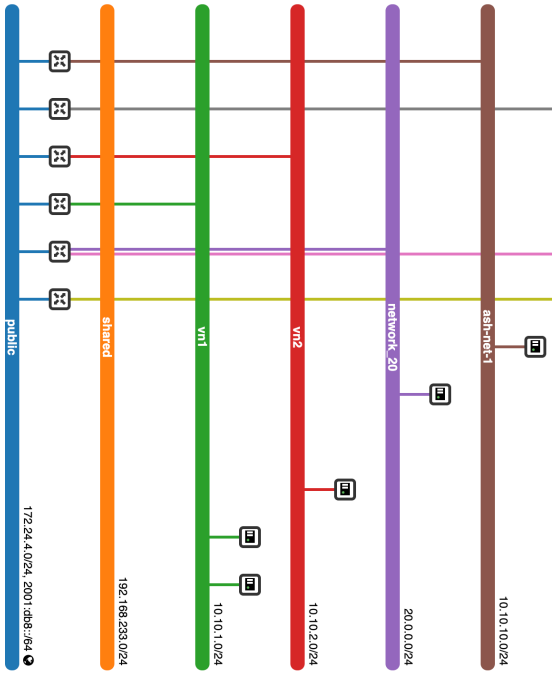
Deleting servers:

```
Virtual Server menu
1. Create
2. Delete
3. List
4. Main menu
Select an option (1-4) or type 'exit': 2

Select the server you want to delete:
1. vm22
2. vm11
3. ash-vm-1
4. vm3
5. vm2
6. vm1
Select a server to delete (1-6): 3
Attempting to delete server: ash-vm-1 (ID: cb8db543-fd0b-490c-ab9e-7169c67cefa6)
Server status: ACTIVE

Are you sure you want to delete server 'ash-vm-1'? (yes/no): yes

Cleaning up....

**********************************************************
Deleted floating IP: 172.24.4.126
Deleted server: ash-vm-1
Deleted security group: ash-vm-1-security-group
**********************************************************
```

List servers:

```

Virtual Server menu
1. Create
2. Delete
3. List
4. Main menu
Select an option (1-4) or type 'exit': 3

**********************************************************
List of Servers:
1. vm22
2. vm11
3. vm3
4. vm2
5. vm1
**********************************************************
```

Final topology – Brown network:



b) All VMs should be accessible from the host server and be able to access the Internet.

Internet access:

```
login as 'cirros' user. default password: 'gocubsgo'. use 'sudo' for root.
ash-vm-1 login: cirros
Password:
$
$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=54 time=10.2 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=54 time=5.63 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=54 time=3.16 ms
^C
--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 3.157/6.328/10.196/2.915 ms
```

3) Automate the security groups and port security configuration to make intra-VN and inter-VN communication possible.

4) Automate spinning up and configuring a Quagga/FRR BGP router as a Docker container.
    a) Automate its BGP configuration to peer with the SDN controller in the next objective.

Creating docker image:

```
stack@nvo-sneha: ~/ash/lab9 (ssh)
(newenv) stack@nvo-sneha:~/ash/lab9$ docker build -t frr-ubuntu .
[+] Building 1579.8s (14/14) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 2.93kB
 => [internal] load metadata for docker.io/library/ubuntu:latest
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [ 1/10] FROM docker.io/library/ubuntu:latest
 => [ 2/10] RUN apt update && apt install -y     git autoconf automake libtool make libreadline-dev texinfo     pkg-config libpam0g
 => [ 3/10] WORKDIR /usr/src
 => [ 4/10] RUN git clone https://github.com/CESNET/libyang.git &&     cd libyang && git checkout v2.1.128 &&     mkdir build && cd
 => [ 5/10] RUN groupadd -r -g 92 frr &&     groupadd -r -g 85 frrvty &&     adduser --system --ingroup frr --home /var/run/frr/
 => [ 6/10] WORKDIR /usr/src
 => [ 7/10] RUN git clone https://github.com/frrouting/frr.git &&     cd frr && ./bootstrap.sh &&     ./configure         --prefix=
 => [ 8/10] RUN install -m 775 -o frr -g frr -d /var/log/frr &&     install -m 775 -o frr -g frrvty -d /etc/frr &&     install -m 6
 => [ 9/10] RUN echo "net.ipv4.ip_forward=1" >> /etc/sysctl.conf &&     echo "net.ipv6.conf.all.forwarding=1" >> /etc/sysctl.conf
 => [10/10] RUN install -m 644 /usr/src/frr/tools/frr.service /etc/systemd/system/frr.service &&     systemctl enable frr || true
 => exporting to image
 => => exporting layers
 => => writing image sha256:d61ca5b02a7bd3280eee310ea13c692eefc4ac4bbcb551ccea17d9bca47fc752
 => => naming to docker.io/library/frr-ubuntu
(newenv) stack@nvo-sneha:~/ash/lab9$
```

Create FRR router:

```
Enter container name: ash-frr-2
Enter Docker image (e.g., frr-ubuntu): frr-ubuntu

How many Docker networks do you want to attach? 1

--- Network #1 ---
Do you want to create a new network for Network #1? (y/n): n

>> Running: docker network ls --format '{{.Name}}'

Available Docker networks:
1. ash-net-1 (Subnet: 10.10.10.0/24)
2. ash-net-2 (Subnet: 20.20.20.0/24)
3. ash-net-3 (Subnet: 30.30.30.0/24)
4. bgp-net (Subnet: 10.0.0.0/24)
5. bgp_network (Subnet: 111.0.0.0/24)
6. bridge (Subnet: 172.17.0.0/16)
7. host (Subnet: N/A)
8. none (Subnet: N/A)
Select a network by number: 1
Assign IP to container in this network (based on 10.10.10.0/24): 10.10.10.4
------------------------------------------------------------------------------------------------------------------------

Starting container...

>> Running: docker run -dit --privileged --name ash-frr-2 --network ash-net-1 --ip 10.10.10.4 frr-ubuntu /bin/bash
5a1d5a0bd3fbed2ab1fe31e1049bb3e77adafa17bc570aeab22b9b14cd9a6e52
------------------------------------------------------------------------------------------------------------------------
```

```
Starting BGP...

>> Running: docker exec ash-frr-2 bash -c "echo 'bgpd=yes' > /etc/frr/daemons"

Enter your BGP AS number: 65004
How many BGP neighbors do you want to configure? 1

--- Neighbor #1 ---
Neighbor IP: 10.10.10.2
Neighbor AS number: 65002

Enter networks to advertise (comma-separated, e.g., 10.10.10.0/24,10.20.20.0/24): 20.20.20.0/24

Writing frr.conf...

>> Running: docker exec ash-frr-2 bash -c "echo 'frr version 10.4-dev-DockerBuild
frr defaults traditional
log syslog informational
no ipv6 forwarding
service integrated-vtysh-config
!
ip prefix-list ALL-IPv4 seq 5 permit 0.0.0.0/0 le 32
route-map ACCEPT_ALL permit 10
 match ip address prefix-list ALL-IPv4
!
router bgp 65004
 neighbor 10.10.10.2 remote-as 65002
!
 address-family ipv4 unicast
  network 20.20.20.0/24
  neighbor 10.10.10.2 route-map ACCEPT_ALL in
  neighbor 10.10.10.2 route-map ACCEPT_ALL out
 exit-address-family
exit
!
end' > /etc/frr/frr.conf"
--------------------------------------------------------------------------------------------------------------------------------
```

```
--------------------------------------------------------------------------------------------------------------------------------

Restarting FRR...

>> Running: docker exec ash-frr-2 env PYTHONWARNINGS='ignore::SyntaxWarning' systemctl restart frr

>> Running: docker exec ash-frr-2 env PYTHONWARNINGS='ignore::SyntaxWarning' systemctl status frr
frr.service - FRRouting
    Loaded: loaded (/etc/systemd/system/frr.service, enabled)
    Active: active (running)
--------------------------------------------------------------------------------------------------------------------------------

==== FRR Running Config ====

>> Running: docker exec ash-frr-2 vtysh -c 'show running-config'
Building configuration...

Current configuration:
!
frr version 10.4-dev-DockerBuild
frr defaults traditional
hostname 5a1d5a0bd3fb
log syslog informational
no ipv6 forwarding
service integrated-vtysh-config
!
ip prefix-list ALL-IPv4 seq 5 permit 0.0.0.0/0 le 32
!
route-map ACCEPT_ALL permit 10
 match ip address prefix-list ALL-IPv4
exit
!
router bgp 65004
 neighbor 10.10.10.2 remote-as 65002
 !
 address-family ipv4 unicast
  network 20.20.20.0/24
  neighbor 10.10.10.2 route-map ACCEPT_ALL in
  neighbor 10.10.10.2 route-map ACCEPT_ALL out
 exit-address-family
exit
!
end
--------------------------------------------------------------------------------------------------------------------------------
```

Testing - between two 2 frr routers:

ash-frr-1:

```
(newenv) stack@nvo-sneha:~/ash/lab9$ docker exec ash-frr-1 vtysh -c 'show run'
Building configuration...

Current configuration:
!
frr version 10.4-dev-DockerBuild
frr defaults traditional
hostname bd528692e69c
log syslog informational
no ipv6 forwarding
service integrated-vtysh-config
!
ip prefix-list ALL-IPv4 seq 5 permit 0.0.0.0/0 le 32
!
route-map ACCEPT_ALL permit 10
 match ip address prefix-list ALL-IPv4
exit
!
router bgp 65002
 neighbor 10.10.10.4 remote-as 65004
 !
 address-family ipv4 unicast
  network 20.20.20.0/24
  neighbor 10.10.10.4 route-map ACCEPT_ALL in
  neighbor 10.10.10.4 route-map ACCEPT_ALL out
 exit-address-family
exit
!
end
```

ash-frr-2:

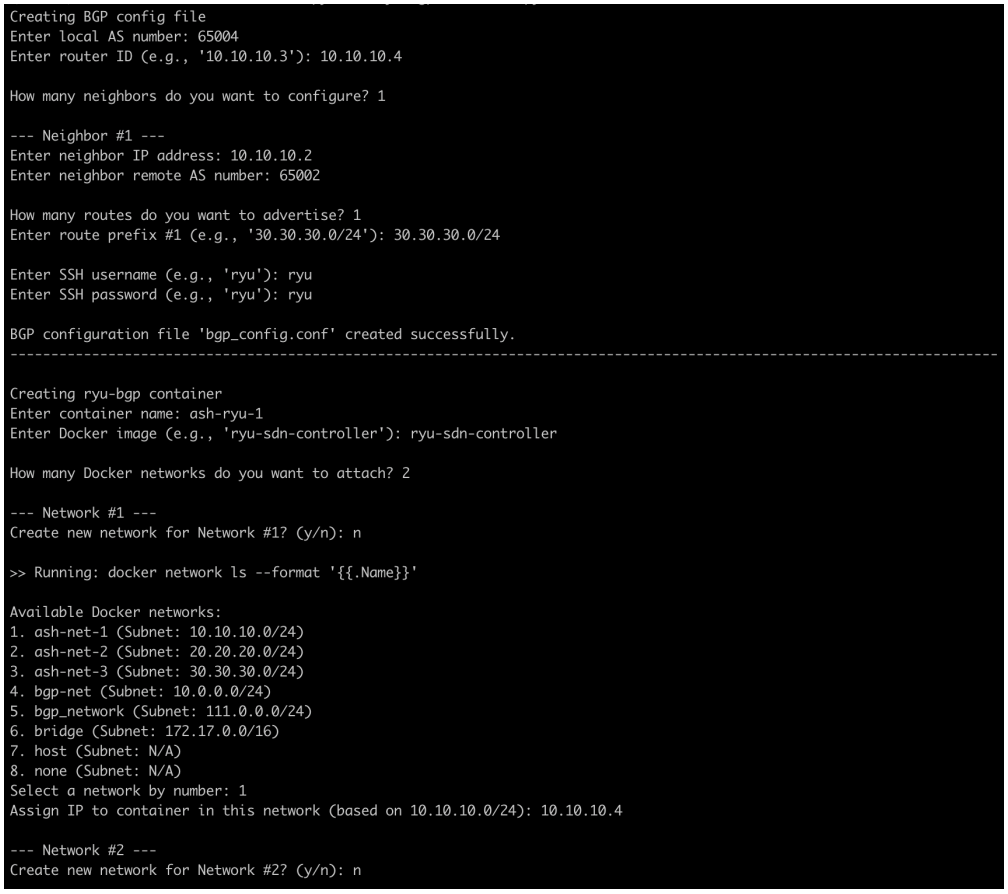
```
(newenv) stack@nvo-sneha:~/ash/lab9$ docker exec ash-frr-2 vtysh -c 'show ip bgp summary'

IPv4 Unicast Summary:
BGP router identifier 10.10.10.4, local AS number 65004 VRF default vrf-id 0
BGP table version 1
RIB entries 1, using 128 bytes of memory
Peers 1, using 24 KiB of memory

Neighbor        V         AS   MsgRcvd   MsgSent   TblVer  InQ OutQ  Up/Down State/PfxRcd   PfxSnt Desc
10.10.10.2      4      65002         7         8        1    0    0 00:01:32            1        1 N/A

Total number of neighbors 1
(newenv) stack@nvo-sneha:~/ash/lab9$ docker exec ash-frr-2 vtysh -c 'show ip route bgp'
Codes: K - kernel route, C - connected, L - local, S - static,
       R - RIP, O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, F - PBR,
       f - OpenFabric, t - Table-Direct,
       > - selected route, * - FIB route, q - queued, r - rejected, b - backup
       t - trapped, o - offload failure

IPv4 unicast VRF default:
B>* 20.20.20.0/24 [20/0] via 10.10.10.2, eth0, weight 1, 00:01:35
```

5) Automate spinning up and configuring an SDN controller as another Docker container.

   a) Automate its BGP speaker configuration to peer with Quagga/FRR.

Creating docker image:

Create RYU-BGP router:



```
>> Running: docker network ls --format '{{.Name}}'

Available Docker networks:
1. ash-net-1 (Subnet: 10.10.10.0/24)
2. ash-net-2 (Subnet: 20.20.20.0/24)
3. ash-net-3 (Subnet: 30.30.30.0/24)
4. bgp-net (Subnet: 10.0.0.0/24)
5. bgp_network (Subnet: 111.0.0.0/24)
6. bridge (Subnet: 172.17.0.0/16)
7. host (Subnet: N/A)
8. none (Subnet: N/A)
Select a network by number: 3
Assign IP to container in this network (based on 30.30.30.0/24): 30.30.30.2

Starting container...

>> Running: docker run -dit --privileged --name ash-ryu-1 --network ash-net-1 --ip 10.10.10.4 ryu-sdn-controller /bin/bash
68a5e36870fdba67b8237080510e6cffaae277c4cae2827fb957b53bf44e2a18

>> Running: docker network connect --ip 30.30.30.2 ash-net-3 ash-ryu-1

Docker container 'ash-ryu-1' created and started successfully.
--------------------------------------------------------------------------------------------------------------------------
Copying config file into the ryu-bgp container

>> Running: docker cp bgp_config.conf ash-ryu-1:/root/bgp_config.conf
Successfully copied 2.05kB to ash-ryu-1:/root/bgp_config.conf

BGP configuration file copied to the container successfully.
--------------------------------------------------------------------------------------------------------------------------
Running ryu-bgp app

Ryu BGP application started successfully in the background.
--------------------------------------------------------------------------------------------------------------------------
```

Testing:

Output of the ryu bgp app:

```
instantiating app /usr/local/lib/python3.9/site-packages/ryu/services/protocols/bgp/application.py of RyuBGPSpeaker
API method core.start called with args: {'waiter': <ryu.lib.hub.Event object at 0x7fdef4886e20>, 'local_as': 65004, 'router_id': '10.10.10.4', 'bgp_server_hosts': ('0.0.0.0', '::'), 'bgp_server_port': 179, 'refresh
_stalepath_time': 0, 'refresh_max_eor_time': 0, 'label_range': (100, 100000), 'allow_local_as_in_count': 0, 'cluster_id': None, 'local_pref': 100}
API method neighbor.create called with args: {'ip_address': '10.10.10.2', 'remote_as': 65002, 'remote_port': 179, 'peer_next_hop': None, 'password': None, 'is_route_server_client': False, 'is_route_reflector_client
': False, 'is_next_hop_self': False, 'connect_mode': 'both', 'cap_enhanced_refresh': False, 'cap_four_octet_as_number': True, 'cap_mbgp_ipv4': True, 'cap_mbgp_ipv6': False, 'cap_mbgp_vpnv4': False, 'cap_mbgp_vpnv6'
: False, 'cap_mbgp_evpn': False, 'cap_mbgp_ipv4fs': False, 'cap_mbgp_ipv6fs': False, 'cap_mbgp_vpnv4fs': False, 'cap_mbgp_vpnv6fs': False, 'cap_mbgp_l2vpnfs': False}
API method network.add called with args: {'prefix': '30.30.30.0/24'}
starting ssh server at 0.0.0.0:4990
Connection to peer: 10.10.10.2 established
```

ssh into bgpd to check the learnt routes:

```
root@68a5e36870fd:~# ssh ryu@localhost -p 4990
The authenticity of host '[localhost]:4990 ([127.0.0.1]:4990)' can't be established.
RSA key fingerprint is SHA256:rXrAaQzzQu7EZ0k5jVv4jAQ7Xyi6T8cDA4kudzG4CbY.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[localhost]:4990' (RSA) to the list of known hosts.

Hello, this is Ryu BGP speaker (version 4.34).

bgpd> show rib all
Status codes: * valid, > best
Origin codes: i - IGP, e - EGP, ? - incomplete
     Network                         Labels   Next Hop            Reason          Metric LocPrf Path
Family: ipv4
 *>  20.20.20.0/24                   None     10.10.10.2          Only Path       0             65002 i
 *>  30.30.30.0/24                   None     0.0.0.0             Local Origin                  i
 *                                   None     10.10.10.2                                        65002 65004 i
```

Deliverable:

Create a personal GitHub page that demonstrates the required functionality.

GitHub link: https://github.com/CUBoulder-Ashwin/Virtualized-Network-Automation