
flokinet-002 - Introduction to the Arista CLI, continued

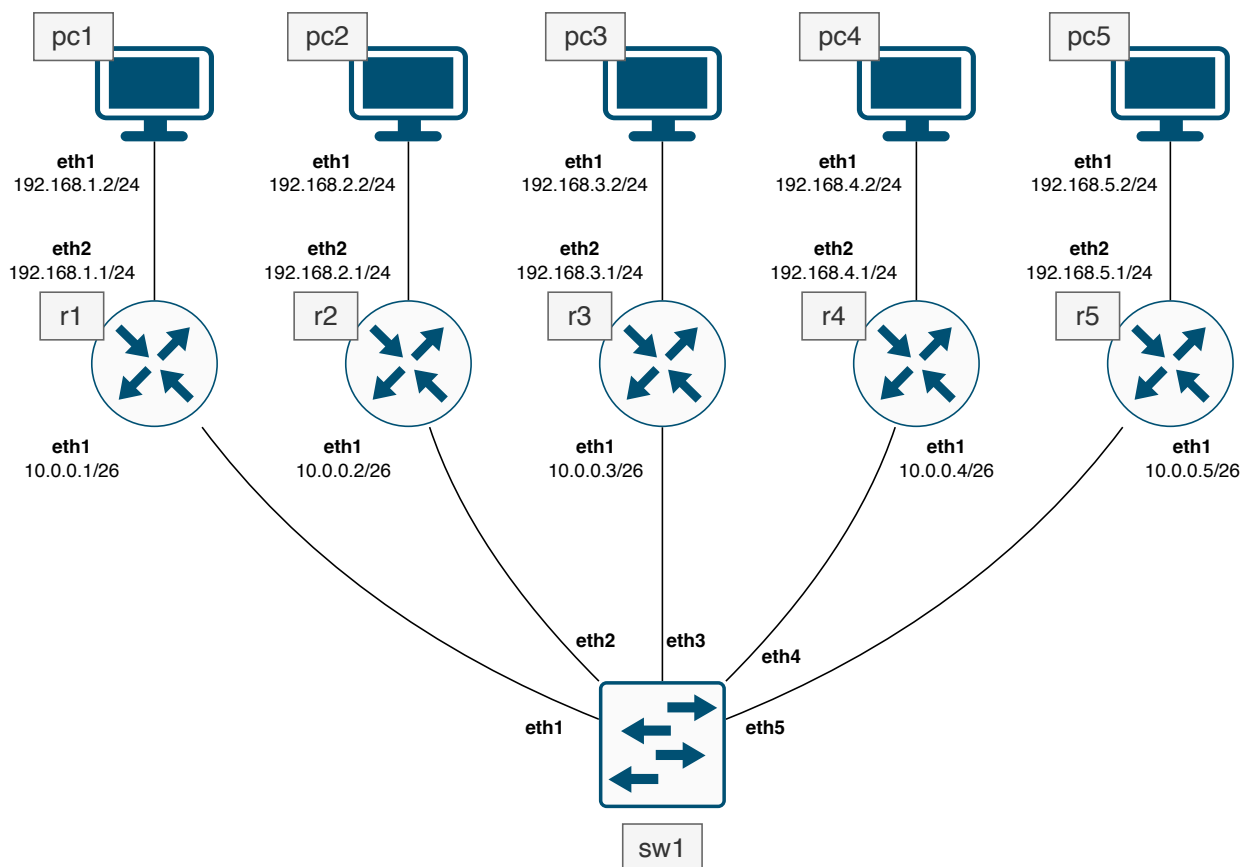
Nicholas Morrison nick@nanocat.net



Connecting to the lab server

- Open your favourite Terminal Emulator
- SSH to the netlab server:
 - `$ ssh-keygen -R netlab.nanocat.net` <- delete the cached fingerprint
(lab server rebuilt frequently)
 - `$ ssh lab@netlab.nanocat.net`
 - Password: (generated fresh each week)
- List the running containerlab devices:
 - `$ sudo containerlab inspect --all`
- Connect to an **Arista** device:
 - `$ sudo docker exec -it clab-device-name Cli`
- .. or connect to a **Linux** device:
 - `$ sudo docker exec -it clab-pcXX-name bash`

Diagram



The Final Goal

- **All PCs (pc1 through pc5) should be able to ping one another.**

- Work step by step!
- Ask questions
- Experiment! You can't break anything.

Step 0.5 - connect to your router

```
$ sudo containerlab inspect --all      <- show all running devices
$ sudo docker exec -it DEVICE-NAME Cli <- connect to your device
```

Step 1 - configure eth1

The first step is to configure the IP address on your router's eth1 interface.

Configure as follows:

```
rX#config                                <- enter configuration mode

rX(config)#interface eth1                <- enter the interface
                                           configuration context

rX(config-if-eth1)#show active           <- shows the configuration
                                           just for this context

rX(config-if-eth1)#description to Switch <- create a human-readable
                                           description for this interface

rX(config-if-eth1)#no switchport         <- make this a routed (L3) port
                                           (default is switched (L2) port)

rX(config-if-eth1)#ip address 10.0.0.x/24 <- refer to the diagram
rX(config-if-eth1)#show active
rX(config-if-eth1)#end                   <- exit configuration mode

rX#show ip interface brief               <- print a list of all interfaces
                                           with an IPv4 (L3) address
```

Step 1.5 - testing

Test by seeing if you can ping another router's IP address. You might need to ask your lab-mates whether they have already configured their devices.

```
rX#ping 10.0.0.X
...
```

Step 2 - configure eth2

This is the interface that connects to the PC.

This interface will serve as the **default gateway** or **gateway of last resort** for the PC.

Configure as follows:

```
rX#config
rX(config)#interface eth2
rX(config-if-eth2)#show active
rX(config-if-eth2)#description to PC
rX(config-if-eth2)#no switchport
rX(config-if-eth2)#ip address 192.168.X.1/24    <- refer to the diagram
rX(config-if-eth2)#show active
rX(config-if-eth2)#end                          <- fully exit configuration mode
rX#show ip interface brief                       <- print a list of all interfaces with
                                                a IPv4 (L3) address
...
```

Step 2.5 - testing

Test to see if you can ping your PC (192.168.X.2)

```
rX#ping 192.168.X.2
```

Step 3 - static routes

- The when the (layer 3) router receives a (layer 3) packet, it checks the destination IP address of the packet to decide which interface to forward it out of
- The router uses a **routing table** to make this decision.
- There are different ways for a routing table to become populated with data.
 - Static routes
 - Dynamic routes

Step 3.1 - configure the static routes

- **Configure** a static route:
 - rX#show ip route <- look at the routing table
 - ...
 - rX#configure
 - rX(config)#ip route 192.168.X.0/24 10.0.0.X <- when I have a packet that is destined for 192.168.X.X, forward it on to 10.0.0.x

```
rX(config)#end
rX#show ip route                <- take another look at the
                                routing table
...
```

- Repeat this for all of the remote networks.

Step 3.5 - testing

Test by pinging a remote PC from your own PC. You might have to check with the other lab members to make sure they're up and running.

```
pcX$ ping 192.168.X.2
```

You can also try a traceroute:

```
pcX$ traceroute -n 192.168.X.2  <- the -n tells traceroute not to try to resolve
                                the IP addresses into names
...
```

Things to try

- Run a tcpdump on your router:
 - rX#bash
 - \$ tcpdump -i eth1 icmp <- listen for traffic on eth1, and only print ICMP (ping) packets
 - ...
 - (press ctrl-c to terminate tcpdump)
- Watch what happens when you send a successful ping
- .. or an unsuccessful ping
- .. or a ping to an unknown network

Questions

- Are the routers in the same network, or different networks?
- Which devices are in the same network as the PCs?