SGNOG
Hackathon 2019
What is a Hackathon?

- Group of people (mostly strangers) collaborating and working intensive together to fix some problem(s), pain points.
  - New ideas, features
  - Manual & repetitive tasks
- Coding, scripting, programming, software development
- Creating solution (prototype) in the end
What actually happened?

- Coding Workshop
- Lab work (setting up shell environment, created some virtual routers)
- Hack starts!
- `while (hack)`
  {
    eat();
    code();
  }
What they did?

- Hear from them!
- 2 examples of the hack ideas to be shared
More network automation engineers

More scripts

More redundant access to the device layer

Faulty code might not close "ssh" sessions

Redundant development effort

ABSTRACTION?
Solution: Network State DB

```
{
  "host1": {
    "bgp": {
      "enabled": bool,
      "bgp": [peer1, peer2, peer3,...],
      peer1: {"local_asn": local_asn, "remote_asn": neigh_as, "state": state_prfxrcd},
      peer2: {"local_asn": local_asn, "remote_asn": neigh_as, "state": state_prfxrcd},
      peer_count: int,
      not_prfx_count: int
    },
    "interface": {
      interfaces: [interface1, interface2, ...]
      interface1: {"desc": description, "status": link_status}
    },
    "lldp": {
      enabled: bool,
      lldp: [local_if1, local_if2, ...]
    },
    "ntp": {
      enabled: bool,
      ntp: [server_name1, server_name2, ...]
    }
  }
}
```
Architecture:

1. SNMP TRAP
2. Change Notification
3. GET <Dev X>
4. Poll info(Netmiko)
5. Send Info(Netmiko)
6. Send Info(JSON)
7. POST <Dev X>

Device Plane

Dev 1

Dev 2

Dev 3

Device Controller

SNMP Server

API Server

Redis Store

REST API
Demo:

CSR5#configure
Enter configuration commands: Continue with Enter command.
CSR5(config)#int g12
CSR5(config-if)#no shut
CSR5(config-if)#

INFO:__main__:Change Detected: Received stateDB update request for: CSR5
INFO:__main__:Updating StateDB for CSR5
Problem Statement - BGP Prefix-Set Modification via Python Script

Customer’s request to update their prefix-set is being done manually through these tedious, repetitive tasks:

- Given the customer Service ID, look for the valid BGP neighbor
- From the BGP configuration, look for applied route policy name
- With the route policy name, derive the prefix-set name
- Update the policy to add/remove the prefix
- Validate the change

Manual configuration of these BGP policies are more prone to human-error
Solution:

Packages Used: netmiko, ciscoconfparse, ipaddress

Source: https://github.com/shhackathon/sgnog19
1. Prefix-set config before change:

```
RP/0/0/CPU0:xrv#show rpl prefix-set Service01_prefix
Thu Jul 11 05:05:41.585 UTC
prefix-set Service01_prefix
  1.1.1.4/22,
  1.1.1.24,
  1.1.1.49/24,
  1.1.1.50/24,
  1.1.1.32/22,
  104.2.0.0/16,
  102.3.0.0/16,
  11.1.10.3/22,
  192.168.0.0/24,
  10.0.0.0/24
end set
```

2. Script Output:

```
edit prefix-set Service01_prefix inline add "15.24.3.0/24"
Thu Jul 11 05:00:41.485 UTC
Proceed with commit (yes/no)? [yes]:
Parsing.
226 bytes parsed in 1 sec (221 bytes/sec)
Committing.
Prepared commit in 0 sec
1 items committed in 2 sec (0 items/sec)
Updating.
Updated commit database in 1 sec
DONE - Prefix 15.24.3.0/24 has been added to Service01_prefix successfully
Commit successfully commit to new set() Diff from New to Old {'15.24.3.0/24'}
```

3. Prefix-set config after change:

```
RP/0/0/CPU0:xrv#show rpl prefix-set Service01_prefix
Thu Jul 11 05:09:25.888 UTC
prefix-set Service01_prefix
  1.1.1.4/22,
  1.1.1.24,
  1.1.1.49/24,
  1.1.1.50/24,
  1.1.1.32/22,
  104.2.0.0/16,
  102.3.0.0/16,
  11.1.10.3/22,
  192.168.0.0/24,
  10.0.0.0/24
```

Demo
Key Takeaways

- Teamwork
- Agility
- Open Source
- Great Food
Questions, Suggestions?

#Fin.