CSCI5840 PROGRESS PRESENTATION

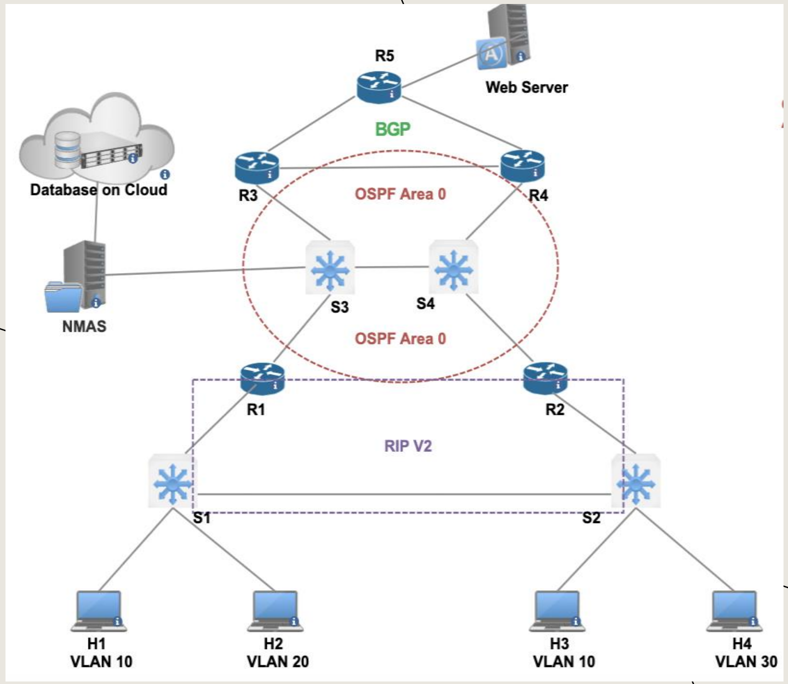
AGENDA

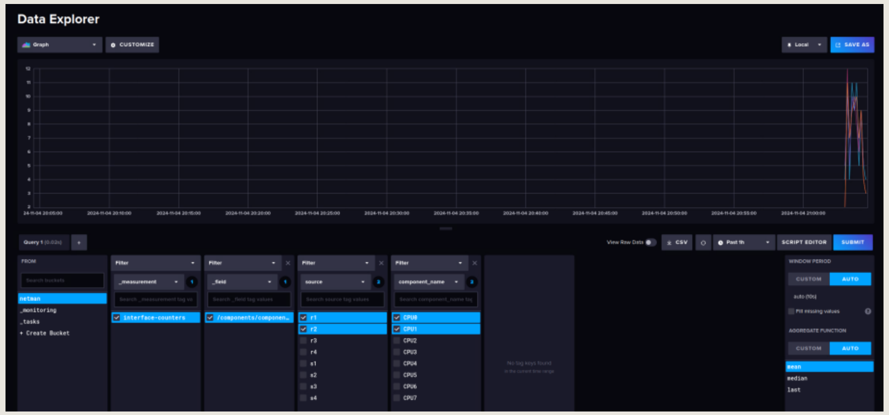
Infrastructure and Visualization

IaC

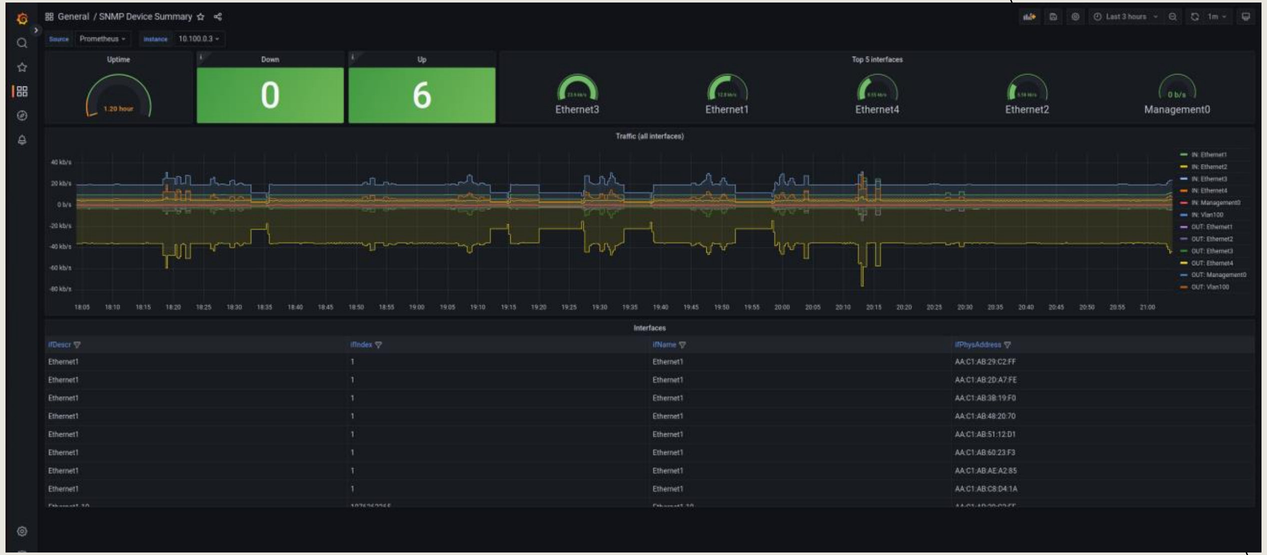
Unit Testing/ Code Coverage

Demos



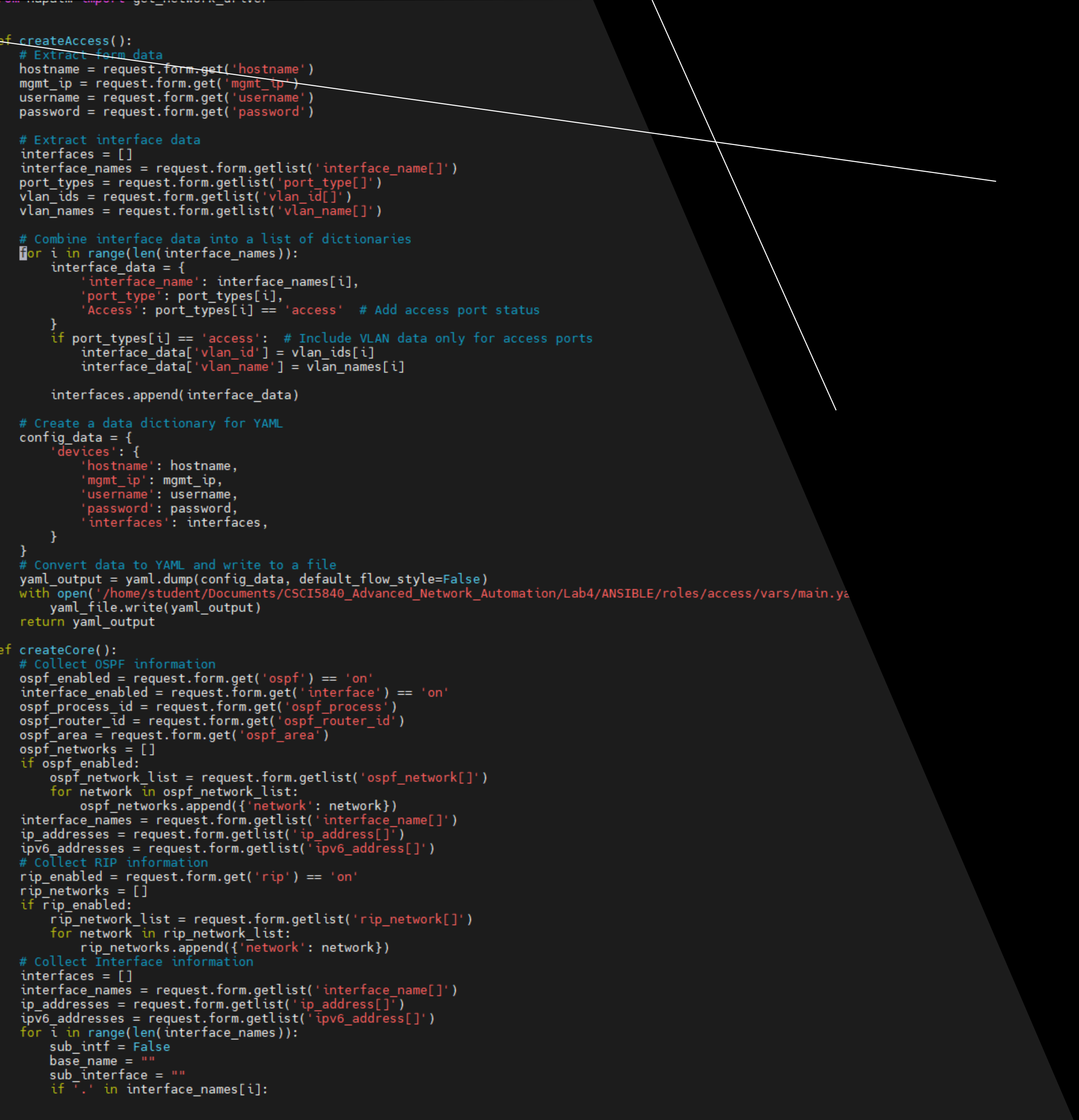


INFRASTRUCTURE AND VISUALIZATION



Interfaces

ifDescr	ifIndex	ifName	ifPhysAddress
Ethernet1	1	Ethernet1	AA:C1:AB:29:C2:FF
Ethernet1	1	Ethernet1	AA:C1:AB:2D:A7:FE
Ethernet1	1	Ethernet1	AA:C1:AB:3B:19:F0
Ethernet1	1	Ethernet1	AA:C1:AB:48:20:70
Ethernet1	1	Ethernet1	AA:C1:AB:51:12:D1
Ethernet1	1	Ethernet1	AA:C1:AB:60:23:F3
Ethernet1	1	Ethernet1	AA:C1:AB:AE:A2:85
Ethernet1	1	Ethernet1	AA:C1:AB:C8:D4:1A



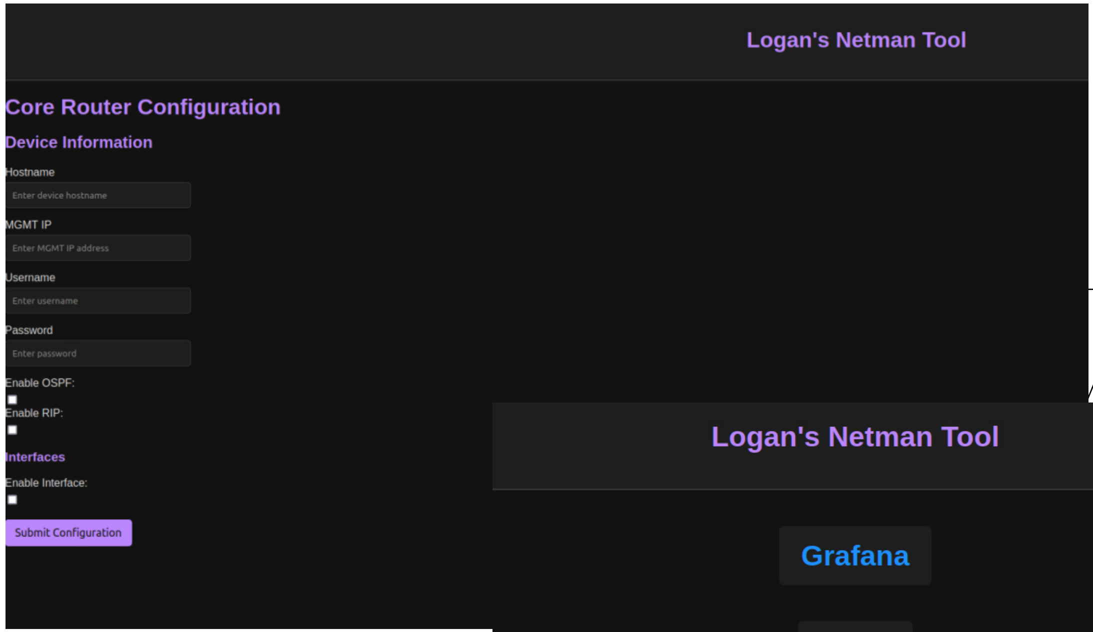
```
om napalm import get_network_driver


ef createAccess():
   # Extract form data
   hostname = request.form.get('hostname')
   mgmt_ip = request.form.get('mgmt_ip')
   username = request.form.get('username')
   password = request.form.get('password')

   # Extract interface data
   interfaces = []
   interface_names = request.form.getlist('interface_name[]')
   port_types = request.form.getlist('port_type[]')
   vlan_ids = request.form.getlist('vlan_id[]')
   vlan_names = request.form.getlist('vlan_name[]')

   # Combine interface data into a list of dictionaries
   for i in range(len(interface_names)):
       interface_data = {
           'interface_name': interface_names[i],
           'port_type': port_types[i],
           'Access': port_types[i] == 'access'  # Add access port status
       }
       if port_types[i] == 'access':  # Include VLAN data only for access ports
           interface_data['vlan_id'] = vlan_ids[i]
           interface_data['vlan_name'] = vlan_names[i]

       interfaces.append(interface_data)

   # Create a data dictionary for YAML
   config_data = {
       'devices': {
           'hostname': hostname,
           'mgmt_ip': mgmt_ip,
           'username': username,
           'password': password,
           'interfaces': interfaces,
       }
   }
   # Convert data to YAML and write to a file
   yaml_output = yaml.dump(config_data, default_flow_style=False)
   with open('/home/student/Documents/CSCI5840_Advanced_Network_Automation/Lab4/ANSIBLE/roles/access/vars/main.ya
       yaml_file.write(yaml_output)
   return yaml_output

ef createCore():
   # Collect OSPF information
   ospf_enabled = request.form.get('ospf') == 'on'
   interface_enabled = request.form.get('interface') == 'on'
   ospf_process_id = request.form.get('ospf_process')
   ospf_router_id = request.form.get('ospf_router_id')
   ospf_area = request.form.get('ospf_area')
   ospf_networks = []
   if ospf_enabled:
       ospf_network_list = request.form.getlist('ospf_network[]')
       for network in ospf_network_list:
           ospf_networks.append({'network': network})
   interface_names = request.form.getlist('interface_name[]')
   ip_addresses = request.form.getlist('ip_address[]')
   ipv6_addresses = request.form.getlist('ipv6_address[]')
   # Collect RIP information
   rip_enabled = request.form.get('rip') == 'on'
   rip_networks = []
   if rip_enabled:
       rip_network_list = request.form.getlist('rip_network[]')
       for network in rip_network_list:
           rip_networks.append({'network': network})
   # Collect Interface information
   interfaces = []
   interface_names = request.form.getlist('interface_name[]')
   ip_addresses = request.form.getlist('ip_address[]')
   ipv6_addresses = request.form.getlist('ipv6_address[]')
   for i in range(len(interface_names)):
       sub_intf = False
       base_name = ""
       sub_interface = ""
       if '.' in interface_names[i]:
```

IAC

IAC

Features:

- Get Golden Configs
- YAML -> Ansible -> J2 Auto configuration
- Fully implemented GUI (with dark mode)
- GitHub Change Management
- IPAM via NetBox
- Password Change System
- Health Checks



Logan's Netman Tool

Core Router Configuration

Device Information

Hostname

Enter device hostname

MGMT IP

Enter MGMT IP address

Username

Enter username

Password

Enter password

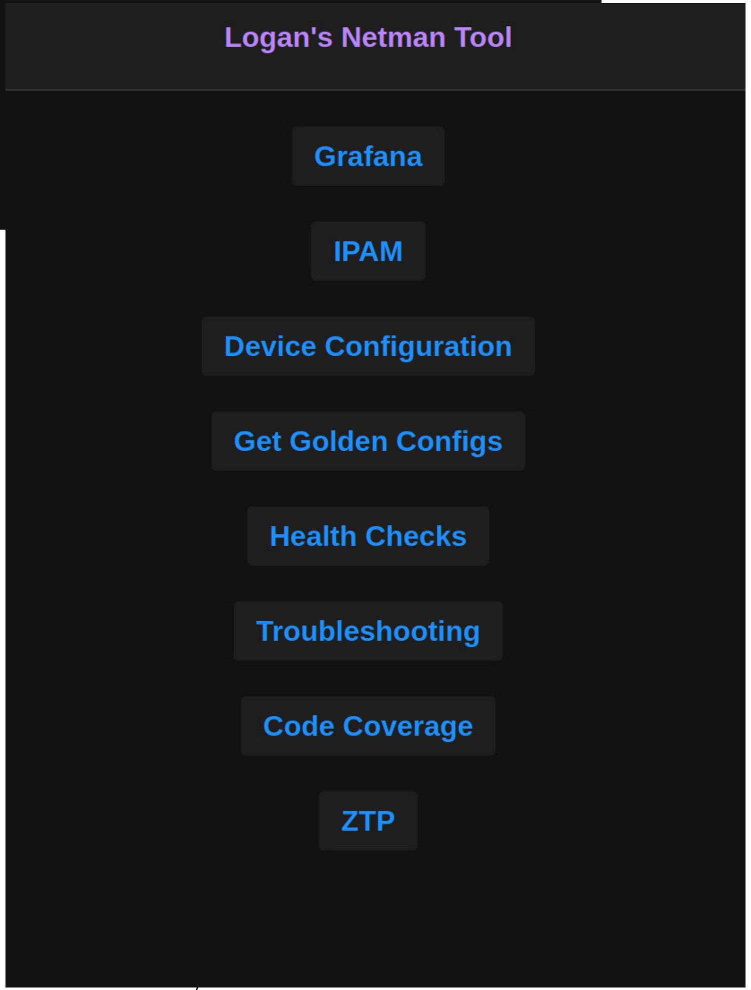
Enable OSPF:

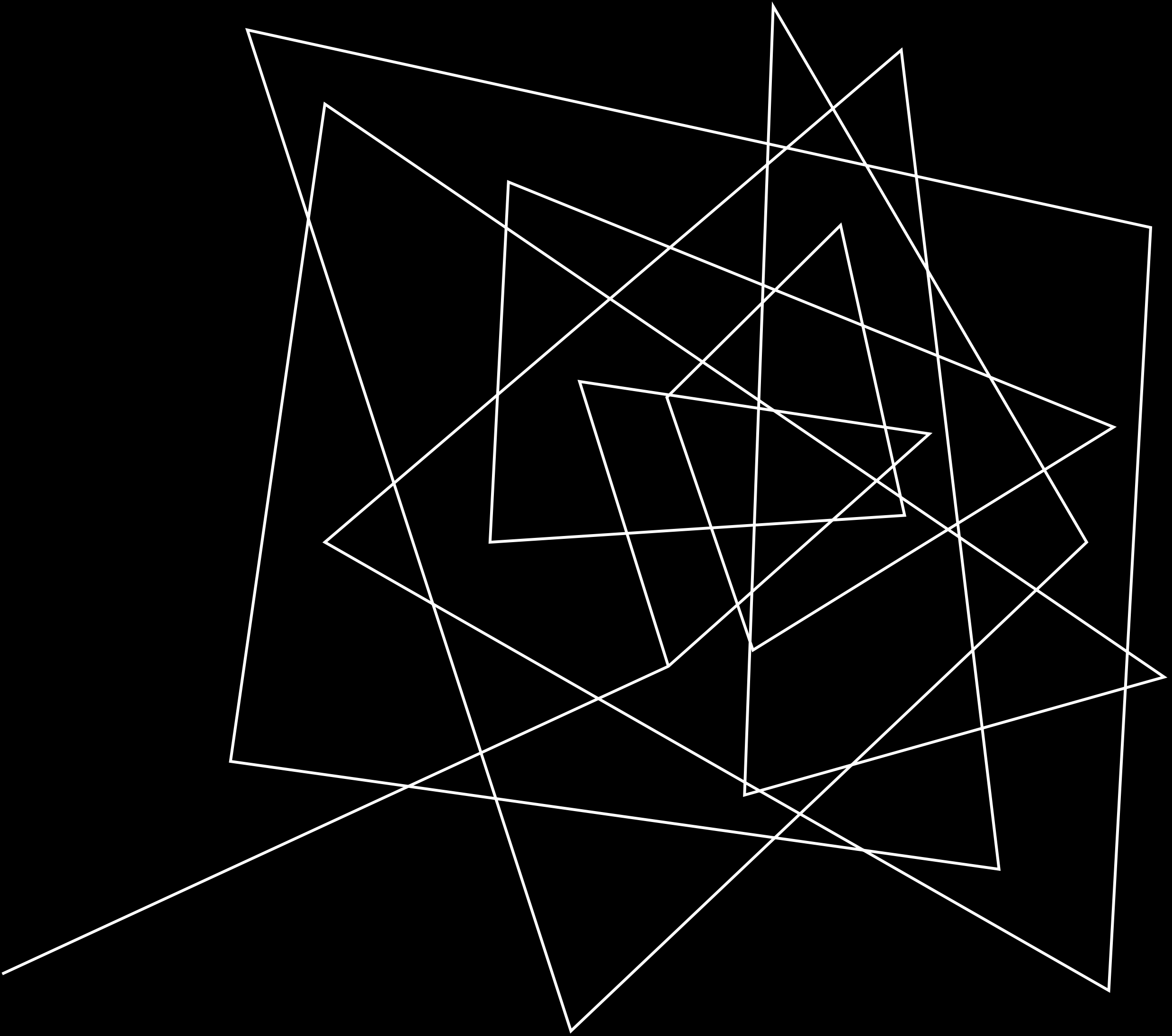
Enable RIP:

Interfaces

Enable Interface:

Submit Configuration





UNIT TESTING / CODE COVERAGE

FEATURES OF THE TOOLS

Unit Testing: GitHub Actions

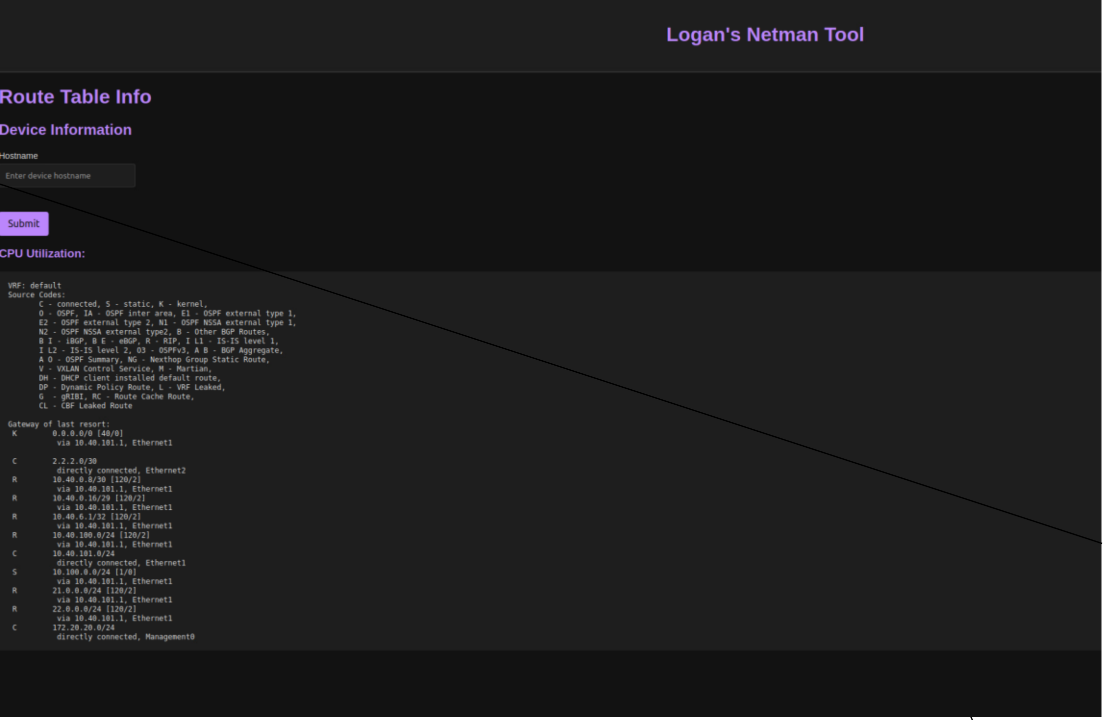
Utilizes a local GitHub Runner to execute code checks like:

- IP Connectivity
- Password Validation
- Expected IaC output
- Ping checks

Code Coverage: IaC automation

Analyzes all IaC and dynamically calculates code coverage %s:

- Calculates for every code and config change
- Displays nicely on GUI


Logan's Netman Tool

Route Table Info

Device Information

Hostname

Enter device hostname

Submit

CPU Utilization:

```
VRF: default
Source Codes:
       C - connected, S - static, K - kernel,
       O - OSPF, IA - OSPF inter area, E1 - OSPF external type 1,
       E2 - OSPF external type 2, N1 - OSPF NSSA external type 1,
       N2 - OSPF NSSA external type2, B - Other BGP Routes,
       B I - iBGP, B E - eBGP, R - RIP, I L1 - IS-IS level 1,
       I L2 - IS-IS level 2, O3 - OSPFv3, A B - BGP Aggregate,
       A O - OSPF Summary, NG - Nexthop Group Static Route,
       V - VXLAN Control Service, M - Martian,
       DH - DHCP client installed default route,
       DP - Dynamic Policy Route, L - VRF Leaked,
       G  - gRIBI, RC - Route Cache Route,
       CL - CBF Leaked Route

Gateway of last resort:
 K        0.0.0.0/0 [40/0]
           via 10.40.101.1, Ethernet1

 C        2.2.2.0/30
           directly connected, Ethernet2
 R        10.40.0.8/30 [120/2]
           via 10.40.101.1, Ethernet1
 R        10.40.0.16/29 [120/2]
           via 10.40.101.1, Ethernet1
 R        10.40.6.1/32 [120/2]
           via 10.40.101.1, Ethernet1
 R        10.40.100.0/24 [120/2]
           via 10.40.101.1, Ethernet1
 C        10.40.101.0/24
           directly connected, Ethernet1
 S        10.100.0.0/24 [1/0]
           via 10.40.101.1, Ethernet1
 R        21.0.0.0/24 [120/2]
           via 10.40.101.1, Ethernet1
 R        22.0.0.0/24 [120/2]
           via 10.40.101.1, Ethernet1
 C        172.20.20.0/24
           directly connected, Management0
```

DEMOS

THANK YOU

Logan Chayet

303-Robo-Networks

logan@robocontrolnetworks.com

www.robocontrolnetworks.com