Lab 5 Guide:

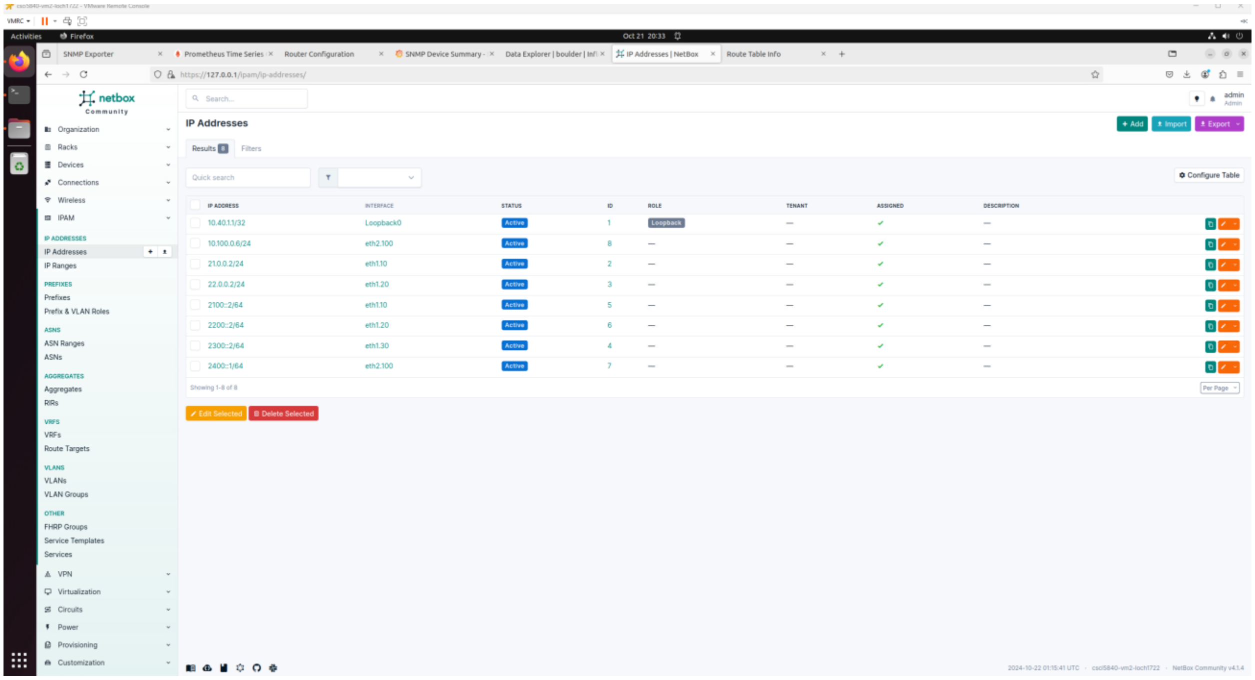
Objective 1: Building the Automation Framework for IaC

IP Address Management (IPAM)

I used Netbox for the IPAM utility. To install this, I followed step by step on the documentation website: https://netboxlabs.com/docs/netbox/en/stable/installation/

Specifically, I used Gunicorn instead of uWSGI as it is ok to use on or the other.

Here's a screenshot of NetBox running:



IP Addresses

IP ADDRESS	INTERFACE	STATUS	ID	ROLE	TENANT	ASSIGNED	DESCRIPTION
10.40.1.1/32	Loopback0	Active	1	Loopback	—	✓	—
10.100.0.6/24	eth2.100	Active	8	—	—	✓	—
21.0.0.2/24	eth1.10	Active	2	—	—	✓	—
22.0.0.2/24	eth1.20	Active	3	—	—	✓	—
2100::2/64	eth1.10	Active	5	—	—	✓	—
2200::2/64	eth1.20	Active	6	—	—	✓	—
2300::2/64	eth1.30	Active	4	—	—	✓	—
2400::1/64	eth2.100	Active	7	—	—	✓	—

Showing 1-8 of 8

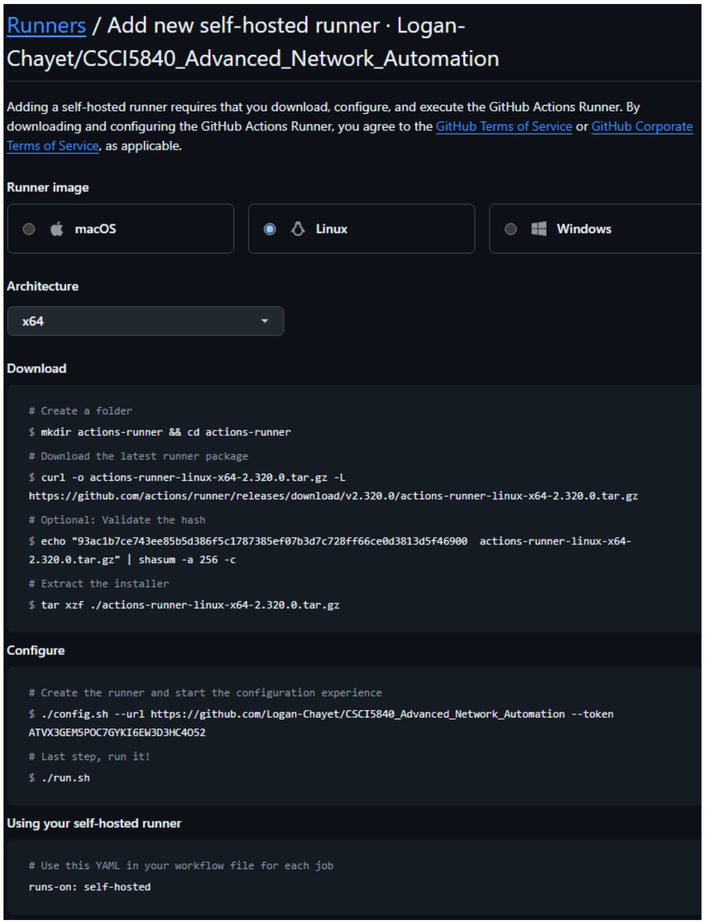
Permissions

For permissions, I have a passwords.py that utilizes a pass_file.csv with device information that creates a random username and password for each device, updates all the device configs, and then writes the new information to the csv. To be changed on a regular basis, I have a cron job running that executes this py file every week. Explanation and code shown in the video.

Objective 2: Building the DevOps Pipeline

For the DevOps Pipeline, I used GitHub Actions.

First, I created a runner that would run locally on the VM:



Runners / Add new self-hosted runner · Logan-Chayet/CSCI5840_Advanced_Network_Automation

Adding a self-hosted runner requires that you download, configure, and execute the GitHub Actions Runner. By downloading and configuring the GitHub Actions Runner, you agree to the GitHub Terms of Service or GitHub Corporate Terms of Service, as applicable.

Runner image

○ macOS ● Linux ○ Windows

Architecture

x64

Download

```
# Create a folder
$ mkdir actions-runner && cd actions-runner
# Download the latest runner package
$ curl -o actions-runner-linux-x64-2.320.0.tar.gz -L
https://github.com/actions/runner/releases/download/v2.320.0/actions-runner-linux-x64-2.320.0.tar.gz
# Optional: Validate the hash
$ echo "93ac1b7ce743ee85b5d386f5c1787385ef07b3d7c728ff66ce0d3813d5f46900  actions-runner-linux-x64-
2.320.0.tar.gz" | shasum -a 256 -c
# Extract the installer
$ tar xzf ./actions-runner-linux-x64-2.320.0.tar.gz
```

Configure

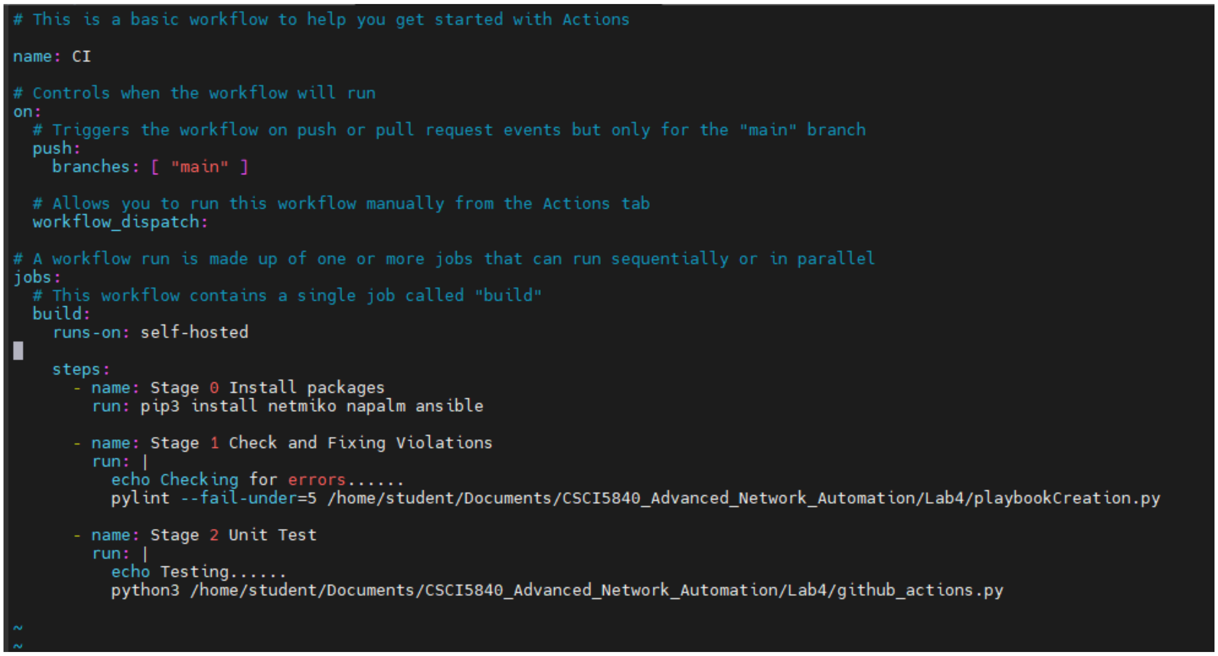
```
# Create the runner and start the configuration experience
$ ./config.sh --url https://github.com/Logan-Chayet/CSCI5840_Advanced_Network_Automation --token
ATVX3GEM5POC7GYKI6EW3D3HC4O52
# Last step, run it!
$ ./run.sh
```

Using your self-hosted runner

```
# Use this YAML in your workflow file for each job
runs-on: self-hosted
```

After these commands, I then went into the same directory and ran sudo svc.sh install to create this runner as a service.

I then created a .yml file that checks for a host of things:



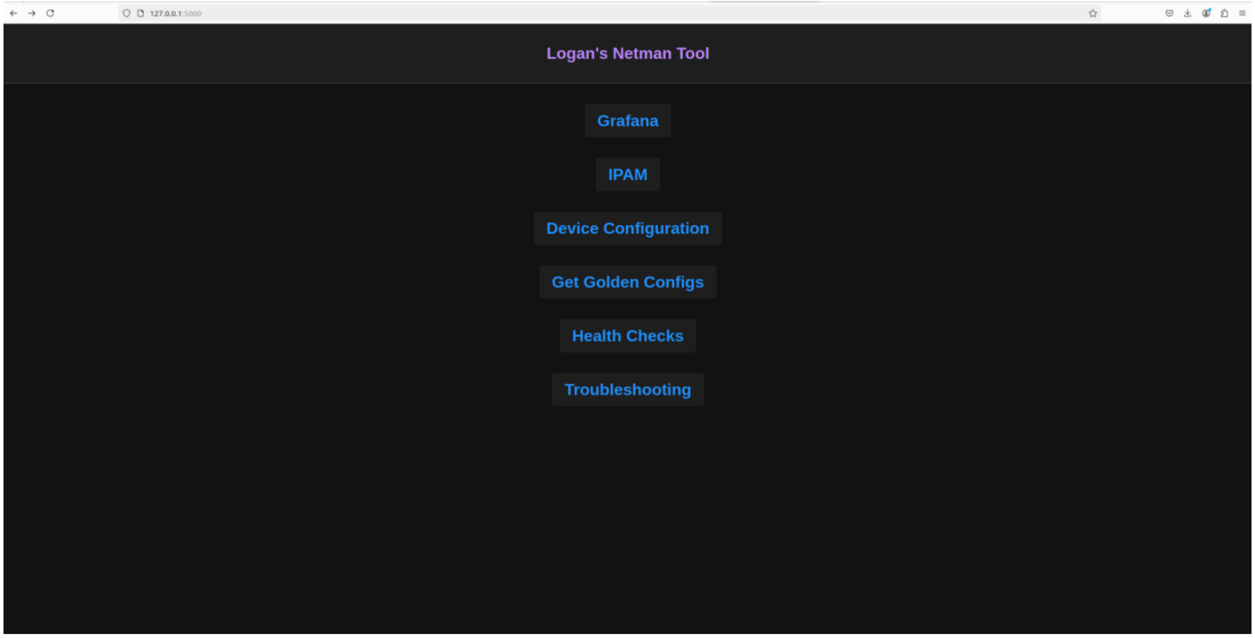
```
# This is a basic workflow to help you get started with Actions

name: CI

# Controls when the workflow will run
on:
  # Triggers the workflow on push or pull request events but only for the "main" branch
  push:
    branches: [ "main" ]

  # Allows you to run this workflow manually from the Actions tab
  workflow_dispatch:

# A workflow run is made up of one or more jobs that can run sequentially or in parallel
jobs:
  # This workflow contains a single job called "build"
  build:
    runs-on: self-hosted

    steps:
      - name: Stage 0 Install packages
        run: pip3 install netmiko napalm ansible

      - name: Stage 1 Check and Fixing Violations
        run: |
          echo Checking for errors......
          pylint --fail-under=5 /home/student/Documents/CSCI5840_Advanced_Network_Automation/Lab4/playbookCreation.py

      - name: Stage 2 Unit Test
        run: |
          echo Testing......
          python3 /home/student/Documents/CSCI5840_Advanced_Network_Automation/Lab4/github_actions.py

~
~
```

First, it checks for required pip3 libraries to make sure they are installed. Then I use pylint on my main playbookCreation.py file that contains all backend code for my website. Finally, I have a python testing framework in github_actions.py that checks for whether the loopback for R1 is 10.40.1.1/32, makes sure that R2 has only one area (area 0), and does a ping test of R1 to the Web Server.

Objective 3: Network Automation

1. Develop the necessary Network Automation Applications

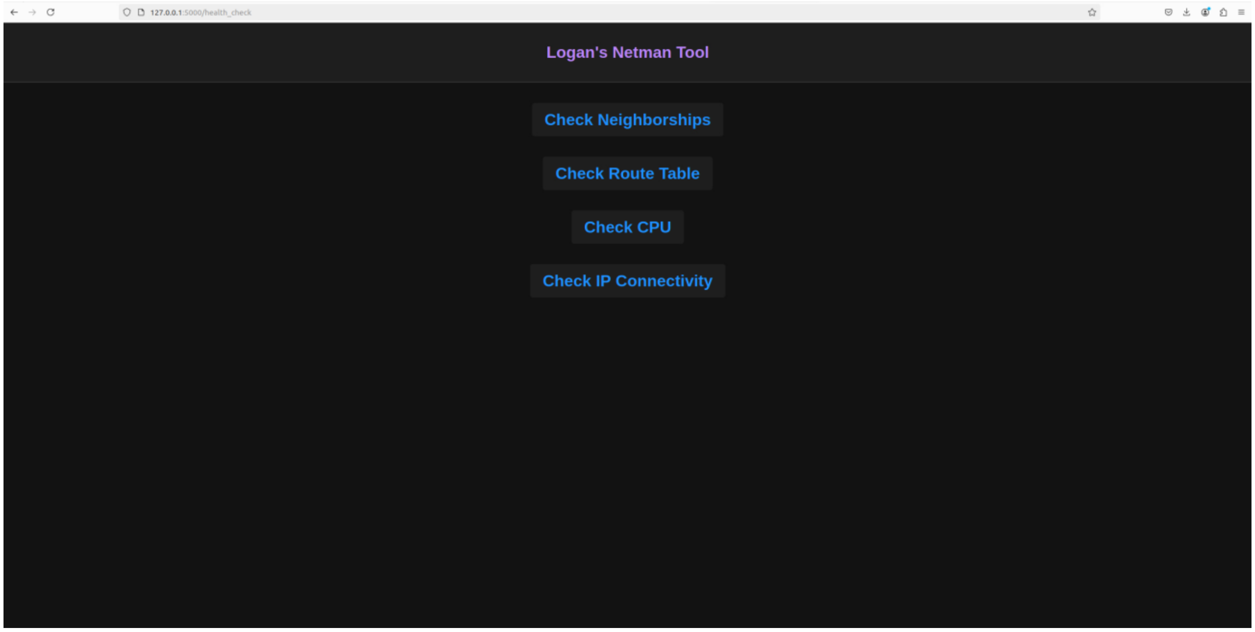
Did a complete UI overhaul of my website. New menu with new options: (IPAM, Health Check, Troubleshooting):



IPAM:

Explained in objective 1.

Health Checks:



Now contains health checks for all things shown above. Checks information for these things on any device in the network with the click of a button.

Troubleshooting:

*To be added in a future lab.