Network Management and Automation

Lab 3
Automation I

University of Colorado Boulder
Interdisciplinary Telecom Program

Professor Levi Perigo, Ph.D.

Summary

Organizations today look for network automation capabilities in their hardware and software. In its simplest form, network automation is aimed to reduce manual effort in routine tasks and automate what used to be typed by network administrators into command line interfaces (CLIs). Network automation can start with scripting and progress to intelligent network control, and ultimately allow efficient translation and deployment of network plans and policies.

If interested, please feel free to contribute to the below Open Source communities.

- Netmiko Source Code: https://github.com/ktbyers/netmiko
- NAPALM Source Code: https://github.com/napalm-automation/napalm
- Paramiko Source Code: https://github.com/paramiko/paramiko
- Pexpect Source Code: https://github.com/pexpect/pexpect

Objectives

- Learn how to remotely access network devices.
- Learn how to parse nested dictionaries and automate device configuration without using CLI.
- Learn how to perform output scraping.
- Learn how to write unit test cases

Problem Statement:

You are a Network Engineer assigned with the task of configuring a hundred new devices in your organization's data center network. The man-hours required for manually configuring a network of this scale will be upwards of 300 hours (if you are being optimistic). Your manager has asked you to come up with a more efficient, less time-consuming approach. You have a week to demonstrate your method on a smaller topology, as a proof of concept (POC).

The majority of data center networks have started using Border Gateway Protocol (BGP) due to several performance advantages over other protocols. For the given topology (**Figure 1**), automate the process of configuring a BGP session between the routers. As the routers are in the same Autonomous System, they will run internal BGP (iBGP) between them.

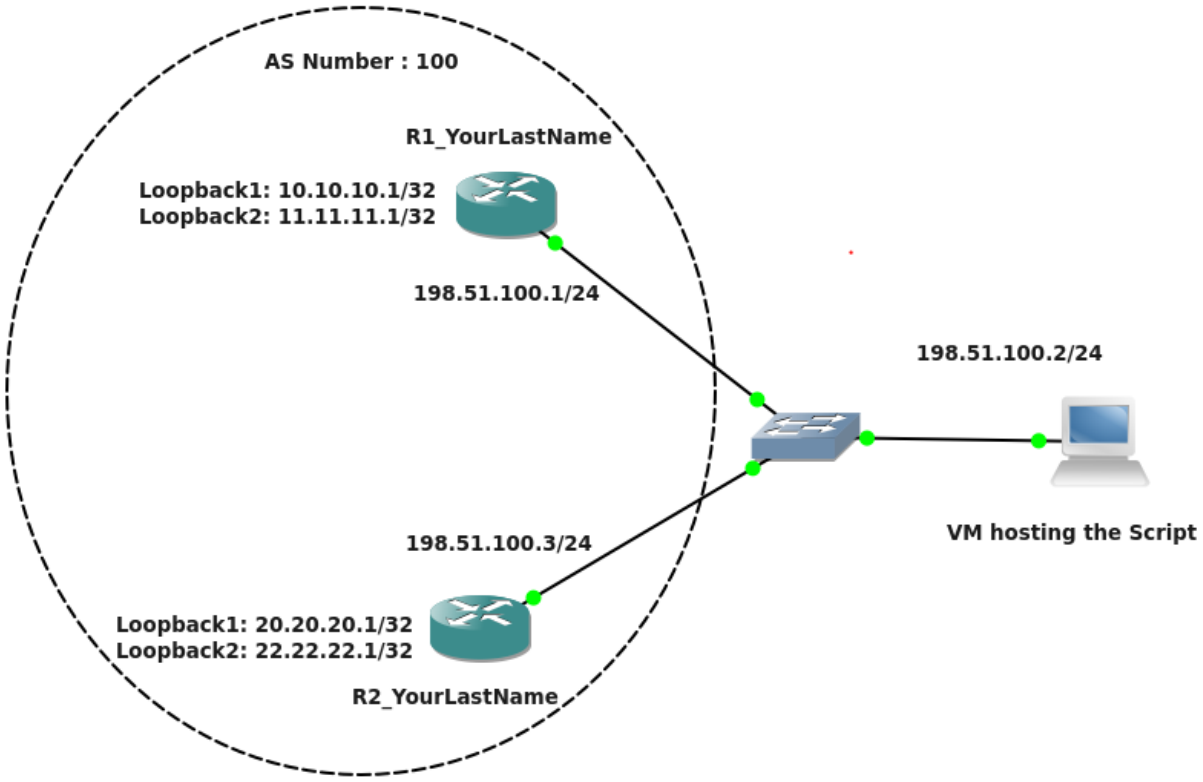


Figure 1

Guidelines:

1. Create the above topology in GNS3 on the NetMan VM. Manually configure the loopbacks and interface IPs on the routers. Follow the IP addressing scheme as mentioned in the above topology.
2. Before writing the code, ensure that SSHv2 is enabled on the routers. Refer: https://www.pluralsight.com/blog/tutorials/configure-secure-shell-ssh-on-cisco-router
3. Avoid hard coding IPs and credentials in your code; use config files wherever required.
4. Use **Netmiko** module for SSH. Also, use functions/classes for clarity of code.
5. While implementing the code, think about "**parallel processing**". The configuration should be deployed on all the devices at the same time. [**20 Points**]

Code Requirements:

For this lab you will have to create modules (.py files) for each of the four Python files (sshInfo.py, validateIP.py, connectivity.py, bgp.py) you write, which can be imported into a lab3main.py file. So, your lab3main.py file should start with –

import sshInfo.py, validateIP.py, connectivity.py, bgp.py

sshInfo.py:

- Create a CSV/JSON file named "sshInfo.csv/json" that will be parsed by your code. The file should include the login credentials (username and password) and the IPs used to SSH into the two routers. Error handling should be performed to check if the file exists. [**10 Points**]

validateIP.py:

- Check that the extracted IPs are valid IP addresses. (Examples of invalid IPs: 10.10.300.10, 192.168.1, etc.) [**5 Points**]

connectivity.py:

- Check if you can reach the parsed SSH IPs from the VM (successful pings) from the code. [**10 Points**]

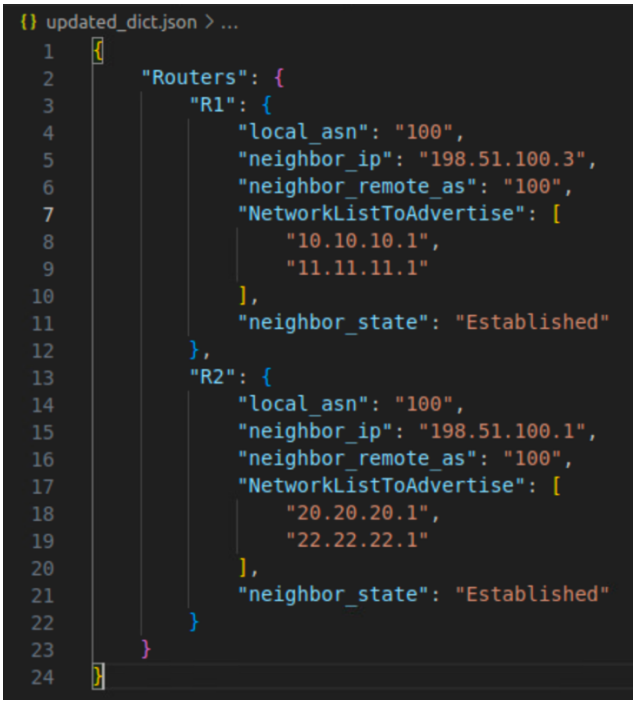
bgp.py:

- SSH into the routers using Netmiko/NAPALM. [**15 Points**]
- Create a “bgp.conf” file with structure and content of your choice. This file could include essential information required to configure iBGP. For example:

  Routerinfo = { “Routers” : { “R1” : { “local_asn”: “100”, “neighbor_ip”: “198.51.100.3”, “neighbor_remote_as”: “100”, “NetworkListToAdvertise” : [x.x.x.x, y.y.y.y] }, “R2” : { “local_asn”: “100”, “neighbor_ip”: “198.51.100.3”, “neighbor_remote_as”: “100”, “NetworkListToAdvertise” : [x.x.x.x, y.y.y.y] }}

  Note: Please use the format mentioned above. You can add any other parameters to the dictionary but the format should be of a nested dictionary.
  The loopback addresses should be included in the advertised networks list. [**15 Points**]
- Using information parsed from the “bgp.conf” file that you created, configure iBGP between the routers on their physical interfaces (198.51.100.0/24). [**25 Points**]

  (You could refer the following link for the configuration commands: https://networklessons.com/bgp/internal-bgp-border-gateway-protocol-explained/)
- While configuring the routers, handle any errors in case a wrong command is entered on the router CLI and throw an exception with a proper error message. [**15 points**]
- Once iBGP is configured, update the dictionary from above with the current BGP Neighbor state for each router(Paste screenshot of updated dictionary) [**10 points]**



- check the output of the "**show ip bgp neighbors**" commands on both routers, and parse the output to display important information in the format as shown below: [**10 Points**]

  R1:

BGP Neighbor IP	BGP Neighbor AS	BGP Neighbor State
198.51.100.3	100	Established

- Print a list of all the BGP routes only(Include only the BGP routes in the output) **[10 points]**
- Finally, retrieve the complete running-config from the routers and save them locally. Include print statements to display the names of the saved files. [**5 Points**]

Extra Credits:[25 points]

Use any unit testing framework to write test cases for

i) ValidateIP.py : cover all possible scenarios when an IP address would not be valid [**10 points]**

ii) Bgp.py : check if the running-config file is as per the required BGP config. Check for any errors (bad config) in the final running-config file. **[15 points]**

Report Question:

Go through this article: http://money.cnn.com/2017/03/02/technology/amazon-s3-outage-human-error/index.html

Do you think the objectives of this lab can help avoid such incidents? Why/why not? [**5 Points**]

Yes, the objectives of this lab can help avoid incidents by reducing human error through automation. By using automation tools like Netmiko to automate configuration pushes, we can minimize the risk of typos and inconsistencies that often occur with manual configuration. If the configurations are tested and verified to work on a small scale, they should be reliable for production devices, thereby reducing the likelihood of incidents.

Total Points _____ / 155