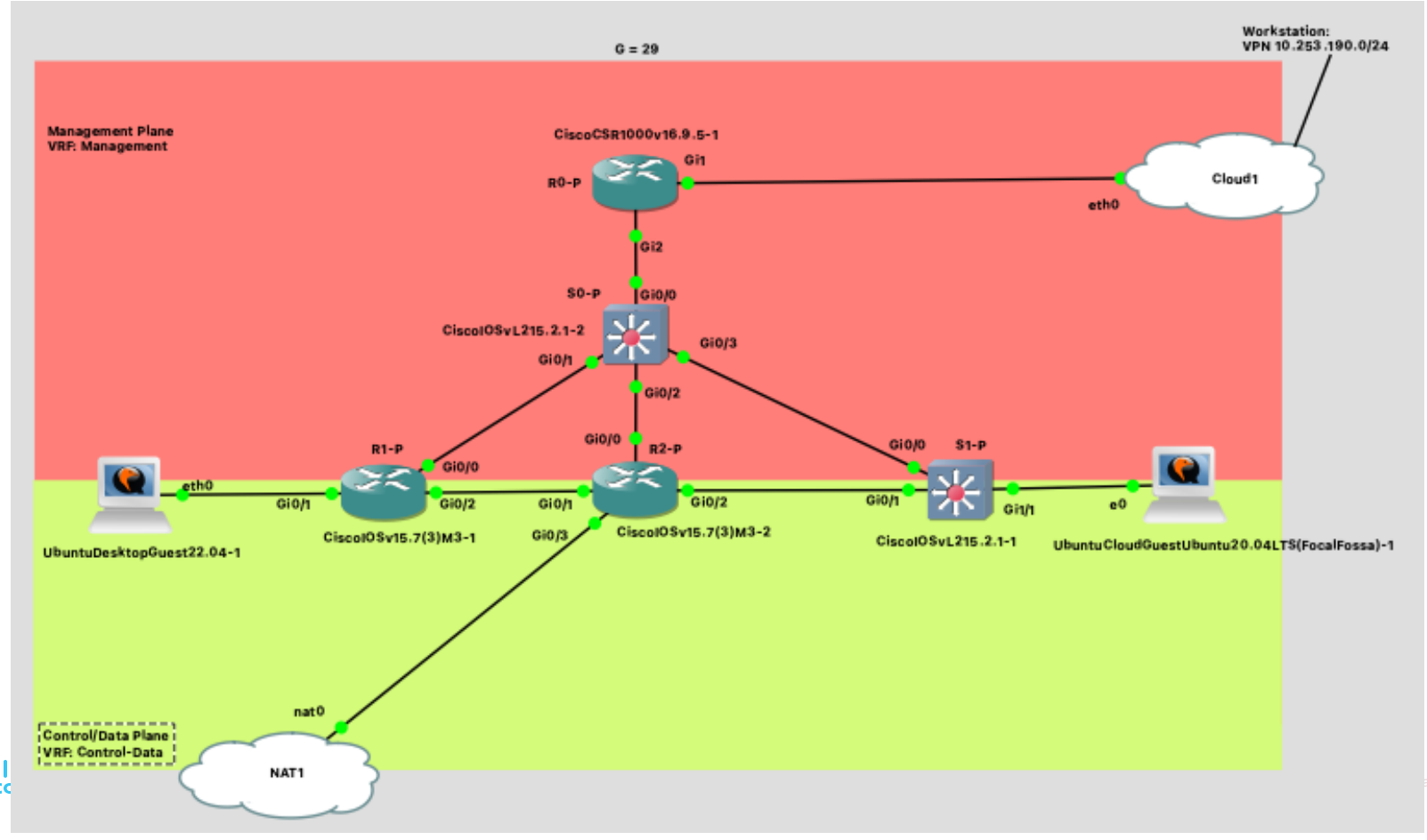
Lab Topology



Lab Instruction – Preparation

- Use IOSv L3 for R0 (Do not use CSR1000v)
- Set R0 G0/1 as a DHCP client to get an IP from Cloud (306).
- The management network for each group is 172.31.Y.0/28.
- Configure Management IPs (see table). S0 and S1 use VLAN 99 for management VLAN
- R1-R2 on management network use 172.31.Y.1 as a default gateway.
- S0-S1 operate only on L2. Disable IP routing and set default-gateway to 172.31.Y.1
- Configure routers (R0-R2) and switches (S0-S1) to allow Telnet and SSH access

Host	Interface	Management IP
R0	G0/0	172.31.Y.1/28
R0	G0/1	10.30.6.XX/24
S0	VLAN99	172.31.Y.2/28
S1	VLAN99	172.31.Y.3/28
R1	G0/0	172.31.Y.4/28
R2	G0/0	172.31.Y.5/28

Remote Access Configuration Example (R0)

Command	Description
(config)#hostname R0	Set hostname to R0
(config)#ip domain-name ipa.com	Set domain name to ipa.com
(config)#ip ssh version 2	Use SSH version 2
(config)#crypto key generate rsa modulus 2048	Generate RSA public/private keys (use 2048 bits)
(config)#username admin privilege 15 password cisco	Set local username “admin” with privilege login (15) and password “cisco”
(config)#line vty 0 4	Enter line vty 0 4
(config-line)#login local	Login to vty 0 4 using local username/password
(config-line)#transport input telnet ssh	Use telnet and SSH as tranport protocols to line vty

Lab Instruction – Preparation (cont.)

- Creates a blank git repository on GitHub. Then clone it to your local git repository.
- Checkpoint:
  - You should be able to ping/telnet/ssh from your DEVASC VM or PC/Notebook in VPN or Lab306 network to all networking devices shown in the topology, using the management IP of each device.
- Note:
  - Mac: You may get an error message when ssh-ing to routers: *"no matching key exchange method found. Their offer: diffie-hellman-group-exchange-sha1,diffie-hellman-group14-sha1"*. Use **ssh -o KexAlgorithms=diffie-hellman-group14-sha1 admin@<IP of your router>**

Lab VRF

- VRF name in the management plane: management
- VRF name in control/data plane: control-data
- **Manually** configure VRF on R1-R2 to separate the Data/Control plane from the Management plane:
  - G0/0 of R1 and R2, Management VRF
  - G0/1-2 of R1 and R2: Control-Data VRF
- Checkpoint:
  - ping from R1 to G0/1 of R0, and G0/0 of R2 should be successful
  - ping from your PC in the VPN or 306 network to G0/0 of R1 should be successful
  - Ping from your PC in the VPN or 306 network to G0/1 of R1 should be fail
  - ping from UbuntuDesktopGuest to G0/1 of R1 should be successful
  - ping from UbuntuDesktopGuest to G0/0 of R1 should be fail
- **Question:** Why do we have to configure VRF manually?

Lab Paramiko

- Install paramiko version 2.8.1 in your virtual environment – pip install paramiko==2.8.1
- Create a git branch named “paramiko”.
- Follow the steps shown in https://networklessons.com/uncategorized/ssh-public-key-authentication-cisco-ios to create public/private keys for username admin (with no passphrase), and configure SSH Public Key authentication on Cisco IOS router to accept user admin and use only publickey based authentication.
- Read Paramiko document **Client.connection()** and write a program named paramikolab.py to SSH login from your PC to R0-R2 and S0-S1 using SSH Public Key authentication
- The following IOS commands to disable password/keyboard ssh authentication will be deprecated soon.

```
R1(config)#no ip ssh server authenticate user password
R1(config)#no ip ssh server authenticate user keyboard
```

```
%% Warning:SSH command accepted but this CLI will be deprecated soon.
Please move to new CLI "ip ssh server algorithm authentication"
Please configure "default ip ssh server authenticate user" to make CLI ineffective.
```

- Cisco IOS uses MD5; use “**ssh-keygen -l -E md5 -f ~/.ssh/id_rsa.pub**” to verify the public key hash configured on router.
- Checkpoint: the paramikolab.py should be able to ssh to R0-R2 and S1-S2 from your PC without entering password (use publickey based authentication).
- Add R0 running-configuration in the local repository(branch paramiko), and merge your “paramiko” branch to the ”main” branch and push to GitHub.

Lab Netmiko

- Create a new git branch named “netmiko”
- Write a netmiko.py script to
  - Configure VLAN 101 for control/data plane on S1 to separate the management plane and control plane
  - Configure OSPF on R1 and R2 on control/data plane. All interfaces in the control/data plane (except G0/3 of R2) and loopback interfaces are in area 0.
  - Advertise a default route to the NAT cloud on R2 into the OSPF at R2.
  - Configure PAT on R2
  - Allow telnet/ssh to R1-2 and S1 only from the management plane IP addresses and the Lab306 network.
- Checkpoint:
  - UbuntuCloudGuest can ping/traceroute www.google.com using path S1->R1->R2->R3.
  - UbuntuCloudGuest can ping UbuntuDesktopGuest
  - UbuntuCloudGuest cannot ping any management plane interfaces
  - UbuntuCloudGuest cannot telnet/ssh to R1-R2 and S1.
- Merge the netmiko branch to the main branch and push to GitHub.

Lab Netmiko-Jinja2

- Create a new git branch named “netmiko-jinja2”
- Write a netmiko-jinja2.py script to refactor netmiko.py using jinja2.
- Checkpoint:
  - UbuntuCloudGuest can ping/traceroute www.google.com using path S1->R1->R2->R3.
  - UbuntuCloudGuest can ping UbuntuDesktopGuest.
  - UbuntuCloudGuest cannot ping any management plane interfaces
  - UbuntuCloudGuest cannot telnet/ssh to R1-R2 and S1.
- Merge the netmiko-jinja2 branch to the main branch and push to GitHub.

Lab Netmiko-RE

- Create a new git branch named “netmiko-re”
- Write a netmiko-re.py to check and display all the active interfaces on routers R1-R2, and their uptime using regular expression.
- Merge the netmiko-re branch to the main branch and push to GitHub.

Lab: Interface/Topology Configuration and Validation

- Create a new git branch named ”textfsm-ntctemplate”
- Install textfsm and ntc-templates in your virtual environment
- Use TDD to write a program – test_textfsm.py using pytest to test the correctness of textfsmlab.py by referring to the network topology.
- Use TDD to write a program – textfsm.py to configure interface description of R1, R2 and S1 on control/data plane and make all testcases pass.
- The description of the interface is in the format of “Connect to <Interface-name> of <Remote device name>”.
- Run ”sh cdp neighbor” to automatically generate descriptions between Cisco devices. For example, the description of interface G0/2 of R1 is “**Connect to G0/1 of R2**”.
- For connection to PC, description is “**Connect to PC**”
- For Interface G0/3 of R2 which is a DHCP client –The interface description is “**Connect to WAN**”.
- Repeat TDD process until it passes all testcases.
- Merge ”netmiko-pytest” branch to the main branch, and push to GitHub.

Lab: Pull Request

- Make the repo public
- Fork your friend’s repo.
- Create a new branch
- Add some features
- Pull request to your friend’s repo.
- Check and approve your friend’s pull request.