Ansible Network Automation Immersion Day Workshop

Overview

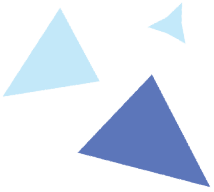
You will use Ansible commands and playbooks to explore and reconfigure a router in a virtual environment. Note, that even though we are using a virtual environment, this is not a requirement, everything we do, can be done on physical devices.

The Immersion Day is meant to be an introduction to Ansible Engine and Ansible Tower. It will expose you to using Ansible Engine as well as Ansible Tower to connect to, explore and configure network devices. In addition, throughout the labs we hope to show you some common features, elements, good and bad practices and patterns to using Ansible for Network Automation.

You will be required to modify some files during the course of this workshop. You will not be required to write your own playbooks as this would require much more time. The playbooks used are open source and thus free to use and modify. That being said, writing playbooks and running them in a test environment is one of the best ways to learn.

Outline

- Part 1: Getting setup
  - Fork Git Repo
  - SSH to jump station
  - Downloading forked Git Repo
- Part 2: Explore the environment
  - Explore the environment with shell and Ansible ad-hoc commands
  - Investigate Ansible's configuration and Inventory
  - Connect to the Routers using Ansible
- Part 3: Backup, Configure, and Explore the Routers facts
  - Lab 1
    - Directory structure review
    - Understand and use facts
    - Run initial playbook to put basic configuration on Router
  - Lab 2
    - Gather banner information and reconfigure the banner



- Backup router configuration
- Restore router configuration from files

- Lab 3
  - Configure DNS and loopback interface
  - Create GRE tunnel and setup routing
  - Secure router by pushing secure configuration file to router
  - Ping another student router from loopback interface
- Lab 4
  - Access Ansible Tower
  - Build a backup job with scheduling
  - Interact with a job containing a Survey

Note: Assume the output shown in the examples below will be different to yours.

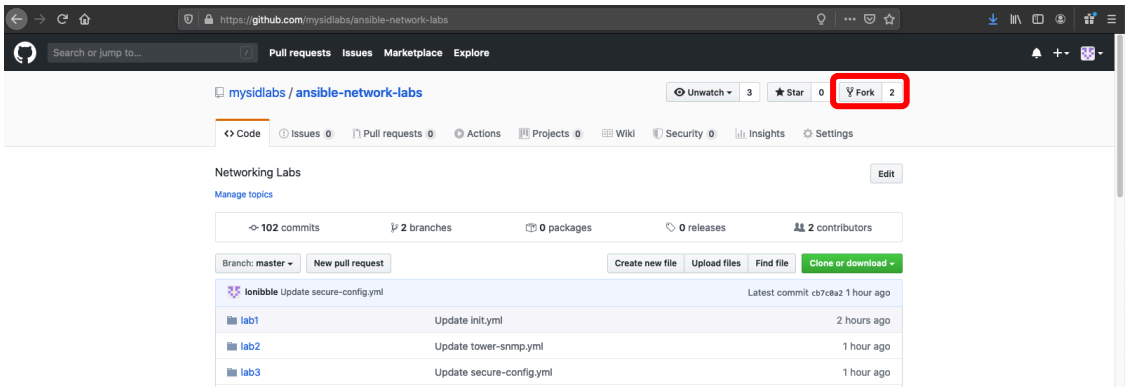
Part 1: Getting setup

Overview

- Fork the lab github repository to your own repository so you can edit and modify.
- Accessing the jump station
- Downloading your forked repository to the jump station

Fork the Sirius ansible networking GitHub repository

1. Login in to Github at https://github.com
2. Go to https://github.com/mysidlabs/ansible-network-labs
3. Click on the Fork button in upper right.



Note: Once forked you can modify all files within GitHub

Connect to the Jump Host

1. SSH to the jump station at the following:

   jump.mysidlabs.com

For MacOS or Linux users the following is an example using the terminal:

$ ssh siduser**<studentID>@jump.mysidlabs.com**

Ex. $ssh siduser101**@jump.mysidlabs.com**

You may get the following message, type **yes** at the prompt:

The authenticity of host 'jump.mysidlabs.com (3.132.28.93)' can't be established.

ECDSA key fingerprint is SHA256: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Are you sure you want to continue connecting (yes/no/[fingerprint])? **Yes**

Warning: Permanently added 'jump.mysidlabs.com,3.132.28.93' (ECDSA) to the list of known hosts.

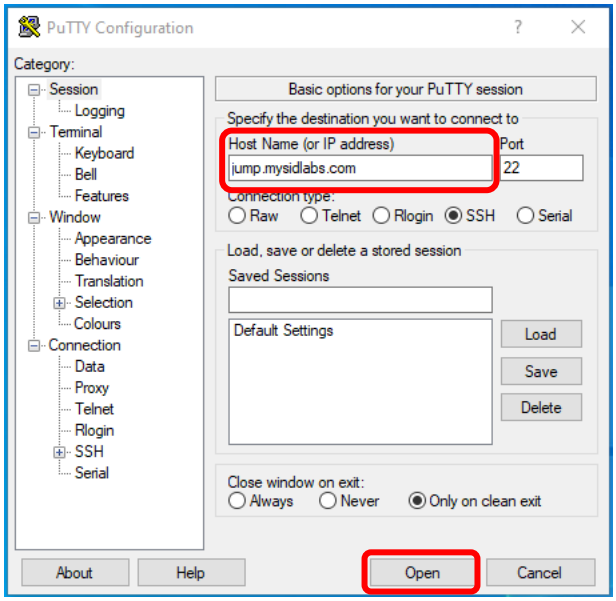
Note: You can remove from known hosts when workshop is completed.

When prompted for your password type in the password the instructor provides

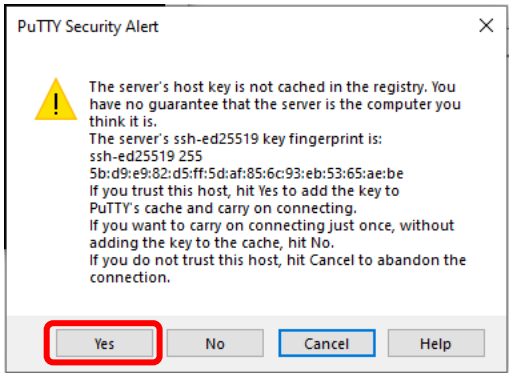
password: *****

For Windows users the following is an example using Putty:

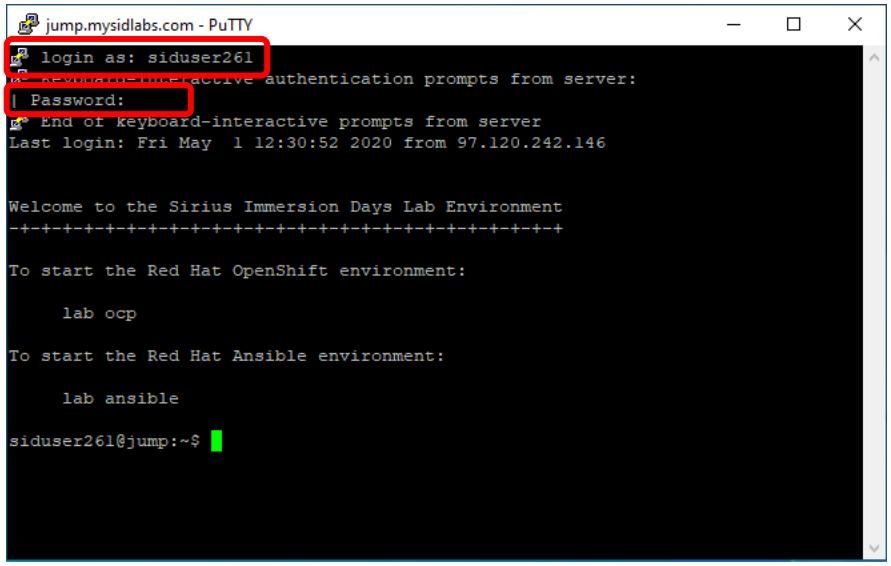
Type jump.mysidlabs.com in the Host Name box and click the Open button



Click the Yes button to accept the ssh key



Type in username and password in the terminal screen at the appropriate prompts



Download repository to jump station

1. Your terminal prompt should change to something like the following:

   siduser101@jump:~$

2. Type in 'lab ansible' at the promt:

   siduser101@jump:~$ **lab ansible**

3. Your terminal prompt should change to something like the following:

   siduser250@toolkit ~ #

4. Clone your repository

   siduser250@toolkit ~ # **git clone** **https://github.com/<<YOUR_GITHUB_USER>>/ansible-network-labs.git**

Tip	The usage of git becomes very important to "infrastructure as code" because it is your source of truth. Everything resides in github including your changes. If you lose connection from the jump box, the repository will be deleted automatically. All you need to do is clone your repository and you are back to where you were.

5. You should now see the repository in your directory

   siduser250@toolkit ~ # **ls**

   ansible-network-labs

6. Move into the ansible-network-labs directory

   siduser250@toolkit ~ # **cd ansible-network-labs**

7. You can now explore the labs directory

   cd = change directory

   ls = list contents

   pwd = display current working directory

   cat = display file

   nano or vim = file editor

   tree = display file structure from current directory

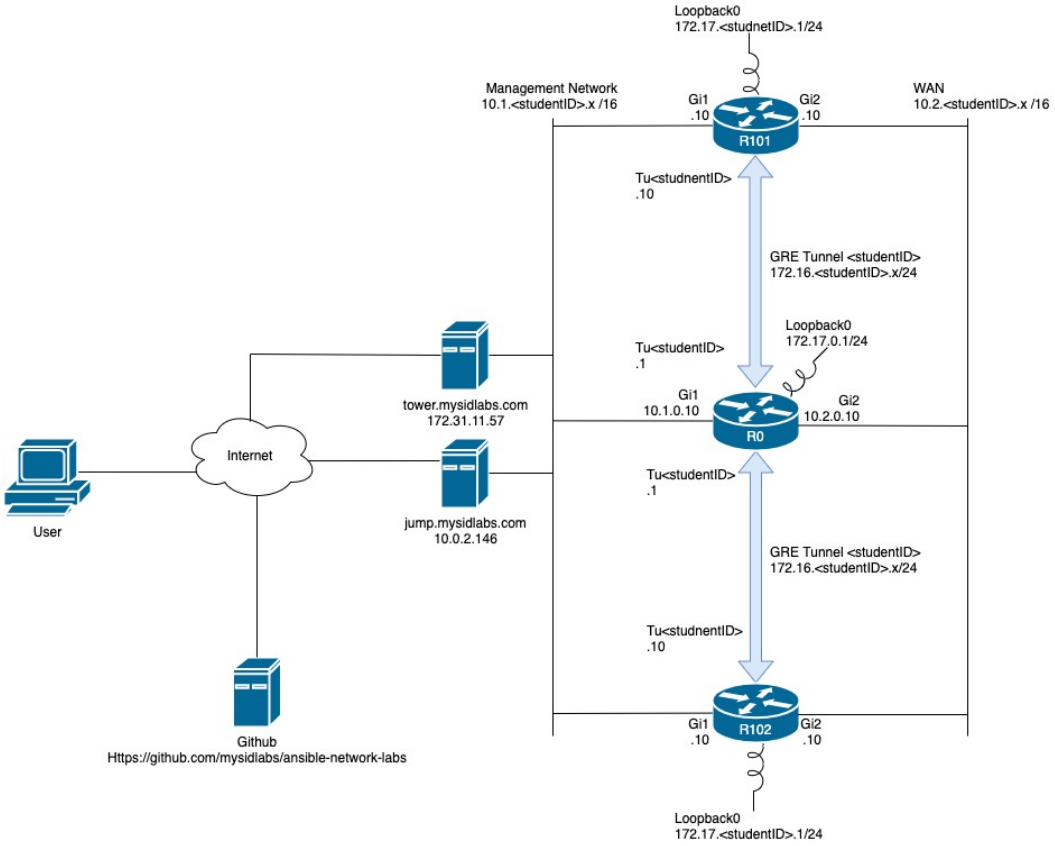
Part 2: Explore the environment with Ansible

Overview

- Review the topology
- Modify your inventory file
- Investigate Ansible's configuration and Inventory files
- Use ansible ad-hoc commands to use the ping, ios_command and cli_command modules to connect to your router

Topology

The topology is simple for the sake of learning some ansible basics. Traffic will traverse the R0 router and by the end you should be able to ping from your loopback interface to other loopback interfaces in the lab. There will be GRE (Generic Routing Encapsulation) tunnel between every student router and R0 by the end which will allow traffic to be routed between devices. The diagram below is an example, it shows the instructor router R0 and 2 student routers but there is a router for every student.



Connect to the Jump Station

1. Explore your ansible configuration

   siduser250@toolkit ~ # **pwd**

   /home/siduser250/ansible-network-labs

2. siduser250@toolkit ~/ansible-network-labs # **ansible --version**

Tip	Your output may differ slightly. You should see a recent version of Ansible (> 2.9) and you can see you are using the **ansible.cfg** file which allows you to customize behavior. ! ansible --version should be run from the “ansible-network-labs" directory.

```
siduser261@jump:~/ansible-network-labs$ ansible --version
ansible 2.9.7
  config file = /home/siduser261/ansible-network-labs/ansible.cfg
  configured module search path = ['/home/siduser261/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/local/lib/python3.7/dist-packages/ansible
  executable location = /usr/local/bin/ansible
  python version = 3.7.6 (default, Dec 19 2019, 09:25:23) [GCC 9.2.1 20191130]
siduser261@jump:~/ansible-network-labs$
```

3. Investigate how your ansible.cfg has been configured

   siduser250@toolkit ~/ansible-network-labs # **cat ansible.cfg**

```
root ~/ansible-network-labs # cat ansible.cfg
[defaults]
deprecation_warnings          = False
gathering                     = explicit
retry_files_enabled           = False
inventory                     = ~/ansible-network-labs/hosts
stdout_callback               = yaml
connection                    = smart
timeout                       = 60

[ssh_connection]
host_key_checking             = False
ssh_args                      = -o ControlMaster=auto -o ControlPersist=30m
ansible_ssh_common_args       = ' -oKexAlgorithms=+diffie-hellman-group1-sha1 -caes128-cbc'

[paramiko_connection]
host_key_auto_add             = True

[persistent_connection]
connect_timeout               = 60
command_timeout               = 60
root ~/ansible-network-labs #
```

Tip	This is the configuration that ansible will use if you are in the ~/ansible-networks-labs directory and if the **inventory** setting was missing this would tell you that Ansible will default to **/etc/ansible/hosts**.

4. Explore your inventory

   siduser250@toolkit ~/ansible-network-labs # **cat hosts**

   Note: That the R<studentID> line, this is not going to work.

```
siduser261@jump:~/ansible-network-labs$ cat hosts
[all:vars]
ansible_user=admin
ansible_pass=ansible
ansible_port=22

[routers:children]
cisco
juniper

[cisco]
R<studentID> ansible_host=10.1.<studentID>.10

[cisco:vars]
ansible_network_os=ios

[juniper]

[juniper:vars]
ansible_network_os=junos

[arista]

[arista:vars]
ansible_network_os=eos
```

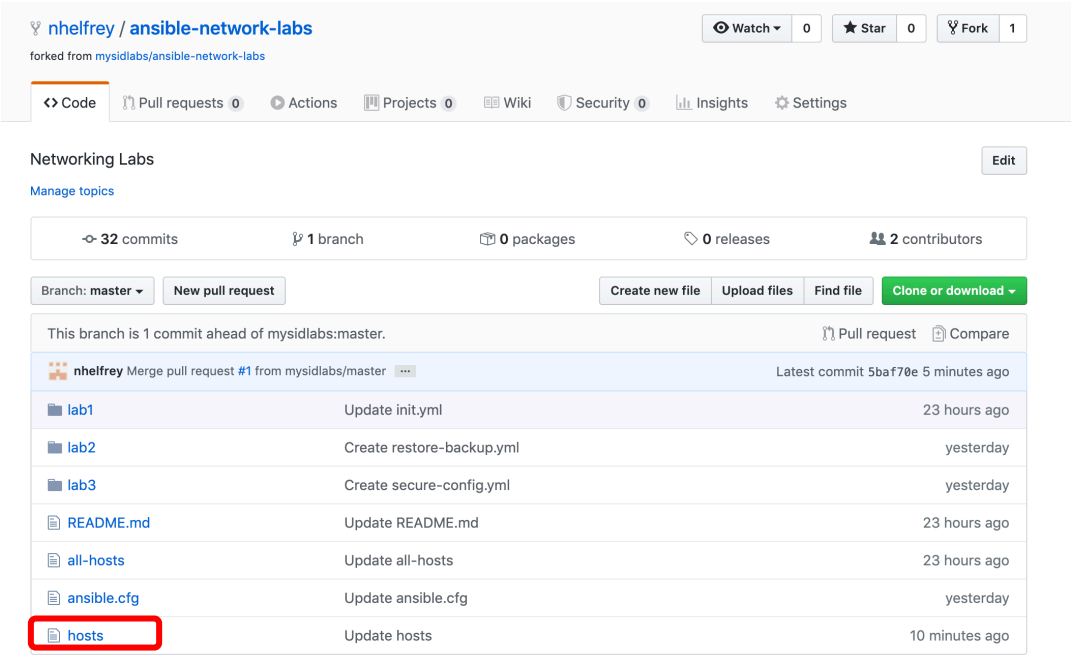
   Note: Ansible uses **ansible_network_os** to inform Ansible which network platform this hosts corresponds too. This is required when using connection plugins network_cli or netconf. We will be using **ansible_connection: network_cli** later which is automatically combined with this.

   Here for example **ansible_network_os=ios** tells Ansible the Networking Operating System is Cisco's ios. Platforms supported by ansible_network_os include:

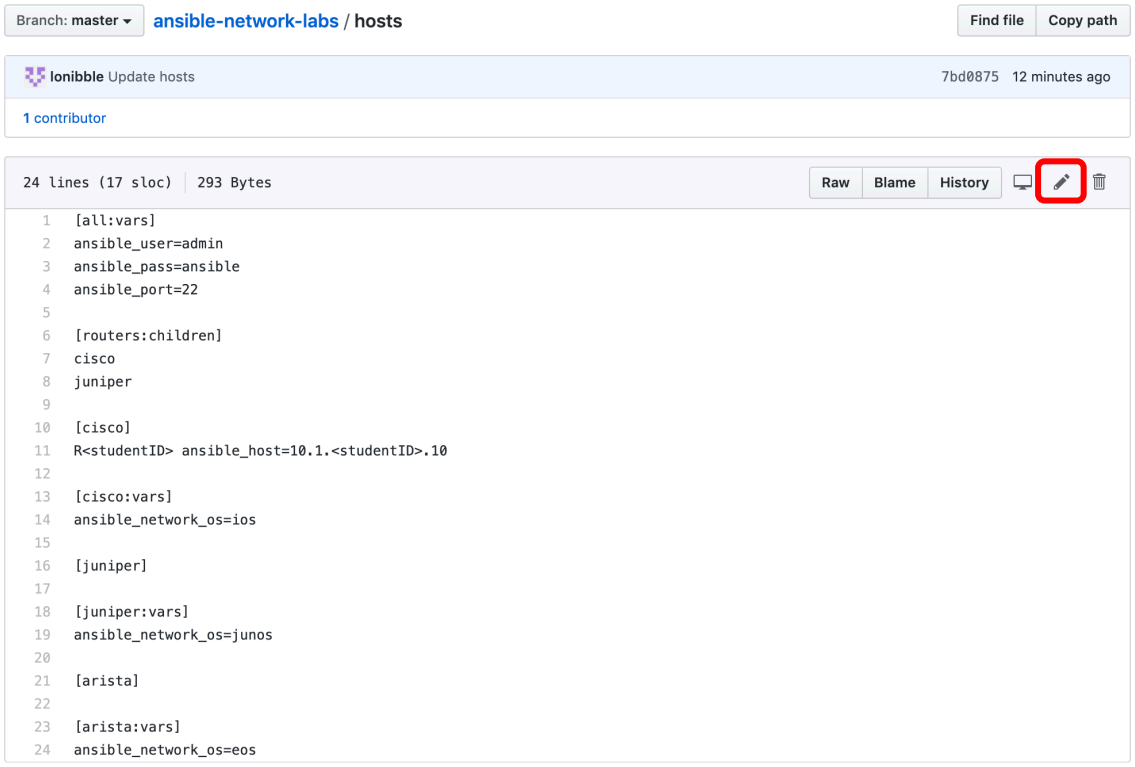
Networking Platform	Ansible Network OS
Cisco IOS	ios
Cisco IOS-XR	iosxr

Cisco NX-OS	nxos
Arista EOS	eos
Juniper JUNOS	junos
F5 BigIP	f5

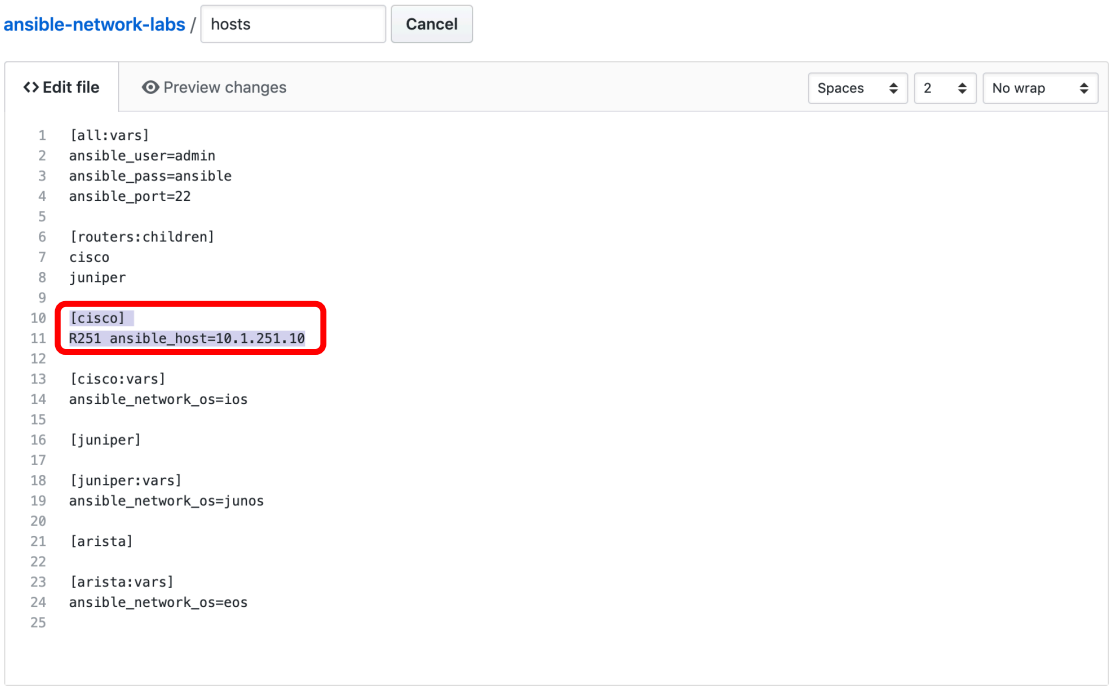
5. Modify your hosts file
    a. Go to your github and go to the ansible-network-labs repository
    b. Click on the hosts file



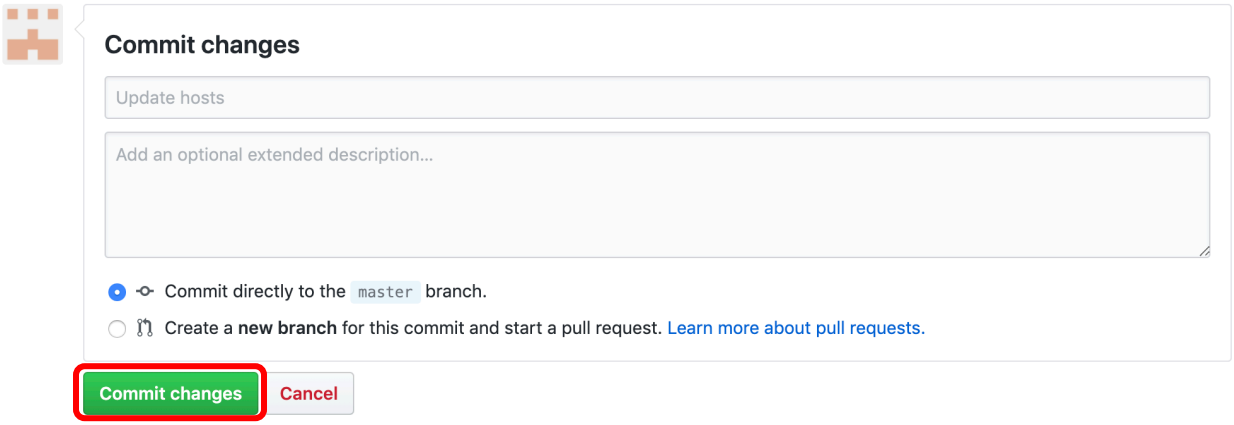
c. Click on the edit/pencil icon to edit your hosts file



d. Modify to match your router, changing <studentID>, with your student number



e. Click “Commit changes” button at bottom of page when you are finished editing



f. After editing the hosts file in github to reflect the proper hostname and IP address of your student ID, pull the changes from your github repository.

siduser250@toolkit ~/ansible-network-labs # **git pull**

Note: Make sure you are in the **~/ansible-network-labs** directory. You can use the 'pwd ' command to see what directory you are in.

g. Check the host file changed:

siduser250@toolkit ~/ansible-network-labs # **cat hosts**

```
siduser261@jump:~/ansible-network-labs$ git pull https://github.com/nhelfrey/ansible-network-labs.git
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/nhelfrey/ansible-network-labs
 * branch            HEAD       -> FETCH_HEAD
Updating 5baf70e..f21d841
Fast-forward
 hosts | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
siduser261@jump:~/ansible-network-labs$ cat hosts
[all:vars]
ansible_user=admin
ansible_pass=ansible
ansible_port=22

[routers:children]
cisco
juniper

[cisco]
R251 ansible_host=10.1.251.10

[cisco:vars]
ansible_network_os=ios

[juniper]

[juniper:vars]
ansible_network_os=junos

[arista]

[arista:vars]
ansible_network_os=eos
siduser261@jump:~/ansible-network-labs$
```

6. Check Ansible can see the hosts in your inventory

   siduser250@toolkit ~/ansible-network-labs # **ansible all --lists-hosts**

```
siduser261@jump:~/ansible-network-labs$ ansible all --list-hosts
  hosts (1):
    R251
siduser261@jump:~/ansible-network-labs$
```

Tip	The **all** field in the previous command refers to the automatically created group 'all'. You can retry the command using another group name or hostname instead. For example, **cisco, routers** or **R<studentID>**. Using Groups liberally in inventory files gives flexibility in the long run and is considered a Best Practice.

Investigate the Network Infrastructure

1. Check if the jump host can connect to the router.

   siduser250@toolkit ~/ansible-network-labs # **ping 10.1.<studentID>.10**

```
siduser261@jump:~/ansible-network-labs$ ping 10.1.251.10
PING 10.1.251.10 (10.1.251.10) 56(84) bytes of data.
64 bytes from 10.1.251.10: icmp_seq=1 ttl=255 time=0.782 ms
64 bytes from 10.1.251.10: icmp_seq=2 ttl=255 time=0.737 ms
64 bytes from 10.1.251.10: icmp_seq=3 ttl=255 time=0.810 ms
64 bytes from 10.1.251.10: icmp_seq=4 ttl=255 time=0.735 ms
64 bytes from 10.1.251.10: icmp_seq=5 ttl=255 time=0.717 ms
64 bytes from 10.1.251.10: icmp_seq=6 ttl=255 time=0.743 ms
^C
--- 10.1.251.10 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5116ms
rtt min/avg/max/mdev = 0.717/0.754/0.810/0.031 ms
siduser261@jump:~/ansible-network-labs$
```

Tip	Mac users: control-c to stop the ping Windows users: ctrl-c to stop the ping

2. Check if you can connect to your student router using an ansible Ad-Hoc command

   siduser250@toolkit ~/ansible-network-labs # **ansible routers -m ping**

   Note: Type “yes” if/when prompted to continue connecting

```
siduser261@jump:~/ansible-network-labs$ ansible cisco -m ping
The authenticity of host '10.1.251.10 (<no hostip for proxy command>)' can't be established.
RSA key fingerprint is SHA256:dn7qBMD1An2gYoQoJpxJ1gyL/c9N4L9KCa7n27ZqnW8.
Are you sure you want to continue connecting (yes/no)? yes
[WARNING]: Unhandled error in Python interpreter discovery for host R251: unexpected output from Python interpreter discovery
[WARNING]: sftp transfer mechanism failed on [10.1.251.10]. Use ANSIBLE_DEBUG=1 to see detailed information
[WARNING]: Platform unknown on host R251 is using the discovered Python interpreter at /usr/bin/python, but future installation of another Python
interpreter could change this. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information.
R251 | FAILED! => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "module_stderr": "Shared connection to 10.1.251.10 closed.\r\n",
    "module_stdout": "\r\n\r\n\r\nLine has invalid autocommand \"/bin/sh -c '/usr/bin/python '\"'\"'Line has invalid autocommand \"/bin/sh -c '\"'\"'\"'\"'
\"'\"'\"'\"'( umask 77 && mkdir -p \"` echo Line has invalid autocommand \"/bin/sh -c '\"'\"'\"'\"'\"'\"'\"'\"'\"'\"'\"'\"'\"'\"'\"'\"'\"'\"'\"'\"'\"'\"'\"
'\"'\"'\"'echo ~ec2-user && sleep 0'\"'\"'\"'\"'\"'\"'\"",
    "msg": "MODULE FAILURE\nSee stdout/stderr for the exact error",
    "rc": 0
}
siduser261@jump:~/ansible-network-labs$
```

Tip	Ansible supports calling modules directly from the command line via the ansible command. These are called Ad-Hoc commands and are often used to establish connectivity as in here with the ping module. Another common use case is to inspect a host or group of hosts with fact gathering modules like **setup**. You can optionally pass parameters to Ad-Hoc commands with the -a option: **ansible localhost -m debug -a "msg='passing a parameter'"**

Note: The ping module, without additional parameters, is unable to successfully communicate with the Routers. By default, Ansible connections are handled transparently by ssh and no connection type needs to be set. This works well in a typical server environment but typically not for network devices. However, Ansible has a pluggable connection architecture which allows it to be extended to connect to these and other devices. Other examples of connection: types include local when connecting to the localhost and winrm which allows Ansible to connect to Microsoft Windows platforms.

3. Retry the ansible ping again selecting the group routers and this time setting the connection type to network_cli

   siduser250@toolkit ~/ansible-network-labs # **ansible routers -m ping -c network_cli**

```
siduser261@jump:~/ansible-network-labs$ ansible cisco -m ping -c network_cli
[WARNING]: Platform linux on host R251 is using the discovered Python interpreter at /usr/bin/python, but future installation of another Python
interpreter could change this. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information.
R251 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
```

Note: Here Ansible combines the connection type **-c network_cli** with the **ansible_network_os=ios** inventory variable and now knows how to successfully communicate with the network device.

Tip	Ansible allows you to pass variables like ansible_network_os on the command line with the -e option and this has the highest precedence, i.e. will override any variable you have set. You could try setting this to an illegal value, for example: **ansible routers -m ping -c network_cli -e ansible_network_os=linux** You can also try setting it to a different vendors network operating system such as junos, the result may at first be surprising. However, 'network_cli' uses similar mechanisms to connect to network devices so a misconfiguration here may not result in failure.

4. Modify hosts files (see previous lesson on editing host file in github and pulling the changes for reference if needed)
   a. Add ansible_connection=network_cli to hosts file in repository and commit the change

```
10  [cisco]
11  R102 ansible_host=10.1.102.10
12
13  [cisco:vars]
14  ansible_network_os=ios
15  ansible_connection=network_cli
```

   b. Pull changes
   siduser250@toolkit ~/ansible-network-labs # **git pull**

c. Validate your changes were pulled down by cat hosts

siduser250@toolkit ~/ansible-network-labs # **`cat hosts`**

```
[cisco:vars]
ansible_network_os=ios
ansible_connection=network_cli
```

Note: A good practice here would be to add the setting ansible_connection=network_cli to your inventory.

5. Retry the ansible ping again

   siduser250@toolkit ~/ansible-network-labs # **ansible routers -m ping**

   Note: When specified in an 'inventory', you need the prefix **ansible_**. By setting it at an inventory level you will no longer have to specify it via -c or in your playbooks. The remainder of the labs assume you have not set this. Feel free to choose your approach.

Using ios_command module vs cli_command module

We are going to look at 2 different modules. The ios_command module is specifically designed to be used with Cisco devices while the cli_command module is designed to be more agnostic.

1. Use another ansible Ad-Hoc command to retrieve the Routers IP interface information.

   siduser250@toolkit ~/ansible-network-labs # **ansible routers -m ios_command -a "commands='sh ip int bri'"**

```
siduser261@jump:~/ansible-network-labs$ ansible cisco -m ios_command -a "commands='show ip interface brief'"
[WARNING]: Platform linux on host R251 is using the discovered Python interpreter at /usr/bin/python, but future installation of another Python
interpreter could change this. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information.
R251 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "stdout": [
        "Interface              IP-Address      OK? Method Status                Protocol\nGigabitEthernet1       10.1.251.10     YES DHCP   up
        up      \nGigabitEthernet2       unassigned      YES unset  administratively down down    \nVirtualPortGroup0      192.168.35.101  YES TFTP   up
                up"
    ],
    "stdout_lines": [
        [
            "Interface              IP-Address      OK? Method Status                Protocol",
            "GigabitEthernet1       10.1.251.10     YES DHCP   up                    up      ",
            "GigabitEthernet2       unassigned      YES unset  administratively down down    ",
            "VirtualPortGroup0      192.168.35.101  YES TFTP   up                    up"
        ]
    ]
}
siduser261@jump:~/ansible-network-labs$
```

Tip	The use of double quotes and single quotes are required for commands that contain spaces. double quote at start before commands, and at end of the command after the single quote, and single quote after the = symbol and at the end of the command. Example: “commands=’sh ip int bri’”

Note: Your student router’s gigabitethernet2 interface IP address is unassigned and administratively down. This interface is required for the “WAN” connectivity and establishment of tunnels later in the lab.

Note: The argument passed via commands, 'sh ip int bri’ is the syntax a Network Operator would use if logged in directly to an ios based router. This is very powerful in that it enables Network Operators to use the <*>_command networking modules to interact with network devices using the syntax with which they are already familiar.

Tip	While there is a more function based ios_banner module available, the optimal way to work with Ansible and network devices is to use higher level modules such as the *_command and *_config modules and allow Network Operators to use their day to day command set. When thinking about which modules to use in your playbooks. The *_command modules are for running commands in “user mode” or “privilege exec mode” such as “show” commands. The *_config modules are for editing and changing device configurations.

Bonus

Below is an example of using all-hosts inventory to run the same command against multiple routers including the instructor router R0. You can see the IP addresses and tunnel interface addresses of the R0 router.

```
siduser261@jump:~/ansible-network-labs$ ansible all -i all-hosts -m ios_command -a "commands='sh ip int bri'"
[WARNING]: Platform linux on host R251 is using the discovered Python interpreter at /usr/bin/python, but future installation of another Python
interpreter could change this. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information.
R251 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "stdout": [
        "Interface              IP-Address      OK? Method Status                Protocol\nGigabitEthernet1       10.1.251.10     YES DHCP   up
          up      \nGigabitEthernet2       unassigned      YES unset  administratively down down    \nVirtualPortGroup0      192.168.35.101  YES TFTP   u
p                 up"
    ],
    "stdout_lines": [
        [
            "Interface              IP-Address      OK? Method Status                Protocol",
            "GigabitEthernet1       10.1.251.10     YES DHCP   up                    up      ",
            "GigabitEthernet2       unassigned      YES unset  administratively down down    ",
            "VirtualPortGroup0      192.168.35.101  YES TFTP   up                    up"
        ]
    ]
}
[WARNING]: Platform linux on host R0 is using the discovered Python interpreter at /usr/bin/python, but future installation of another Python interpreter
could change this. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information.
R0 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "stdout": [
        "Interface              IP-Address      OK? Method Status                Protocol\nGigabitEthernet1       10.1.0.10       YES DHCP   up
          up      \nGigabitEthernet2       10.2.0.10       YES DHCP   up                    up      \nLoopback0              172.17.0.1      YES manual u
p                 up      \nTunnel101              172.16.101.1    YES manual up                    up      \nTunnel102              172.16.102.1    Y
ES manual up                    up      \nVirtualPortGroup0      192.168.35.101  YES TFTP   up                    up"
    ],
    "stdout_lines": [
        [
            "Interface              IP-Address      OK? Method Status                Protocol",
            "GigabitEthernet1       10.1.0.10       YES DHCP   up                    up      ",
            "GigabitEthernet2       10.2.0.10       YES DHCP   up                    up      ",
            "Loopback0              172.17.0.1      YES manual up                    up      ",
            "Tunnel101              172.16.101.1    YES manual up                    up      ",
            "Tunnel102              172.16.102.1    YES manual up                    up      ",
            "VirtualPortGroup0      192.168.35.101  YES TFTP   up                    up"
        ]
    ]
}
siduser261@jump:~/ansible-network-labs$
```

2. Try to "port" the above ad hoc command to use the cli_command module. The difference between ios_command and cli_command is

   siduser250@toolkit ~/ansible-network-labs # **ansible all -i all-hosts -m cli_command -a "commands='sh ip int bri'"**

```
siduser261@jump:~/ansible-network-labs$ ansible all -i all-hosts -m cli_command -a "commands='sh ip int bri'"
[WARNING]: Platform linux on host R0 is using the discovered Python interpreter at /usr/bin/python, but future installation of another Python
interpreter could change this. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information.
R0 | FAILED! => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "msg": "Unsupported parameters for (cli_command) module: commands Supported parameters include: answer, check_all, command, newline, prompt, sendonly
"
}
[WARNING]: Platform linux on host R251 is using the discovered Python interpreter at /usr/bin/python, but future installation of another Python
interpreter could change this. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information.
R251 | FAILED! => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "msg": "Unsupported parameters for (cli_command) module: commands Supported parameters include: answer, check_all, command, newline, prompt, sendonly
"
}
```

   While this fails look at the error message carefully. The source of the failure is a very minor syntax change in the module argument. Try again replacing the argument “commands” omitting the “s”:

   Now try your updated *ad hoc* command.

   siduser250@toolkit ~/ansible-network-labs # **ansible all -i all-hosts -m cli_command -a "command='sh ip int bri'"**

   Take a moment to use the ansible-doc command to explore cli_command in more depth

   siduser250@toolkit ~/ansible-network-labs # **ansible-doc cli_command**

Tip	Type q to quit from ansible docs and <space bar> to page down

<<OUTPUT OMITTED>>

Part 3: Explore, Configure and Backup using playbooks

Lab 1 Overview

- Directory structure review
- Understand and use facts
- Run initial playbook to put basic configuration on Router

Review Directory structure on Jump Host

For these labs, the directory in which the "ansible-playbook" command is executed, is very important. We will take a moment to review the folder structure on the jump host and the location of the hosts, all-hosts, and ansible.cfg file. The command "ansible-playbook" needs to be run from the same directory that the hosts, all-hosts, and ansible.cfg file exist for the variables and configuration information from these files to be used.

Tip	If ansible-playbook is executed from the wrong directory your playbook will fail. Settings in the ansible.cfg file tell ansible what inventory file use and how to operate. Without these setting ansible will use the default ansible.cfg file in /etc/ansible/ folder.

Gather facts from router

Like servers it is possible to gather **facts** for networking devices including physical, virtual, and software configuration. Unlike Linux and UNIX servers the traditional 'setup' module does not gather facts about network devices and can be turned off in your playbook header section with 'gather_facts: False'

1. Look at the 1.0-router-facts.yml file

   siduser250@toolkit ~/ansible-network-labs # **cat 1.0-router-facts.yml**

   ```
   ---
   - name: Show router configurations
     hosts: routers
     connection: network_cli
     gather_facts: no

     tasks:
   ```

```
- name: gather ios_facts
  ios_facts:
  register: facts

- name: print out the results of ios_facts
  debug:
    msg: "{{ facts }}"
```

…

Note: In the last line we are using jinja2 substitution to tell ansible to use the hostvars for each target node in the inventory group (inventory_hostname) and extract the value of ansible_net_model.

2. Run the 1.0-router-facts.yml playbook and see what the output is

   siduser250@toolkit ~/ansible-network-labs # **ansible-playbook 1.0-router-facts.yml**

```
siduser261@jump:~/ansible-network-labs$ ansible-playbook lab1/cisco_facts.yml

PLAY [Show router configurations] ***********************************************************************************************

TASK [gather ios_facts] *********************************************************************************************************
[WARNING]: default value for `gather_subset` will be changed to `min` from `!config` v2.11 onwards
[WARNING]: Platform linux on host R251 is using the discovered Python interpreter at /usr/bin/python, but future installation of another Python
interpreter could change this. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information.
ok: [R251]

TASK [print out the results of ios_facts] ***************************************************************************************
ok: [R251] =>
  msg:
    ansible_facts:
      ansible_net_all_ipv4_addresses:
      - 192.168.35.101
      - 10.1.251.10
      ansible_net_all_ipv6_addresses: []
      ansible_net_api: cliconf
      ansible_net_filesystems:
      - 'bootflash:'
      ansible_net_filesystems_info:
        'bootflash:':
          spacefree_kb: 5342868.0
          spacetotal_kb: 6138880.0
      ansible_net_gather_network_resources: []
      ansible_net_gather_subset:
      - hardware
      - default
      - interfaces
      ansible_net_hostname: ip-10-1-251-10
      ansible_net_image: boot:packages.conf
```

Tip	You can edit the ansible.cfg file parameter “stdout_callback = json” if you’d prefer the output to be formatted in JSON.

3. Explore the output to see what is retrieved from the router. These collected facts can be parsed and filtered

4. Now try filtering the information we want to see. Look at the 1.1-facts2.yml and run it.

   siduser250@toolkit ~/ansible-network-labs # **ansible-playbook 1.1-facts2.yml**

```
siduser261@jump:~/ansible-network-labs$ ansible-playbook lab1/facts2.yml

PLAY [gather information from routers] *****************************************************************************************************************

TASK [gather router facts] *****************************************************************************************************************************
[WARNING]: default value for `gather_subset` will be changed to `min` from `!config` v2.11 onwards
[WARNING]: Platform linux on host R251 is using the discovered Python interpreter at /usr/bin/python, but future installation of another Python
interpreter could change this. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information.
ok: [R251]

TASK [display version] *********************************************************************************************************************************
ok: [R251] =>
  msg: 'The IOS version is: 17.01.01'

TASK [display serial number] ***************************************************************************************************************************
ok: [R251] =>
  msg: The serial number is:9BVM9L1AHBH

PLAY RECAP *********************************************************************************************************************************************
R251                       : ok=3    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

siduser261@jump:~/ansible-network-labs$
```

5. Now run the 1.1-facts2.yml playbook against multiple hosts

   siduser250@toolkit ~/ansible-network-labs # **ansible-playbook -i all-hosts 1.1-facts2.yml**

```
siduser261@jump:~/ansible-network-labs$ ansible-playbook -i all-hosts lab1/facts2.yml

PLAY [gather information from routers] *****************************************************************************************************************

TASK [gather router facts] *****************************************************************************************************************************
[WARNING]: default value for `gather_subset` will be changed to `min` from `!config` v2.11 onwards
[WARNING]: Platform linux on host R251 is using the discovered Python interpreter at /usr/bin/python, but future installation of another Python
interpreter could change this. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information.
ok: [R251]
[WARNING]: Platform linux on host R0 is using the discovered Python interpreter at /usr/bin/python, but future installation of another Python
interpreter could change this. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information.
ok: [R0]

TASK [display version] *********************************************************************************************************************************
ok: [R0] =>
  msg: 'The IOS version is: 17.01.01'
ok: [R251] =>
  msg: 'The IOS version is: 17.01.01'

TASK [display serial number] ***************************************************************************************************************************
ok: [R251] =>
  msg: The serial number is:9BVM9L1AHBH
ok: [R0] =>
  msg: The serial number is:9D4L46RDXZ7

PLAY RECAP *********************************************************************************************************************************************
R0                         : ok=3    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
R251                       : ok=3    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Tip	The “-i" parameter tells ansible to overide the inventory file in the ansible.cfg file.

Note: The difference here is running the command against multiple devices simultaneously.

Tip	Each of the major networking platforms has their own facts module which is simple the name of the ansible_networking_os prefixing _facts. In this case ios_facts, with Juniper devices the equivalent module would be junos_facts. Here we use a common pattern in Ansible to capture the results of running the ios_facts module in a register variable we have chosen to call facts. Then in the second, debug, task we output the contents of the variable facts

6. Look at 1.2.1-adv-facts.yml and run 1.2.1-adv-facts.yml against default inventory file and with all-hosts inventory file.

   siduser250@toolkit ~/ansible-network-labs # **cat 1.2.1-adv-facts.yml**

   <<OUTPUT OMITTED>>

   siduser250@toolkit ~/ansible-network-labs # **ansible-playbook -i all-hosts 1.2.1-adv-facts.yml**

   <<OUTPUT OMITTED>>

****BONUS****

Look at 1.2.2-adv-facts-filter.yml and run 1.2.2-adv-facts-filter.yml

Try to decipher what it is doing.

Do the initial configuration of your router

We will now run a playbook to put a base configuration on your router. This will configure the hostname, enable GigabitEthernet2, set the IP on the interface and add a user.

1. Modify the username/password in the 1.3-init.yml in the lab1 folder in your repository, pull down your change and validate that you have successfully pulled the change by looking at the 1.3-init.yml on your jump box.

```
33
34      - name: add username and password to router
35        ios_config:
36          lines:
37            - username siduserXXX privilege 15 secret password
```

Note: Storing Usernames and Passwords in git is a very bad idea. We do this for lab purposes, this should never be done otherwise!

Tip	Pay attention to indentation. Ansible provides a way to validate syntax before running a playbook. This will not validate that you are using a module correctly. It only validates you are using proper syntax structure. **ansible-playbook 1.3-init.yml --syntax-check**

2. Run the 1.3-init.yml playbook against your router.

   siduser250@toolkit ~/ansible-network-labs # ansible-playbook 1.3-init.yml

   Note: While you can run your 1.3-init.yml and other configuration playbooks on all the routers in the lab we ask that you only run the configuration playbooks on your assigned router.

```
siduser261@jump:~/ansible-network-labs$ ansible-playbook -i all-hosts lab1/facts2.yml

PLAY [gather information from routers] *****************************************************************************************************************

TASK [gather router facts] *****************************************************************************************************************************
[WARNING]: default value for `gather_subset` will be changed to `min` from `!config` v2.11 onwards
[WARNING]: Platform linux on host R251 is using the discovered Python interpreter at /usr/bin/python, but future installation of another Python
interpreter could change this. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information.
ok: [R251]
[WARNING]: Platform linux on host R0 is using the discovered Python interpreter at /usr/bin/python, but future installation of another Python
interpreter could change this. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information.
ok: [R0]

TASK [display version] *********************************************************************************************************************************
ok: [R0] =>
  msg: 'The IOS version is: 17.01.01'
ok: [R251] =>
  msg: 'The IOS version is: 17.01.01'

TASK [display serial number] ***************************************************************************************************************************
ok: [R251] =>
  msg: The serial number is:9BVM9L1AHBH
ok: [R0] =>
  msg: The serial number is:9D4L46RDXZ7

PLAY RECAP *********************************************************************************************************************************************
R0                         : ok=3    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
R251                       : ok=3    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

3. Now SSH into the router from the jump station and check the configuration

   Note: The 'XXX' in the example below should be replaced with your ID. The password is the password you configured before.

   siduser250@toolkit ~/ansible-network-labs # ssh siduserXXX@10.1.XXX.10

```
siduser261@jump:~/ansible-network-labs$ ssh siduserXXX@10.1.251.10
Password:



R251#
R251#
R251#sho ip int brief
Interface              IP-Address      OK? Method Status                Protocol
GigabitEthernet1       10.1.251.10     YES DHCP   up                    up
GigabitEthernet2       10.2.251.10     YES DHCP   up                    up
VirtualPortGroup0      192.168.35.101  YES TFTP   up                    up
R251#sho run | i username
username ec2-user privilege 15
username siduserXXX privilege 15 secret 9 $9$wXxFI8MtOHl6IE$bjyyG5Vioq9o6uRXghloE.9l5mQ70B0yjZDaRJ8AQ6c
```

4. Ping the instructor router R0 10.2.0.10

   R250# **ping 10.2.0.10**

```
R251#ping 10.2.0.10
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.2.0.10, timeout is 2 seconds:
..!!!
Success rate is 60 percent (3/5), round-trip min/avg/max = 1/1/1 ms
R251#
```

5. Type exit to leave the router cli

   R250# **exit**

****BONUS****

Run playbook 1.4 using ansible vault to encrypt and decrypt a variable file containing the usernames and passwords.

Step 1

Look at encrypt.yml in your forked github account. modify the file with your username and password from lab 1.3

username: siduserXXX
password: password

On the jump station

siduser150@toolkit ~/ansible-f5-labs # **git pull**

Step 2

Encrypt the encrypt.yml file and look at it again.

siduser150@toolkit ~/ansible-f5-labs # **ansible-vault encrypt encrypt.yaml**

Vault password: 123
encryption successful

Step 3

siduser150@toolkit ~/ansible-network-labs # **cat encrypt.yml**

```
$ANSIBLE_VAULT;1.1;AES256
62653464356261393636373266376463393239373830303535303230306437363333666436306139
61643934666163323433353232316339663135303832643 20a3564666538636235343037396236 33
64653836316166316134383264336535613937646135366161343031643963373166393563613133
35366231373665326 10a6266356361333539336666366436538613534626633326537333133613 4
64316532376636637643462386362626466306536366139656463666539366165 39
```

Step 4

Use view command to unencrypt and view the file.

siduser150@toolkit ~/ansible-f5-labs # **ansible-vault view encrypt.yml**

Vault password:

username: siduser150

password: password

Step 5

Use rekey command to change the file encryption key.

siduser150@toolkit ~/ansible-f5-labs # **ansible-vault rekey encrypt.yml**

Vault password:

New Vault password:

Confirm New Vault password:

Rekey successful

Step 6

Use the "- - help" command to see what other options are available. Also review the ansible-vault documentation. https://docs.ansible.com/ansible/latest/user_guide/vault.html#vault-ids-and-multiple-vault-passwords

siduser150@toolkit ~/ansible-f5-labs # **ansible-vault --help**

Step 7

Run the 1.4 playbook with the encrypted variable file using the --ask-vault-pass parameter.
siduser150@toolkit ~/ansible-f5-labs # **ansible-playbook 1.4-user-setup.yml --ask-vault-pass**

```
siduser150@toolkit ~/ansible-network-labs # ansible-playbook 1.4-user-setup.yml --ask-vault-pass
Vault password:
PLAY [Add User]*****************************************************************************

TASK [add username and password to router] ***************************************************
changed: [R150]

PLAY RECAP ***********************************************************************************
R150                       : ok=1    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Lab 2 Overview

- Configure SNMP
- Gather banner information and reconfigure the banner
- Backup router configuration

Configure SNMP settings

1. Look at the 2.0-snmp.yml file and examine the contents. Modify playbook with your own values in the repository using github, pull down changes to jump station and validate changes successfully downloaded.

   siduser250@toolkit ~/ansible-network-labs # **cat 2.0-snmp.yml**

   <<OUTPUT OMITTED>>

2. Run the playbook

   siduser250@toolkit ~/ansible-network-labs # **ansible-playbook 2.0-snmp.yml**

```
siduser261@jump:~/ansible-network-labs$ ansible-playbook lab2/snmp.yml

PLAY [snmp ro/rw string configuration] *****************************************************************************************************

TASK [ensure that the desired snmp strings are present] ************************************************************************************
[WARNING]: Platform linux on host R251 is using the discovered Python interpreter at /usr/bin/python, but future installation of another Python
interpreter could change this. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information.
changed: [R251]

PLAY RECAP *********************************************************************************************************************************
R251                       : ok=1    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Tip	Notice how this playbook does not use variables. It is best practice to use variable when specific parameters are subject to change such as a security requirement to change snmp community strings every 3 months. This is also where Ansible Tower can help with the use of surveys.

3. Validate your router has the snmp configuration
   a. **Bonus** #1: Try using ansible ad-hoc to validate configuration
   b. **Bonus** #2: Try writing a playbook to validate configuration

Configure routers MOTD

1. Check that the message of the day (MOTD) is set

   siduser250@toolkit ~/ansible-network-labs # **ansible routers -m ios_command -a "commands='sh banner motd'"**

```
siduser261@jump:~/ansible-network-labs$ ansible cisco -m ios_command -a "commands='sh banner motd'"
[WARNING]: Platform linux on host R251 is using the discovered Python interpreter at /usr/bin/python, but future installation of another Python
interpreter could change this. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information.
R251 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "stdout": [
        "oops, you ran this playbook with out reading it first."
    ],
    "stdout_lines": [
        [
            "oops, you ran this playbook with out reading it first."
        ]
    ]
}
```

2. Look at the 2.1-banner.yml file in the lab2 directory and examine the contents. Modify the banner variable to something you choose in the repository and pull-down changes to jump station and validate changes successfully downloaded.

14 lines (12 sloc) | 353 Bytes — Raw | Blame | History

```
1  ---
2  - name: Update banner message
3    hosts: routers
4    connection: network_cli
5    gather_facts: no
6
7    vars:
8      banner_message: Restricted to authorized users only. All activities on this system are logged.
9
10   tasks:
11   - name: "Update banner message to '{{ banner_message }}'"
12     ios_config:
13       lines:
14         - "banner motd %{{ banner_message }}%"
```

Note: Notice the use of vars to setup a variable banner_message. The use vars is important to making playbooks more reuseable and allows easier editing. The variable can be reset at the command line by using **ansible-playbook banner.yml -e "banner_message='my new message'"**. The -e option allows you to over-write any variable as it has the highest priority in the Ansible variable structure.

3. Run the playbook

   siduser250@toolkit ~/ansible-network-labs # **ansible-playbook 2.1-banner.yml**

Tip	Notice how the variable message was substituted in the Task "**- name:**" message

4. Check that the message was updated as expected

   siduser250@toolkit ~/ansible-network-labs # **ansible routers -m ios_command -a "commands='sh banner motd'"**

```
siduser261@jump:~/ansible-network-labs$ ansible-playbook lab2/banner.yml

PLAY [Update banner message] ***************************************************************************************************************************

TASK [Update banner message to 'Sirius Immersion Days are Super Fun and Cool!!!'] ***********************************************************************
[WARNING]: Platform linux on host R251 is using the discovered Python interpreter at /usr/bin/python, but future installation of another Python
interpreter could change this. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information.
changed: [R251]

PLAY RECAP *********************************************************************************************************************************************
R251                       : ok=1    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

siduser261@jump:~/ansible-network-labs$ ansible cisco -m ios_command -a "commands='sh banner motd'"
[WARNING]: Platform linux on host R251 is using the discovered Python interpreter at /usr/bin/python, but future installation of another Python
interpreter could change this. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information.
R251 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "stdout": [
        "Sirius Immersion Days are Super Fun and Cool!!!"
    ],
    "stdout_lines": [
        [
            "Sirius Immersion Days are Super Fun and Cool!!!"
        ]
    ]
}
```

5. Challenge – If time permits, change banner.yml or create a new playbook to use the cli_config module

Tip	The 'ansible-doc' command not only defines the task parameters that 'cli_config' takes but also gives examples. With a few minor changes, banner.yml can be moved to the more agnostic cli_config module. Also, watch out for the usage of quotes. If you use quote where you are not supposed to, your playbook will not work even though syntax is correct.

Backing Up:

Now you will do a common networking task, backing up the configuration of the router. In this case, if you are going to back up one router config, you might as well back them all up. We will use the -i parameter to override the default hosts file with the all-hosts file to backup all the routers configuration.

1. We will start by using an older method of backing up a router config. Look at the 2.2-backup.yml file and examine the contents. Notice that you are not specifying a name or where to backup the file.

```
siduser261@jump:~/ansible-network-labs$ cat lab2/backup.yml
---
- name: backup ios router configurations
  hosts: cisco
  connection: network_cli
  gather_facts: no

  tasks:

  - name: backup router configuration
    ios_config:
      backup: yes
```

2. Run the playbook

   siduser250@toolkit ~/ansible-network-labs # **ansible-playbook -i all-hosts 2.2-backup.yml**

```
siduser261@jump:~/ansible-network-labs$ ansible-playbook -i all-hosts lab2/backup.yml

PLAY [backup ios router configurations] ****************************************************************************************************

TASK [backup router configuration] *********************************************************************************************************
[WARNING]: Platform linux on host R0 is using the discovered Python interpreter at /usr/bin/python, but future installation of another Python
interpreter could change this. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information.
changed: [R0]
[WARNING]: Platform linux on host R101 is using the discovered Python interpreter at /usr/bin/python, but future installation of another Python
interpreter could change this. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information.
changed: [R101]
[WARNING]: Platform linux on host R251 is using the discovered Python interpreter at /usr/bin/python, but future installation of another Python
interpreter could change this. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information.
changed: [R251]

PLAY RECAP *********************************************************************************************************************************
R0                         : ok=1    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
R101                       : ok=1    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
R251                       : ok=1    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

3. Explore the resulting backup files which can be found in the backup sub-directory where the backup.yml file is

   siduser250@toolkit ~/ansible-network-labs # **ls**
   << Output Omitted >>
   (Notice the 'backup' directory)

siduser250@toolkit ~/ansible-network-labs # **ls backup**
(Notice the file name structure)
<< Output Omitted >>

4. View the backed-up configuration file. (Your filename will be different than the example below)

   siduser250@toolkit ~/ansible-network-labs # **cat backup/R250_config.2020-04-30@18:07:40**
   << Output Omitted >>

Tip	Filenames are case sensitive in Linux

You can open the text-based backup files which can be used to restore a router or as a starting point for a configuration file.

Backing up using the cli_command module

While the **ios_config** module has a convenient backup parameter it is of course very much vendor specfic and similar modules exist for the major Network Platforms. Beginning in Ansible 2.7 there are 2 new, more generic, modules: **cli_config** and **cli_command**. The advantage here being, simpler, more generic playbooks can be developed to more cleanly support a multi-vendor environment. See Appendix A for link to deeper dive into **cli_command**.

1. Look at the 2.3-cli-backup.yml file and examine the contents.
   siduser250@toolkit ~/ansible-network-labs # **cat 2.3-cli-backup.yml**
   << Output Omitted >>

2. Run the playbook

   siduser250@toolkit ~/ansible-network-labs # **ansible-playbook -i all-hosts 2.3-cli-backup.yml**

```
siduser261@jump:~/ansible-network-labs$ ansible-playbook -i all-hosts lab2/cli-backup.yml

PLAY [backup router configuration with cli_command module] *********************************************************************************************

TASK [backup config] ***********************************************************************************************************************************
[WARNING]: Platform linux on host R251 is using the discovered Python interpreter at /usr/bin/python, but future installation of another Python interpreter
could change this. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information.
ok: [R251]
[WARNING]: Platform linux on host R0 is using the discovered Python interpreter at /usr/bin/python, but future installation of another Python interpreter
could change this. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information.
ok: [R0]
[WARNING]: Platform linux on host R101 is using the discovered Python interpreter at /usr/bin/python, but future installation of another Python interpreter
could change this. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information.
ok: [R101]

TASK [move to file] ************************************************************************************************************************************
changed: [R251]
changed: [R101]
changed: [R0]

PLAY RECAP *********************************************************************************************************************************************
R0                         : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
R101                       : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
R251                       : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

siduser261@jump:~/ansible-network-labs$ ls lab2/backup
R0_config.2020-05-01@14:02:37  R101_config.2020-05-01@14:02:37  R251_config.2020-05-01@14:02:37
siduser261@jump:~/ansible-network-labs$
```

3. Explore the resulting backup files which can be found in the backup directory.

   siduser250@toolkit ~/ansible-network-labs # **ls -l backup/**
   << Output Omitted >>

4. Compare your new backups with the ones you made earlier if you would like.

Tip	The sum command can be used to generate a checksum of each file and allow you to verify if your backups differ.

   siduser250@toolkit ~/ansible-network-labs # **sum backup/R250***

***** Bonus *****

Take a look at the 2.4-adv-cli-backup.yml file. This playbook adds the feature of cleaning out configuration portions that would cause a restore to fail. Otherwise removing the top 2 lines of the backup file would be manual prior to a restore.

Take a look at the 2.5-restore-backup.yml file. This is a simple way to restore a configuration.

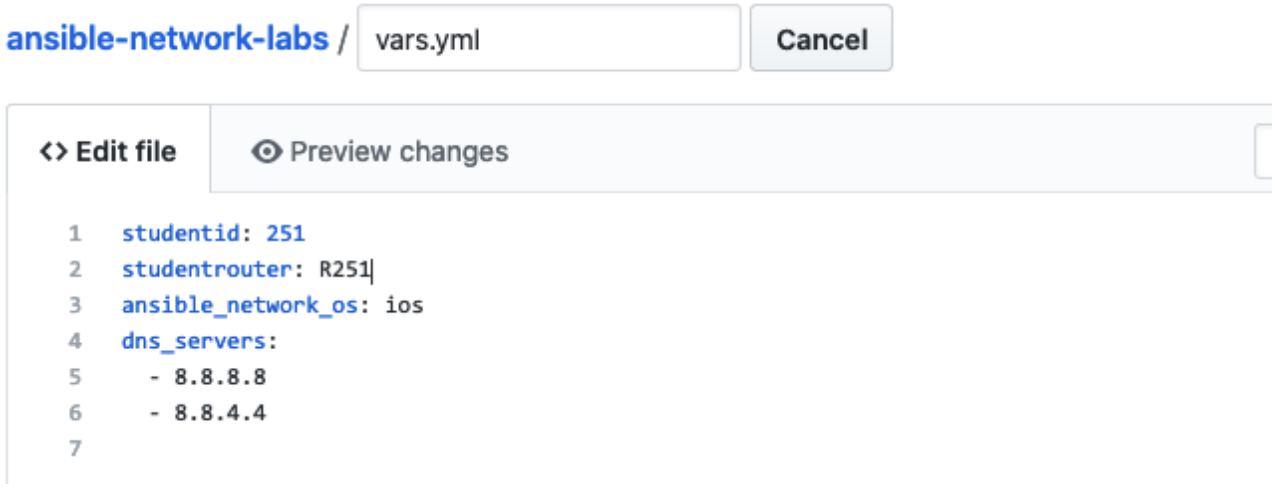
Tip	SCP needs to be installed “pip install scp --user" for the 2.5 playbook to run

Lab 3 Overview

- Configure DNS and loopback interface
- Create GRE tunnel and setup routing
- Secure router by pushing secure configuration file to router
- Ping another student router from loopback interface

Configure DNS and Loopback 0

1. Edit the vars.yml file in your github repository <your github username>/ansible-network-labs/vars.yml. Change << Student ID Number >> and <<Router Hostname >> to your values, see screenshot below for an example. This variable file will be referenced in the playbooks and will be used in the remainder of the labs. Yes, we could have done this from the beginning, but it wasn't time to learn until now.



Tip	Best practice is to use variables. There are many places to put variables and there is a hierarchy of variables that will take priority. We are using an example here of a usage of variables, but this location and position are not necessarily best practice. Please read the document referenced by the link in the Appendix A. It is important to learn and understand using variables.

2. Commit the changes and from your jump box, pull the changes.
   siduser250@toolkit ~/ansible-network-labs # **git pull**
   << Output Omitted >>

3. Verify your changes were successfully pulled.
   siduser250@toolkit ~/ansible-network-labs # **cat vars.yml**
   << Output Omitted >>

4. Run the 3.0-dns-cfg.yml playbook with the –i all-hosts parameter. (In this lab we are showing 1 way of making a playbook device specific even when other devices exist in the inventory)

   siduser250@toolkit ~/ansible-network-labs # **ansible-playbook -i all-hosts 3.0-dns-cfg.yml**

```
siduser261@jump:~/ansible-network-labs$ ansible-playbook -i all-hosts lab3/dns-cfg.yml

PLAY [Router Configurations] ***************************************************************************************************************************************

TASK [configure name servers] **************************************************************************************************************************************
skipping: [R0] => (item=8.8.8.8)
skipping: [R0] => (item=8.8.4.4)
skipping: [R101] => (item=8.8.8.8)
skipping: [R101] => (item=8.8.4.4)
changed: [R251] => (item=8.8.8.8)
changed: [R251] => (item=8.8.4.4)
[WARNING]: Platform linux on host R251 is using the discovered Python interpreter at /usr/bin/python, but future installation of another Python interpreter
could change this. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information.

TASK [enable LoopBack0 interface] **********************************************************************************************************************************
skipping: [R0]
skipping: [R101]
changed: [R251]

TASK [Configure L0] ************************************************************************************************************************************************
skipping: [R0]
skipping: [R101]
ok: [R251]

PLAY RECAP *********************************************************************************************************************************************************
R0                         : ok=0    changed=0    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
R101                       : ok=0    changed=0    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
R251                       : ok=3    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

siduser261@jump:~/ansible-network-labs$
```

Note: Notice how the other routers from the all-hosts inventory file were skipped.

Tip	The ‘block:’ statement in the playbook allows us to group multiple related tasks together, here we are using it combined with the when statement to group tasks together for your router. The example is not typical, but we wanted to illustrate a few different features. Notice the use of loop to iterate through the 2 values of the list variable dns_servers. Prior to Ansible 2.5 we would have had to use with_items:.

Configure GRE tunnel and Routing

1. SSH into your router. Ping the instructor's loopback interface from your loopback interface.

   R250#**ping 172.17.0.1 source l0**

```
R251#ping 172.17.0.1 source l0
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.17.0.1, timeout is 2 seconds:
Packet sent with a source address of 172.17.251.1
.....
Success rate is 0 percent (0/5)
```

   Note: This fails because we do not have a GRE Tunnel and routing existing between your router and the instructor router.

2. Disconnect from your router by typing exit

   R250#**exit**

3. Run the 3.0-gre.yml playbook

   siduser250@toolkit ~/ansible-network-labs # **ansible-playbook 3.0-gre.yml**

```
siduser261@jump:~/ansible-network-labs$ ansible-playbook lab3/gre.yml

PLAY [Configure GRE Tunnel] ****************************************************************************************************************************************************

TASK [create tunnel interface to R0] *******************************************************************************************************************************************
[WARNING]: Platform linux on host R251 is using the discovered Python interpreter at /usr/bin/python, but future installation of another Python interpreter
could change this. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information.
changed: [R251]

TASK [configure OSPF] **********************************************************************************************************************************************************
changed: [R251]

PLAY RECAP *********************************************************************************************************************************************************************
R251                       : ok=2    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

4. SSH into your router. Ping the instructor's loopback interface from your loopback interface.

   R250#**ping 172.17.0.1 source l0**

```
R251#ping 172.17.0.1 source l0
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.17.0.1, timeout is 2 seconds:
Packet sent with a source address of 172.17.251.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms
```

5. Disconnect from your router by typing exit

   R250#**exit**

Configure security setting via file

1. Look at the secure-config.cfg file in the lab3 folder

   siduser250@toolkit ~/ansible-network-labs # **cat 3.2.1-secure-config.cfg**

   << Output Omitted >>

2. Look at the secure-config.yml file in the lab3 folder

   siduser250@toolkit ~/ansible-network-labs # **cat 3.2-secure-config.yml**

   << Output Omitted >>

3. Run the secure-config.yml file

   siduser250@toolkit ~/ansible-network-labs # **ansible-playbook 3.2-secure-config.yml**

```
siduser261@jump:~/ansible-network-labs$ ansible-playbook lab3/secure-config.yml

PLAY [HARDEN IOS ROUTERS] ******************************************************************************************************************************************

TASK [ENSURE THAT ROUTERS ARE SECURE] ******************************************************************************************************************************
[WARNING]: Platform linux on host R251 is using the discovered Python interpreter at /usr/bin/python, but future installation of another Python interpreter
could change this. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information.
changed: [R251]

PLAY RECAP *********************************************************************************************************************************************************
R251                       : ok=1    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

4. Validate the router has the new configuration

Ping another student's loopback address

1. SSH into your router and ‘ping << another students loopback ip >> source l0’

```
R251#ping 172.17.101.1 source l0
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.17.101.1, timeout is 2 seconds:
Packet sent with a source address of 172.17.251.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms
```

2. Disconnect from your router by typing exit

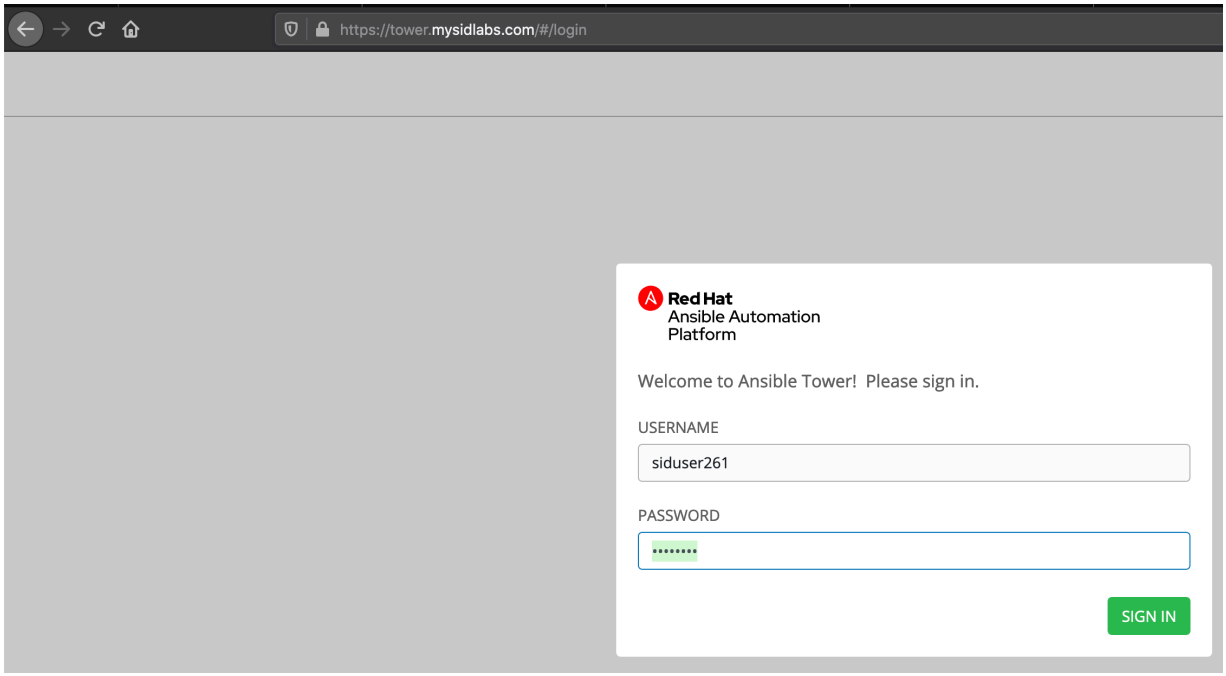
   `exit`

Lab 4 Overview

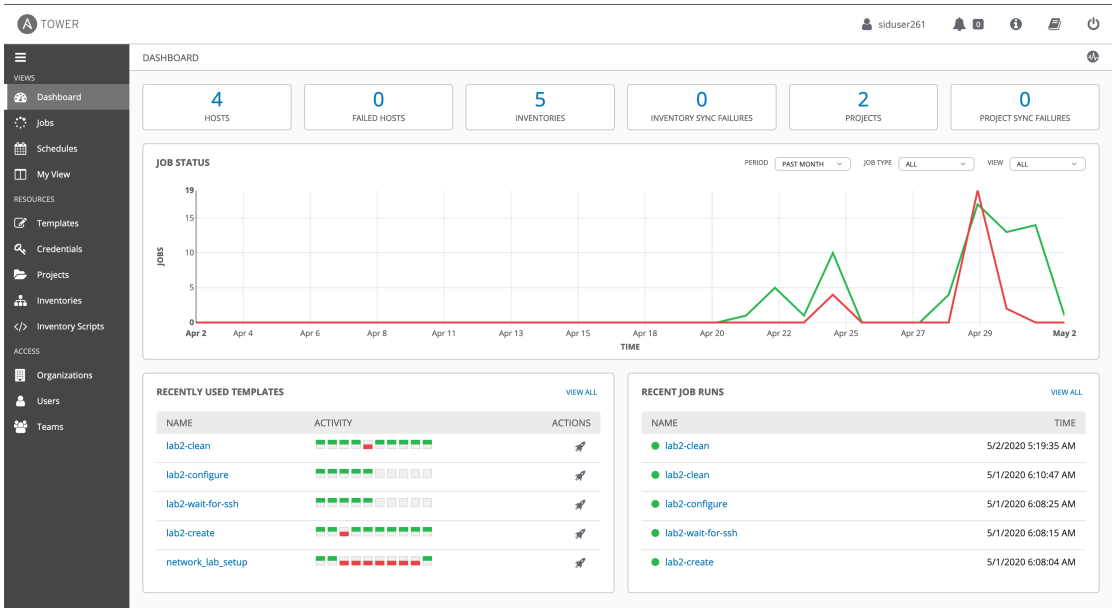
- Access Ansible Tower
- Build a backup job with scheduling
- Interact with a job containing a Survey

Access Ansible Tower

1. Open web browser and go to https://tower.mysidlabs.com and enter in your username and password.



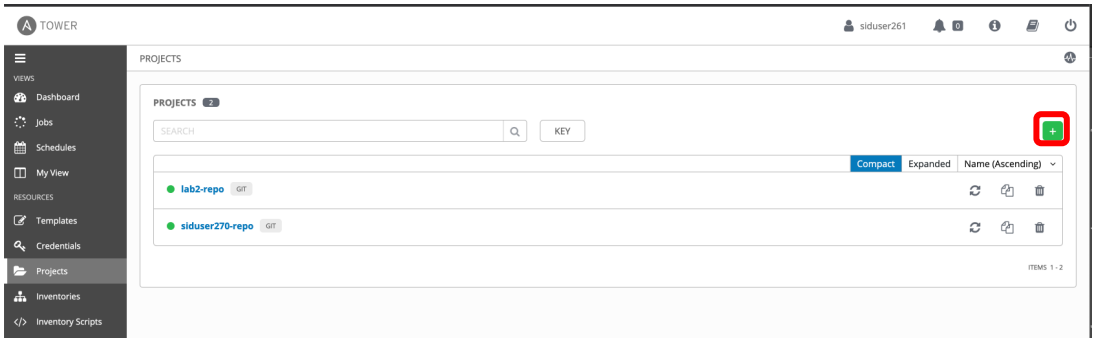
2. Once logged in explore the interface



Build a backup job

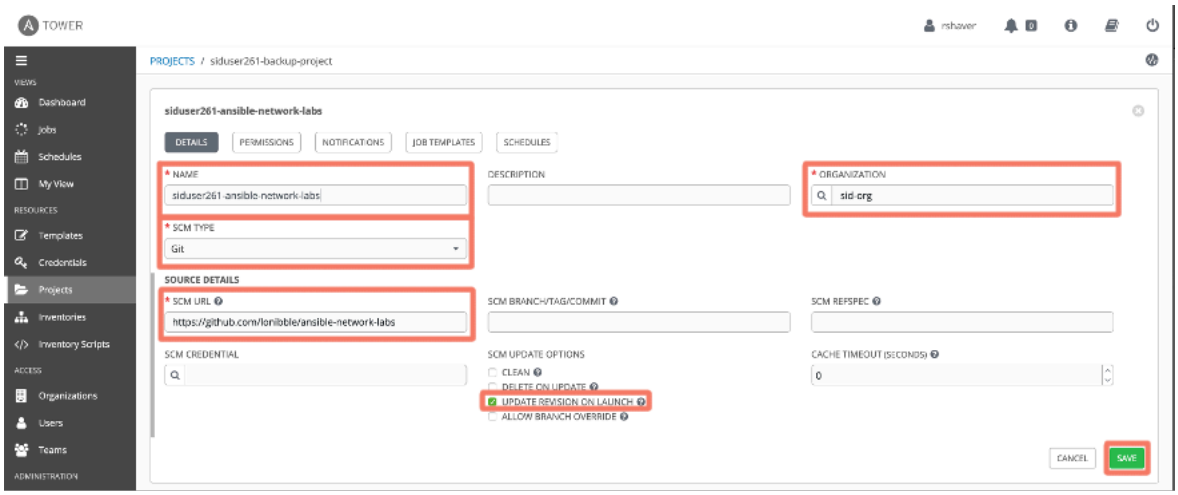
1. **Go into your github repository and modify the tower-hosts file with your router information**. Similar to what you did at the beginning of this workshop.

2. Create a new project. Click 'Projects' on the left-hand side of the page. Then click the plus '+' icon to add a new inventory.

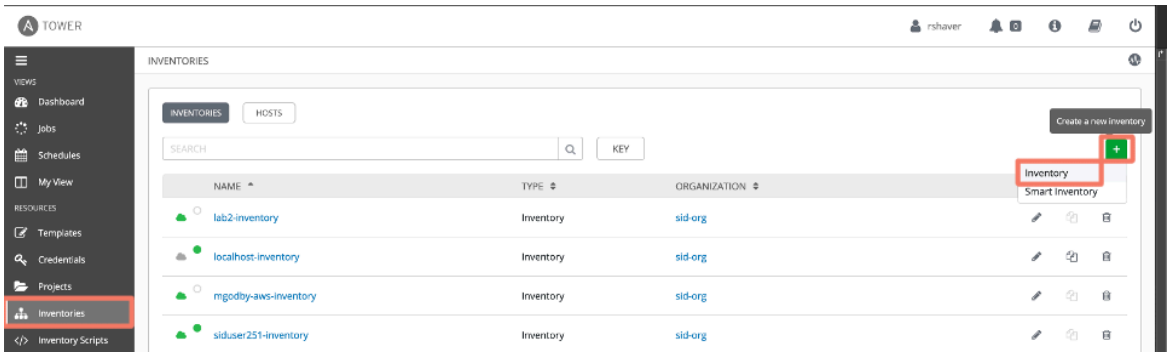


3. Fill in the information and click the ‘SAVE’ Button.

Name	<<siduserID>>-ansible-network-labs
Organization	Sid-org
SCM Type	Git
SCM URL	https://github.com/<<User name>>/ansible-network-labs
Update Revision on Launch	check

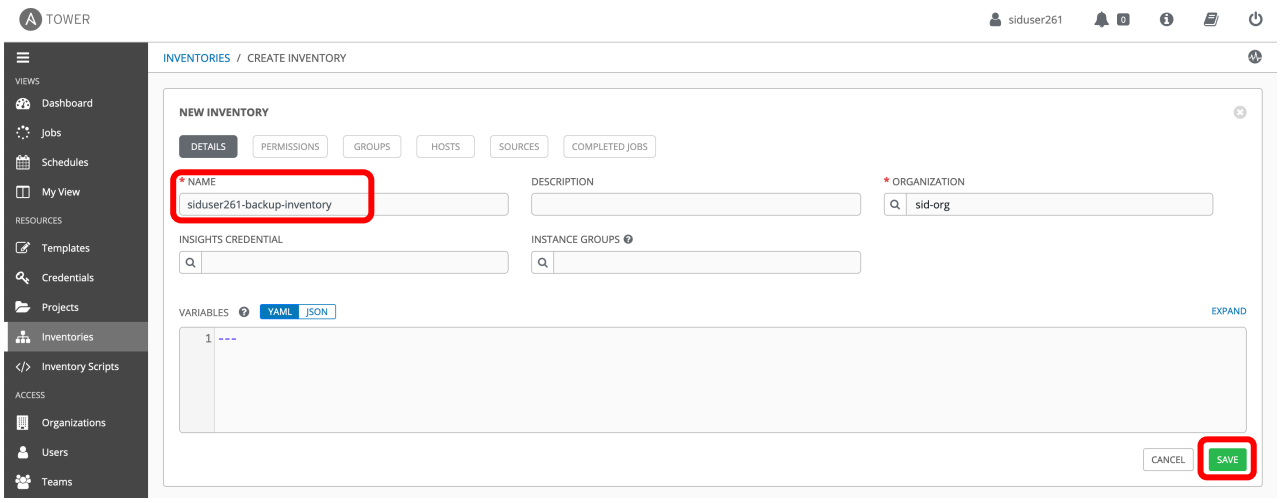


4. Then we need to create our inventory. Click ‘Inventories’ on the left-hand side of the page. Then click the plus ‘+’ icon to add a new inventory. Select Inventory from the drop-down menu.

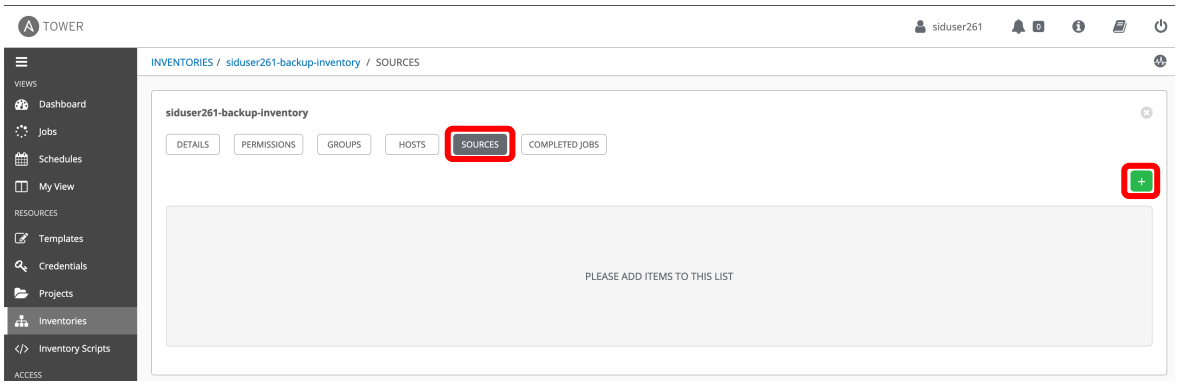


5. Fill in the information and click the ‘SAVE’ Button.

Name	<<siduserID>>-inventory
Organization	Sid-org



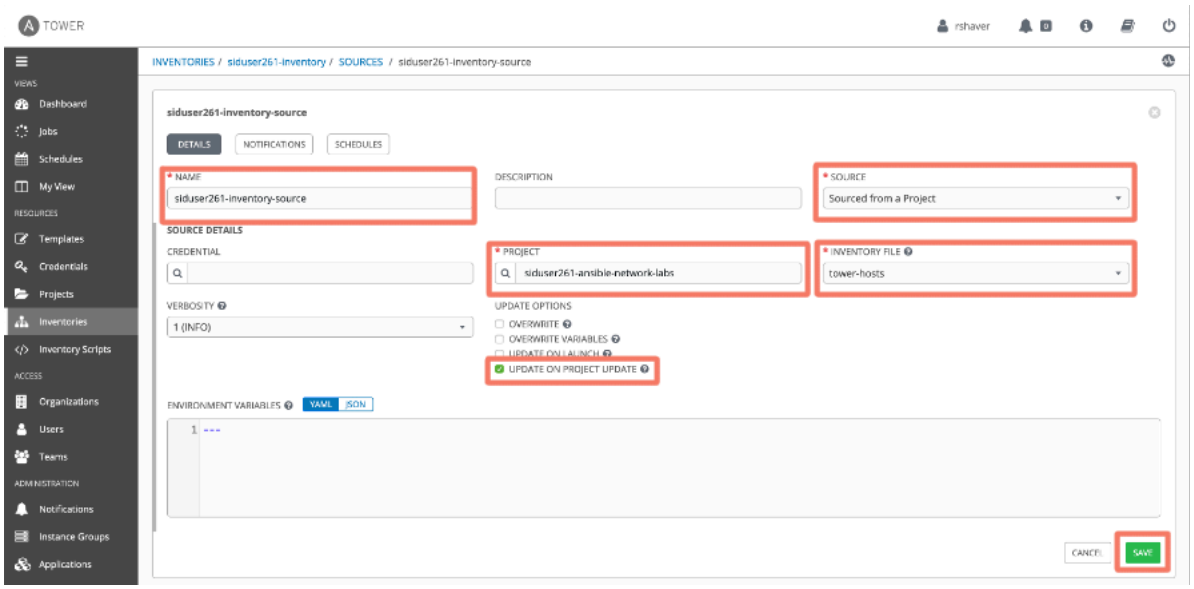
6. Now click on the ‘SOURCES’ button. Then click on the plus ‘+’ icon.



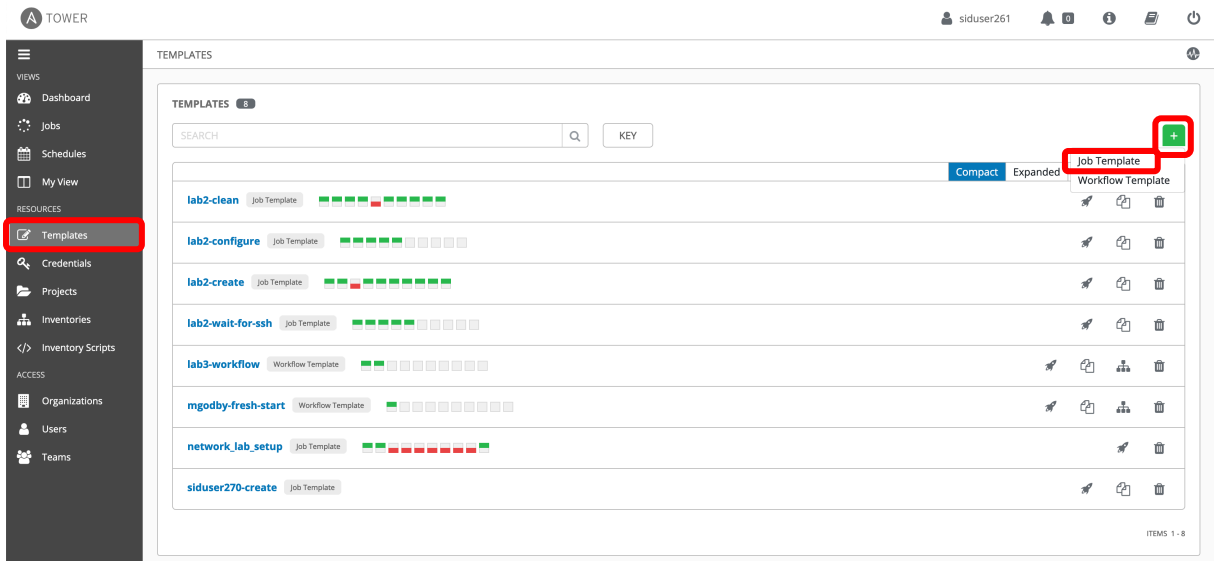
Tip	Take your inventory to the next level by integrating with a dynamic CMDB such as netbox.io, servicenow or Remedy.

7. Fill in the information and click the ‘SAVE’ Button

Name	<<siduserID>>-inventory-source
Source	Sourced from a Project
Project	<<siduserID>>-ansible-network-labs
Inventory File	tower-hosts
Update on Project Update	Checked

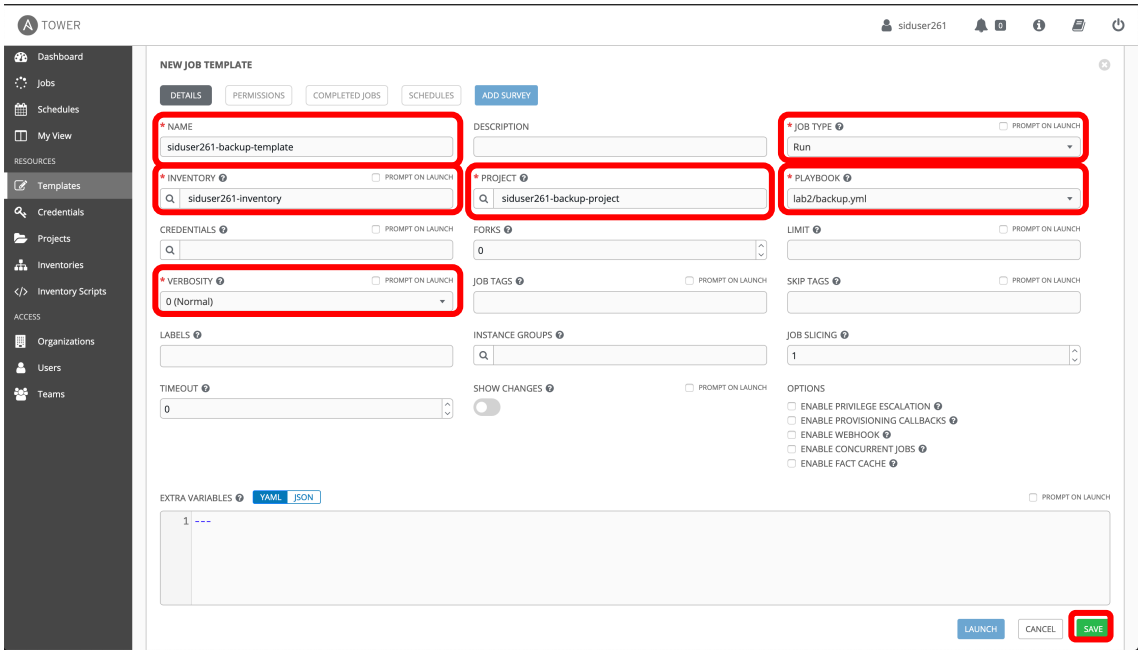


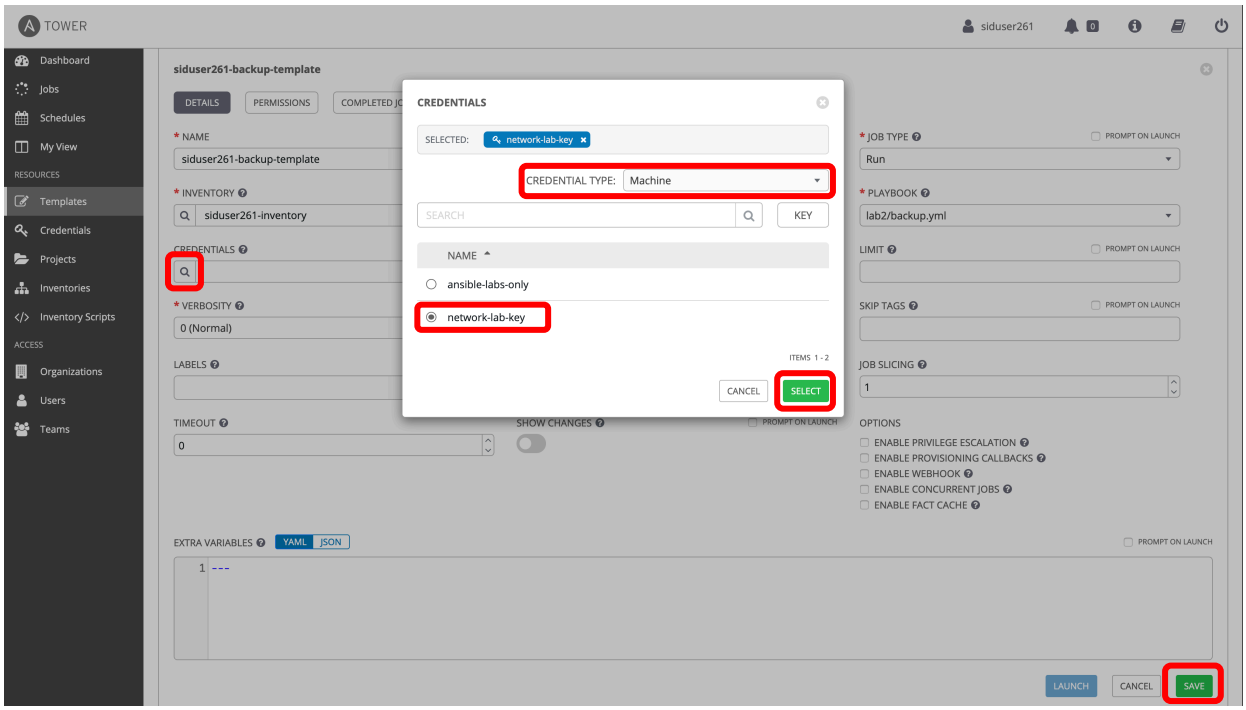
8. Create a new Template. Click ‘Templates’ on the left-hand side of the page. Then click the plus ‘+’ icon to add a new template. Select ‘Job Template’ from the drop down.



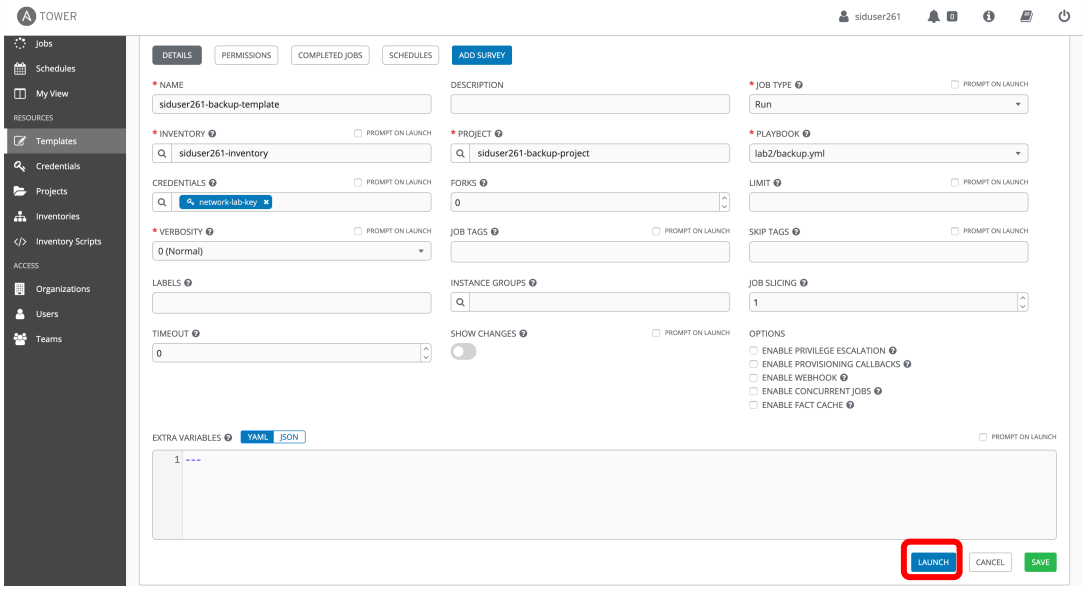
9. Fill in the information and click the ‘SAVE’ Button

Name	<<siduserID>>-backup
Jab Type	Run
Inventory	<<siduserID>>-inventory
Project	<<siduserID>>-ansible-network-labs
Playbook	4.0-tower-backup.yml
Credentials	(Credential Type: Machine) (Name: network-lab-key)

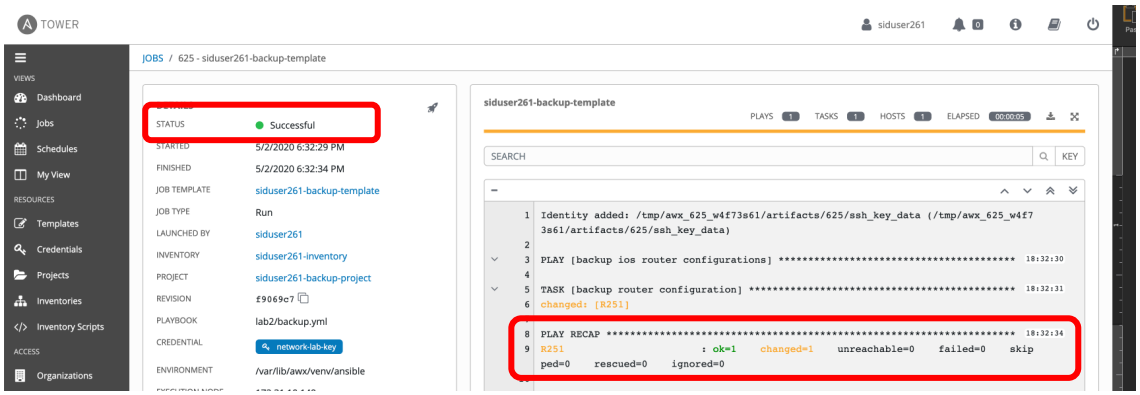




10. Click the Launch Button



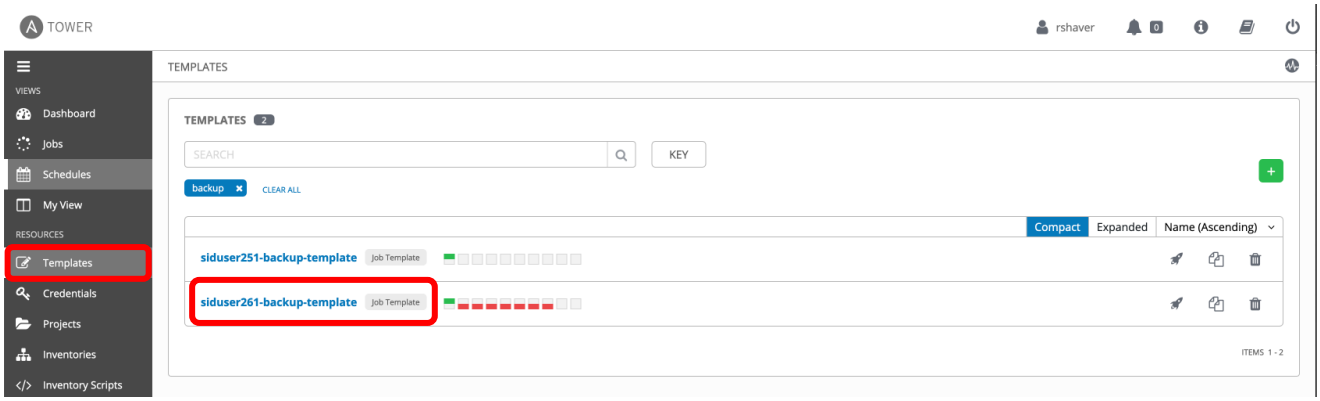
11. Verify job completes successfully



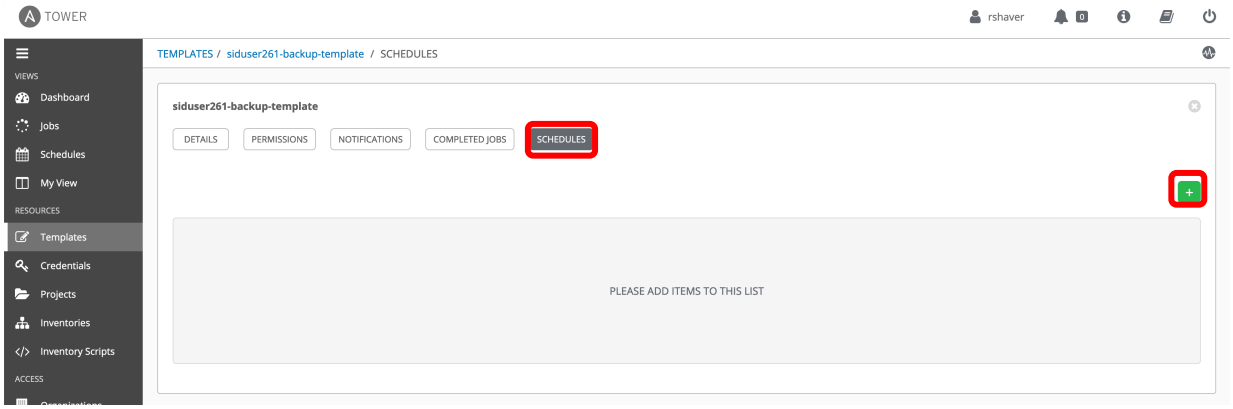
Tip	Think of 'Projects' as containing your code repository. Think of 'Templates' as the job or task you want to run. If you want to run multiple jobs create a workflow template. For an example of a workflow template, look at lab3-workflow and click on the 'WORKFLOW VISUALIZER' button.

Scheduling a job

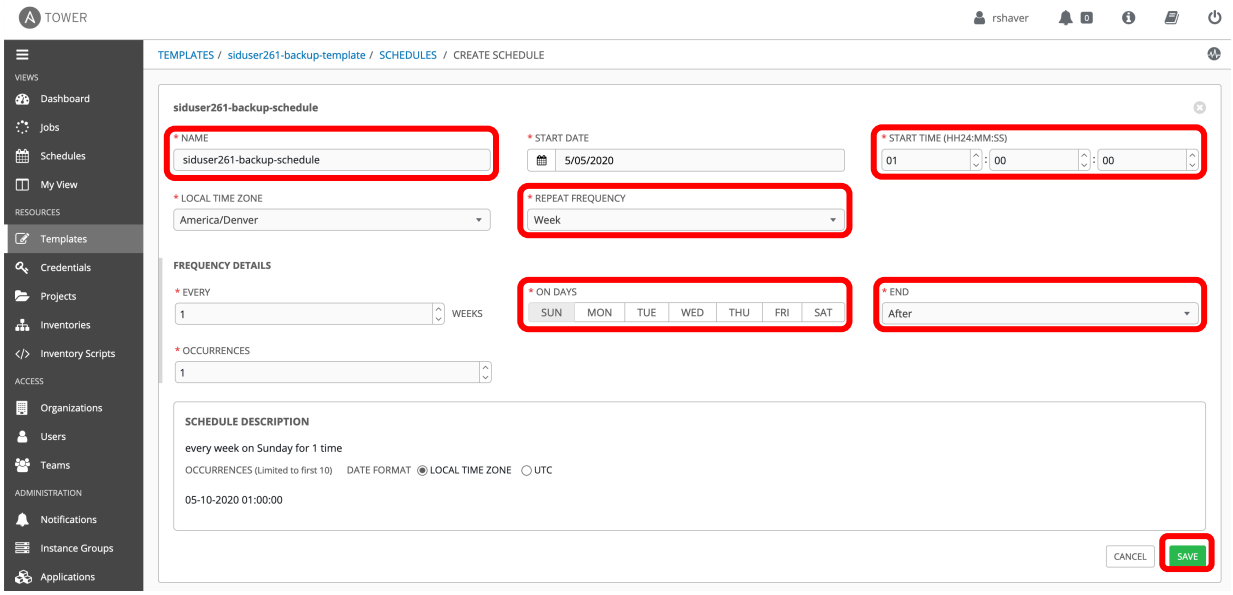
1. Now let's create a schedule for this backup job. Click on 'Templates' on the left-hand side and select on your template.



2. Click on ‘SCHEDULES’, then the plus ‘+’ button.

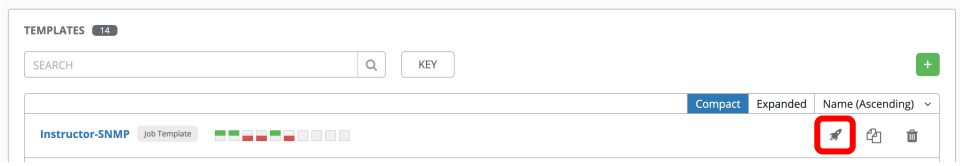


3. In the ‘NAME’ field type in ‘siduser<<StudentID>>-backup-schedule’. In the ‘START TIME (HH24:MM:SS)’ hour field specify 01:00:00. In the ‘REPEAT FREQUENCY’ drop down field select ‘Week’. In the ‘ON DAYS’ field select ‘SUN’. In the ‘END’ drop down field, select ‘After’. Now click the ‘SAVE’ button.

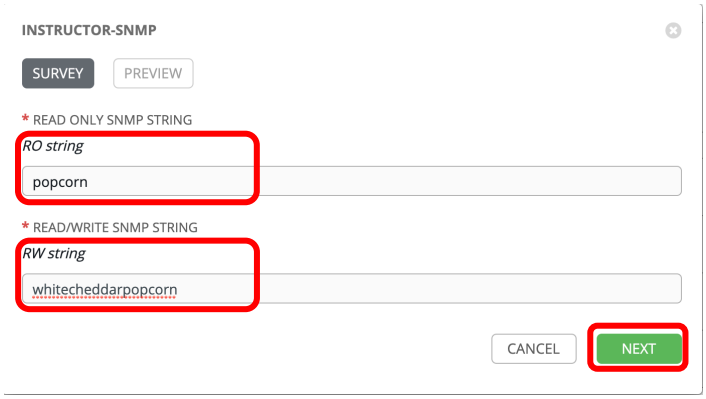


Interact with a job that has a Survey

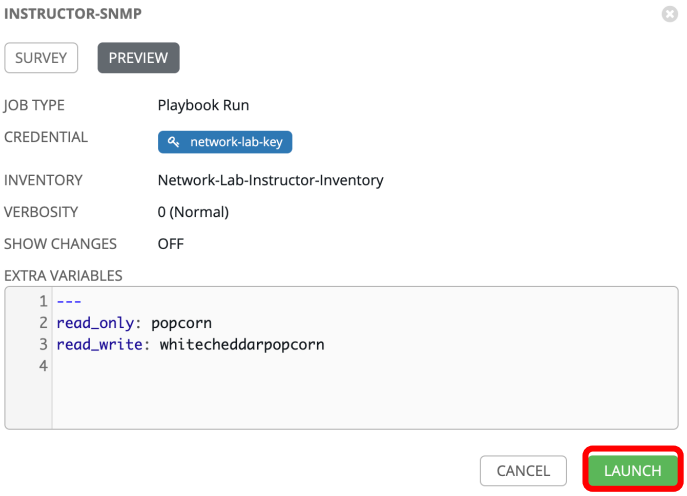
1. In “Templates”, select the rocket ship icon next to the “Instructor-SNMP” job template.



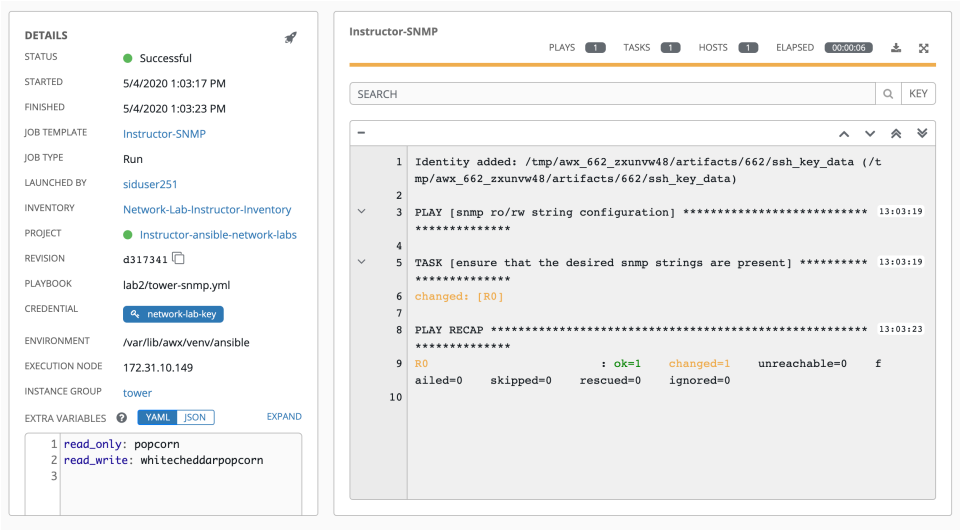
2. Fill out the survey with your own creative SNMP RO and RW strings and select “NEXT”



3. Review the job before selecting “LAUNCH”



4. Review the output which should look similar to the below:



Tip	Our general Sirius Ansible Immersion days are focused more heavily with tower. If you'd like a deeper dive into Ansible Tower, talk to your CE about when those immersion days are coming up for your area.

Success - Congratulations.

Appendix A:

Useful resource links and information

Links:

Ansible Best Practices

https://docs.ansible.com/ansible/latest/user_guide/playbooks_best_practices.html

Ansible Network troubleshooting

https://docs.ansible.com/ansible/latest/network/user_guide/network_debug_troubleshooting.html

Ansible cli_command module information

https://www.ansible.com/blog/deep-dive-on-cli-command-for-network-automation

Variable precedence

https://docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html#variable-precedence-where-should-i-put-a-variable

Additional Notes:

- Remember YAML is very sensitive to correct indentation
- **`Hostvars`** allow us to access meta-data about our inventory hosts.
- The use of an Ansible role is best practice when there is a well-defined scope with a high possibility of re-use.
- If you copy and paste text for a playbook you may get indentation issues. Ansible provides a simple syntax checker, try ansible-playbook --syntax-check backup.yml to verify. A Best Practice is to use a linter, for example ansible-review. Ansible provides excellent online documentation, which is also available from the command line, for example ansible-doc ios_config. For a full list of modules try ansible-doc –l
- There where multiple ways of implementing a playbook where specific tasks or groups of tasks execute against specific hosts. For example, we could have used 1 playbook for configuring every router in the lab utilizing the “when:” statement to ensure specific tasks are only applied to a specific router. Although this is not necessarily following best practices.
- The use of `handlers:` which can be used in any playbook. A handler is a special way of calling a task whenever an action needs to be taken after a previous task. For example, both installing and configuring an application may require a restart. A handler would be notified by both tasks but would only run once when the playbook finishes.