Lab 02 – Ansible

## Nicholas Ferguson

[GitHub](https://github.com/Nicholas-Ferguson/School/tree/main/lab-02-ansible-assignment)

## 991142920

**TABLE OF CONTENTS**

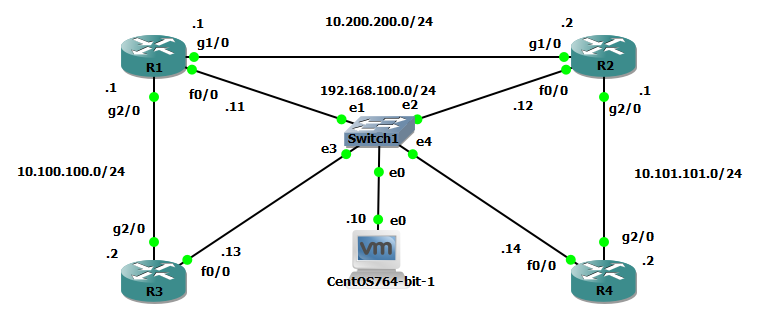
|  |  |
| --- | --- |
| *Page 1* | *Table of Contents* |
| *Page 2* | *Intro* |
| *Page 3* | *Network Deployment* |
| *Page 4* | *Inventory File* |
| *Page 5* | *Playbook Execution* |
| *Page 6* | *Playbook Code* |

**INTRO**

Ansible is a very powerful tool for automation on any type of network. This lab has introduced me to a lot of new concepts and skills to manage hosts on a network. Although I am limited to the number of hosts I can use and run on my computer, I can really start to understand the need for ansible going forward, and why so many professionals use it. When it comes to running playbooks and learning modules, it seems like the sky is the limit for ansible. The most difficult thing for me to learn was different types of variables and the syntax and placement for the different types. In this report I will go over different plays I used, different new concepts I learned, and how ansible has made the configuration and management of devices much easier.

**NETWORK DEPLOYMENT**

For my network I chose to keep it simple and use devices that I was comfortable with while I’m still learning ansible concepts. I used four Cisco 7200 routers and a CentOS 7 Virtual Machine. I did the entire lab on GNS3, including the VM. First, I added my VM to GNS3. The VM I used is a machine I had already used before for the previous exercise. I decided to use a VM instead of a docker file because my machine is a little older and I am very familiar with VM’s because of the number of things that I can customize. I am pretty good with navigating around using only CLI, and learning ansible is very similar. I then added four Cisco routers to GNS3 and connected them all together, with a switch for the management ports. I wasn’t sure what the best practice is for using an ansible workstation with multiple hosts, so I just kept it simple with a switch. After that, I logged into each router individually to configure the management ports so that I was able to ping each machine from my ansible workstation. When I was able to ping each host, I was then able to start writing my plays to configure them. The management subnet was assigned IPs in 192.168.100.0/24. The R1 and R2 subnet was assigned IPs in 10.200.200.0/24 with BGP. The R1 and R3 subnet was assigned IPs in 10.100.100.0/24 with OSPF. And R2 and R4 subnet was assigned IPs in 10.101.101.0/24 with OSPF.



**INVENTORY FILE**

My inventory was pretty straightforward because I used four routers of the same make and model. First, I defined the variables to be used for all the routers: user, password, become enable and enable password. Because I decided to add enable passwords and secrets, in order to make changes to the routers I needed to define ansible become and ansible become password as variables that the cisco module would recognize without having to type them every time. I know there is a way to ask for password input, but I decided to use a variable instead. Also, this is not a very safe way to put password variables, it is better to have them hidden in a file that is not local to the playbooks and inventory. Second, I defined cisco as a child of routers, this would be beneficial if I were to have different groups of routers that would use different ansible modules. Third, I defined a group called cisco with each router hostname and management IP address to be used individually or as a group called cisco. Fourth, I defined the variables for the group cisco, these variables are defined by cisco module to be used when I call on the hosts group cisco, so that the routers know how to handle the ansible plays properly. And fifth, I defined another group called dc1, this group just has the hostnames of the routers, I never ended up using this group but it was more for testing purposes.

**PLAYBOOK EXECUTION**

To run my plays, I decided to create individual playbooks for each type of configuration, hostname, interfaces, gather facts, etc. I also made a “parent” playbook that could run all of them at once if I decided to. For this set up I’m not sure if it is best practice or not. Each playbook has different set ups, some call on the cisco hosts group because the plays were pretty universal and could be applied to all the routers at once, others were set up for individual routers because of the specific configurations they needed. There are ways to use when loops to set up the routers with one play, but I’m not sure how to approach the syntax so that each would use a different variable, especially for unique things like interfaces, subnets and individually assigned IP addresses. Also, for playbooks that were universal, I was able to apply the playbook to specific hosts only using the --limit argument when calling on the ansible playbook. This made testing very easy without having to wait as long for it to run against all hosts. Another argument that was very useful was the --check argument. The check argument helped me complete my plays without error, it would check to make sure that playbook had the correct arguments and syntax without applying the changes. There were a couple times when I had to see what items were being changed, the check argument was useful, as well as using the verbose -v argument along with it. This gave me a step by step of the things that were being changed and the differences between the play and what was already configured. To run the playbooks, I used the command ansible-playbook and then the yaml file, after that I would add arguments depending on what I was trying to accomplish.

**PLAYBOOK CODE**

For each individual playbook I included comments describing the code in the YAML file. I will describe each playbook and then paste the code below. All playbooks and files will be attached in my assignment submission, as well as on my GitRepo <https://github.com/Nicholas-Ferguson/School/tree/main/lab-02-ansible-assignment>.

This playbook is the “parent” playbook and runs all of the configuration playbooks at once.

lab-02-ansible-assignment.yaml

---

# output to localhost for a direct output message while running playbook

- hosts: localhost

# assigning variables to use in this playbook

vars:

my\_name: Nicholas Ferguson

my\_sid: 991142920

tasks:

- debug:

# output message using the variables defined in this playbook

msg: "Lab 02 Ansible completed by {{ my\_name }}, student ID {{ my\_sid }}"

# running multiple playbooks from one playbook, can be used to run multiple playbooks or can run each playbook individually from command line

- name: Set Hostnames

include: hostname.yaml

- name: Set Interfaces

include: interfaces.yaml

- name: Set OSPF

include: ospf.yaml

- name: Set BGP

include: bgp.yaml

- name: Set Login Banner

include: banner.yaml

# write memory to cisco devices if config has been modified

- hosts: cisco

gather\_facts: no

tasks:

- name: Write Memory

cisco.ios.ios\_config:

save\_when: modified

This playbook sets the hostnames for the routers.

hostname.yaml

---

# specific hosts defined in the inventory file to run this play

- hosts: R1

gather\_facts: no

tasks:

- name: Configure R1 Hostname

# using the cisco system module to config the hostname for each individual router

cisco.ios.ios\_system:

hostname: R1

- hosts: R2

gather\_facts: no

tasks:

- name: Configure R2 Hostname

cisco.ios.ios\_system:

hostname: R2

- hosts: R3

gather\_facts: no

tasks:

- name: Configure R3 Hostname

cisco.ios.ios\_system:

hostname: R3

- hosts: R4

gather\_facts: no

tasks:

- name: Configure R4 Hostname

cisco.ios.ios\_system:

hostname: R4

This playbook configures the interfaces for all the routers.

interfaces.yaml

---

# specific hosts defined in the inventory file to run this playbook

- name: Configure R1 IPv4

hosts: R1

gather\_facts: no

tasks:

- name: Configure R1 GigabitEthernet1

# using the cisco config module to configure the interfaces on each router. each router is different so i decided to separate the tasks to individual host plays

cisco.ios.ios\_config:

lines:

- description GigabitEthernet1

- ip address 10.200.200.1 255.255.255.0

- no shutdown

# using parent to specify the interface to configure

parents: interface g1/0

- name: Configure R1 GigabitEthernet2

cisco.ios.ios\_config:

lines:

- description GigabitEthernet2

- ip address 10.100.100.1 255.255.255.0

- no shutdown

parents: interface g2/0

- name: Configure R1 Loopback0

cisco.ios.ios\_config:

lines:

- description Loopback0

- ip address 1.1.1.1 255.255.255.255

parents: interface Lo0

- name: Configure R2 IPv4

hosts: R2

gather\_facts: no

tasks:

- name: Configure R2 GigabitEthernet1

cisco.ios.ios\_config:

lines:

- description GigabitEthernet1

- ip address 10.200.200.2 255.255.255.0

- no shutdown

parents: interface g1/0

- name: Configure R2 GigabitEthernet2

cisco.ios.ios\_config:

lines:

- description GigabitEthernet2

- ip address 10.101.101.1 255.255.255.0

- no shutdown

parents: interface g2/0

- name: Configure R2 Loopback0

cisco.ios.ios\_config:

lines:

- description Loopback0

- ip address 2.2.2.2 255.255.255.255

parents: interface Lo0

- name: Configure R3 IPv4

hosts: R3

gather\_facts: no

tasks:

- name: Configure R3 GigabitEthernet2

cisco.ios.ios\_config:

lines:

- description GigabitEthernet2

- ip address 10.100.100.2 255.255.255.0

- no shutdown

parents: interface g2/0

- name: Configure R3 Loopback0

cisco.ios.ios\_config:

lines:

- description Loopback0

- ip address 3.3.3.3 255.255.255.255

parents: interface Lo0

- name: Configure R4 IPv4

hosts: R4

gather\_facts: no

tasks:

- name: Configure R4 GigabitEthernet2

cisco.ios.ios\_config:

lines:

- description GigabitEthernet2

- ip address 10.101.101.2 255.255.255.0

- no shutdown

parents: interface g2/0

- name: Configure R4 Loopback0

cisco.ios.ios\_config:

lines:

- description Loopback0

- ip address 4.4.4.4 255.255.255.255

parents: interface Lo0

This playbook configures OSPF for the routers.

ospf.yaml

---

# specific hosts defined in the inventory file to run this playbook

- hosts: R1

gather\_facts: no

tasks:

- name: Configure R1 Cisco OSPF

# use the cisco config module to configure ospf to specific commands, each host is different so i chose to separate each host into individual tasks

cisco.ios.ios\_config:

lines:

- network 10.100.100.0 0.0.0.255 area 0

- network 1.1.1.1 0.0.0.0 area 0

- redistribute bgp 65000 subnets

- redistribute connected subnets

# must use parents: when running commands without a designated module

parents: router ospf 1

- hosts: R2

gather\_facts: no

tasks:

- name: Configure R2 Cisco OSPF

cisco.ios.ios\_config:

lines:

- network 10.101.101.0 0.0.0.255 area 0

- network 2.2.2.2 0.0.0.0 area 0

- redistribute bgp 65001 subnets

- redistribute connected subnets

parents: router ospf 1

- hosts: R3

gather\_facts: no

tasks:

- name: Configure R3 Cisco OSPF

cisco.ios.ios\_config:

lines:

- network 10.100.100.0 0.0.0.255 area 0

- network 3.3.3.3 0.0.0.0 area 0

parents: router ospf 1

- hosts: R4

gather\_facts: no

tasks:

- name: Configure R4 Cisco OSPF

cisco.ios.ios\_config:

lines:

- network 10.101.101.0 0.0.0.255 area 0

- network 4.4.4.4 0.0.0.0 area 0

parents: router ospf 1

This playbook configures BGP on two of the routers.

bgp.yaml

---

# specific hosts defined in the inventory file to run this playbook

- hosts: R1

gather\_facts: no

tasks:

# using the cisco bgp module to config bgp on each specific router

- name: configure global bgp as 65000

cisco.ios.ios\_bgp:

config:

bgp\_as: 65000

router\_id: 1.1.1.1

log\_neighbor\_changes: true

neighbors:

- neighbor: 10.200.200.2

remote\_as: 65001

address\_family:

- afi: ipv4

neighbors:

- neighbor: 10.200.200.2

next\_hop\_self: yes

redistribute:

- protocol: ospf

id: 1

networks:

- prefix: 10.0.0.0

masklen: 8

operation: merge

- hosts: R2

gather\_facts: no

tasks:

- name: configure global bgp as 65001

cisco.ios.ios\_bgp:

config:

bgp\_as: 65001

router\_id: 2.2.2.2

log\_neighbor\_changes: true

neighbors:

- neighbor: 10.200.200.1

remote\_as: 65000

address\_family:

- afi: ipv4

neighbors:

- neighbor: 10.200.200.1

next\_hop\_self: yes

redistribute:

- protocol: ospf

id: 1

networks:

- prefix: 10.0.0.0

masklen: 8

operation: merge

This playbook configures the banner message on all of the routers at once using variables.

banner.yaml

---

# configure all cisco hosts defined in the inventory file to run this playbook, can use --limit while running command to specify individual hosts

- hosts: cisco

gather\_facts: yes

tasks:

- name: Configure Cisco login banner

# use the cisco banner module to config the banner

cisco.ios.ios\_banner:

banner: login

# This inputs the banner with text, using predefined ansible\_net\_ modules to input text from each routers specific variables using gather facts

text: "This is router {{ ansible\_net\_hostname }} with loopback address {{ ansible\_net\_interfaces.Loopback0.ipv4.0.address }}"

state: present

This playbook gathers facts from all of the routers.

getfacts.yaml

---

- name: gather information from routers

# specific hosts used in this playbook, can limit specific hosts with --limit argument

hosts: cisco

gather\_facts: false

tasks:

- name: gather router facts

# uses cisco ios facts module to gather facts

cisco.ios.facts:

# each variable is unique and displays the output of the specific key

# for nested variables you have to use dot form to specify the nested variable

- name: display version

debug:

msg: "The IOS version is:{{ ansible\_net\_version }}"

- name: display serial number

debug:

msg: "The serial number is:{{ ansible\_net\_serialnum }}"

- name: display model

debug:

msg: "The model is:{{ ansible\_net\_model }}"

- name: display hostname

debug:

msg: "The hostname is:{{ ansible\_net\_hostname }}"

- name: display all ipv4 address

debug:

msg: "All running ipv4 address are:{{ ansible\_net\_all\_ipv4\_addresses }}"

- name: display loopback0

debug:

# nest variable displayed

msg: "The loopback0 is:{{ ansible\_net\_interfaces.Loopback0.ipv4.0.address }}"

This playbook takes the config fact and outputs to a file.

config\_to\_file.yaml

---

# specific hosts defined in the inventory file to run this playbook, can use --limit argument to run against an individual host

- name: gather information from routers

hosts: cisco

gather\_facts: false

tasks:

# using cisco facts module to gather all facts

- name: gather router facts

cisco.ios.facts:

gather\_subset: all

# copy the running config facts to a cfg file in the current directory

- name: Copy running config to file.

copy:

# using the ansible net variable to be saved

content: "{{ ansible\_net\_config }}"

# saving the content to a local file to be saved or used later

dest: $PWD/{{ ansible\_net\_hostname }}\_runningconfig.cfg