Name: Jessie Robert Lazo	**Date Performed: 10/7/2024**
Course/Section:CPE 212-CPE31S2	**Date Submitted:10/7/2024**
Instructor: Engr. Robin Valenzuela	**Semester and SY:**

Activity 7: Managing Files and Creating Roles in Ansible

1. **Objectives:**

1.1 Manage files in remote servers

1.2 Implement roles in ansible

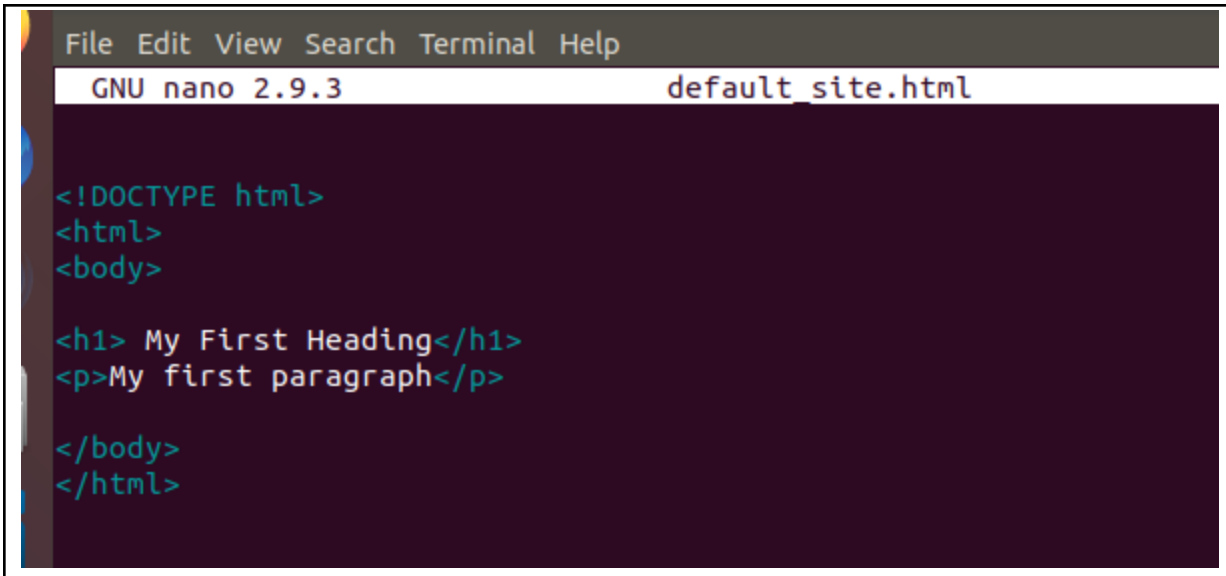
2. **Discussion**:

In this activity, we look at the concept of copying a file to a server. We are going to create a file into our git repository and use Ansible to grab that file and put it into a particular place so that we could do things like customize a default website, or maybe install a default configuration file. We will also implement roles to consolidate plays.

Task 1: Create a file and copy it to remote servers

1. Using the previous directory we created, create a directory, and named it "*files*." Create a file inside that directory and name it "*default_site.html*." Edit the file and put basic HTML syntax. Any content will do, as long as it will display text later. Save the file and exit.

```
jessielazo@Desktop:~$ ls
Desktop    Downloads         HOA6  Lazo_PrelimExam  Pictures  Templates
Documents  examples.desktop  HOA7  Music            Public    Videos
jessielazo@Desktop:~$ cd HOA7
jessielazo@Desktop:~/HOA7$ ls
files  inventory  inventory.ini  site.retry  site.yml  site.ymlcd
jessielazo@Desktop:~/HOA7$ cd files
jessielazo@Desktop:~/HOA7/files$ ls
default_site.html
jessielazo@Desktop:~/HOA7/files$
```



2. Edit the *site.yml* file and just below the *web_servers* play, create a new file to copy the default html file for site:
   - name: copy default html file for site

     tags: apache, apache2, httpd
     copy:
       src: default_site.html
       dest: /var/www/html/index.html
       owner: root
       group: root
       mode: 0644

```
GNU nano 2.9.3                          site.yml


- name: start httpd (CentOS)
  tags: apache,centos,httpd
  service:
    name: httpd
    state: started
  when: ansible_distribution == "CentOS"

- name: copy default html file for site
  tags: apache,apache2,httpd
  copy:
    src: default_site.html
    dest: /var/www/html/index.html
    owner: root
    group: root
    mode: 0644

- hosts: file_servers
```

3. Run the playbook *site.yml*. Describe the changes.

```
PLAY RECAP *********************************************************************
*
192.168.56.101             : ok=5    changed=0    unreachable=0    failed=0
192.168.56.102             : ok=5    changed=0    unreachable=0    failed=0
192.168.56.108             : ok=7    changed=1    unreachable=0    failed=0
```

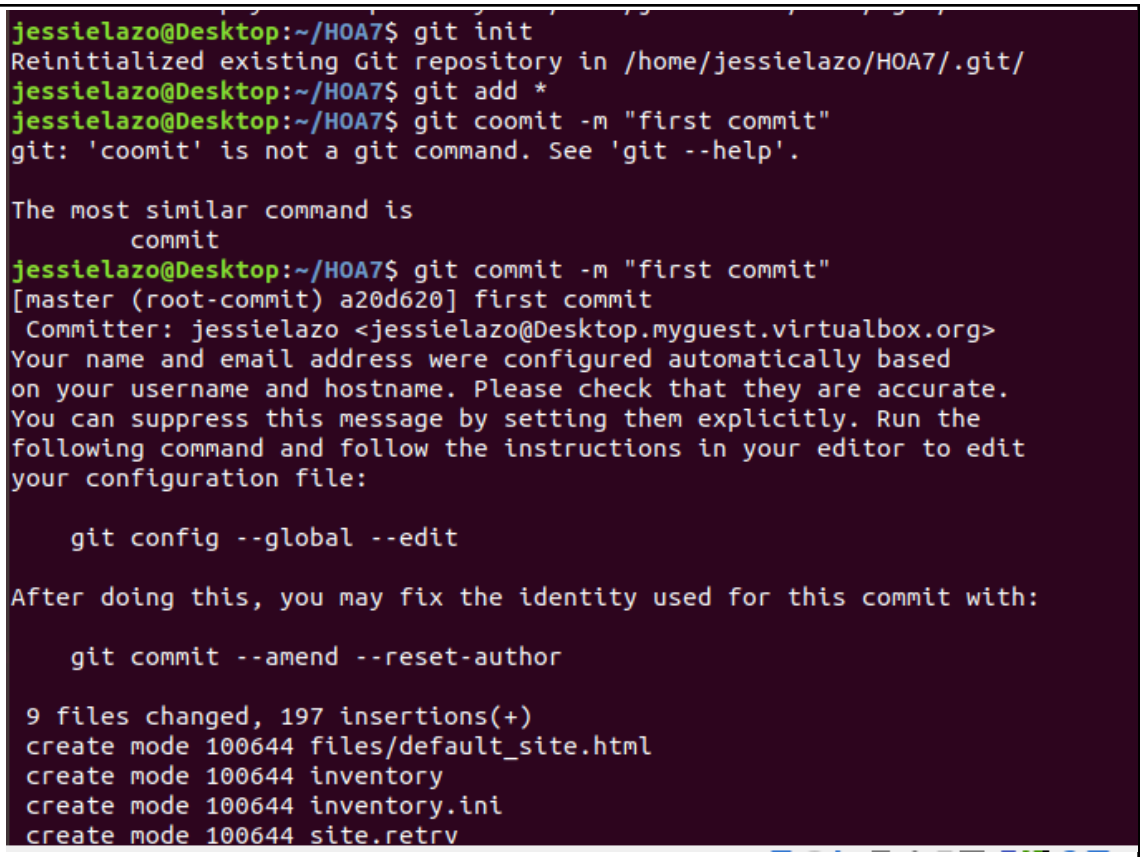
no error appears.

4. Go to the remote servers (*web_servers*) listed in your inventory. Use cat command to check if the index.html is the same as the local repository file (*default_site.html*). Do both for Ubuntu and CentOS servers. On the CentOS server, go to the browser and type its IP address. Describe the output.

```
jessieserve@Server2:~$ cat /var/www/html/index.html

<!DOCTYPE html>
<html>
<body>

<h1> My First Heading</h1>
<p>My first paragraph</p>

</body>
</html>

jessieserve@Server2:~$
```

```
File  Edit  View  Search  Terminal  Help
jessieserver2@Server2:~$ cat /var/www/html/index.html

<!DOCTYPE html>
<html>
<body>

<h1> My First Heading</h1>
<p>My first paragraph</p>

</body>
</html>

jessieserver2@Server2:~$
```

5. Sync your local repository with GitHub and describe the changes.



Task 2: Download a file and extract it to a remote server

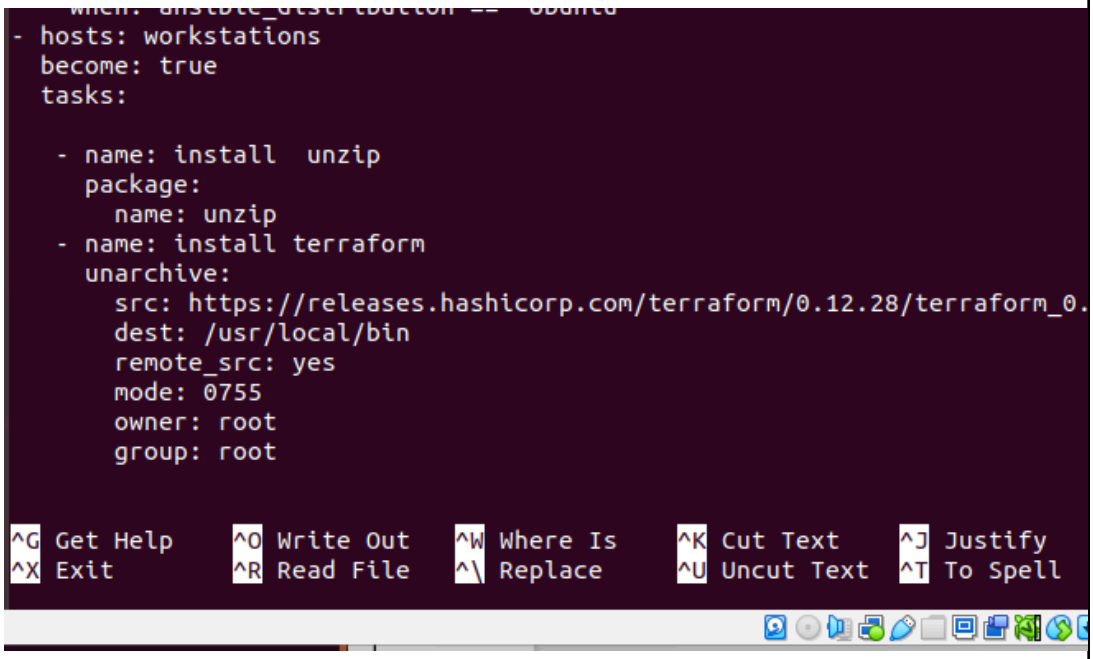
1. Edit the site.yml. Just before the web_servers play, create a new play:
   - hosts: workstations
     become: true
     tasks:

     - name: install unzip
       package:
         name: unzip

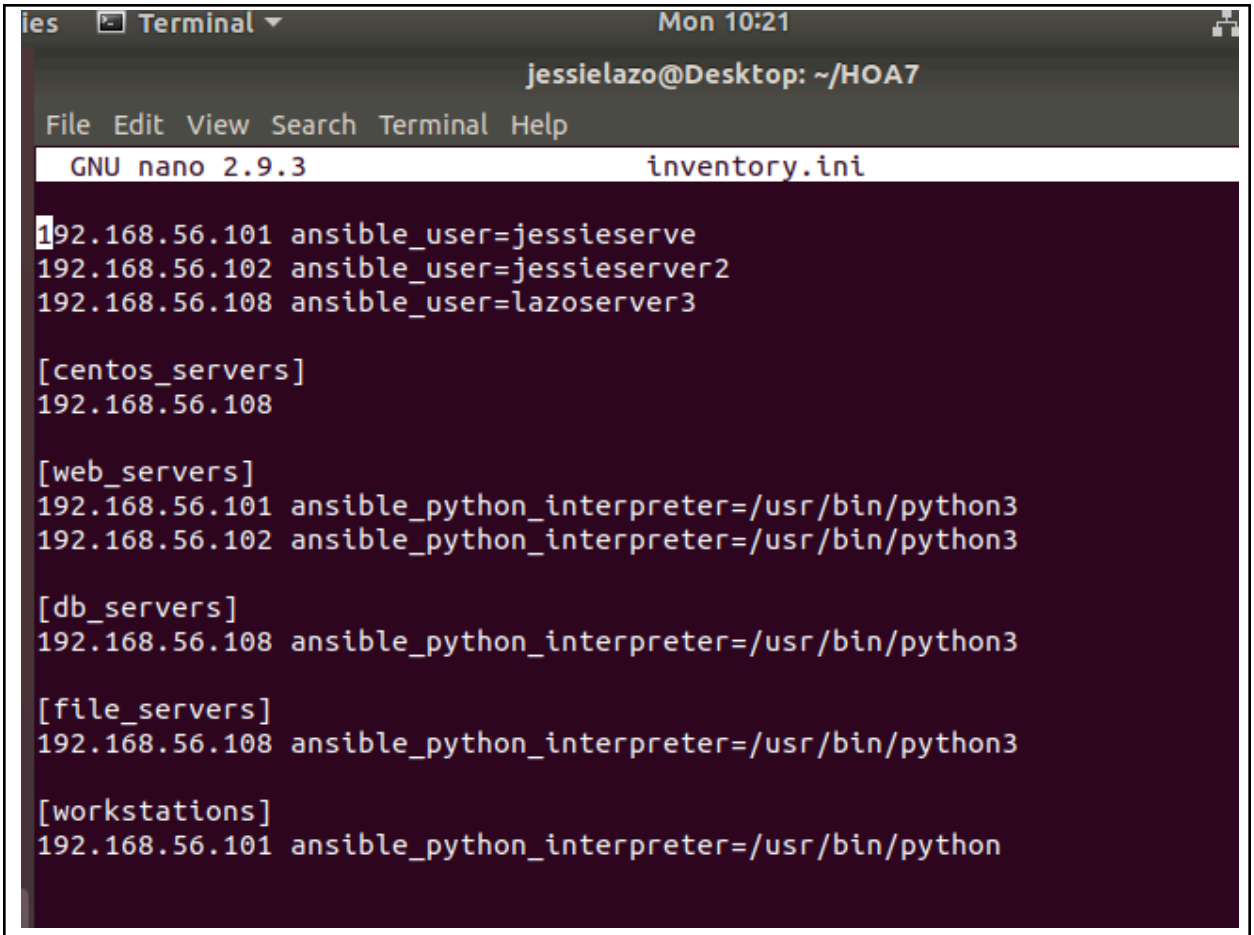
     - name: install terraform
       unarchive:
         src: https://releases.hashicorp.com/terraform/0.12.28/terraform_0.12.28_linux_amd64.zip
         dest: /usr/local/bin
         remote_src: yes
         mode: 0755
         owner: root

group: root

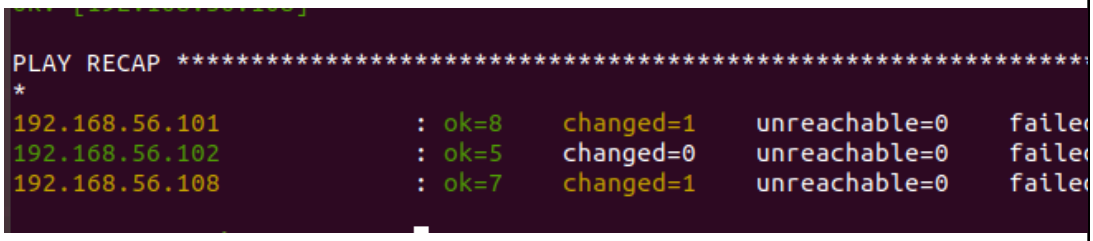
no error terraform has been installed.



2. Edit the inventory file and add workstations group. Add any Ubuntu remote server. Make sure to remember the IP address.

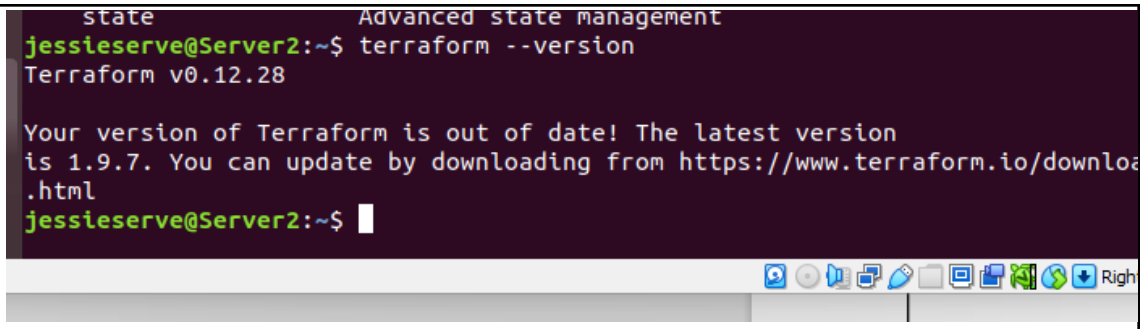


3. Run the playbook. Describe the output.



no error, terraform has been installed.

4. On the Ubuntu remote workstation, type terraform to verify installation of terraform. Describe the output.

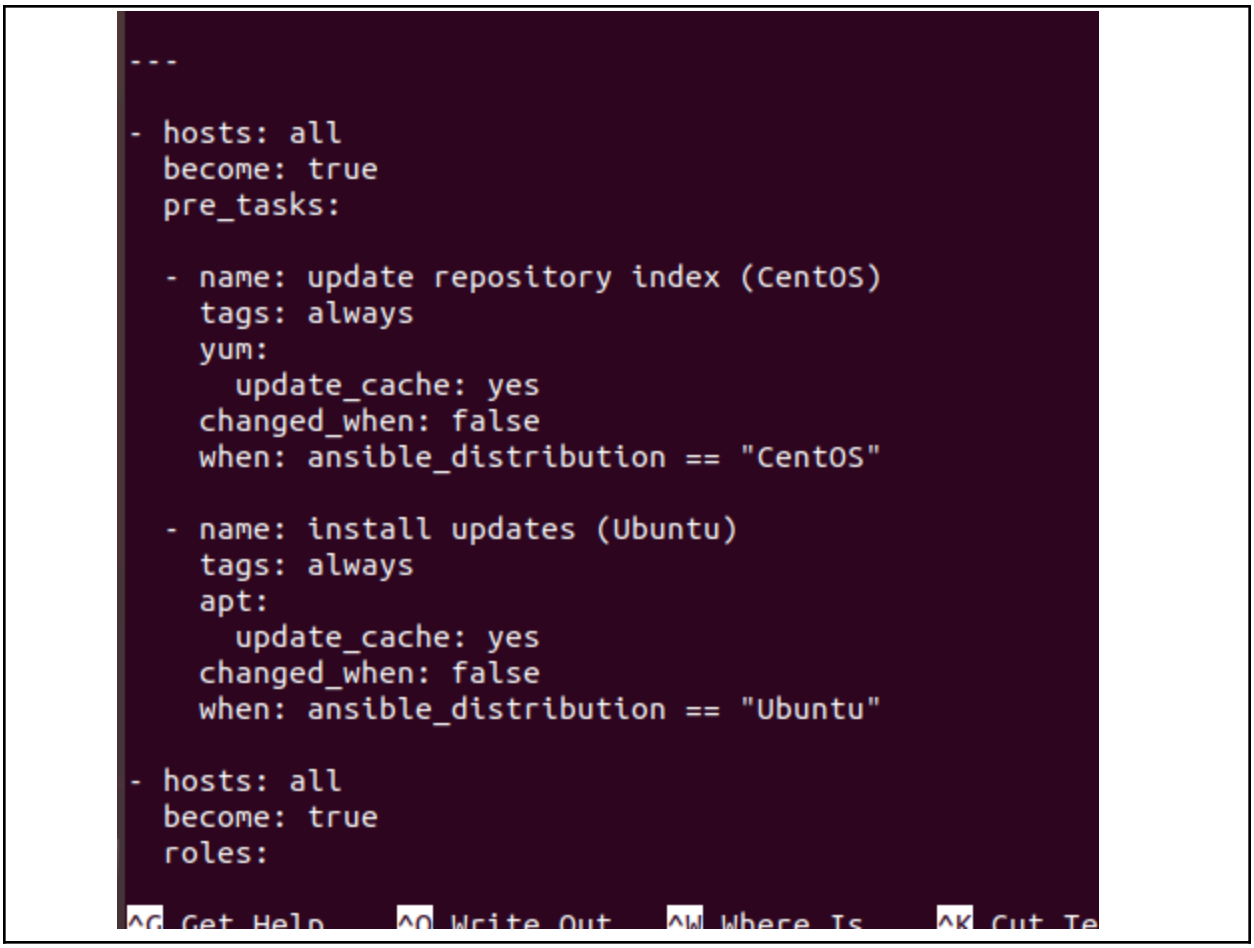


terraform has been successfully installed.

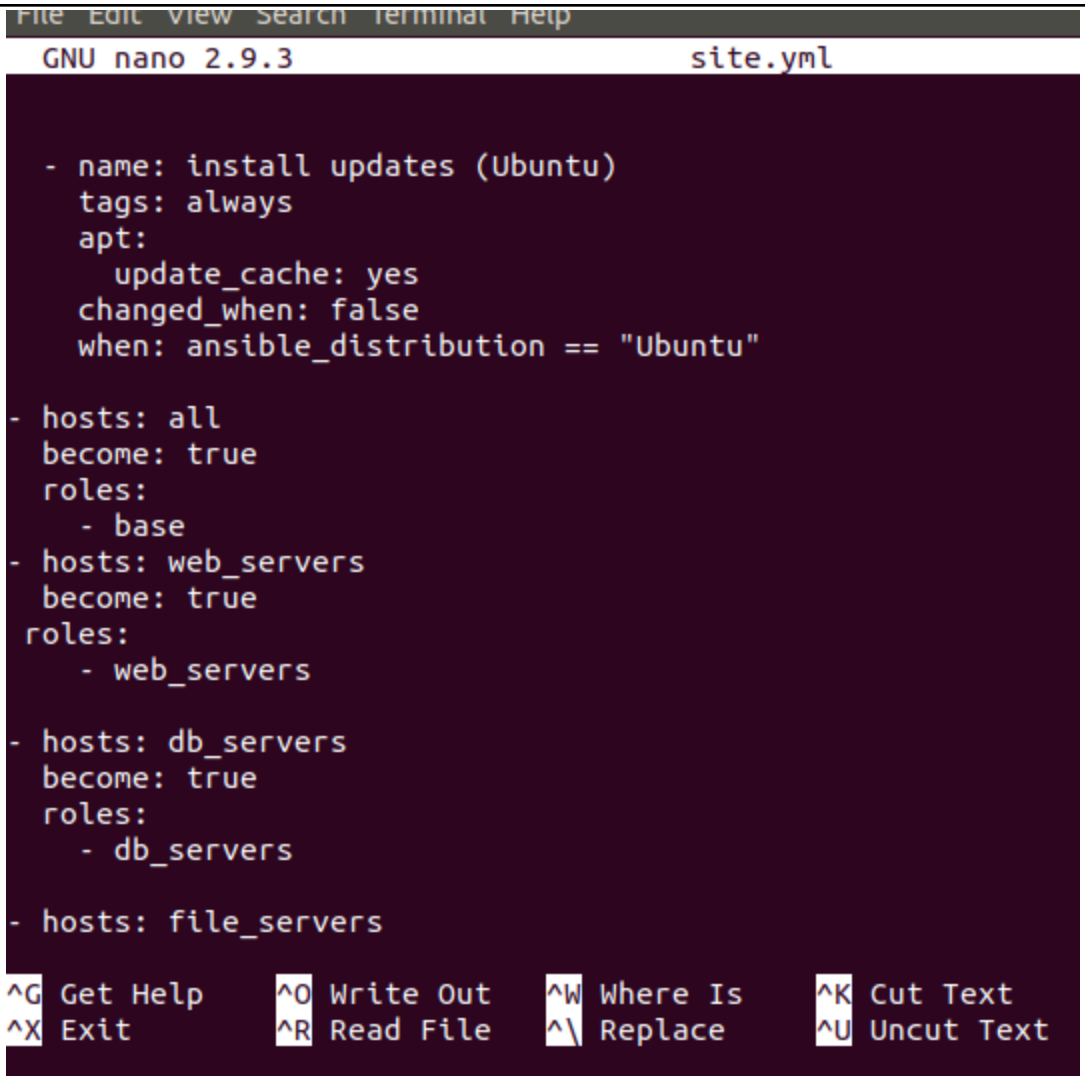
Task 3: Create roles

1. Edit the site.yml. Configure roles as follows: (make sure to create a copy of the old site.yml file because you will be copying the specific plays for all groups)

```
---
- hosts: all
  become: true
  pre_tasks:

  - name: update repository index (CentOS)
    tags: always
    dnf:
      update_cache: yes
    changed_when: false
    when: ansible_distribution == "CentOS"
  - name: install updates (Ubuntu)
    tags: always
    apt:
      update_cache: yes
    changed_when: false
    when: ansible_distribution == "Ubuntu"

- hosts: all
  become: true
  roles:
    -  base

- hosts: workstations
  become: true
  roles:
    - workstations

- hosts: web_servers
  become: true
  roles:
    - web_servers

- hosts: db_servers
  become: true
  roles:
    - db_servers

- hosts: file_servers
  become: true
  roles:
    - file_servers
```

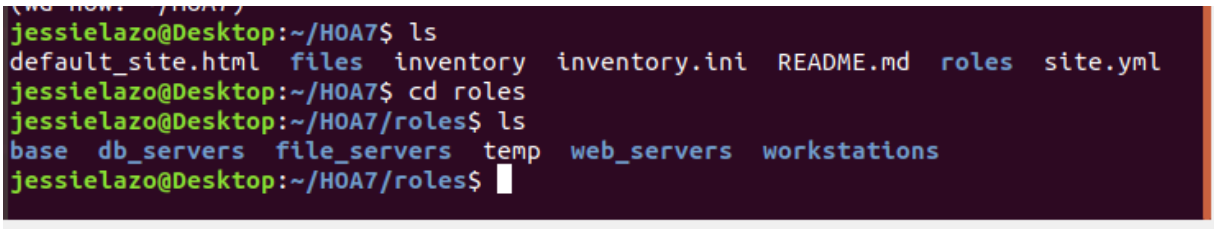


```
---

- hosts: all
  become: true
  pre_tasks:

  - name: update repository index (CentOS)
    tags: always
    yum:
      update_cache: yes
    changed_when: false
    when: ansible_distribution == "CentOS"

  - name: install updates (Ubuntu)
    tags: always
    apt:
      update_cache: yes
    changed_when: false
    when: ansible_distribution == "Ubuntu"

- hosts: all
  become: true
  roles:

^G Get Help    ^O Write Out    ^W Where Is    ^K Cut Te
```



Save the file and exit.

2. Under the same directory, create a new directory and name it roles. Enter the roles directory and create new directories: base, web_servers, file_servers, db_servers and workstations. For each directory, create a directory and name it tasks.

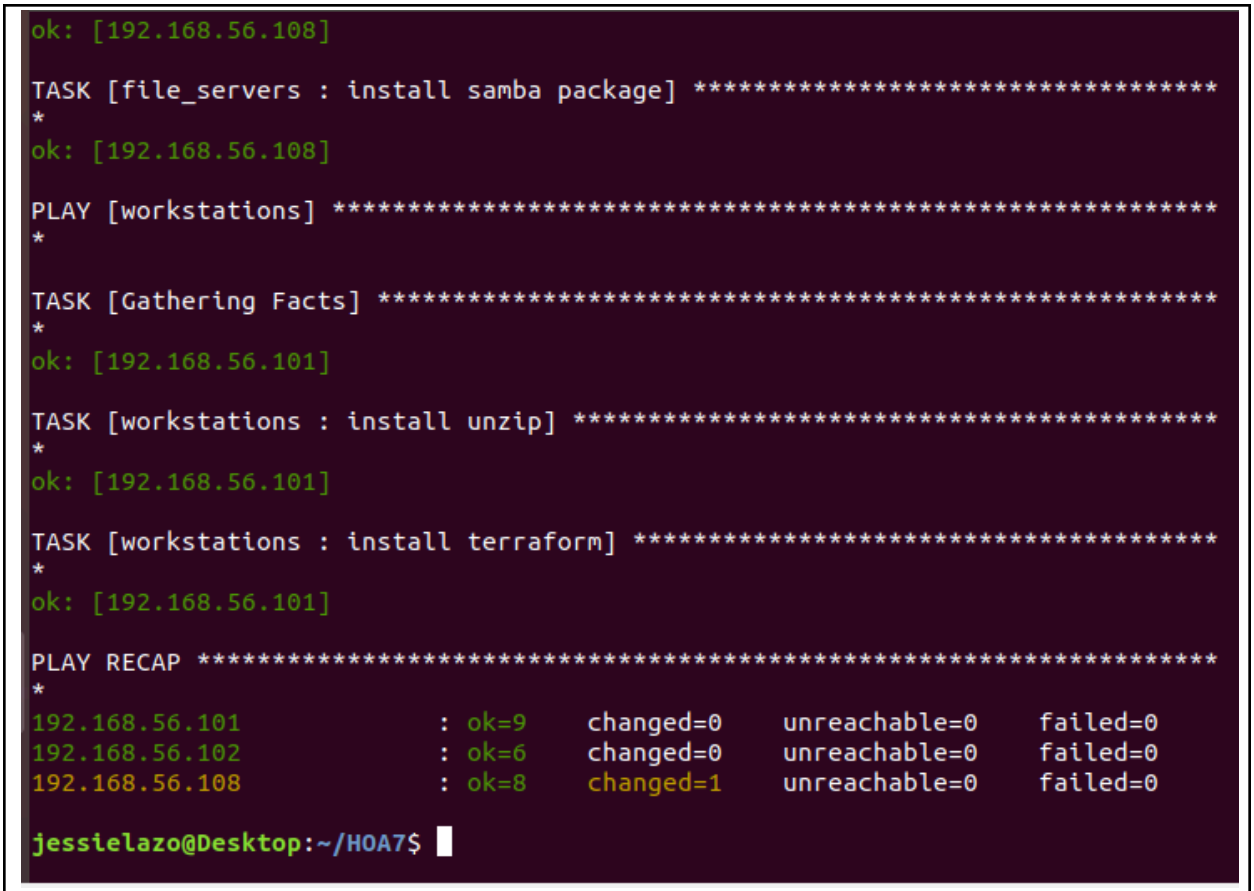


3. Go to tasks for all directory and create a file. Name it main.yml. In each of the tasks for all directories, copy and paste the code from the old site.yml file. Show all contents of main.yml files for all tasks.

```
jessielazo@Desktop:~/HOA7/roles$ ls
base  db_servers  file_servers  temp  web_servers  workstations
jessielazo@Desktop:~/HOA7/roles$ cd base
jessielazo@Desktop:~/HOA7/roles/base$ ls
tasks
jessielazo@Desktop:~/HOA7/roles/base$ cd ..
jessielazo@Desktop:~/HOA7/roles$ cd db_servers
jessielazo@Desktop:~/HOA7/roles/db_servers$ ls
tasks
jessielazo@Desktop:~/HOA7/roles/db_servers$ cd ..
jessielazo@Desktop:~/HOA7/roles$ cd file_servers
jessielazo@Desktop:~/HOA7/roles/file_servers$ ls
tasks
jessielazo@Desktop:~/HOA7/roles/file_servers$ cd ..
jessielazo@Desktop:~/HOA7/roles$ cd web_servers
jessielazo@Desktop:~/HOA7/roles/web_servers$ ls
tasks
jessielazo@Desktop:~/HOA7/roles/web_servers$ cd ..
jessielazo@Desktop:~/HOA7/roles$ cd workstations
jessielazo@Desktop:~/HOA7/roles/workstations$ ls
tasks
jessielazo@Desktop:~/HOA7/roles/workstations$
```

I have already created the directories and now just showing all of the directories that were made by using mkdir.

4. Run the site.yml playbook and describe the output.



Upon making a lot of configurations and directories, the playbook successfully runs without errors.

Reflections:

Answer the following:

1. What is the importance of creating roles?

The importance of making roles is that it promotes a more organized management of files and commands for each existing group and role.

2. What is the importance of managing files?

Imagine making a playbook file with so much contents, package installation, and a lot of different machines. When you do not manage files and organize codes with roles, it is much more difficult to debug for errors and find where you are lacking and which line is the source of conflict error.

https://github.com/Jessie-Lazo/HOA7/tree/main