Name: Hazel Manuel	**Date Performed: 10/3**
Course/Section: CPE31S4	**Date Submitted: 10/3**
Instructor: Engr. Valenzuela	**Semester and SY: 1st**

Activity 7: Managing Files and Creating Roles in Ansible

1. Objectives:

1.1 Manage files in remote servers

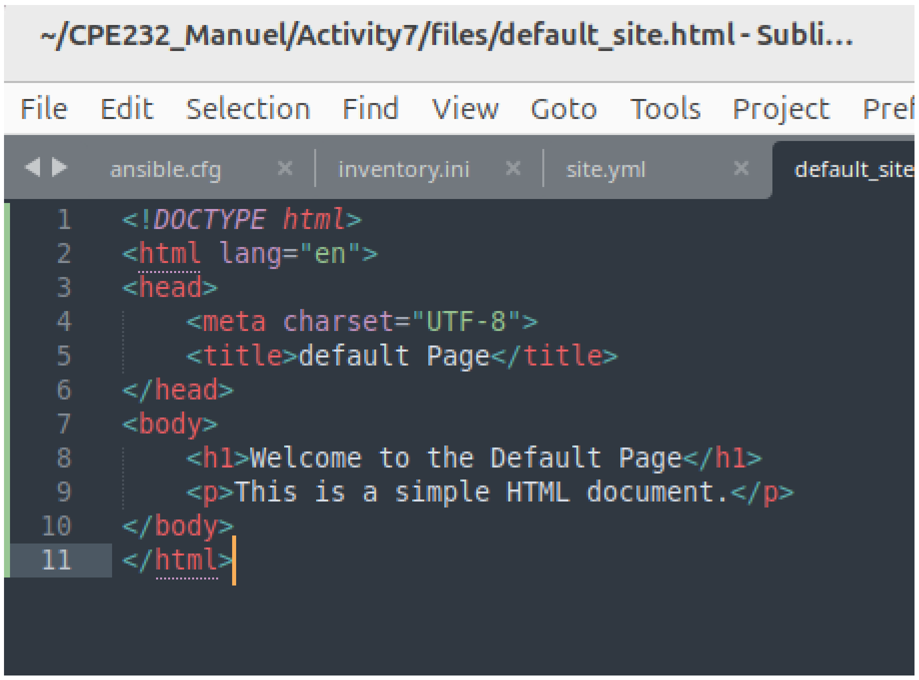
1.2 Implement roles in ansible

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.

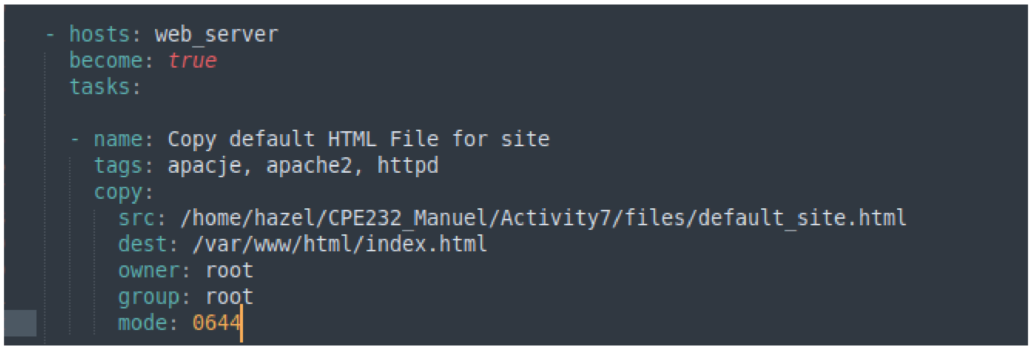

~/CPE232_Manuel/Activity7/files/default_site.html - Subli...
File Edit Selection Find View Goto Tools Project Pref
ansible.cfg
inventory.ini
site.yml
default_site

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>default Page</title>
</head>
<body>
    <h1>Welcome to the Default Page</h1>
    <p>This is a simple HTML document.</p>
</body>
</html>
```

Figure 1: Creating an HTML with basic syntax in it.

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:

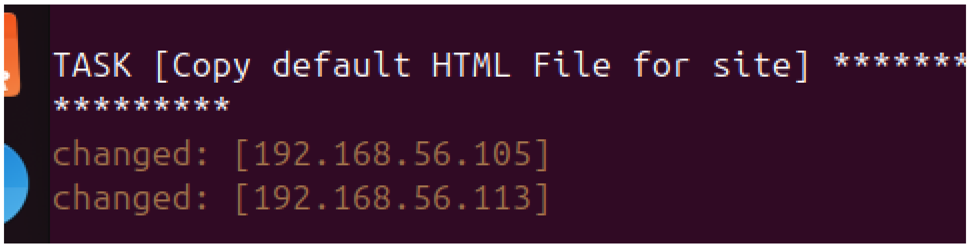
```
- name: copy default html file for site

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



```
- hosts: web_server
  become: true
  tasks:

  - name: Copy default HTML File for site
    tags: apacje, apache2, httpd
    copy:
      src: /home/hazel/CPE232_Manuel/Activity7/files/default_site.html
      dest: /var/www/html/index.html
      owner: root
      group: root
      mode: 0644
```

Figure 2: Copying a file from a local machine to a remote one.

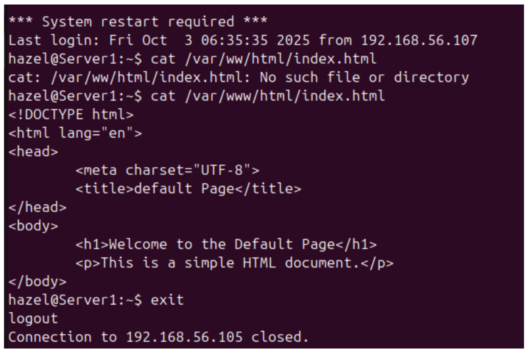
3. Run the playbook *site.yml*. Describe the changes.



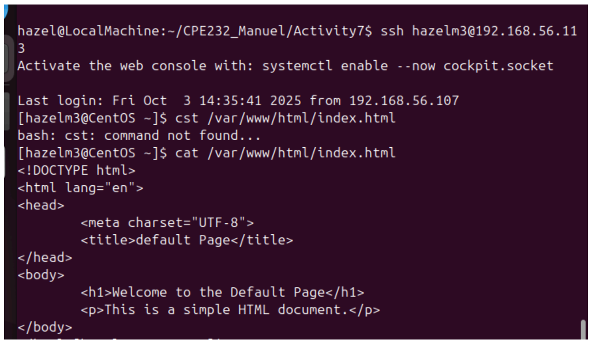
```
TASK [Copy default HTML File for site] *******
*********
changed: [192.168.56.105]
changed: [192.168.56.113]
```

Figure 3: New task was added upon running and indicates that the task is run successfully.

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.



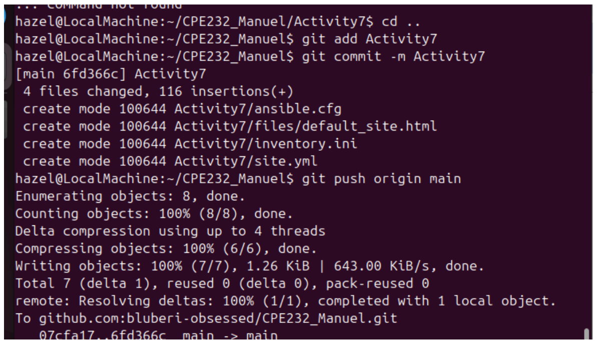
```
*** System restart required ***
Last login: Fri Oct  3 06:35:35 2025 from 192.168.56.107
hazel@Server1:~$ cat /var/ww/html/index.html
cat: /var/ww/html/index.html: No such file or directory
hazel@Server1:~$ cat /var/www/html/index.html
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <title>default Page</title>
</head>
<body>
        <h1>Welcome to the Default Page</h1>
        <p>This is a simple HTML document.</p>
</body>
hazel@Server1:~$ exit
logout
Connection to 192.168.56.105 closed.
```



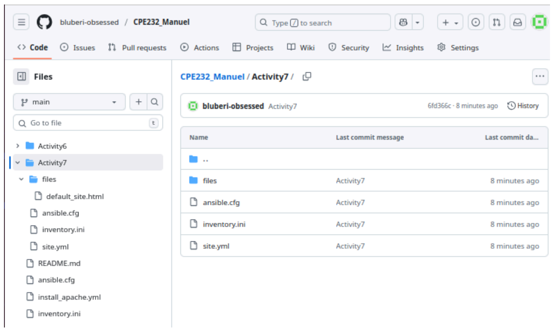
```
hazel@LocalMachine:~/CPE232_Manuel/Activity7$ ssh hazelm3@192.168.56.11
3
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Fri Oct  3 14:35:41 2025 from 192.168.56.107
[hazelm3@CentOS ~]$ cst /var/www/html/index.html
bash: cst: command not found...
[hazelm3@CentOS ~]$ cat /var/www/html/index.html
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <title>default Page</title>
</head>
<body>
        <h1>Welcome to the Default Page</h1>
        <p>This is a simple HTML document.</p>
</body>
```

Figure: Checking the servers in the local machine using ssh. This allowed me to confirm that the task to copy the html to different workstations indeed work.

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



```
hazel@LocalMachine:~/CPE232_Manuel/Activity7$ cd ..
hazel@LocalMachine:~/CPE232_Manuel$ git add Activity7
hazel@LocalMachine:~/CPE232_Manuel$ git commit -m Activity7
[main 6fd366c] Activity7
 4 files changed, 116 insertions(+)
 create mode 100644 Activity7/ansible.cfg
 create mode 100644 Activity7/files/default_site.html
 create mode 100644 Activity7/inventory.ini
 create mode 100644 Activity7/site.yml
hazel@LocalMachine:~/CPE232_Manuel$ git push origin main
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (7/7), 1.26 KiB | 643.00 KiB/s, done.
Total 7 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:bluberi-obsessed/CPE232_Manuel.git
   07cfa17..6fd366c  main -> main
```



bluberi-obsessed Activity7 6fd366c · 8 minutes ago

Name	Last commit message	Last commit da...
..		
files	Activity7	8 minutes ago
ansible.cfg	Activity7	8 minutes ago
inventory.ini	Activity7	8 minutes ago
site.yml	Activity7	8 minutes ago

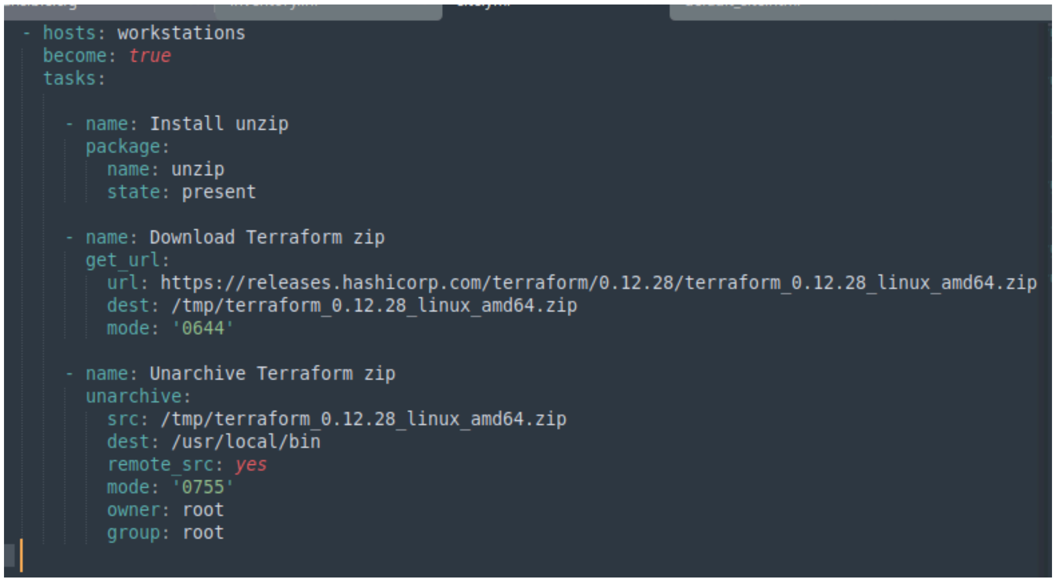
Figure 5: Syncing the current changes we have done to the local repository to github.

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
```

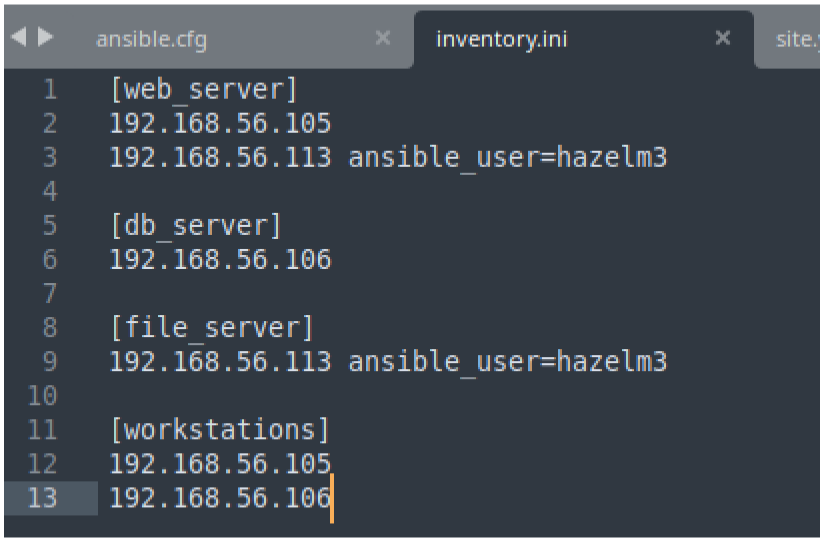
```
dest: /usr/local/bin
remote_src: yes
mode: 0755
owner: root
group: root
```



```yaml
- hosts: workstations
  become: true
  tasks:

    - name: Install unzip
      package:
        name: unzip
        state: present

    - name: Download Terraform zip
      get_url:
        url: https://releases.hashicorp.com/terraform/0.12.28/terraform_0.12.28_linux_amd64.zip
        dest: /tmp/terraform_0.12.28_linux_amd64.zip
        mode: '0644'

    - name: Unarchive Terraform zip
      unarchive:
        src: /tmp/terraform_0.12.28_linux_amd64.zip
        dest: /usr/local/bin
        remote_src: yes
        mode: '0755'
        owner: root
        group: root
```

Figure 6: Creating a new play where it installs a file and extract it.

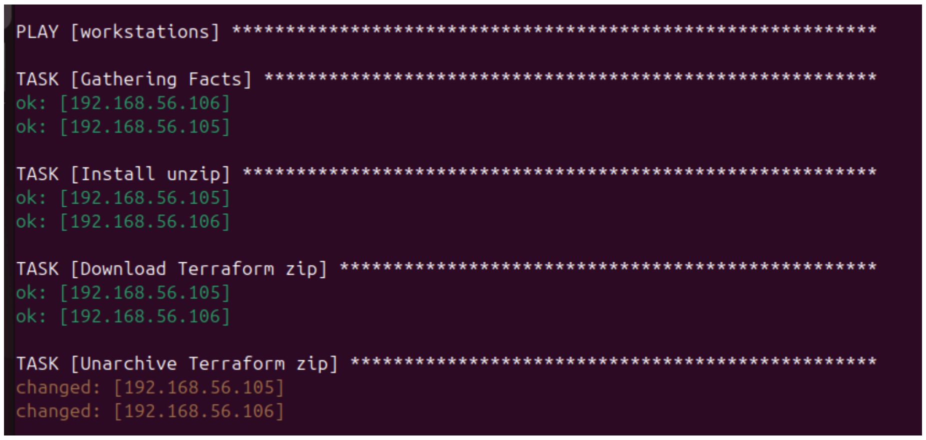
2. Edit the inventory file and add workstations group. Add any Ubuntu remote server. Make sure to remember the IP address.



```
[web_server]
192.168.56.105
192.168.56.113 ansible_user=hazelm3

[db_server]
192.168.56.106

[file_server]
192.168.56.113 ansible_user=hazelm3

[workstations]
192.168.56.105
192.168.56.106
```

Figure 7: adding a new group in the inventory file

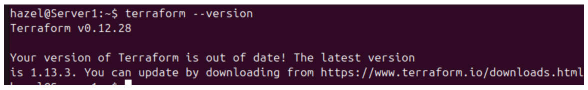
3. Run the playbook. Describe the output.



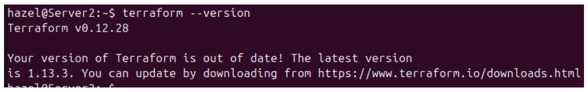
```
PLAY [workstations] ************************************************************

TASK [Gathering Facts] *********************************************************
ok: [192.168.56.106]
ok: [192.168.56.105]

TASK [Install unzip] ***********************************************************
ok: [192.168.56.105]
ok: [192.168.56.106]

TASK [Download Terraform zip] **************************************************
ok: [192.168.56.105]
ok: [192.168.56.106]

TASK [Unarchive Terraform zip] *************************************************
changed: [192.168.56.105]
changed: [192.168.56.106]
```

Figure 8: It shows that it is able to download the unzip and the zip file of Terraform to the remote servers. After that it unarchives it to fully install the package

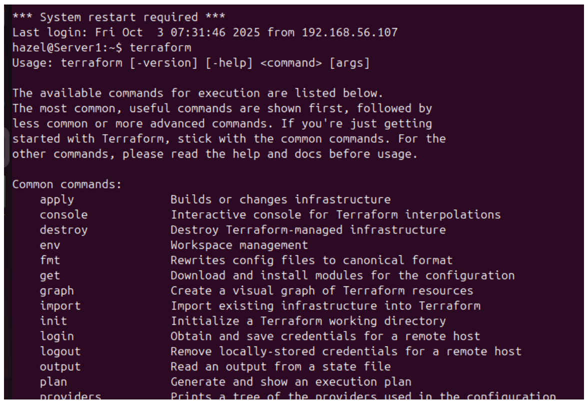
4. On the Ubuntu remote workstation, type terraform to verify installation of terraform. Describe the output.



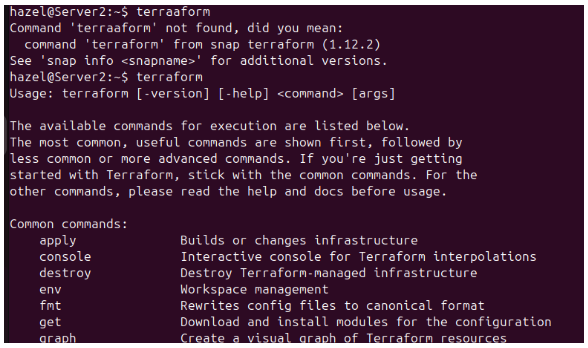
```
hazel@Server1:~$ terraform --version
Terraform v0.12.28

Your version of Terraform is out of date! The latest version
is 1.13.3. You can update by downloading from https://www.terraform.io/downloads.html
```



```
hazel@Server2:~$ terraform --version
Terraform v0.12.28

Your version of Terraform is out of date! The latest version
is 1.13.3. You can update by downloading from https://www.terraform.io/downloads.html
```



```
*** System restart required ***
Last login: Fri Oct  3 07:31:46 2025 from 192.168.56.107
hazel@Server1:~$ terraform
Usage: terraform [-version] [-help] <command> [args]

The available commands for execution are listed below.
The most common, useful commands are shown first, followed by
less common or more advanced commands. If you're just getting
started with Terraform, stick with the common commands. For the
other commands, please read the help and docs before usage.

Common commands:
    apply              Builds or changes infrastructure
    console            Interactive console for Terraform interpolations
    destroy            Destroy Terraform-managed infrastructure
    env                Workspace management
    fmt                Rewrites config files to canonical format
    get                Download and install modules for the configuration
    graph              Create a visual graph of Terraform resources
    import             Import existing infrastructure into Terraform
    init               Initialize a Terraform working directory
    login              Obtain and save credentials for a remote host
    logout             Remove locally-stored credentials for a remote host
    output             Read an output from a state file
    plan               Generate and show an execution plan
    providers          Prints a tree of the providers used in the configuration
```

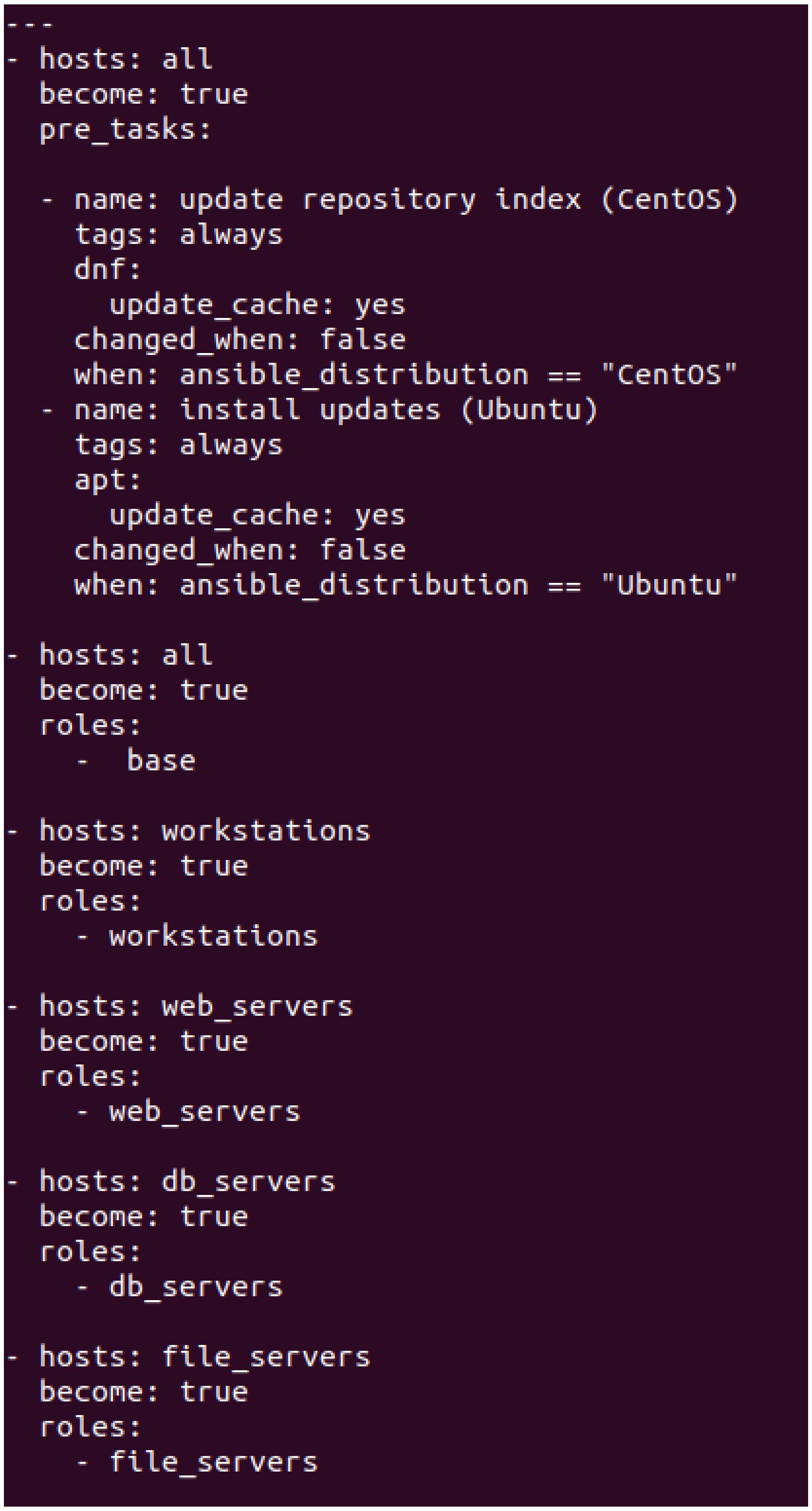


```
hazel@Server2:~$ terraaform
Command 'terraaform' not found, did you mean:
  command 'terraform' from snap terraform (1.12.2)
See 'snap info <snapname>' for additional versions.
hazel@Server2:~$ terraform
Usage: terraform [-version] [-help] <command> [args]

The available commands for execution are listed below.
The most common, useful commands are shown first, followed by
less common or more advanced commands. If you're just getting
started with Terraform, stick with the common commands. For the
other commands, please read the help and docs before usage.

Common commands:
    apply              Builds or changes infrastructure
    console            Interactive console for Terraform interpolations
    destroy            Destroy Terraform-managed infrastructure
    env                Workspace management
    fmt                Rewrites config files to canonical format
    get                Download and install modules for the configuration
    graph              Create a visual graph of Terraform resources
```

Figure 9: This shows us that the terraform has been extracted successfully. The images above show the common commands of terraform as well as its version.

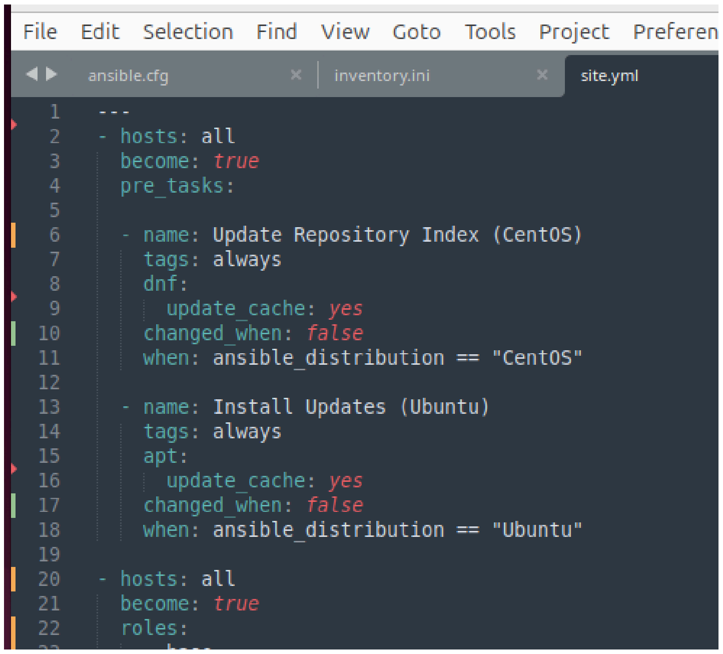
Task 3: Create roles

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
```

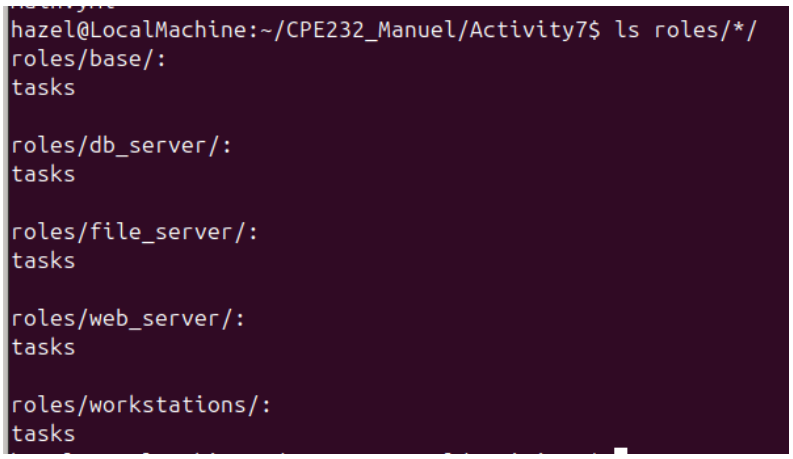
Save the file and exit.



```
---
- 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:
```

Figure 10: Modifying the ansible playbook to create roles.

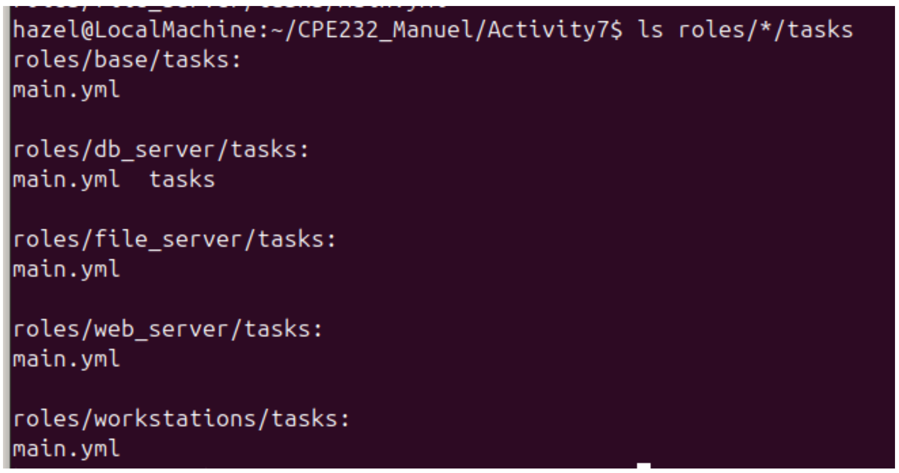
2. Under the same directory, create a new directory and name it roles. Enter the roles directory and create new directories: base, web_servers, file_servers, db_servers and workstations. For each directory, create a directory and name it tasks.



```
hazel@LocalMachine:~/CPE232_Manuel/Activity7$ ls roles/*/
roles/base/:
tasks

roles/db_server/:
tasks

roles/file_server/:
tasks

roles/web_server/:
tasks

roles/workstations/:
tasks
```

Figure 11: Creating a new directory for roles and have each role have a tasks directory.

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.



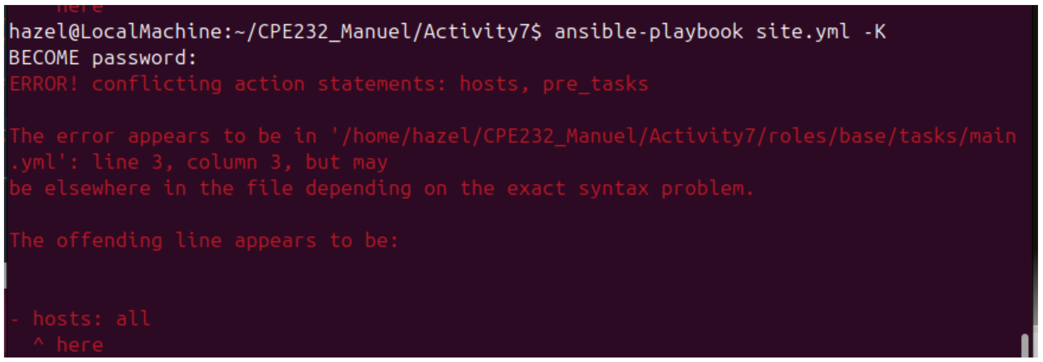
```
hazel@LocalMachine:~/CPE232_Manuel/Activity7$ ls roles/*/tasks
roles/base/tasks:
main.yml

roles/db_server/tasks:
main.yml  tasks

roles/file_server/tasks:
main.yml

roles/web_server/tasks:
main.yml

roles/workstations/tasks:
main.yml
```

Figure 12: Copying old_site.yml and rename it to main.yml to all of the roles

Commands

```
for dir in roles/*/tasks/; do
  cp old_site.yml "$dir"
done

for dir in roles/*/tasks/; do
  mv "$dir/old_site.yml" "$dir/main.yml"
done
```

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



```
hazel@LocalMachine:~/CPE232_Manuel/Activity7$ ansible-playbook site.yml -K
BECOME password:
ERROR! conflicting action statements: hosts, pre_tasks

The error appears to be in '/home/hazel/CPE232_Manuel/Activity7/roles/base/tasks/main
.yml': line 3, column 3, but may
be elsewhere in the file depending on the exact syntax problem.

The offending line appears to be:


- hosts: all
  ^ here
```

Figure 13: Results to an error as the playbook has many plays in one yml file. The cause of this errors is because there is too many "- hosts:" which resulted to having parsing errors.

Reflections:

Answer the following:

1. What is the importance of creating roles?

   Roles play a huge role as this allows us to set and manage the user’s permissions and access control. This allows for the machine to be much more secure

2. What is the importance of managing files?

   Managing file is crucial as this allows us to easily find certain files that are important. This can also be helpful when we manage permissions of files