Name: Alexis Neil D. Deniega	**Date Performed:** October 3rd, 2025
Course/Section: CPE31S4	**Date Submitted:** October 3rd, 2025
Instructor: Engr. Robin Valenzuela	**Semester and SY:** SY 2025-26, Sem 1

Activity 7: Managing Files and Creating Roles in Ansible

1. **Objectives:**

1.1 Manage files in remote servers

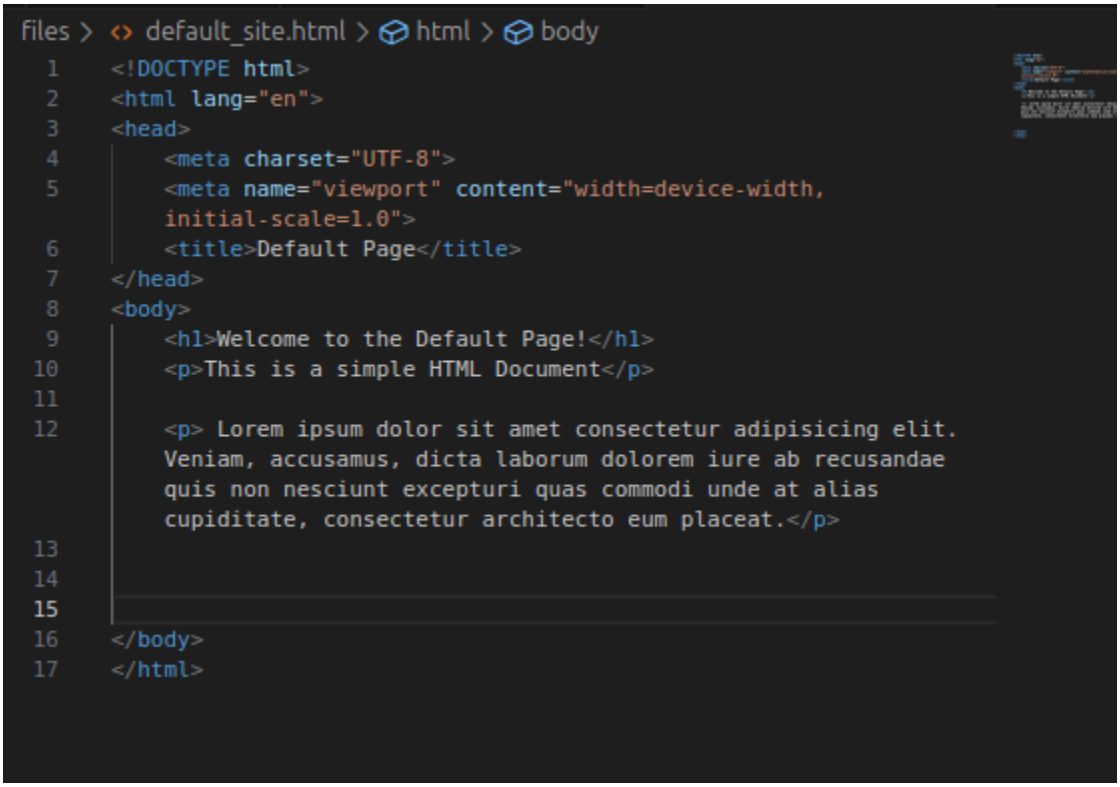
1.2 Implement roles in ansible

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.



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

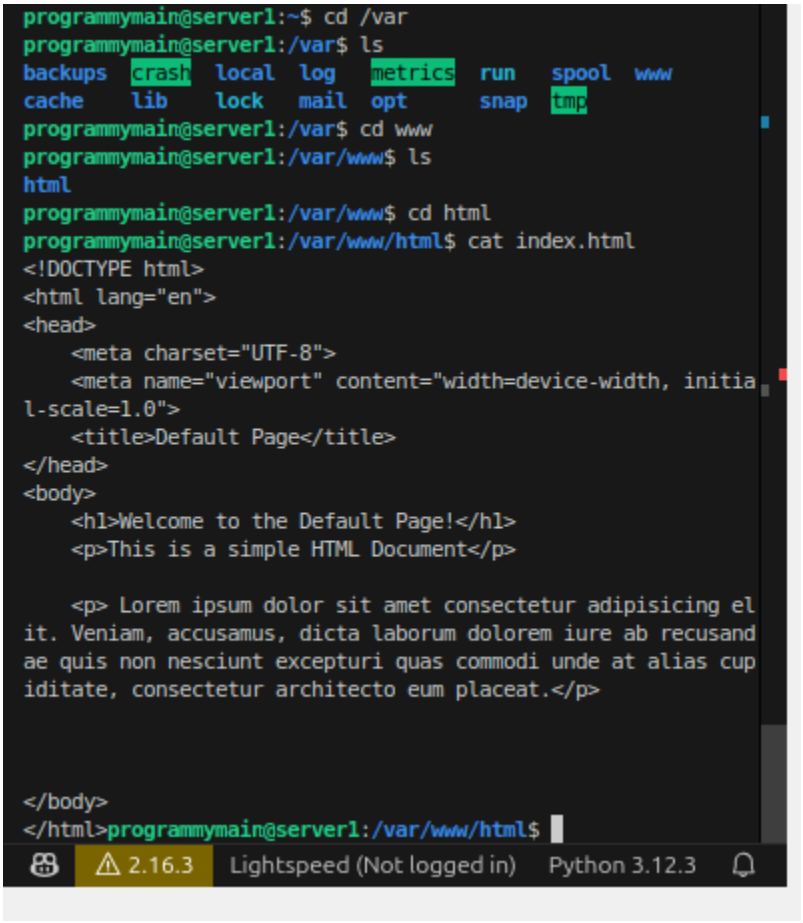
```
    service:
      name: httpd
      state: started
      enabled: true
    when: ansible_distribution == "CentOS"

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

- hosts: db_servers
  become: true
  tasks:
  - name: install mariadb package (centos)
    tags: centos, db, mariadb
```

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

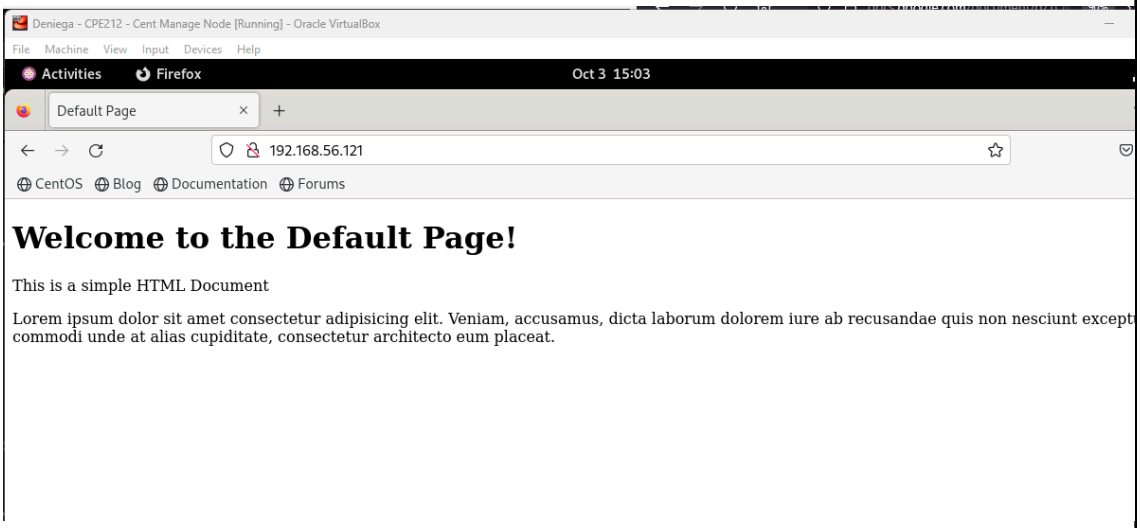
```
TASK [start httpd (centos)] ********************************
********************
skipping: [192.168.56.116]
skipping: [192.168.56.117]
ok: [192.168.56.121]

TASK [copy default html file for site] **********************
********************
changed: [192.168.56.121]
changed: [192.168.56.116]
changed: [192.168.56.117]

PLAY [db_servers] ******************************************
********************

TASK [Gathering Facts] *************************************
********************
```

It says changed.

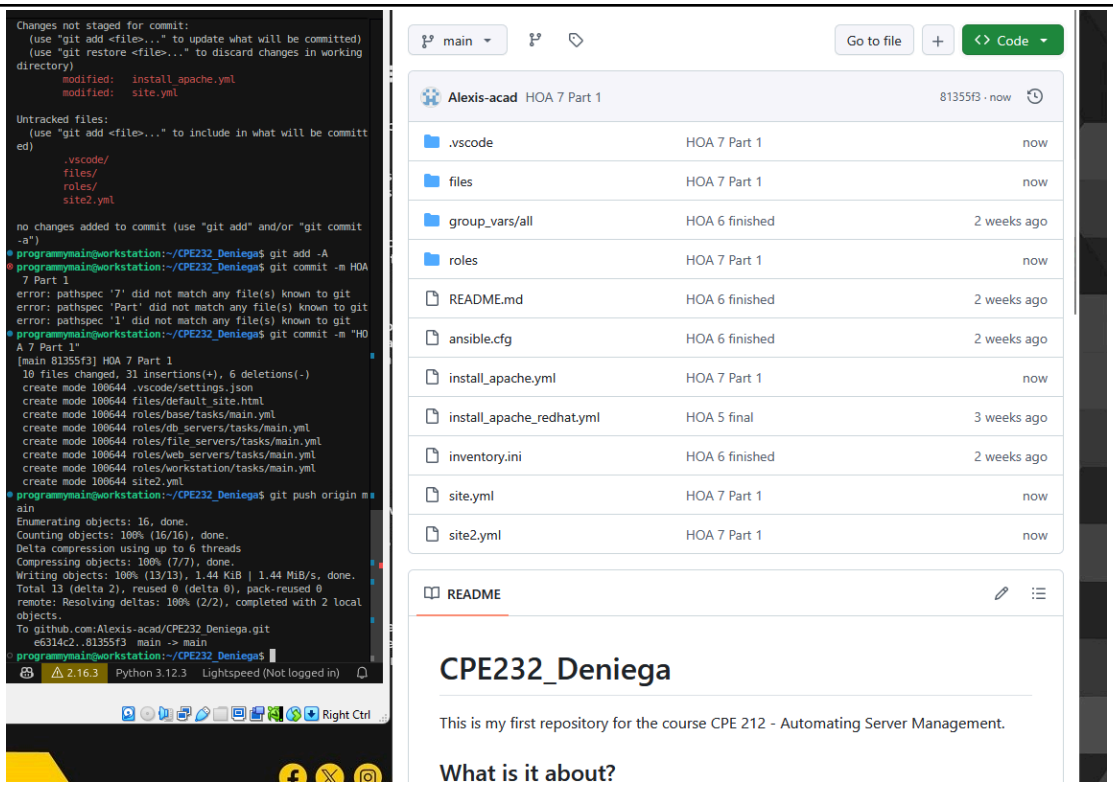
4. Go to the remote servers (*web_servers*) listed in your inventory. Use cat command to check if the index.html is the same as the local repository file (*default_site.html*). Do both for Ubuntu and CentOS servers. On the CentOS server, go to the browser and type its IP address. Describe the output.



```
Last login: Fri Oct  3 14:52:43 2025 from 192.168.56.115
programmymain@server2:~$ cat /var/www/html/index.html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initia
l-scale=1.0">
    <title>Default Page</title>
</head>
<body>
    <h1>Welcome to the Default Page!</h1>
    <p>This is a simple HTML Document</p>

    <p> Lorem ipsum dolor sit amet consectetur adipisicing el
it. Veniam, accusamus, dicta laborum dolorem iure ab recusand
ae quis non nesciunt excepturi quas commodi unde at alias cup
iditate, consectetur architecto eum placeat.</p>



</body>
</html>programmymain@server2:~$
 ⚠ 2.16.3   Lightspeed (Not logged in)   Python 3.12.3
```



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
```

```
        owner: root
        group: root
        mode: 0644

  - 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

  - hosts: db_servers
    become: true
    tasks:
```

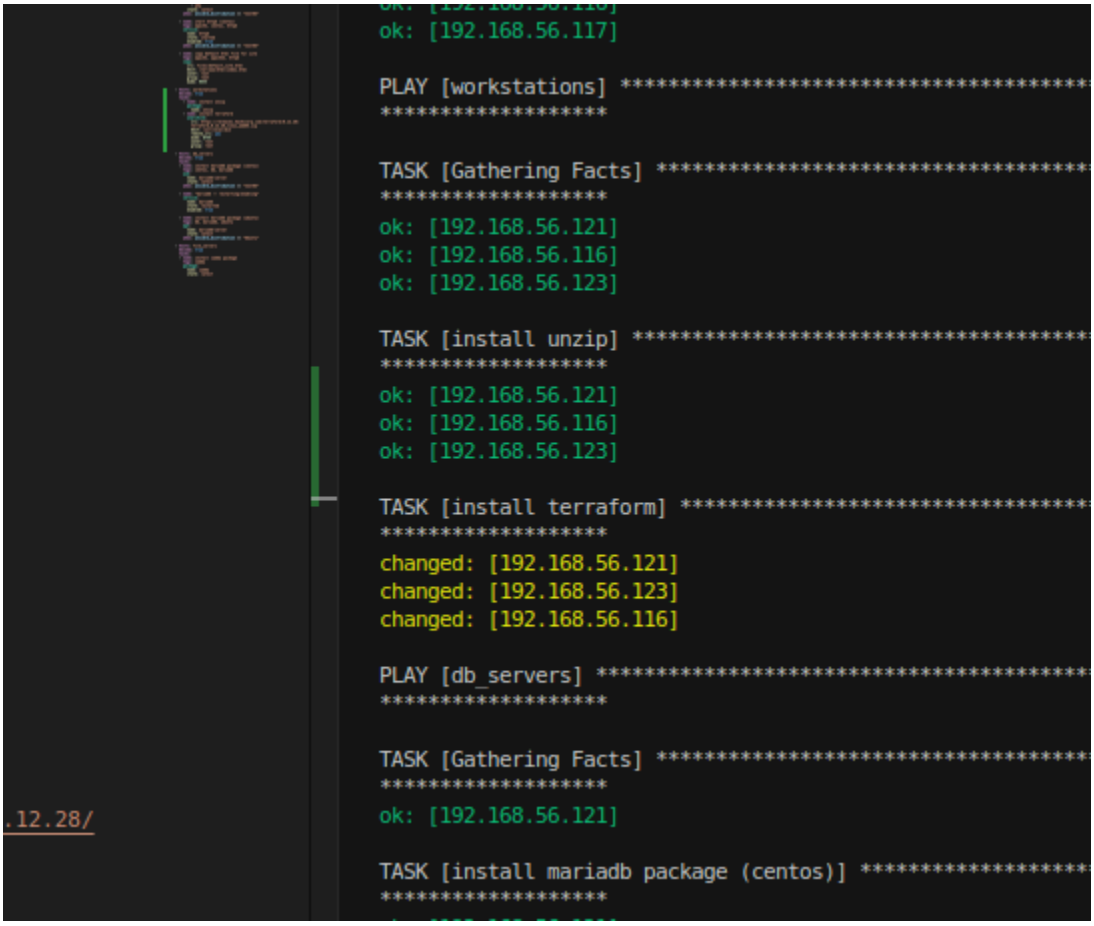
2. Edit the inventory file and add workstations group. Add any Ubuntu remote server. Make sure to remember the IP address.

inventory.ini M | site.yml M | site2.yml | default_site.h

inventory.ini

```
[web_servers]
192.168.56.116
192.168.56.117
192.168.56.121 ansible_user=alexis-cent

[workstations]
192.168.56.116
192.168.56.121 ansible_user=alexis-cent
192.168.56.123

[db_servers]
192.168.56.121 ansible_user=alexis-cent

[file_servers]
192.168.56.123

[all:vars]
ansible_ssh_common_args='-o GSSAPIAuthentication=no'
```

3. Run the playbook. Describe the output.



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

```
programmymain@workstation:~/CPE232_Deniega$ ssh programmymain
@192.168.56.123
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-29-generic x8
6_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled
.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates
.
See https://ubuntu.com/esm or run: sudo pro status

*** System restart required ***
Last login: Fri Oct  3 15:11:39 2025 from 192.168.56.115
programmymain@server3:~$ 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 t
he
other commands, please read the help and docs before usage.
```

It's just the help thing, but it works…

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.

site2.yml

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

  - name: update repo 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
```

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.

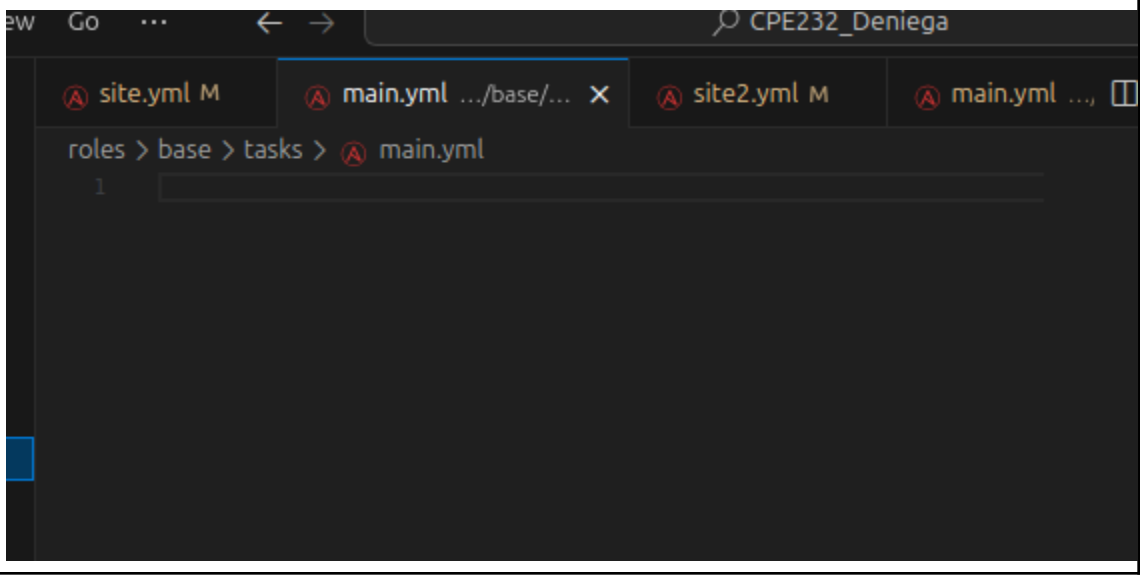
```
programmymain@workstation:~/CPE232_Deniega$ mkdir -p roles/{w
orkstation,base,web_servers,file_servers,db_servers}/tasks
```

```
Processing triggers for man-db (2.12.0-4build2) ...
programmymain@workstation:~/CPE232_Deniega$ tree
.
├── ansible.cfg
├── group_vars
│   └── all
│       └── vault.yaml
├── install_apache_redhat.yml
├── install_apache.yml
├── inventory.ini
├── README.md
├── roles
│   ├── base
│   │   └── tasks
│   ├── db_servers
│   │   └── tasks
│   ├── file_servers
│   │   └── tasks
│   ├── web_servers
│   │   └── tasks
│   └── workstation
│       └── tasks
└── site.yml
```

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.

```
programmymain@workstation:~/CPE232_Deniega$ touch roles/{workstation,base,web_servers,file_servers,db_servers}/tasks/main.yml
programmymain@workstation:~/CPE232_Deniega$ tree
.
├── ansible.cfg
├── files
│   └── default_site.html
├── group_vars
│   └── all
│       └── vault.yaml
├── install_apache_redhat.yml
├── install_apache.yml
├── inventory.ini
├── README.md
├── roles
│   ├── base
│   │   └── tasks
│   │       └── main.yml
│   ├── db_servers
│   │   └── tasks
│   │       └── main.yml
│   ├── file_servers
│   │   └── tasks
│   │       └── main.yml
│   ├── web_servers
│   │   └── tasks
│   │       └── main.yml
│   └── workstation
│       └── tasks
│           └── main.yml
└── site.yml

15 directories, 13 files
```



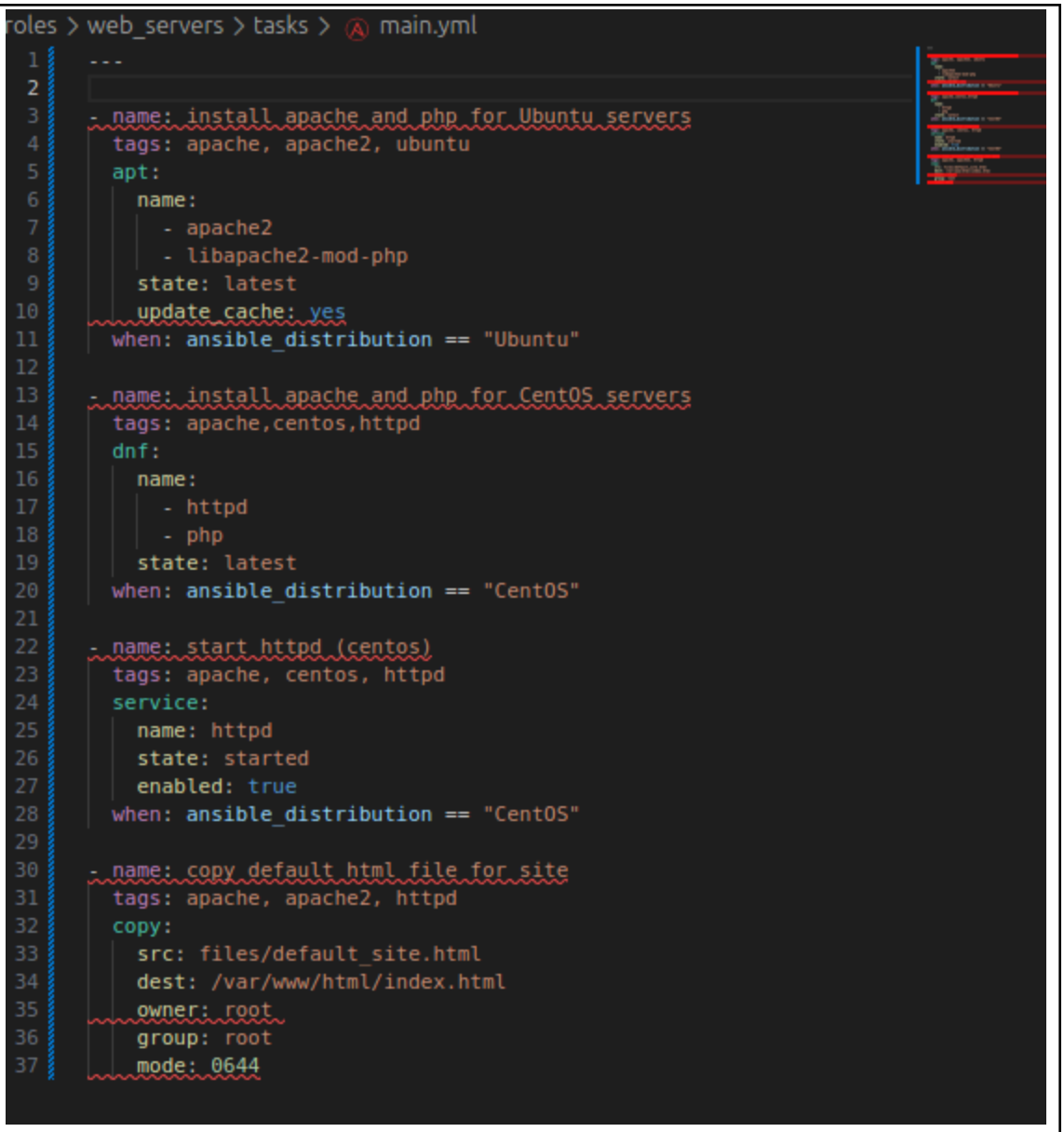
roles > db_servers > tasks > main.yml

```yaml
---

- name: install mariadb package (centos)
  tags: centos, db, mariadb
  yum:
    name: mariadb-server
    state: latest
  when: ansible_distribution == "CentOS"

- name: "mariadb -- restarting/enabling"
  service:
    name: mariadb
    state: restarted
    enabled: true

- name: install mariadb package (ubuntu)
  tags: db, mariadb, ubuntu
  apt:
    name: mariadb-server
    state: latest
  when: ansible_distribution == "Ubuntu"
```

roles > file_servers > tasks > main.yml

```yaml
---

- name: install samba package
  tags: samba
  package:
    name: samba
    state: latest

```

roles > web_servers > tasks > main.yml



```yaml
---

- name: install apache and php for Ubuntu servers
  tags: apache, apache2, ubuntu
  apt:
    name:
      - apache2
      - libapache2-mod-php
    state: latest
    update_cache: yes
  when: ansible_distribution == "Ubuntu"

- name: install apache and php for CentOS servers
  tags: apache,centos,httpd
  dnf:
    name:
      - httpd
      - php
    state: latest
  when: ansible_distribution == "CentOS"

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

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

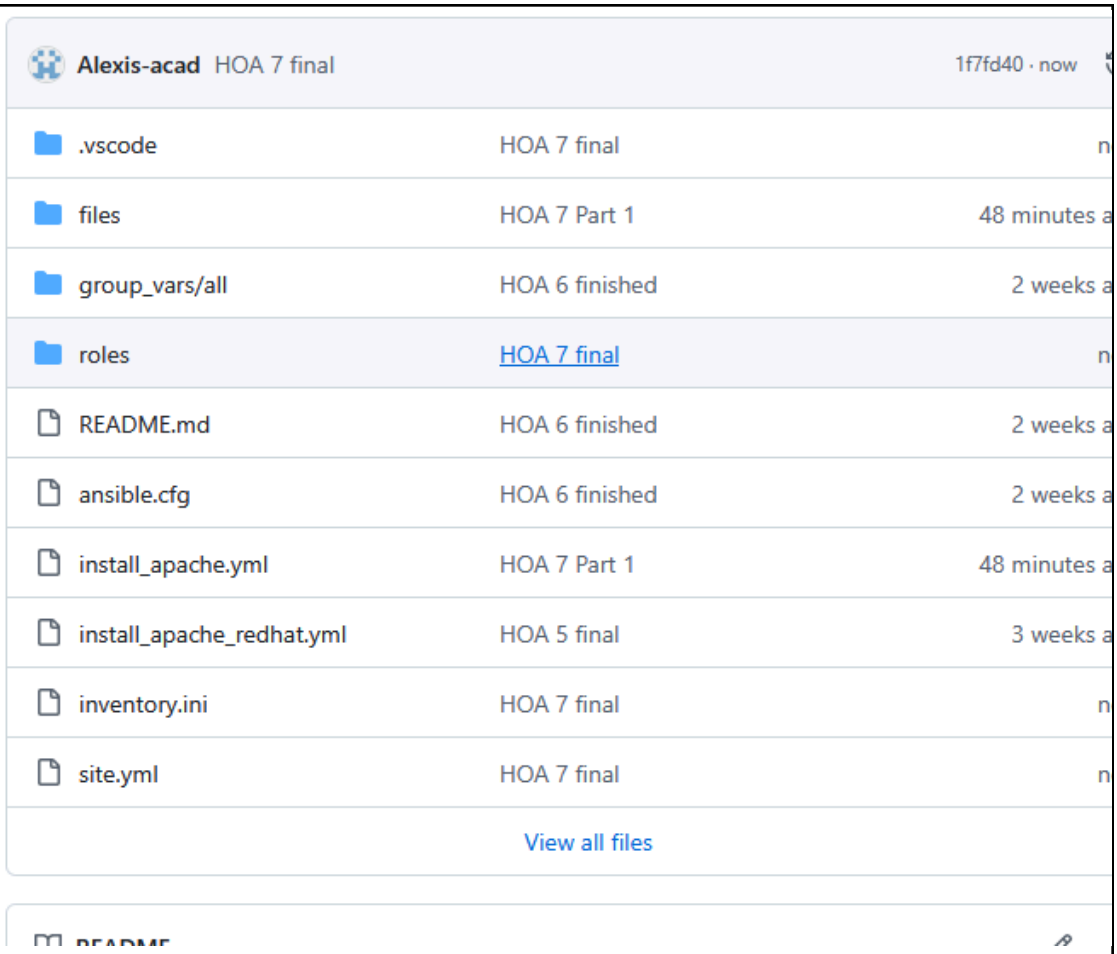
roles > workstations > tasks > main.yml

```yaml
---

- 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
```

NOTE: I know that there are a lot of squiggly lines here but trust me this is working!

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

As you can see here, each play called the role's .yml file instead of putting it in one big file.



also just pushed it just in case

Reflections:

Answer the following:

1. What is the importance of creating roles?

Roles are ultimately a set of goals assigned to a person or a group of people. It is essential to a great team. This is the same for a group of remote servers. They only need certain files for certain needs. Putting all plays into one big file ultimately causes problems like having to scroll endlessly trying to find that one error (true story) but splitting it into multiple directories makes it more manageable and makes it more efficient in finding errors.

2. What is the importance of managing files?

   An operating system's greatest supply are files, they are the ones that keep the hardware and software going. Managing files remotely not only allows for better control of the server, but also just makes it easier if done correctly (unlike putting it all in one folder called "to be sorted later")