Name: Clarence B. Chavez	**Date Performed:** Oct. 7, 2024
Course/Section: CPE31S2	**Date Submitted:** Oct. 9, 2024
Instructor: Engr. Robin Valenzuela	**Semester and SY:** 1st Sem, 2024-2025

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**:

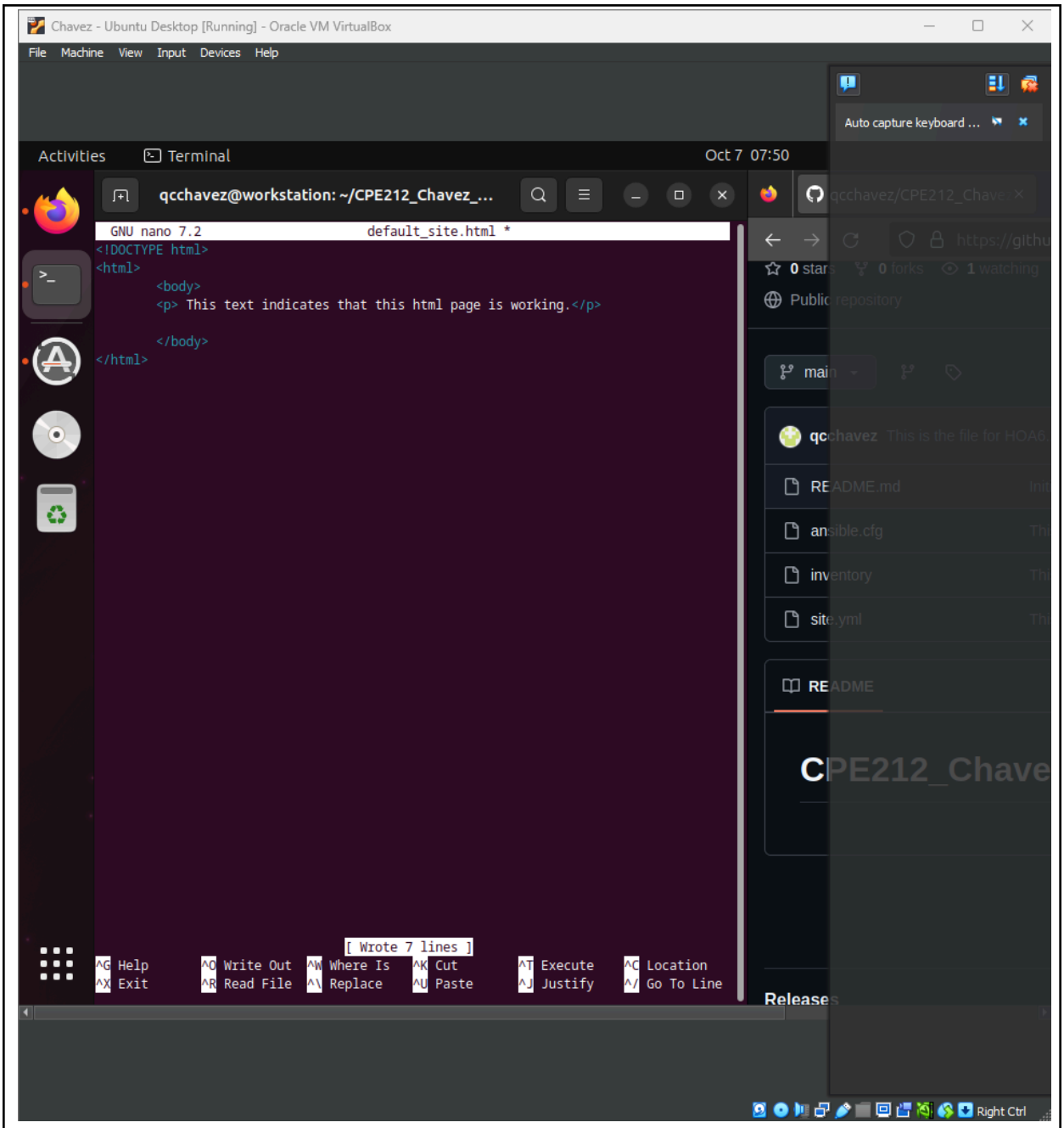
In this activity, we look at the concept of copying a file to a server. We are going to create a file into our git repository and use Ansible to grab that file and put it into a particular place so that we could do things like customize a default website, or maybe install a default configuration file.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

```
qcchavez@workstation:~/CPE212_Chavez_HOA6.1$ sudo mkdir files
qcchavez@workstation:~/CPE212_Chavez_HOA6.1$ ls
ansible.cfg  files  inventory  README.md  site.yml
qcchavez@workstation:~/CPE212_Chavez_HOA6.1$ cd files
qcchavez@workstation:~/CPE212_Chavez_HOA6.1/files$ sudo nano default_site.html
qcchavez@workstation:~/CPE212_Chavez_HOA6.1/files$ ls
default_site.html
qcchavez@workstation:~/CPE212_Chavez_HOA6.1/files$
```

- In this screenshot, I've created a directory named "files" and inside that directory, I've created an html file named "default_site".

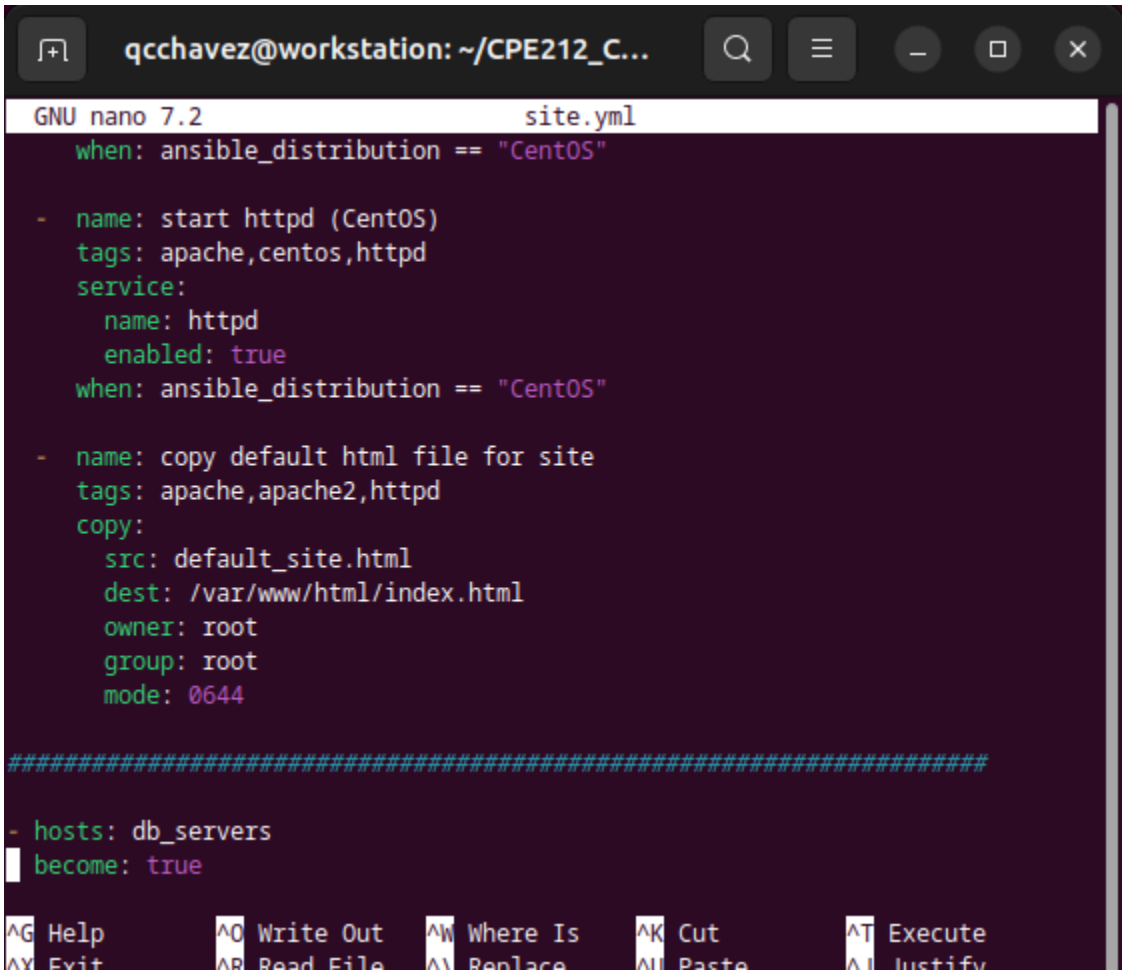


- This is the content of my **default_site.html**

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:

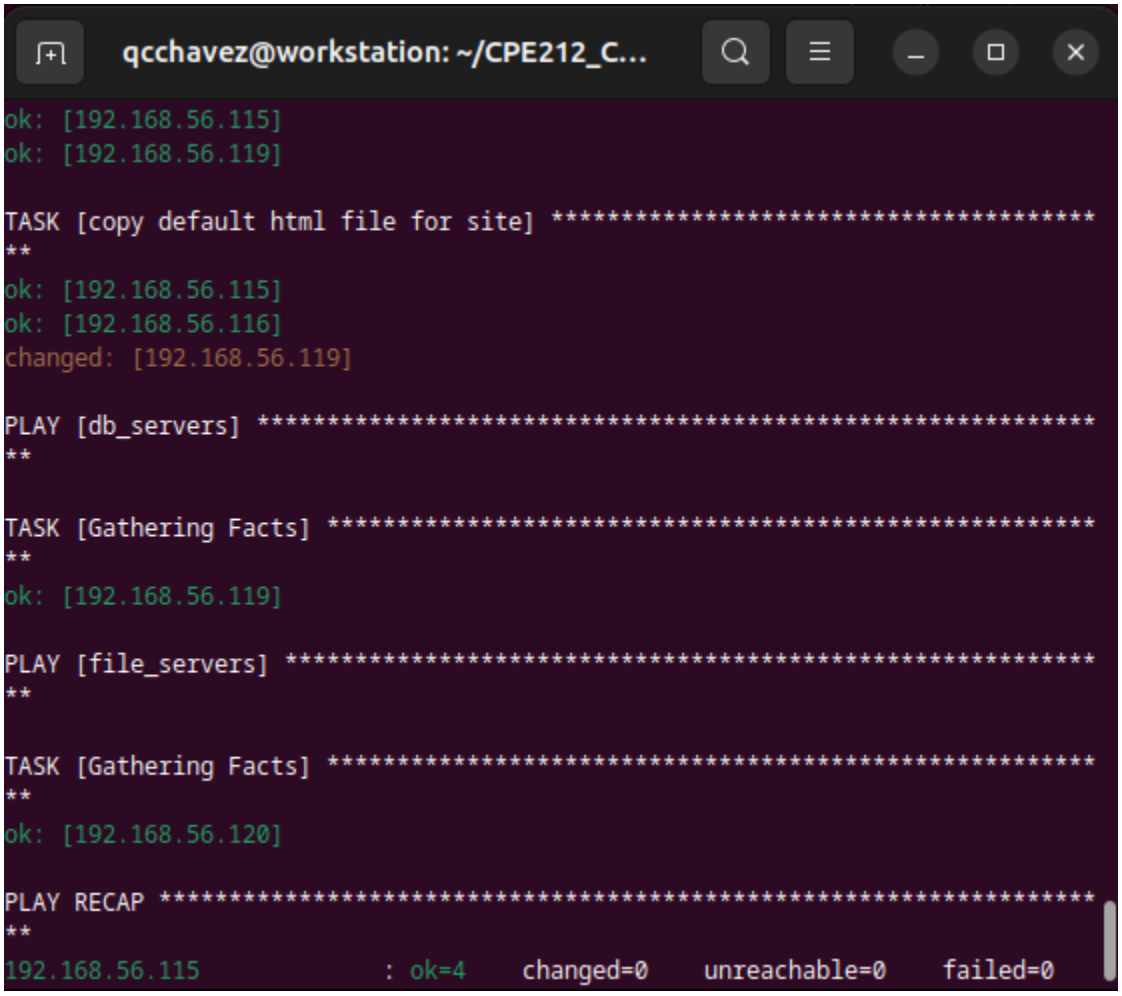
   - name: copy default html file for site

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



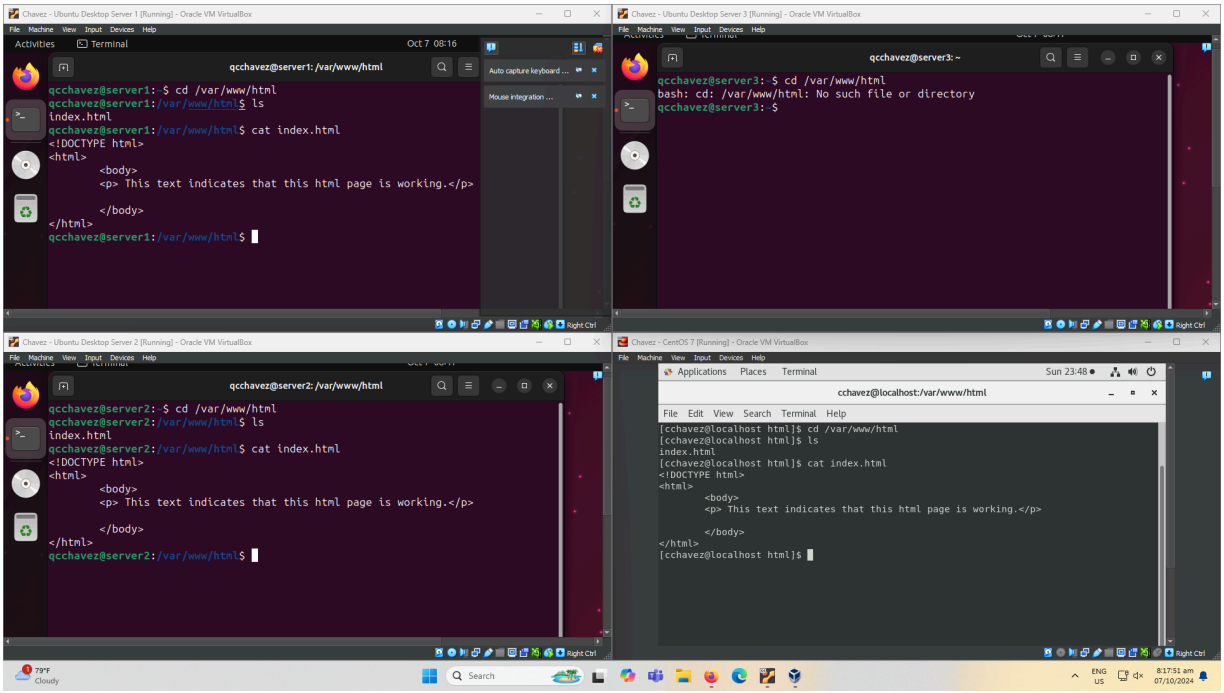
- In this screenshot, I've added the necessary lines of codes in the **site.yml** playbook based on the required task.

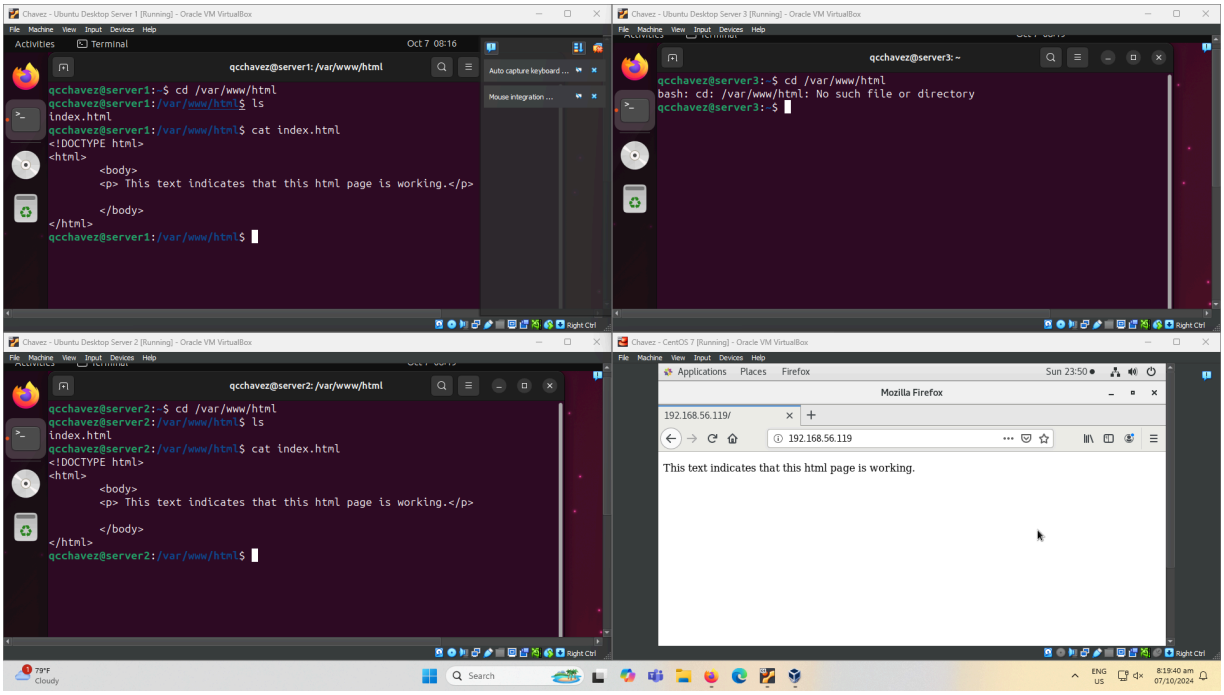
3. Run the playbook *site.yml*. Describe the changes.



- In this screenshot, it shows that copying the default html file for site to the target remote servers was done successfully.

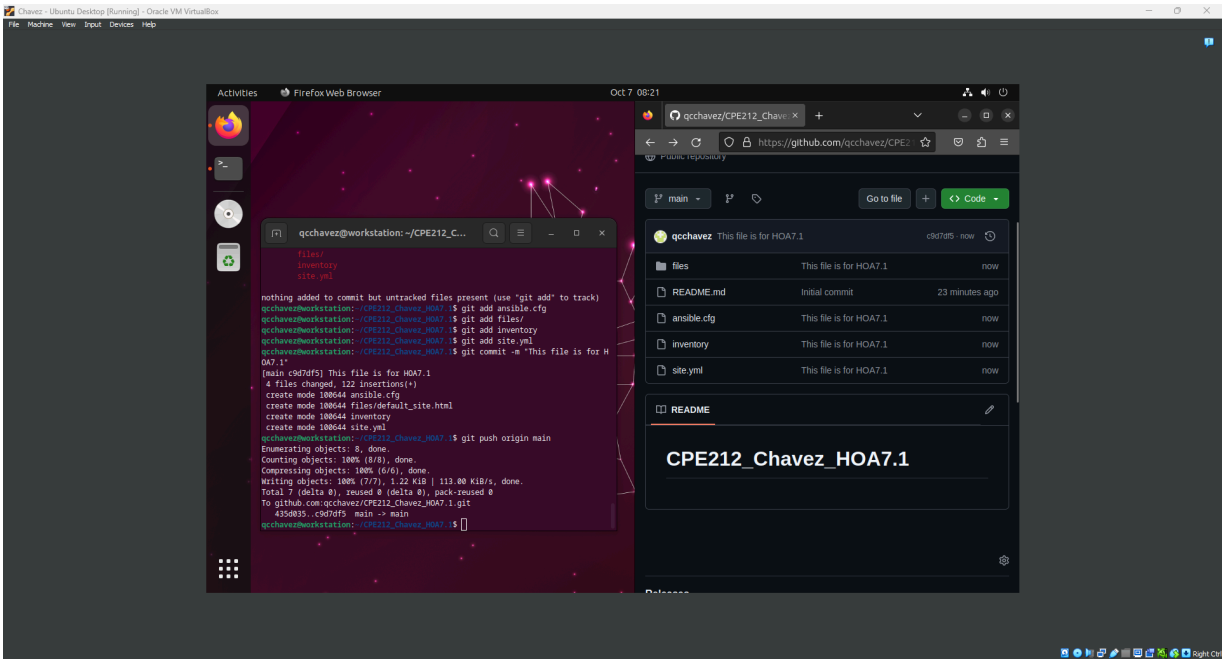
4. Go to the remote servers (*web_servers*) listed in your inventory. Use cat command to check if the index.html is the same as the local repository file (*default_site.html*). Do both for Ubuntu and CentOS servers. On the CentOS server, go to the browser and type its IP address. Describe the output.





- In this screenshot, I've shown if the index.html was applied to the remote servers in the **web_servers** group.

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



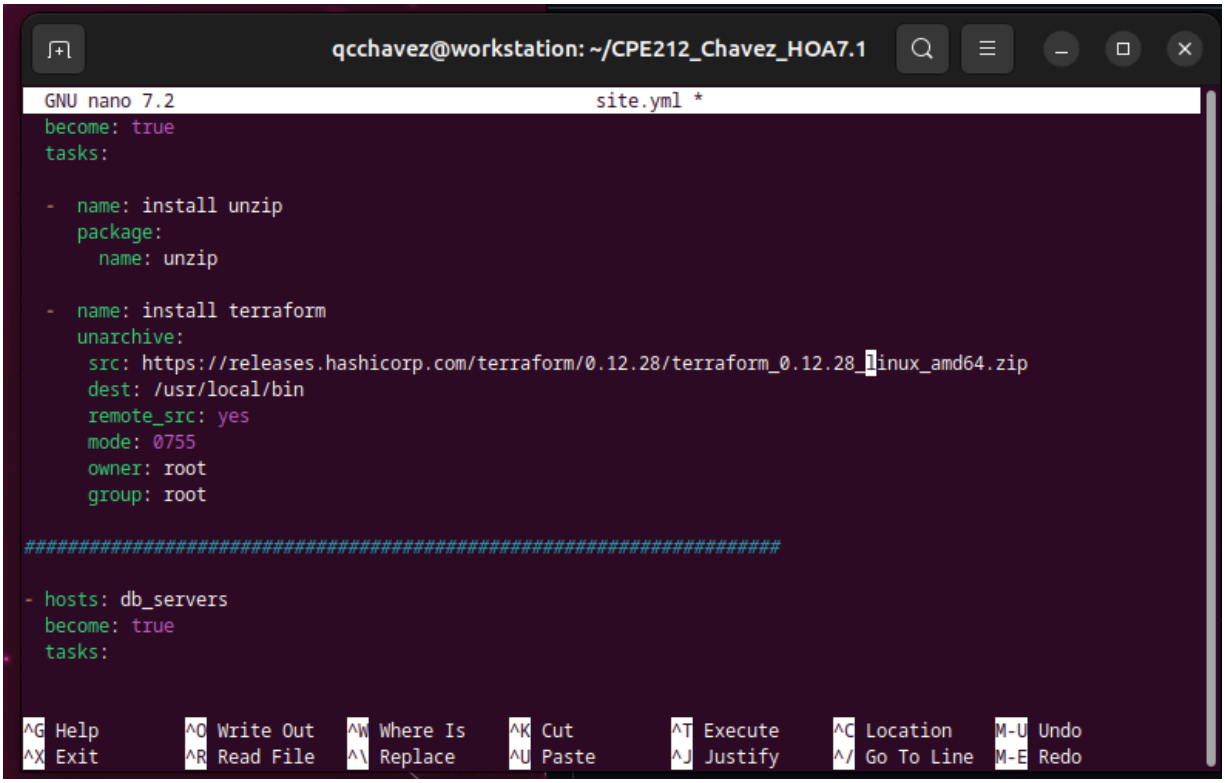
- In this screenshot, I've committed to the GitHub to comply with the task 1.

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:

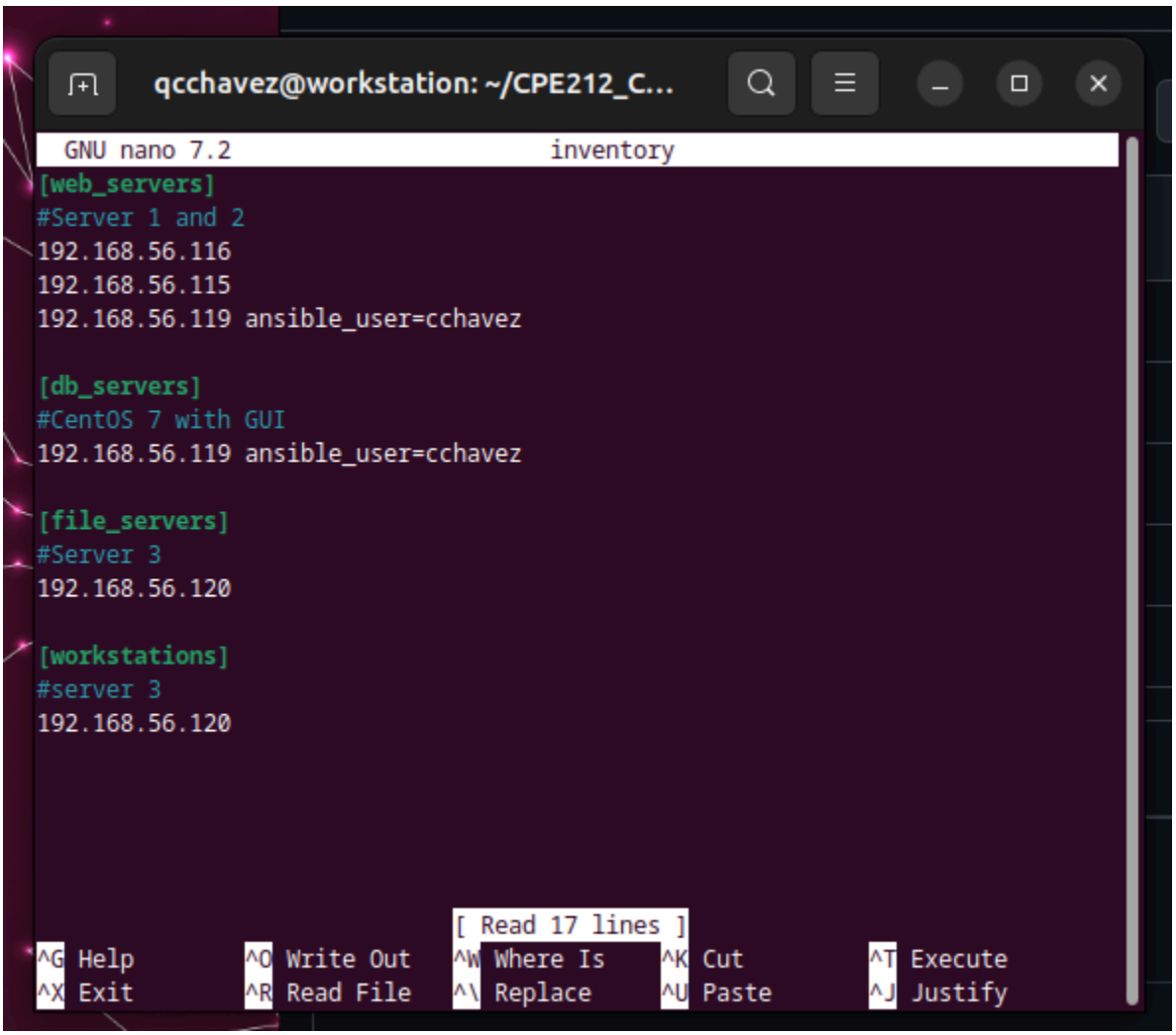
```
- hosts: workstations
  become: true
  tasks:

  - 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



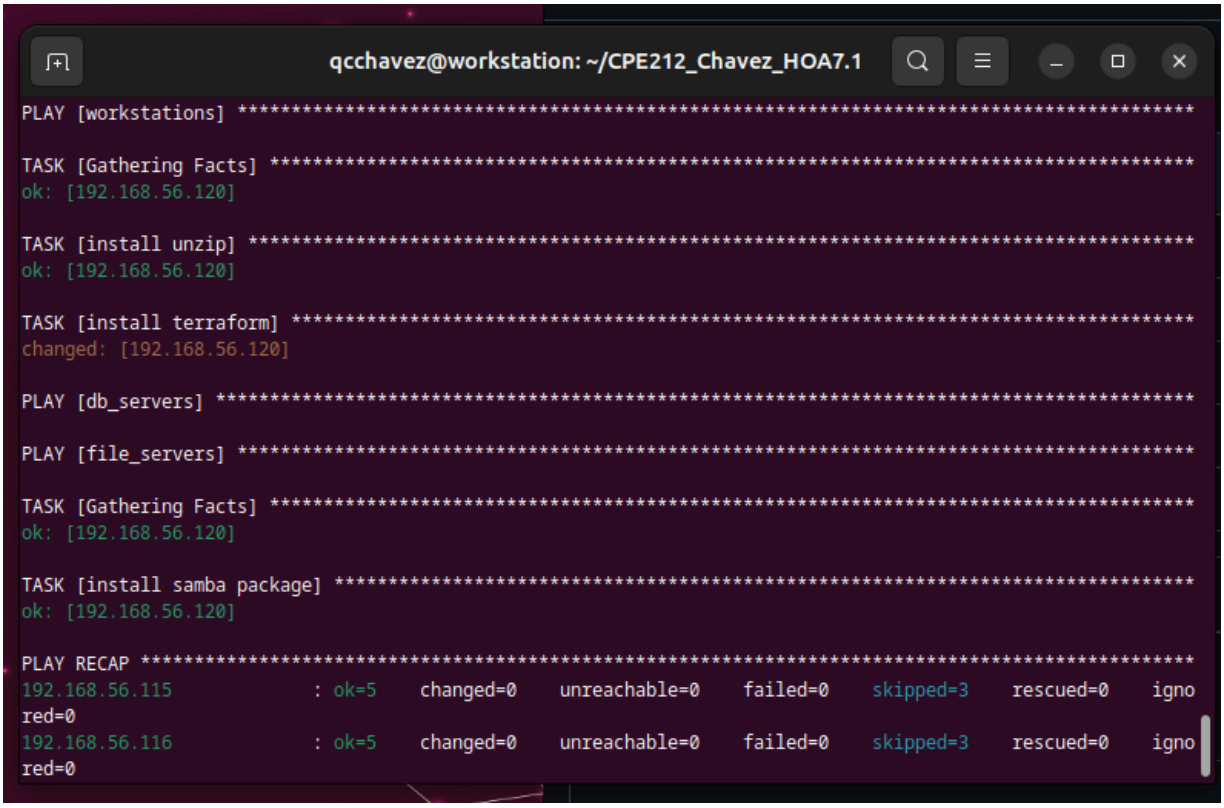
- In this screenshot, I've added the necessary lines of codes for the workstations group.

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



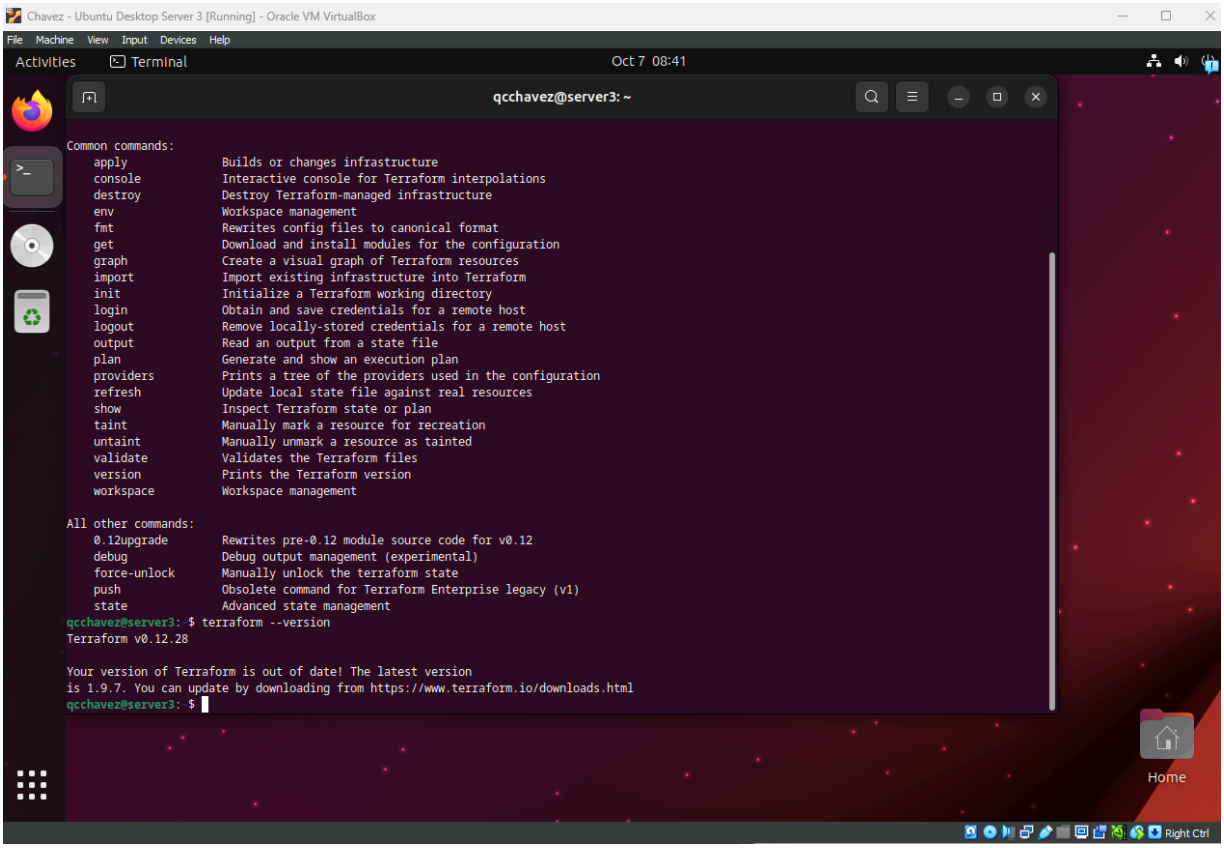
- In this screenshot, I've added the **server3 IP address** inside the **workstations** group.

3. Run the playbook. Describe the output.



- After running the playbook, it shows that the **install terraform** task was successfully done and made changes to the target remote server.

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



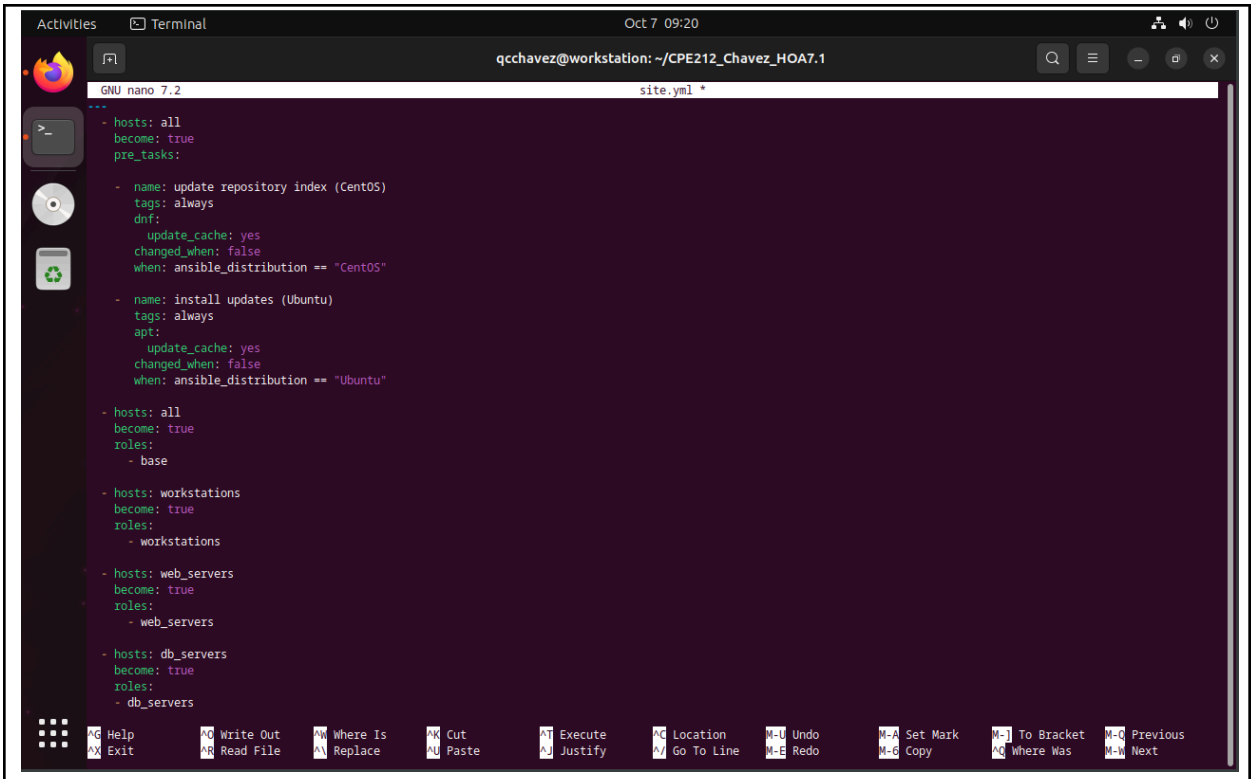
- After the successful changes, it was confirmed that the terraform was successfully installed in the target remote server which is the **Server3.**

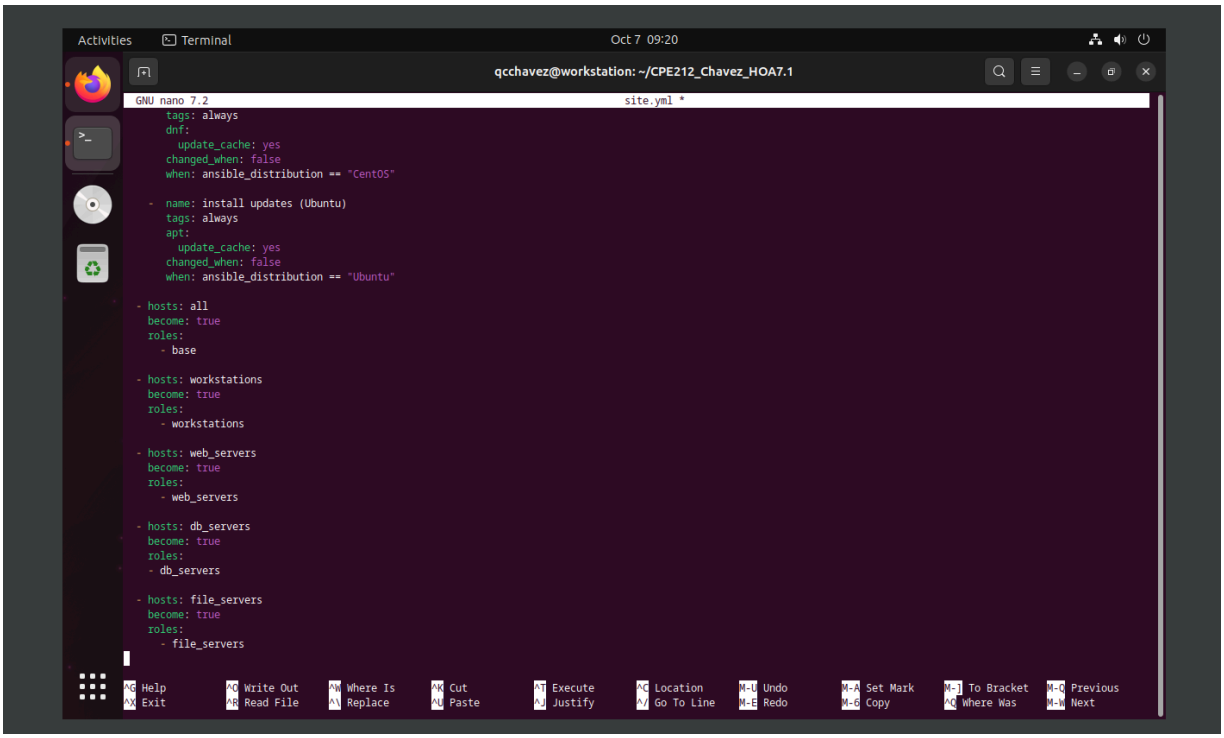
Task 3: Create roles

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
```

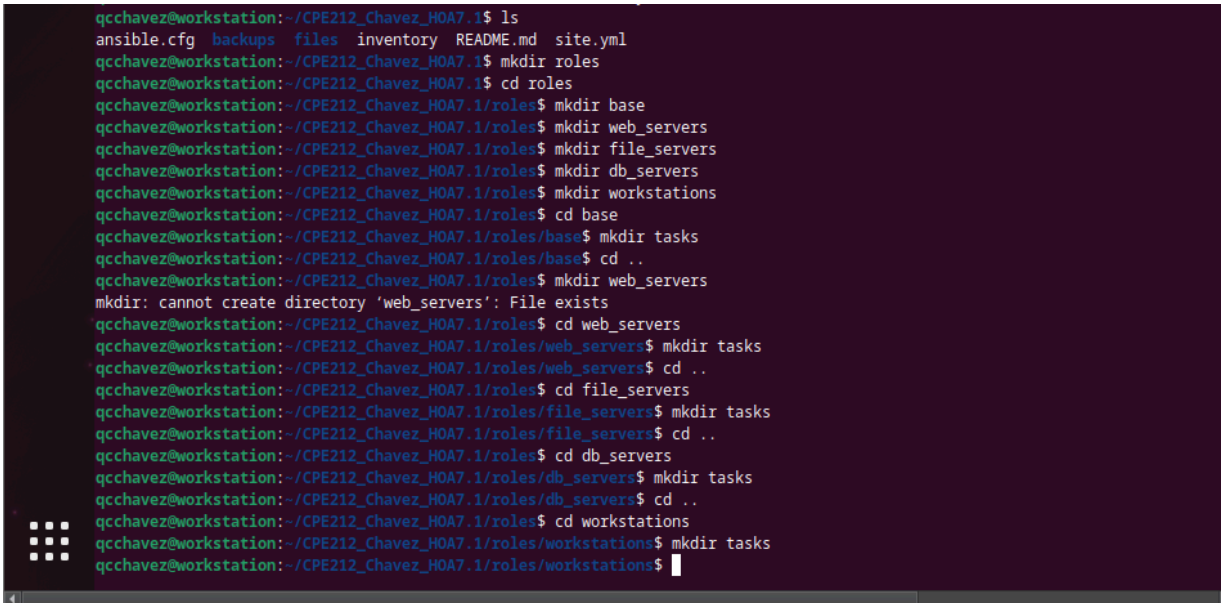
Save the file and exit.





- In these screenshots, I've made a whole change in the **site.yml** file and encoded the necessary lines of codes based on the screenshots given above.

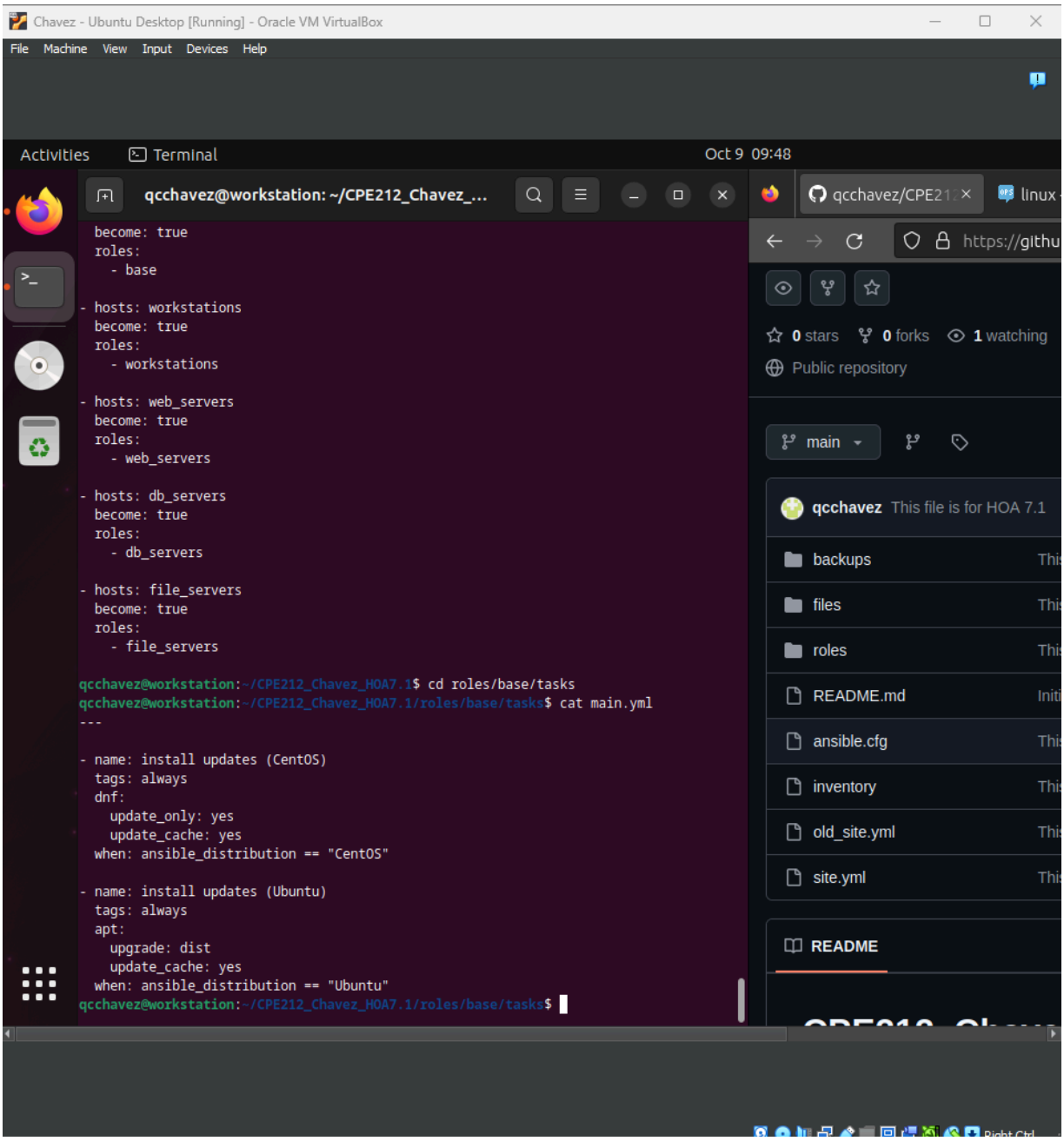
2. Under the same directory, create a new directory and name it roles. Enter the roles directory and create new directories: base, web_servers, file_servers, db_servers and workstations. For each directory, create a directory and name it tasks.



- In this screenshot, I've created the necessary directories and created **tasks** directories in each one of them.

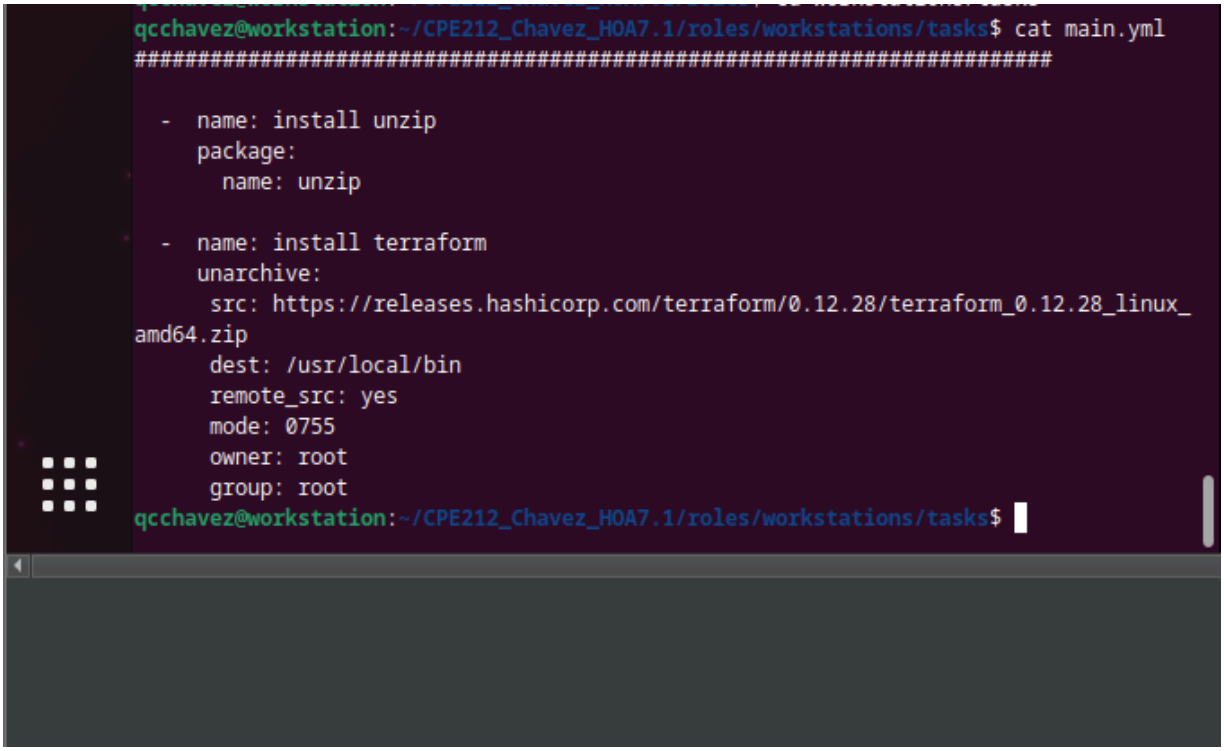
3. Go to tasks for all directory and create a file. Name it main.yml. In each of the tasks for all directories, copy and paste the code from the old site.yml file. Show all contents of main.yml files for all tasks.

base



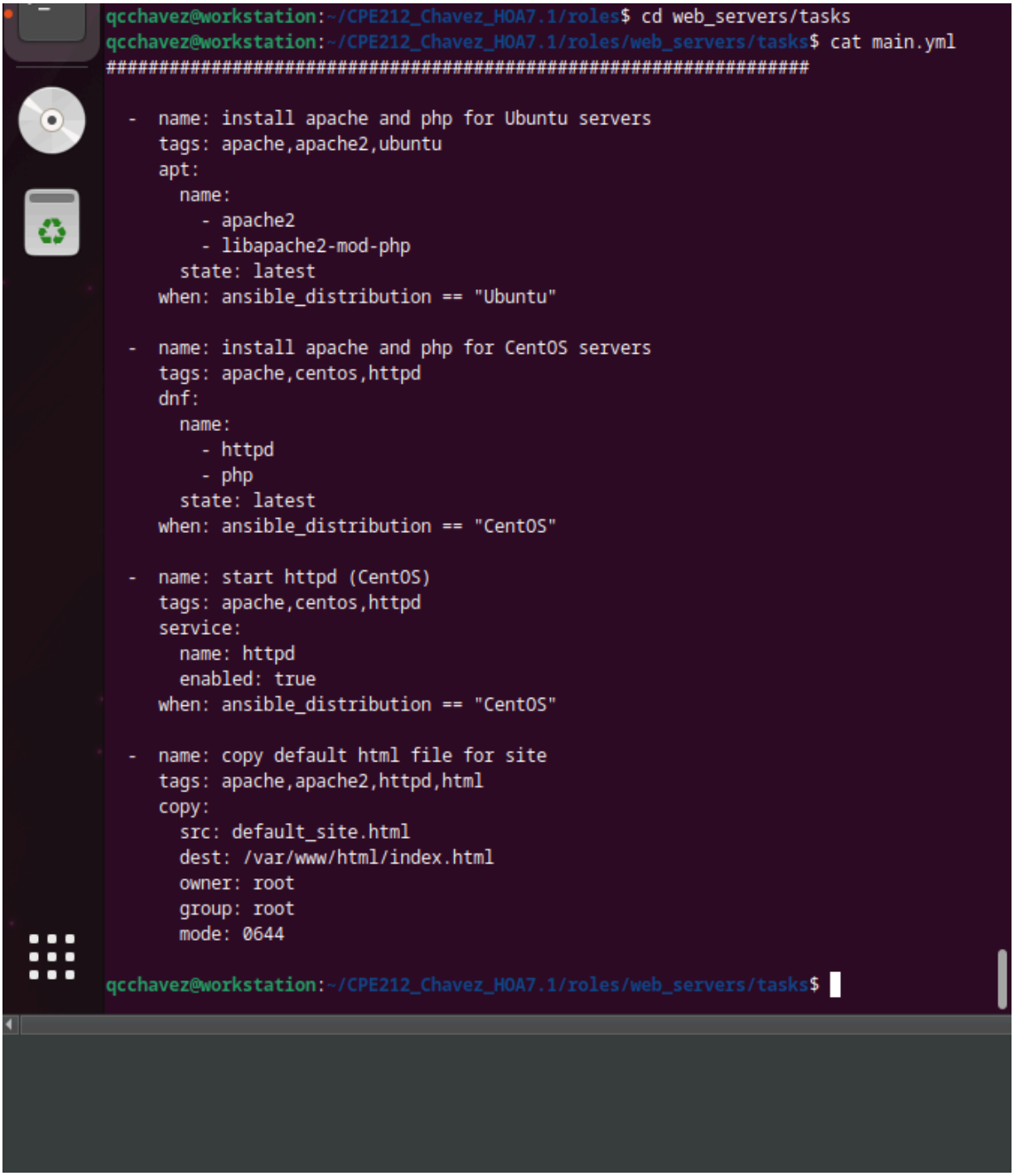
- In this screenshot, I've created a **main.yml** file and copy-pasted the necessary codes for the **base** role.

workstations



- In this screenshot, I've created a **main.yml** file and copy-pasted the necessary codes for the **workstations** role.

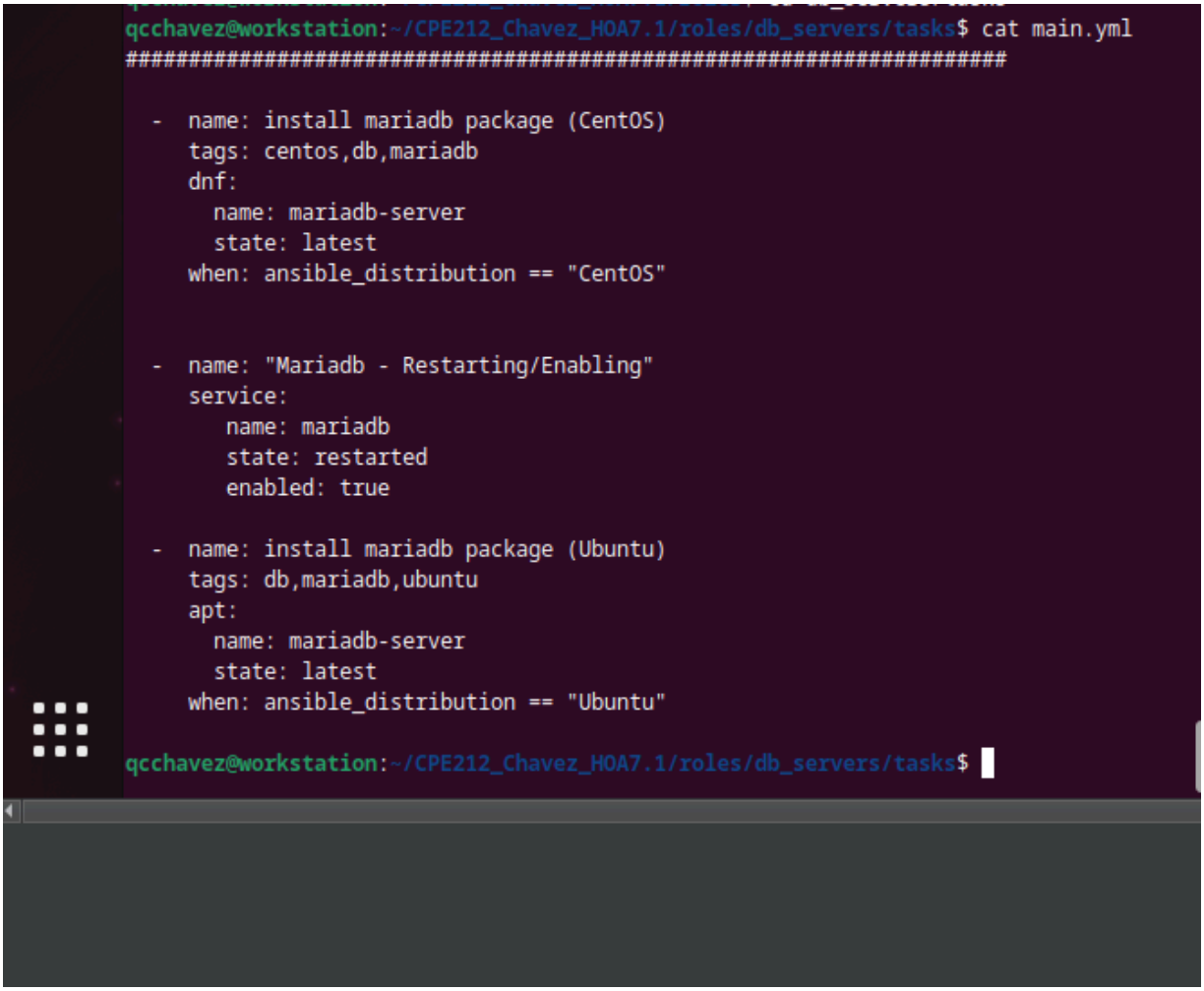
web_servers



```
qcchavez@workstation:~/CPE212_Chavez_HOA7.1/roles$ cd web_servers/tasks
qcchavez@workstation:~/CPE212_Chavez_HOA7.1/roles/web_servers/tasks$ cat main.yml
###################################################################

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

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

qcchavez@workstation:~/CPE212_Chavez_HOA7.1/roles/web_servers/tasks$
```

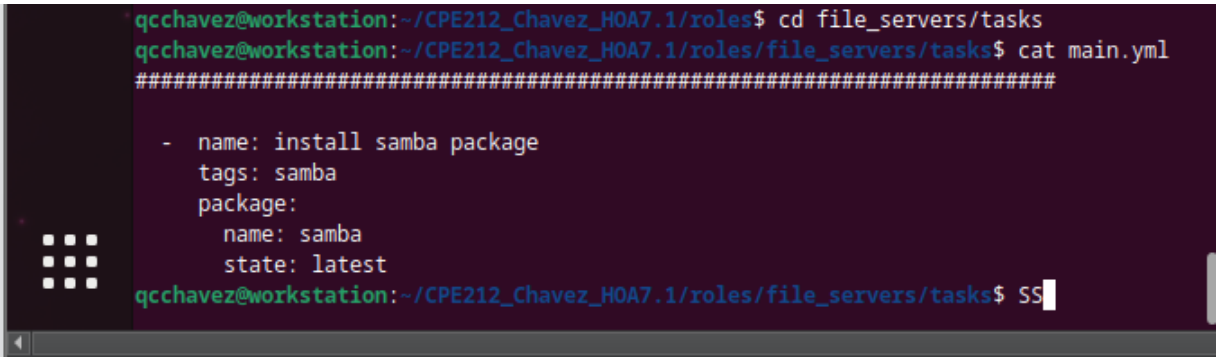
- In this screenshot, I've created a **main.yml** file and copy-pasted the necessary codes for the **web_servers** role.

db_servers



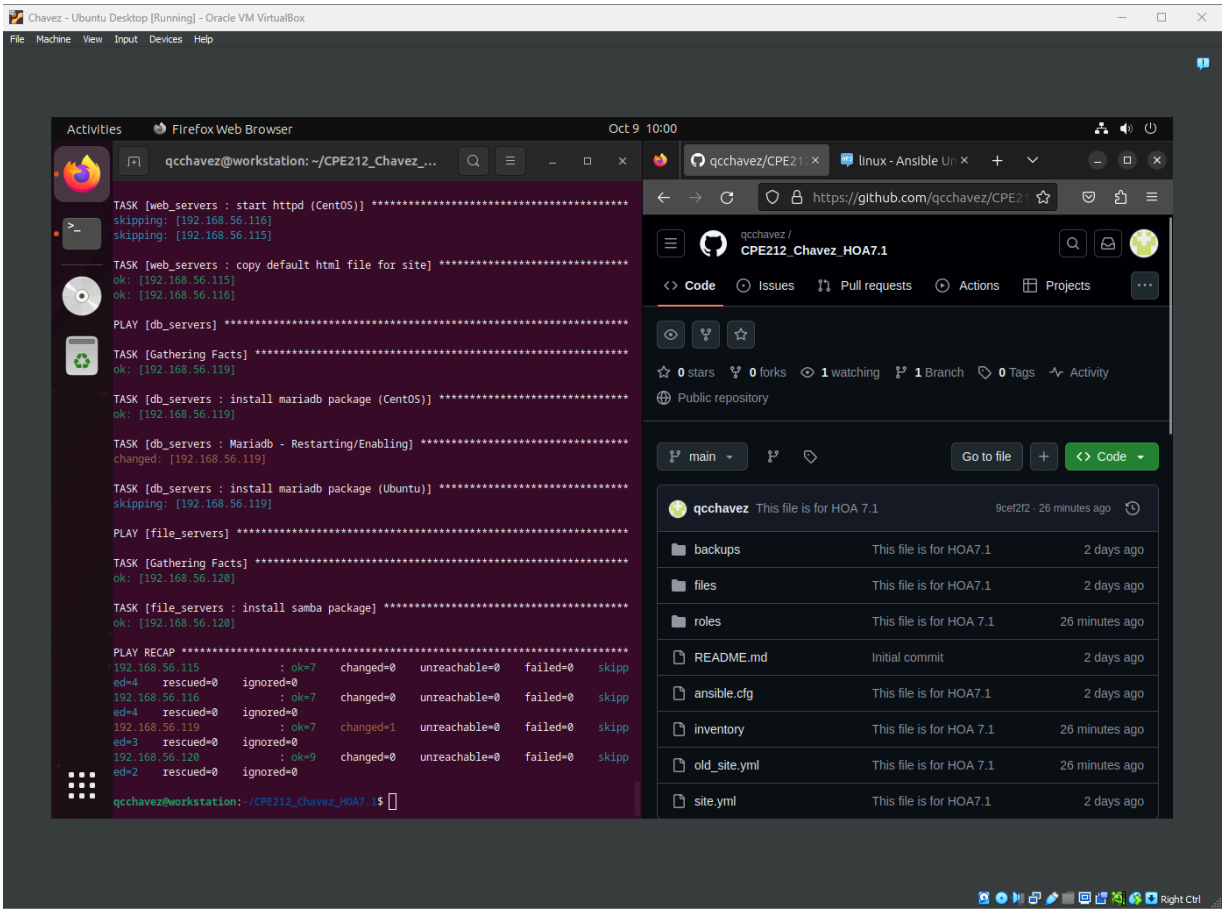
- In this screenshot, I've created a **main.yml** file and copy-pasted the necessary codes for the **db_servers** role.

file_servers



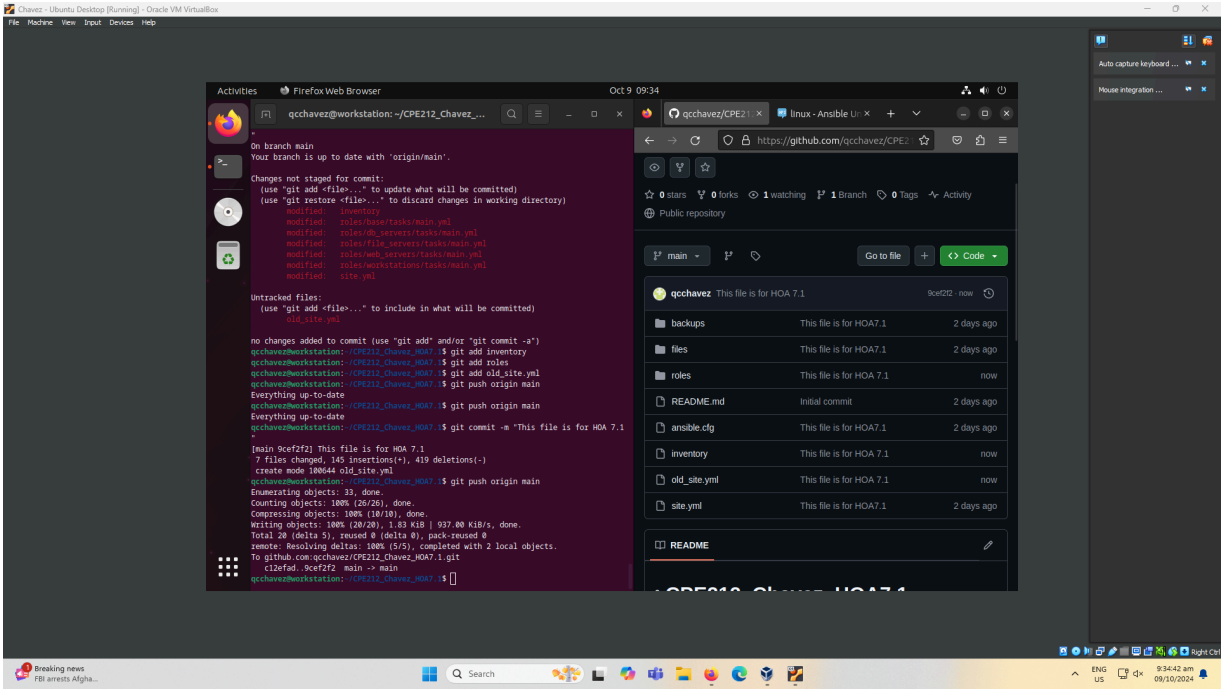
- In this screenshot, I've created a **main.yml** file and copy-pasted the necessary codes for the **file_servers** role.

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



- In this screenshot, I've ran the **site.yml** playbook and it shows that it worked successfully.

Committing to GitHub



- In this screenshot, I've committed my repository to GitHub

Reflections:

Answer the following:

1. What is the importance of creating roles?
   - The importance of creating roles is that it can be maintained properly, can be reused over and over, and also modular since every role contains a .yml file that is specifically made for that exact role.

2. What is the importance of managing files?
   - The importance of managing files is that it is more neat, organized, and less room for making mistakes. Properly managed files also means that it is easier to detect the cause of the errors in case some of them occur in the future.