Name: Ilagan, Carlo Hideki D.	Date Performed: 10-7-2024
Course/Section: CPE31S2	Date Submitted: 10-10-2024
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

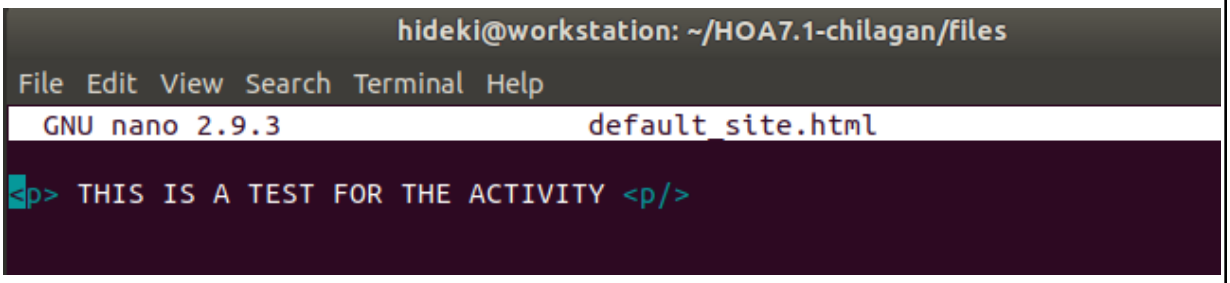
1.2 Implement roles in ansible

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.



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
```

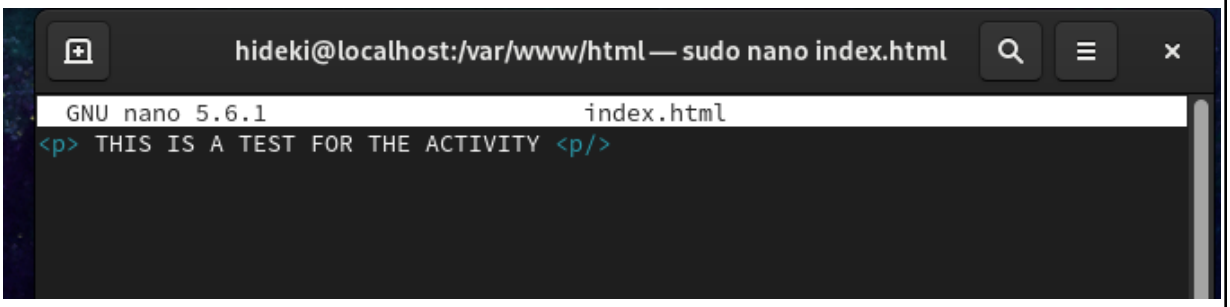
```
hideki@workstation: ~/HOA7.1-chilagan
File  Edit  View  Search  Terminal  Help
 GNU nano 2.9.3                        site.yml

    when: ansible_distribution == "CentOS"

  - 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: db_servers
```

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

```
hideki@workstation: ~/HOA7.1-chilagan
File  Edit  View  Search  Terminal  Help
skipping: [192.168.56.111]

TASK [install apache and php for CENTOS servers] *******************************
ok: [192.168.56.111]

TASK [start httpd (CentOS)] ****************************************************
changed: [192.168.56.111]

TASK [copy default html file for site] *****************************************
changed: [192.168.56.111]

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

TASK [Gathering Facts] *********************************************************
ok: [192.168.56.111]

TASK [install mariadb package in CENTOS] ***************************************
ok: [192.168.56.111]

TASK [install mariadb package in Ubuntu] ***************************************
skipping: [192.168.56.111]

TASK [Mariadb- Restart/Enable] *************************************************
changed: [192.168.56.111]
```

- There were added files for the servers inside the group web_servers.

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.



- The text inside the default_site.html was copied to the remote server file named index.html since this is where the text was directed to.

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

- In checking repositories, this will now provide information that HOA 7.1 includes an HTML file inside the repository.

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

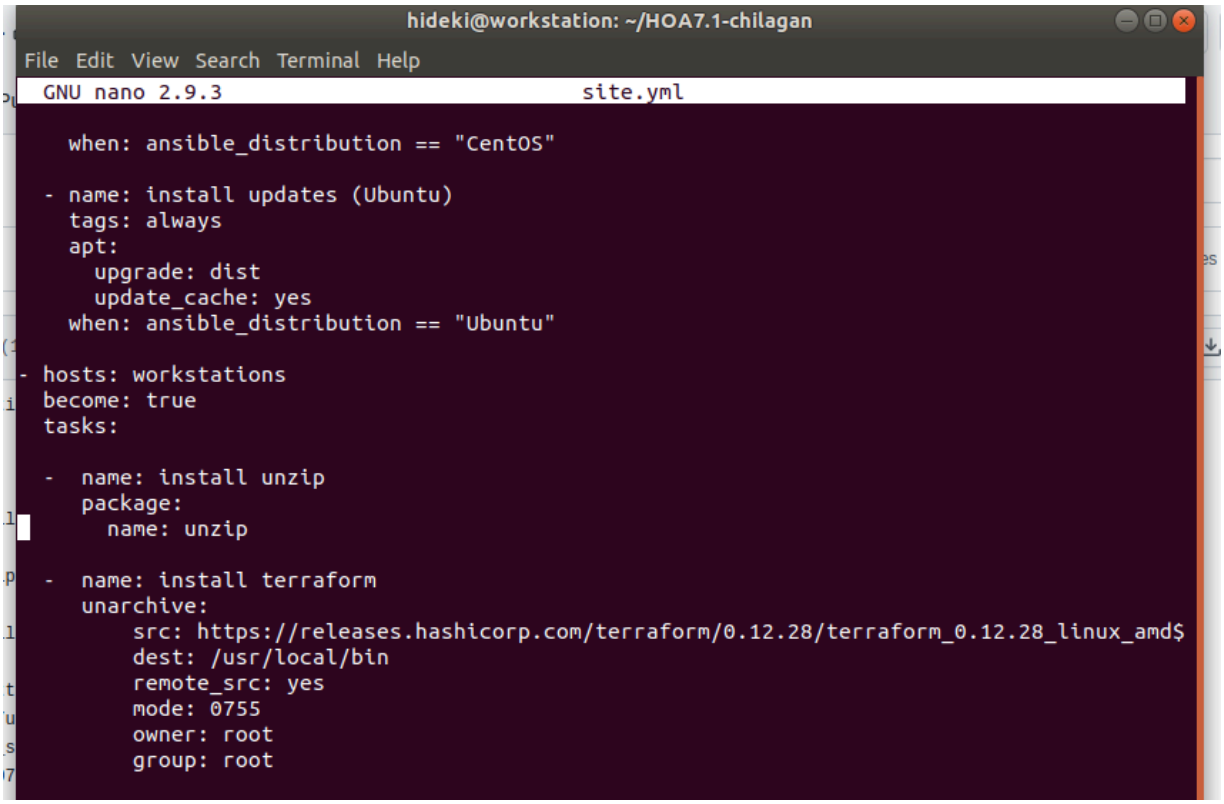
1. Edit the site.yml. Just before the web_servers play, create a new play:
   - hosts: workstations
     become: true
     tasks:

     - name: install unzip
       package:
         name: unzip

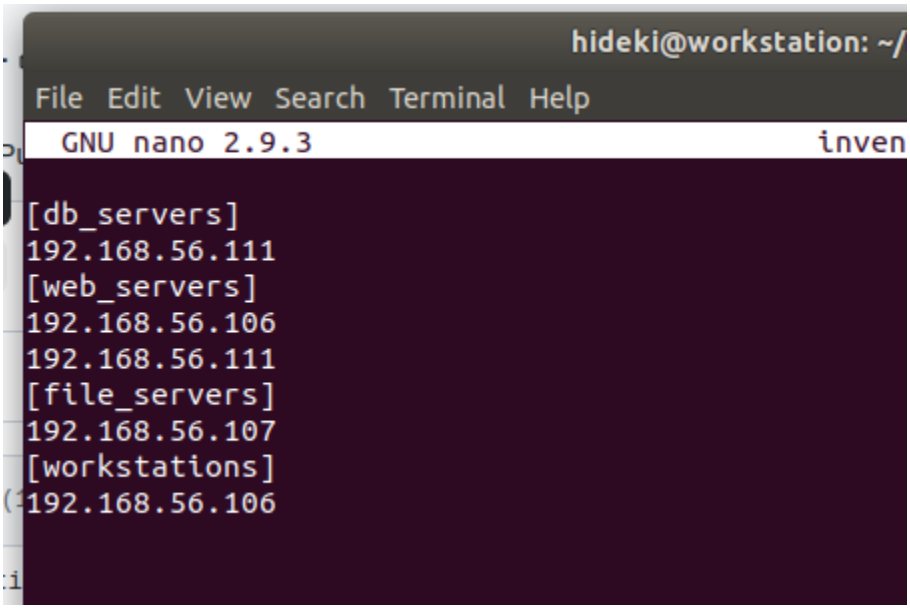
     - name: install terraform
       unarchive:

       src: https://releases.hashicorp.com/terraform/0.12.28/terraform_0.12.28_linux_amd64.zip

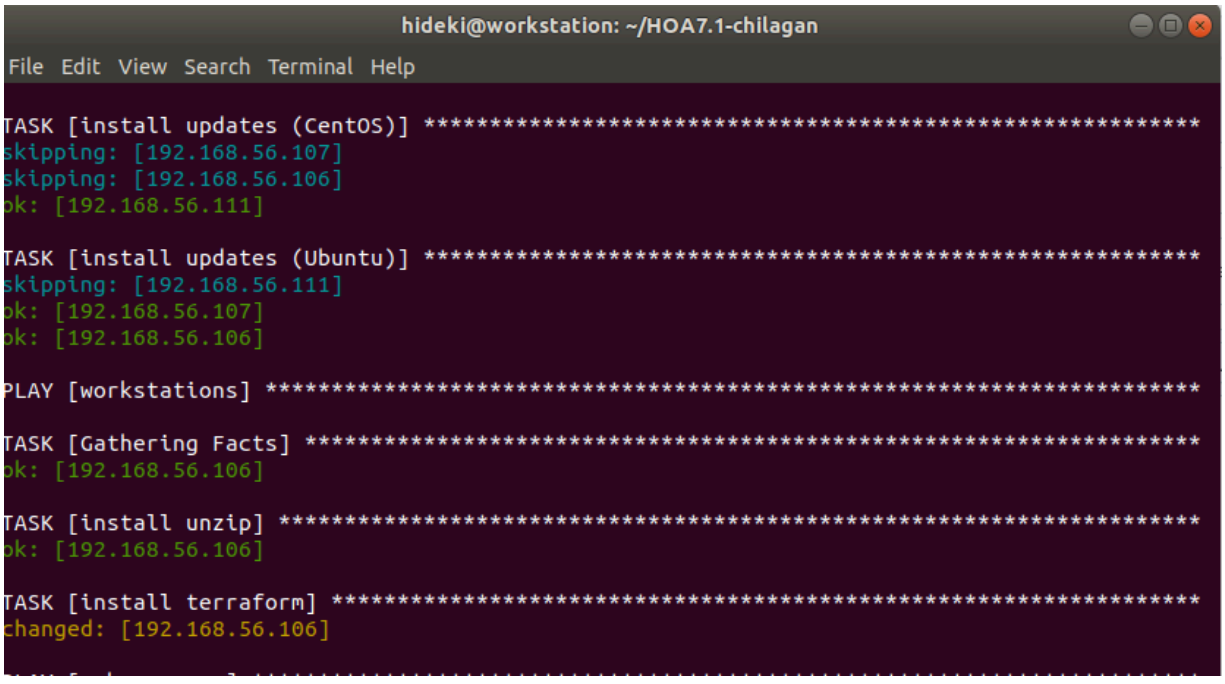
dest: /usr/local/bin
remote_src: yes
mode: 0755
owner: root
group: root



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.



- There were changes after install updates for the ubuntu which was the installation of terraform to the ubuntu remote server.

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

```
hideki@server1:~$ terraform version
Terraform v0.12.28
```

- Terraform was installed to the ubuntu server

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.

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.

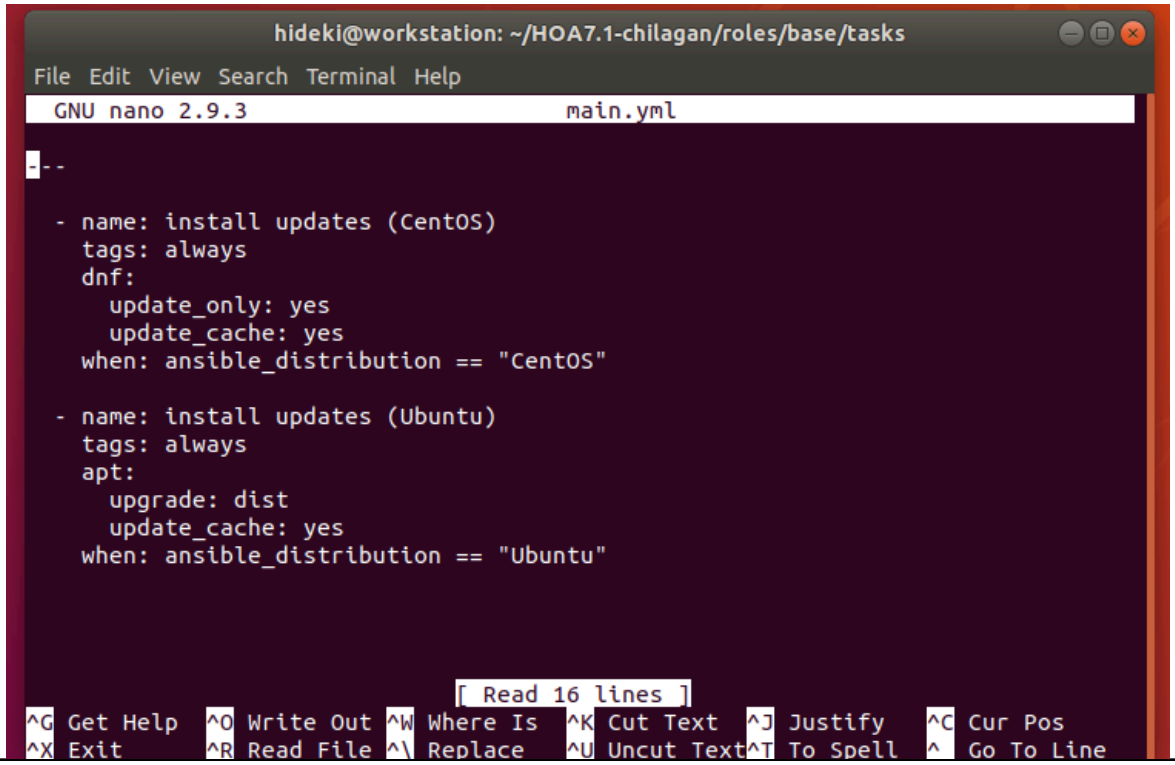
```
hideki@workstation:~/HOA7.1-chilagan$ tree roles
roles
├── base
│   └── tasks
├── db_servers
│   └── tasks
├── file_servers
│   └── tasks
├── web_servers
│   └── tasks
└── workstations
    └── tasks
```

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.

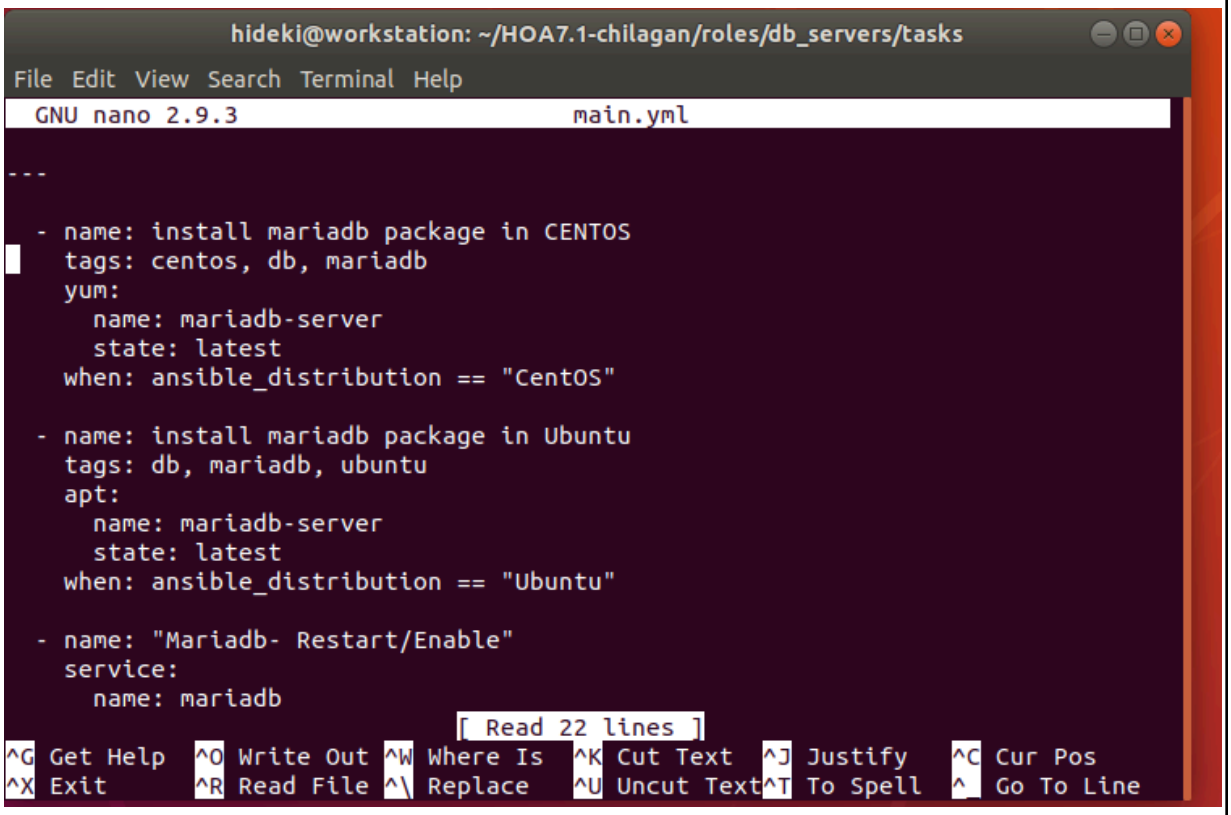
```
hideki@workstation:~/HOA7.1-chilagan$ cat roles/base/tasks/main.yml
---
- hosts: all
  become: true
  pre_tasks:

  - name: install updates (CentOS)
    tags: always
    dnf:
      update_only: yes
      update_cache: yes
    when: ansible_distribution == "CentOS"

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

- hosts: workstations
  become: true
  tasks:

  -  name: install unzip
     package:
       name: unzip
```

```
hideki@workstation:~/HOA7.1-chilagan$ tree roles
roles
├── base
│   └── tasks
│       └── main.yml
├── db_servers
│   └── tasks
│       └── main.yml
├── file_servers
│   └── tasks
│       └── main.yml
├── web_servers
│   └── tasks
│       └── main.yml
└── workstations
    └── tasks
        └── main.yml

10 directories, 5 files
```

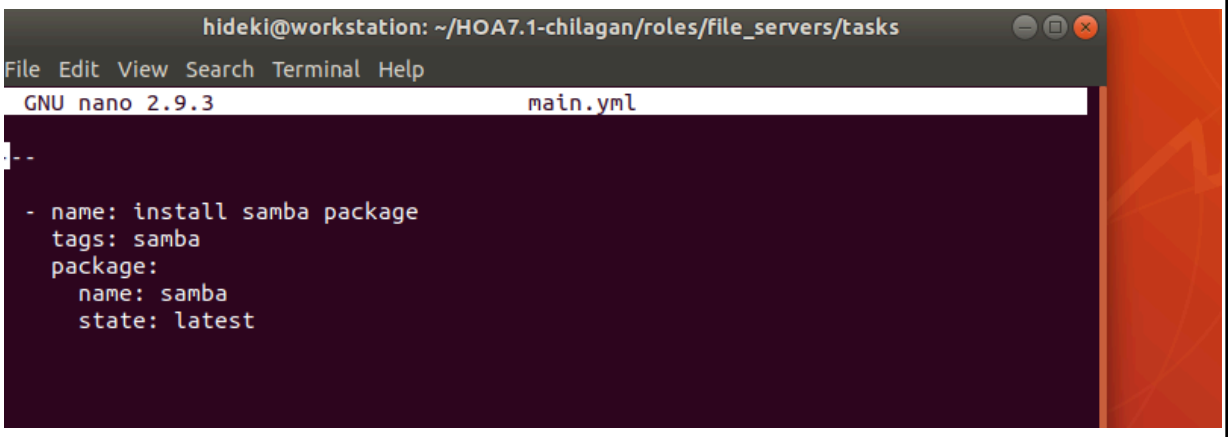
main.yml for base:



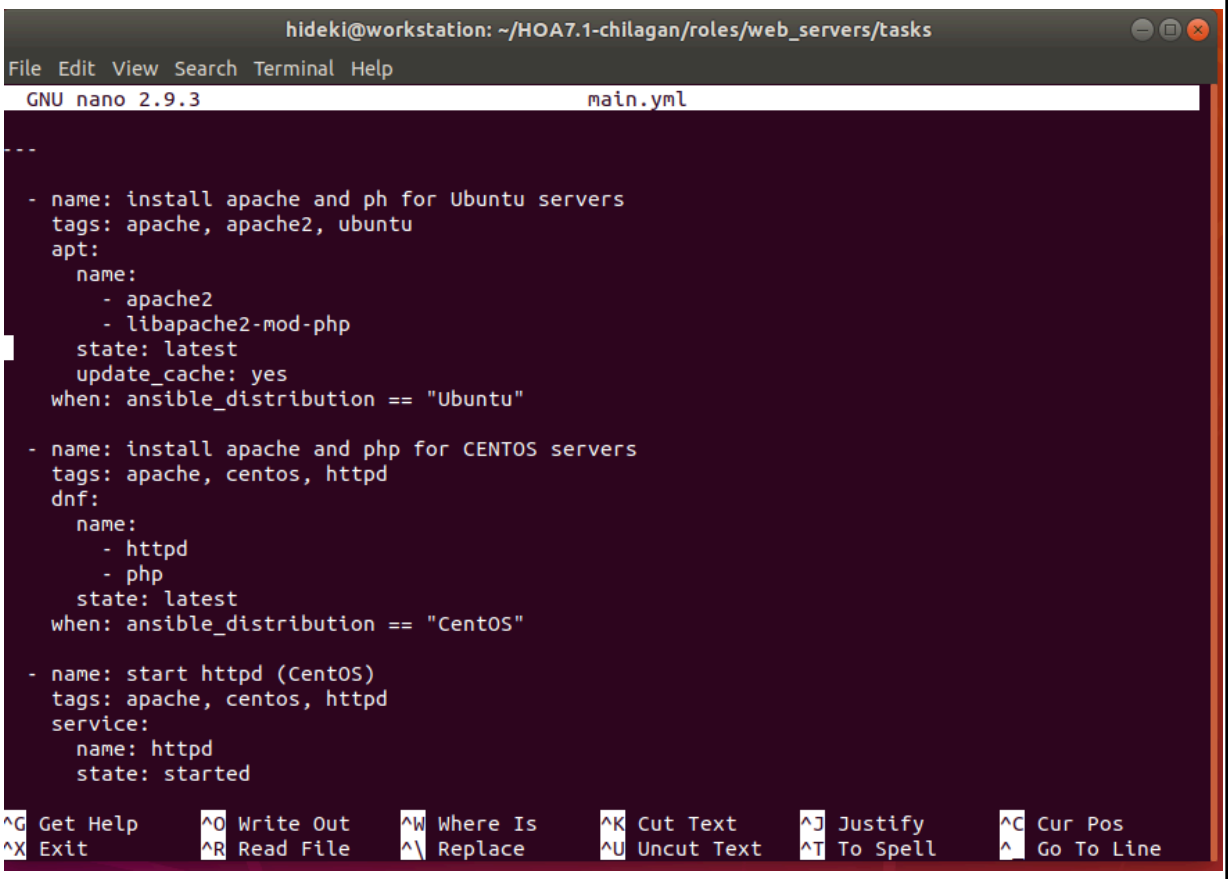
main.yml for db_servers:



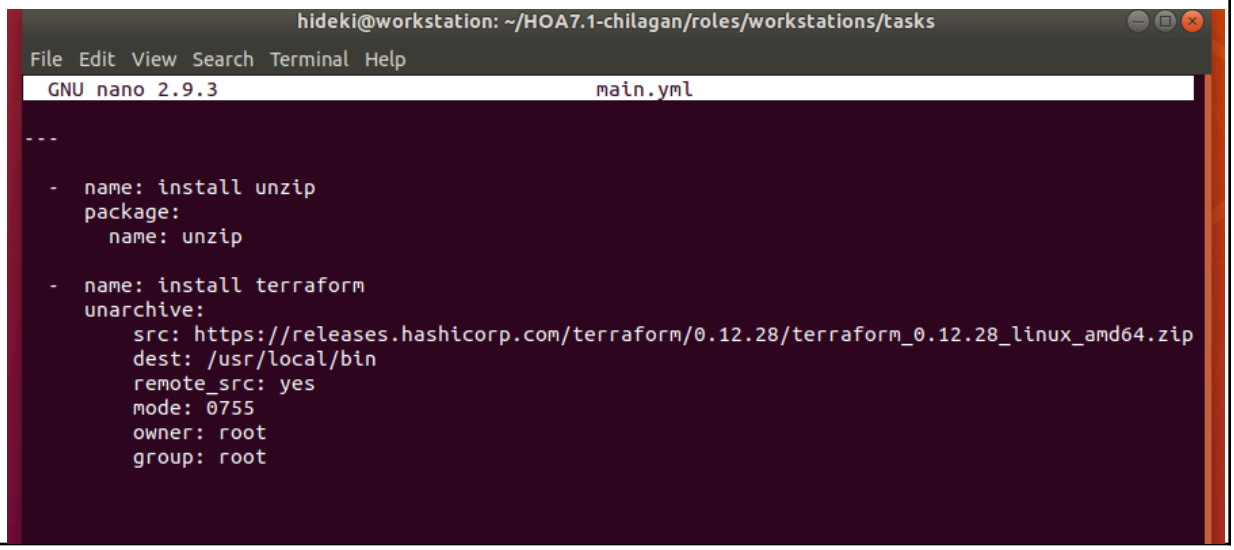
main.yml for file_servers:



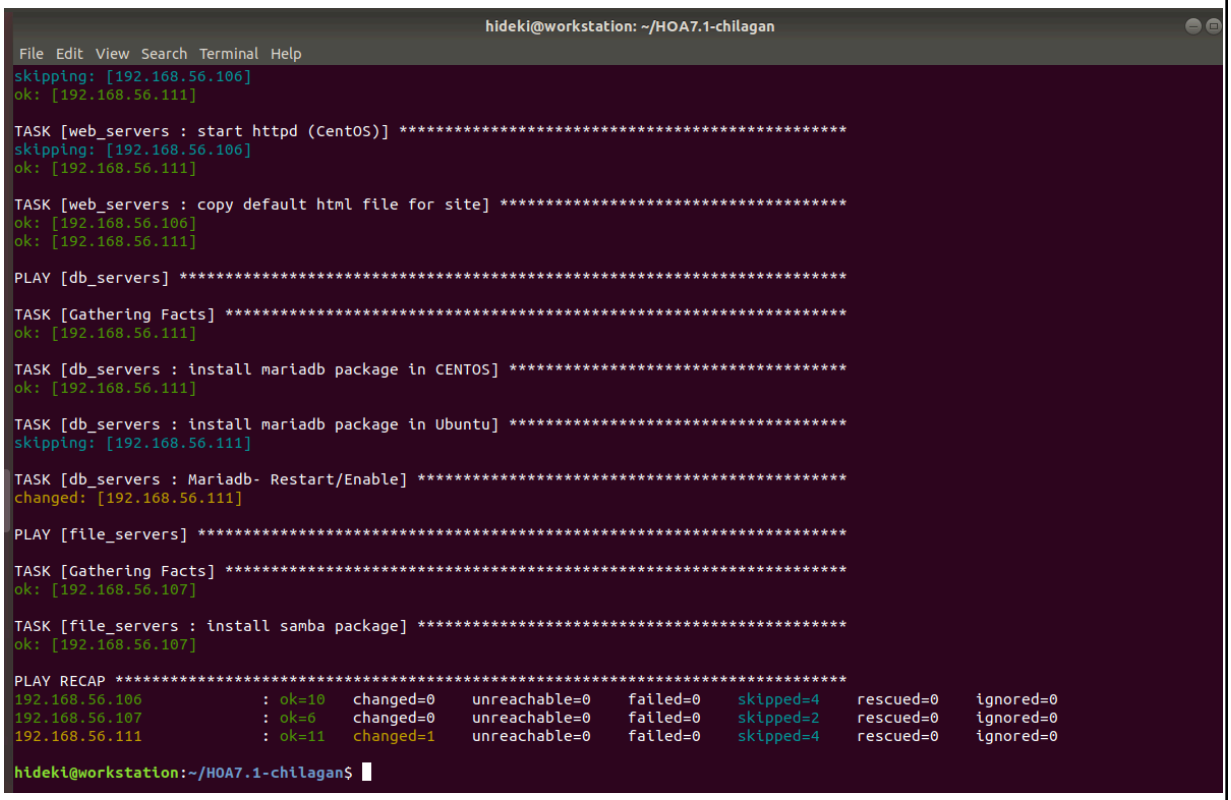
main.yml for web_servers:



main.yml for workstations:



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



- Each host ran its specific tasks included.

Reflections:

In this activity, we were able to utilize and learn how to copy content of files to a server's file in which we copied an html content from control node to the managed nodes without accessing the server directly. We were also able to run different tasks for different servers using roles. Not only is this type of process resistant to errors, but it is also a time efficient way to do commands.

Answer the following:

1. What is the importance of creating roles?
   Creating roles will provide an efficiency for the administrator in managing its task per server. For example, if the role of server 1 and 2 is to have a mariadb, you can add a command or inventory to install mariadb servers in this 2 server by using roles.

2. What is the importance of managing files?
    - Managing files can also provide an efficiency for the administrator since in this type of process, files from different servers can be easily manipulated, modified, and saved or deleted using the control node.