| Name: Reyes, Alexzander J. | Date Performed: 03/10/2025 |
|---|---|
| Course/Section: CPE 212- CPE31S2 | Date Submitted: 03/10/2025 |
| 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**:

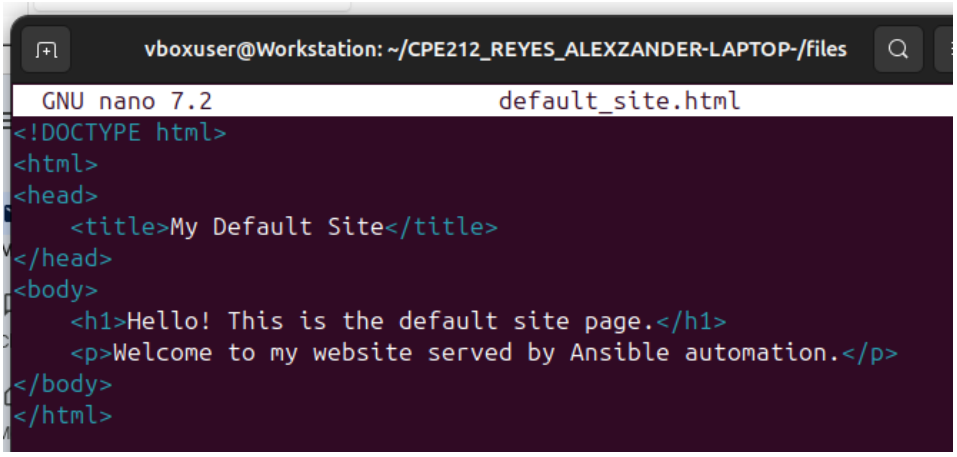
In this activity, we look at the concept of copying a file to a server. We are going to create a file into our git repository and use Ansible to grab that file and put it into a particular place so that we could do things like customize a default website, or maybe install a default configuration file.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

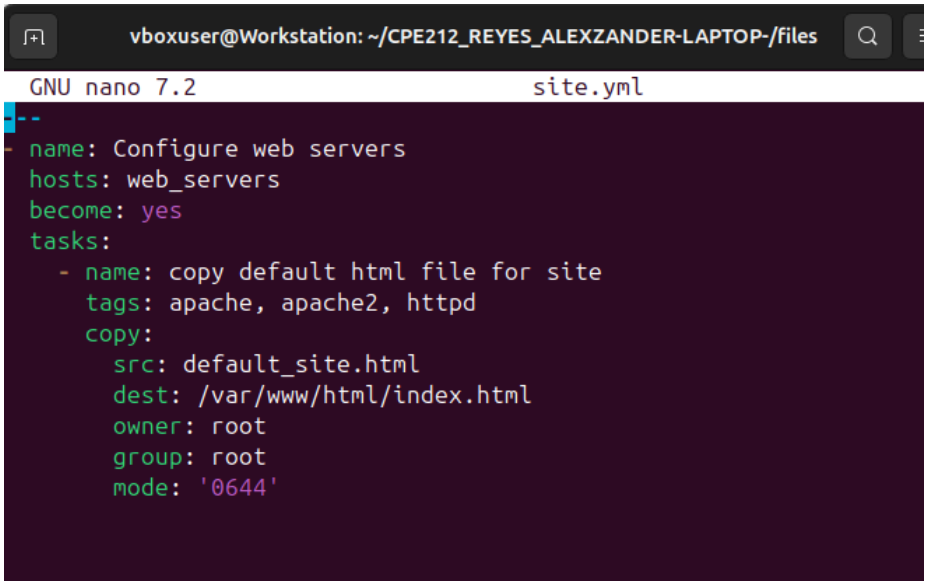
```
vboxuser@Workstation:~/CPE212_REYES_ALEXZANDER-LAPTOP-$ rm files
vboxuser@Workstation:~/CPE212_REYES_ALEXZANDER-LAPTOP-$ mkdir files
vboxuser@Workstation:~/CPE212_REYES_ALEXZANDER-LAPTOP-$ cs files
Command 'cs' not found, but can be installed with:
sudo apt install csound
vboxuser@Workstation:~/CPE212_REYES_ALEXZANDER-LAPTOP-$ cd files
vboxuser@Workstation:~/CPE212_REYES_ALEXZANDER-LAPTOP-/files$ sudo nano default_
site.html
vboxuser@Workstation:~/CPE212_REYES_ALEXZANDER-LAPTOP-/files$ ls
default_site.html
```

```
vboxuser@Workstation:~/CPE212_REYES_ALEXZANDER-LAPTOP-/files$ ls
default_site.html  inventory.ini  site.yml
```




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

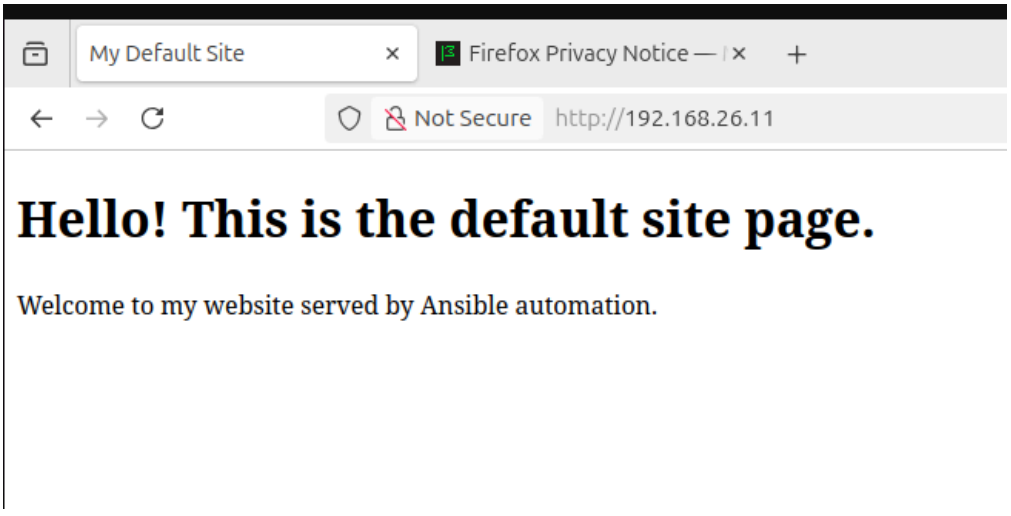



3. Run the playbook *site.yml*. Describe the changes.

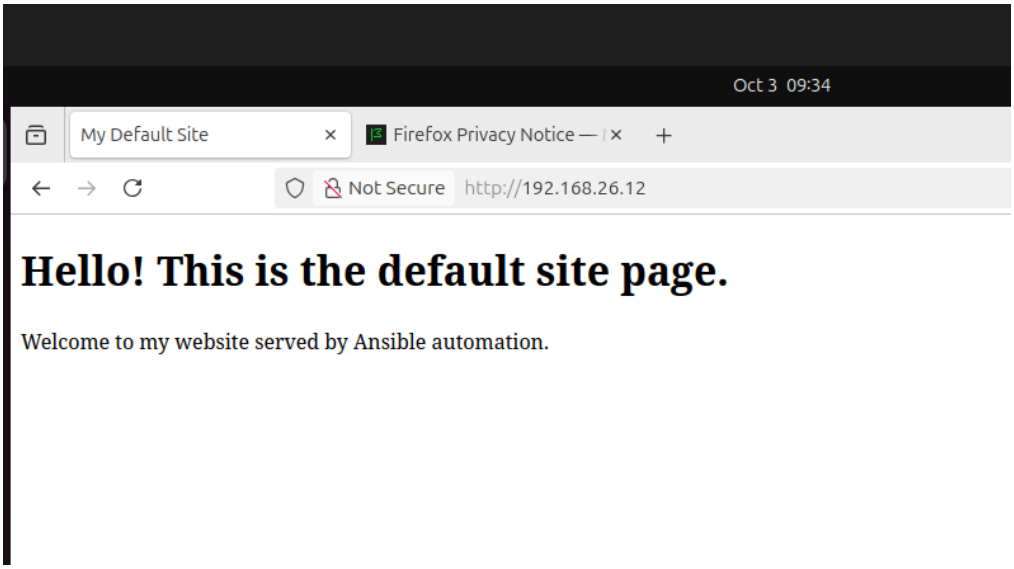
```
vboxuser@Workstation:~/CPE212_REYES_ALEXZANDER-LAPTOP-/files$ ansible-playbook -
i inventory.ini site.yml -K
BECOME password:

PLAY [Configure web servers] ***************************************************

TASK [Gathering Facts] *********************************************************
ok: [192.168.26.11]
ok: [192.168.26.12]

TASK [copy default html file for site] *****************************************
changed: [192.168.26.11]
changed: [192.168.26.12]

PLAY RECAP *********************************************************************
192.168.26.11              : ok=2    changed=1    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0
192.168.26.12              : ok=2    changed=1    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0
```

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.

```
vboxuser@Server1:~$ cat /var/www/html/index.html
<!DOCTYPE html>
<html>
<head>
    <title>My Default Site</title>
</head>
<body>
    <h1>Hello! This is the default site page.</h1>
    <p>Welcome to my website served by Ansible automation.</p>
</body>
</html>
```



```
vboxuser@Server2:~$ cat /var/www/html/index.html
<!DOCTYPE html>
<html>
<head>
    <title>My Default Site</title>
</head>
<body>
    <h1>Hello! This is the default site page.</h1>
    <p>Welcome to my website served by Ansible automation.</p>
</body>
</html>
```



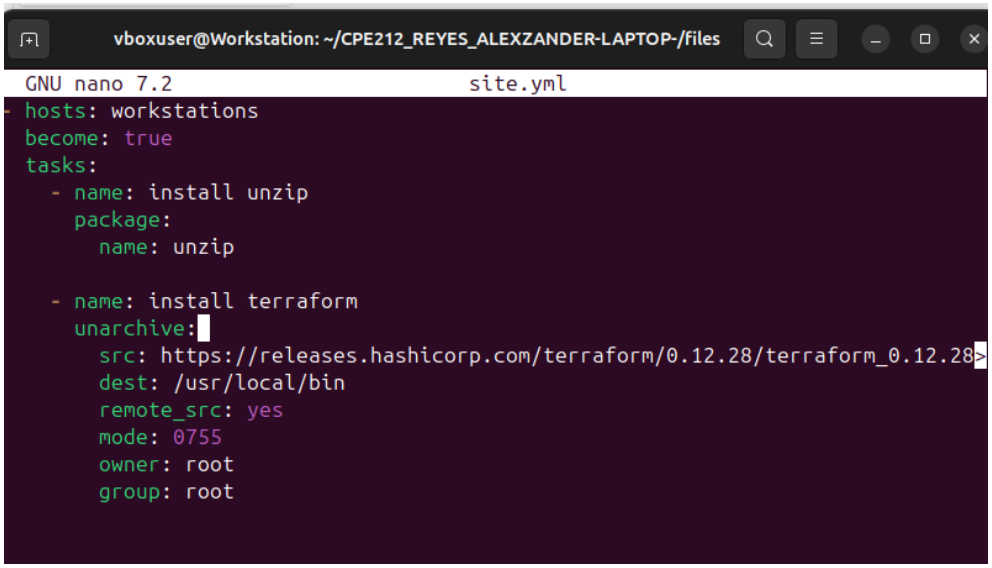

5. Sync your local repository with GitHub and describe the changes.

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

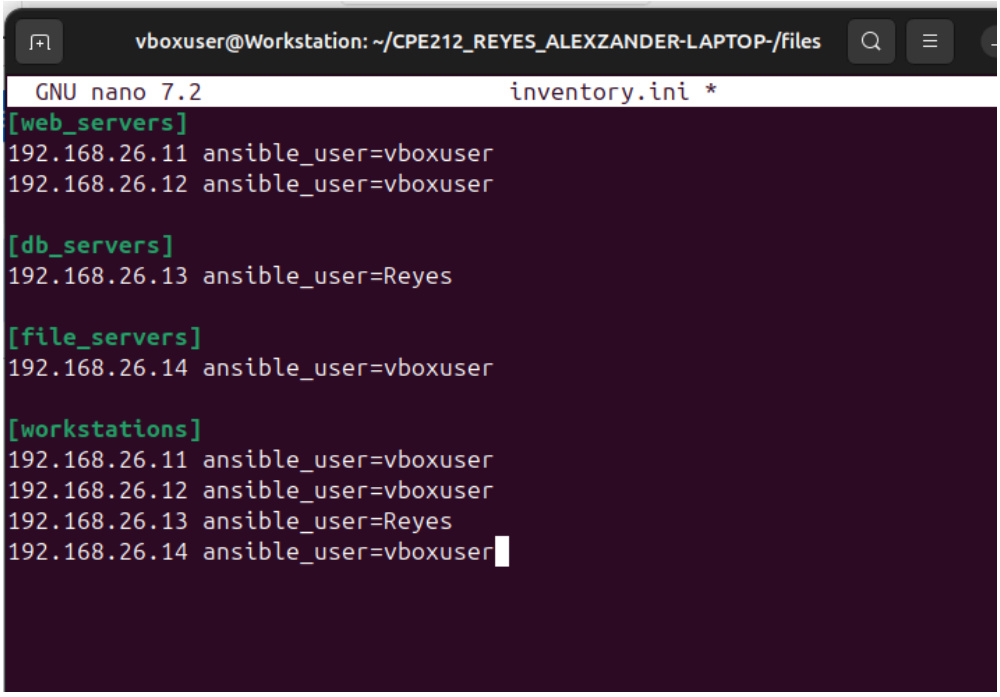
1. Edit the site.yml. Just before the web_servers play, create a new play:

```
- 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




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.

```
vboxuser@Workstation:~/CPE212_REYES_ALEXZANDER-LAPTOP-/files$ ansible-playbook -
i inventory.ini site.yml -K
BECOME password:

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

TASK [Gathering Facts] *********************************************************
ok: [192.168.26.11]
ok: [192.168.26.13]
ok: [192.168.26.12]
ok: [192.168.26.14]

TASK [install unzip] ***********************************************************
ok: [192.168.26.14]
ok: [192.168.26.11]
ok: [192.168.26.12]
ok: [192.168.26.13]

TASK [install terraform] *******************************************************
changed: [192.168.26.14]
changed: [192.168.26.12]
changed: [192.168.26.11]
changed: [192.168.26.13]
```

```
PLAY RECAP *********************************************************************
192.168.26.11              : ok=3    changed=1    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0
192.168.26.12              : ok=3    changed=1    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0
192.168.26.13              : ok=3    changed=1    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0
192.168.26.14              : ok=3    changed=1    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0
```

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

5.

```
vboxuser@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
```

```
vboxuser@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
    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
```

**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.

```
vboxuser@Workstation:~/CPE212_REYES_ALEXZANDER-LAPTOP-/files$ cd
vboxuser@Workstation:~$ cd CPE212_REYES_ALEXZANDER-LAPTOP-/
vboxuser@Workstation:~/CPE212_REYES_ALEXZANDER-LAPTOP-$ mkdir roles
vboxuser@Workstation:~/CPE212_REYES_ALEXZANDER-LAPTOP-$ cd roles
vboxuser@Workstation:~/CPE212_REYES_ALEXZANDER-LAPTOP-/roles$ mkdir web_servers
file_servers db_servers workstations
vboxuser@Workstation:~/CPE212_REYES_ALEXZANDER-LAPTOP-/roles$ ls
db_servers  file_servers  web_servers  workstations
vboxuser@Workstation:~/CPE212_REYES_ALEXZANDER-LAPTOP-/roles$ mkdir base
vboxuser@Workstation:~/CPE212_REYES_ALEXZANDER-LAPTOP-/roles$ ls
base  db_servers  file_servers  web_servers  workstations
vboxuser@Workstation:~/CPE212_REYES_ALEXZANDER-LAPTOP-/roles$ mkdir base/tasks
vboxuser@Workstation:~/CPE212_REYES_ALEXZANDER-LAPTOP-/roles$ mkdir db_servers/t
asks
vboxuser@Workstation:~/CPE212_REYES_ALEXZANDER-LAPTOP-/roles$ mkdir file_servers
/tasks
vboxuser@Workstation:~/CPE212_REYES_ALEXZANDER-LAPTOP-/roles$ mkdir web_servers/
tasks
vboxuser@Workstation:~/CPE212_REYES_ALEXZANDER-LAPTOP-/roles$ mkdir 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.

```
vboxuser@Workstation:~/CPE212_REYES_ALEXZANDER-LAPTOP-/roles/workstations/tasks$
 ls
main.yml
```

```
vboxuser@Workstation:~/CPE212_REYES_ALEXZANDER-LAPTOP-/roles/base/tasks$ ls
main.yml
```

```
vboxuser@Workstation:~/CPE212_REYES_ALEXZANDER-LAPTOP-/roles/web_servers/tasks$
ls
main.yml
```

```
vboxuser@Workstation:~/CPE212_REYES_ALEXZANDER-LAPTOP-/roles/db_servers/tasks$ l
s
main.yml
```

## web_servers

vboxuser@Workstation: ~/CPE212_REYES_ALEXZANDER-LAPTOP-/roles/web_ser...

GNU nano 7.2 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"
```

## db_servers

vboxuser@Workstation: ~/CPE212_REYES_ALEXZANDER-LAPTOP-/roles/db_serv...

GNU nano 7.2 main.yml

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

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

- name: Mariadb - Restarting/Enabling
  service:
    name: mariadb
    state: restarted
    enabled: true
```

## file_servers

vboxuser@Workstation: ~/CPE212_REYES_ALEXZANDER-LAPTOP-/roles/file_ser...

GNU nano 7.2 main.yml

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

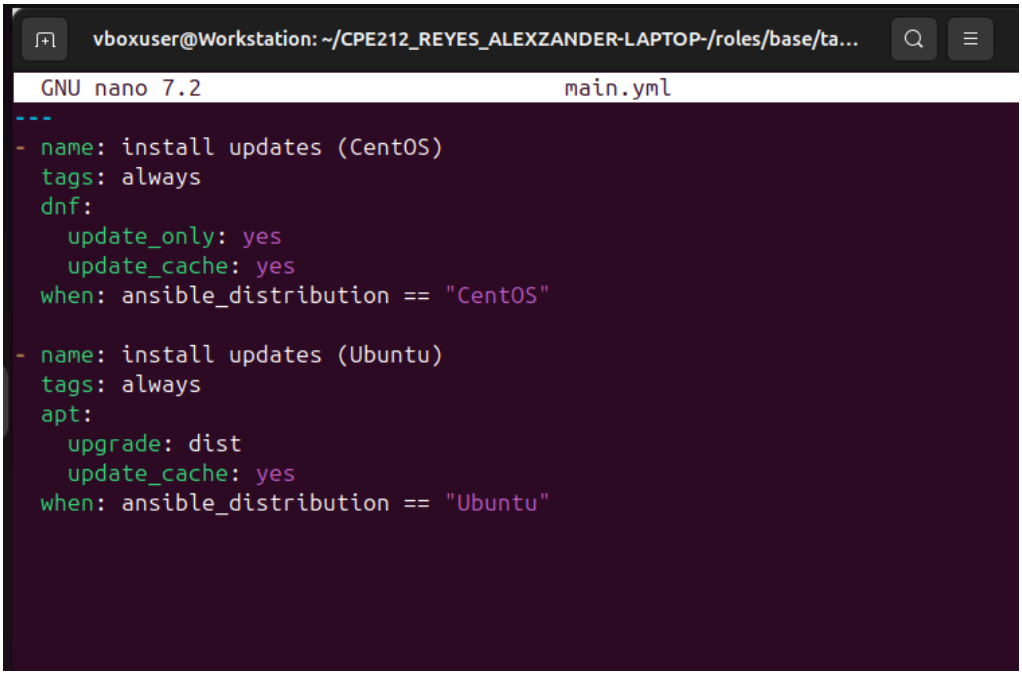
## workstations

vboxuser@Workstation: ~/CPE212_REYES_ALEXZANDER-LAPTOP-/roles/worksta...

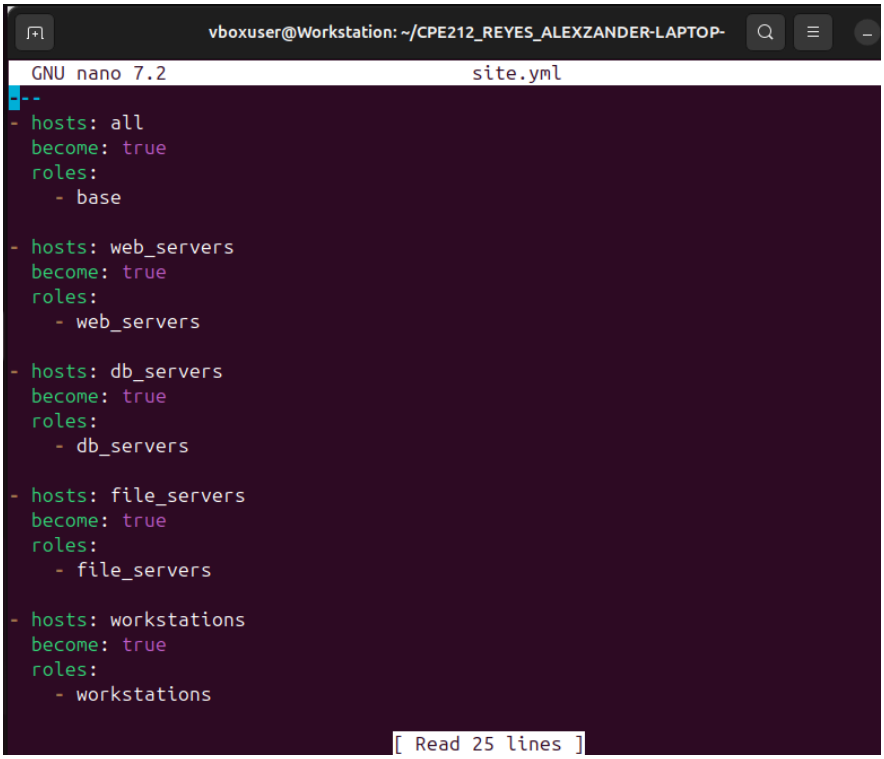
GNU nano 7.2 main.yml

```
---
- name: install unzip
  package:
    name: unzip
    state: present

- name: install terraform
  unarchive:
    src: https://releases.hashicorp.com/terraform/0.12.28/terraform_0.12.28_linux_am>
    dest: /usr/local/bin
    remote_src: yes
    mode: '0755'
    owner: root
    group: root
```

## base




## site.yml



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

```
vboxuser@Workstation:~/CPE212_REYES_ALEXZANDER-LAPTOP-$ ansible-playbook -i inventory
.ini site.yml -K
BECOME password:

PLAY [all] *********************************************************************

TASK [Gathering Facts] *********************************************************
ok: [192.168.26.11]
ok: [192.168.26.13]

TASK [base : install updates (CentOS)] *****************************************
skipping: [192.168.26.11]
ok: [192.168.26.13]

TASK [base : install updates (Ubuntu)] *****************************************
skipping: [192.168.26.13]
ok: [192.168.26.11]

PLAY [web_servers] *************************************************************

TASK [Gathering Facts] *********************************************************
ok: [192.168.26.11]
ok: [192.168.26.13]

TASK [web_servers : install apache and php for Ubuntu servers] *****************
skipping: [192.168.26.13]
ok: [192.168.26.11]

TASK [web_servers : install apache and php for CentOS servers] *****************
```

```
TASK [web_servers : install apache and php for CentOS servers] *****************
skipping: [192.168.26.11]
ok: [192.168.26.13]

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

TASK [Gathering Facts] *********************************************************
ok: [192.168.26.13]

TASK [db_servers : install mariadb package (CentOS)] ***************************
ok: [192.168.26.13]

TASK [db_servers : install mariadb package (Ubuntu)] ***************************
skipping: [192.168.26.13]

TASK [db_servers : Mariadb - Restarting/Enabling] ******************************
changed: [192.168.26.13]

PLAY [file_servers] ************************************************************

TASK [Gathering Facts] *********************************************************
ok: [192.168.26.11]
ok: [192.168.26.13]

TASK [file_servers : install samba package] ************************************
ok: [192.168.26.11]
ok: [192.168.26.13]
```


**Reflections:**

Answer the following:

1. What is the importance of creating roles?
    - Creating roles in Ansible is important because it helps organize and structure playbooks into smaller, manageable parts. Each role can handle specific server functions, such as web, database, or file servers, which makes the automation process more organized and easier to maintain. Roles promote reusability, allowing the same configurations or tasks to be applied across multiple systems without rewriting the code. They also improve collaboration, as different team members can work on separate roles simultaneously, and make it easier to scale automation in larger environments.
2. What is the importance of managing files?
    - Managing files, on the other hand, is essential for ensuring consistency, reliability, and security across all systems. It allows administrators to automatically deploy, update, or modify configuration and application files on multiple machines efficiently. Proper file management prevents configuration drift, maintains uniform setups, and helps automate repetitive tasks that would otherwise require manual effort. Additionally, it ensures that files are stored securely with the right permissions, making the overall system administration process more efficient and controlled.