Name: Potestades, North Nygel G.	**Date Performed:** 9/19/25
Course/Section: CPE31S2	**Date Submitted:** 10/3/25
Instructor: Engr. Robin Valenzuela	**Semester and SY:** 1st Semester 2025-2026

Activity 7: Managing Files and Creating Roles in Ansible

1. Objectives:

1.1 Manage files in remote servers

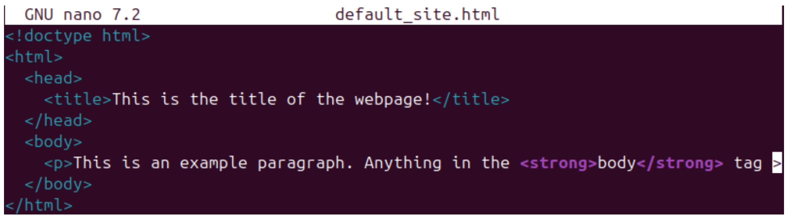
1.2 Implement roles in ansible

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.

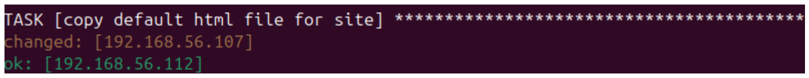


```
 GNU nano 7.2                    default_site.html
<!doctype html>
<html>
  <head>
    <title>This is the title of the webpage!</title>
  </head>
  <body>
    <p>This is an example paragraph. Anything in the <strong>body</strong> tag >
  </body>
</html>
```

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
```

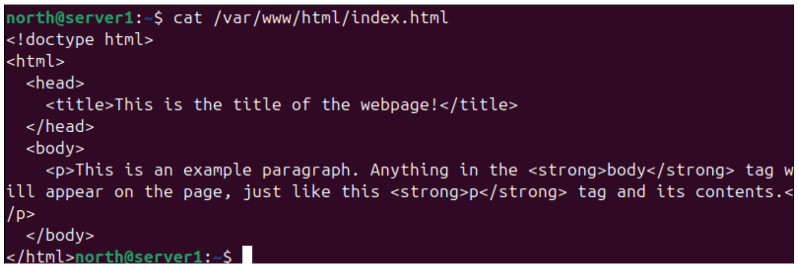
3. Run the playbook *site.yml*. Describe the changes.



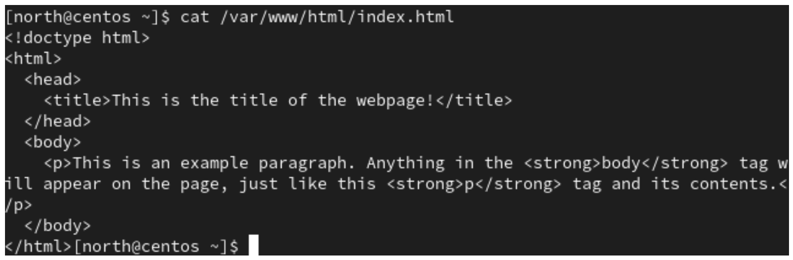
```
TASK [copy default html file for site] ******************************************
changed: [192.168.56.107]
ok: [192.168.56.112]
```

<u>Other than the updates to packages as we have not opened our nodes recently, the html file is now copied to the web_servers virtual machines. One is okay as I changed the inventory file to fit the wanted output</u>

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.



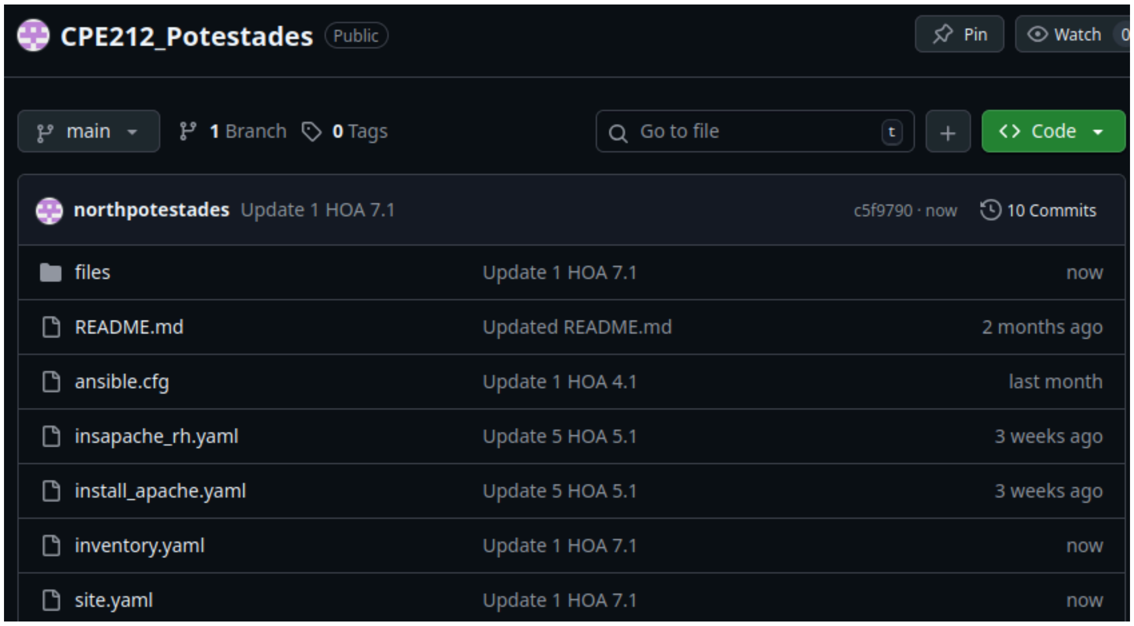
```
north@server1:~$ cat /var/www/html/index.html
<!doctype html>
<html>
  <head>
    <title>This is the title of the webpage!</title>
  </head>
  <body>
    <p>This is an example paragraph. Anything in the <strong>body</strong> tag w
ill appear on the page, just like this <strong>p</strong> tag and its contents.<
/p>
  </body>
</html>north@server1:~$
```



```
[north@centos ~]$ cat /var/www/html/index.html
<!doctype html>
<html>
  <head>
    <title>This is the title of the webpage!</title>
  </head>
  <body>
    <p>This is an example paragraph. Anything in the <strong>body</strong> tag w
ill appear on the page, just like this <strong>p</strong> tag and its contents.<
/p>
  </body>
</html>[north@centos ~]$
```

The file can be found in the specified location on both nodes, showing that the copy operation was successful.

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



northpotestades Update 1 HOA 7.1 c5f9790 · now 10 Commits

files	Update 1 HOA 7.1	now
README.md	Updated README.md	2 months ago
ansible.cfg	Update 1 HOA 4.1	last month
insapache_rh.yaml	Update 5 HOA 5.1	3 weeks ago
install_apache.yaml	Update 5 HOA 5.1	3 weeks ago
inventory.yaml	Update 1 HOA 7.1	now
site.yaml	Update 1 HOA 7.1	now

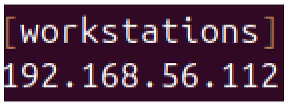
The modified files are now updated, and the files folder and the file inside of it were committed into the GitHub repository.

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:

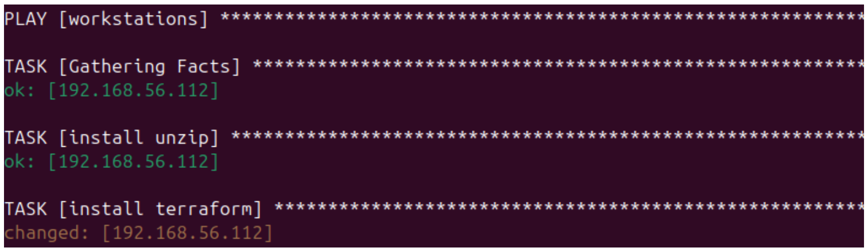
```
- hosts: workstations
  become: true
  tasks:

  - 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.



```
[workstations]
192.168.56.112
```

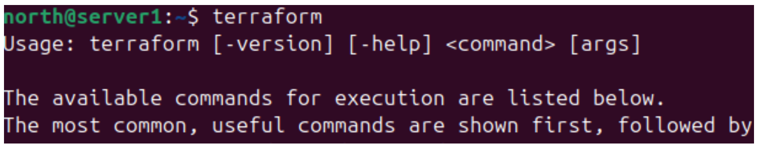
3. Run the playbook. Describe the output.



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

TASK [Gathering Facts] ********************************************************
ok: [192.168.56.112]

TASK [install unzip] **********************************************************
ok: [192.168.56.112]

TASK [install terraform] ******************************************************
changed: [192.168.56.112]
```

We can see that unzip was already installed on the system, and that terraform was installed as something was changed.

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

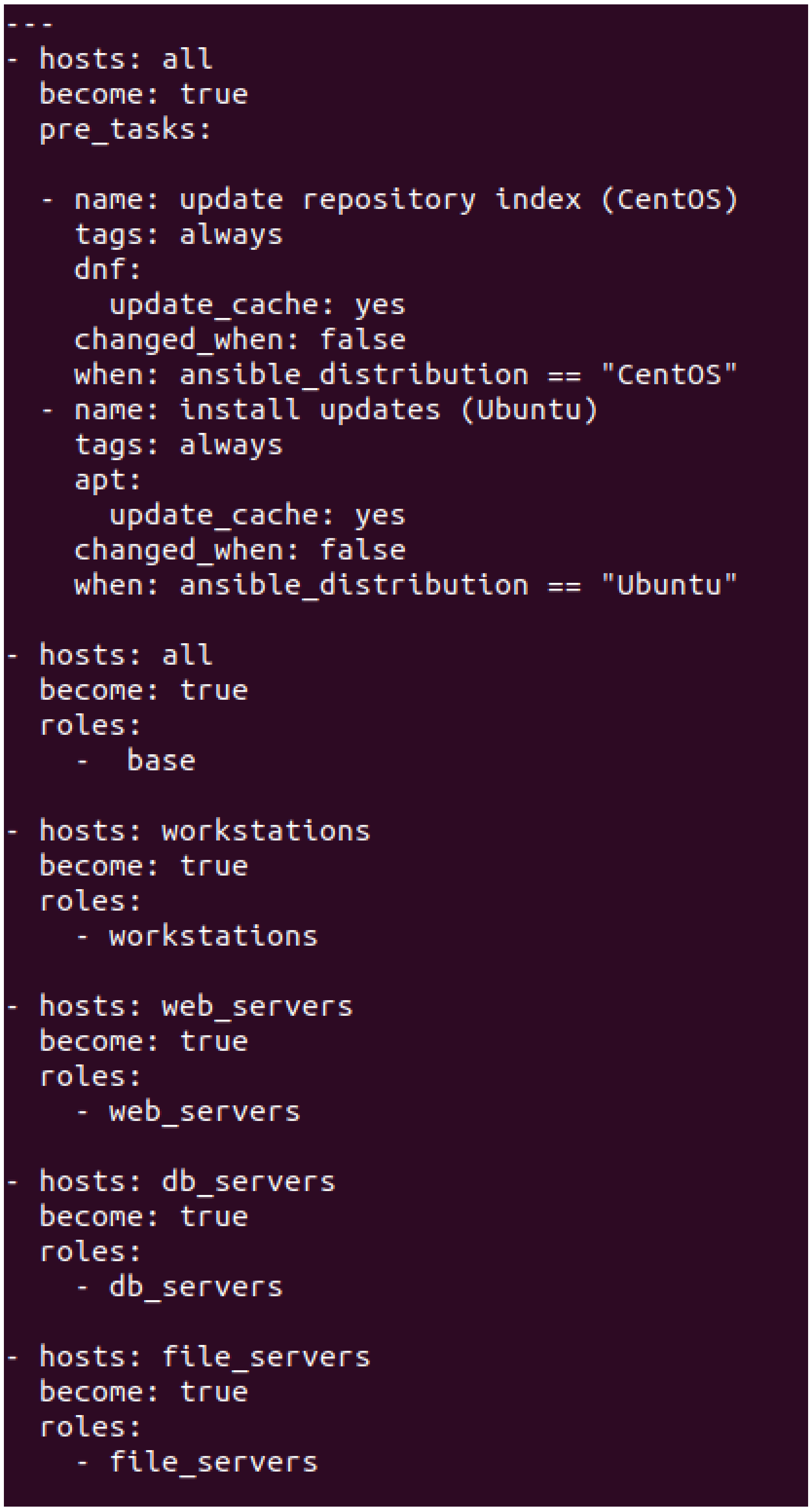


```
north@server1:~$ terraform
Usage: terraform [-version] [-help] <command> [args]

The available commands for execution are listed below.
The most common, useful commands are shown first, followed by
```

Based on this screenshot, we can confirm that terraform has been successfully installed, as it lists the different options for the terraform command.

Task 3: Create roles

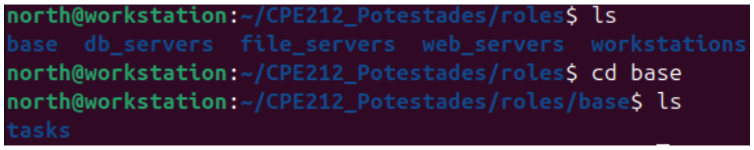
1. Edit the site.yml. Configure roles as follows: (make sure to create a copy of the old site.yml file because you will be copying the specific plays for all groups)



```
---
- 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
```

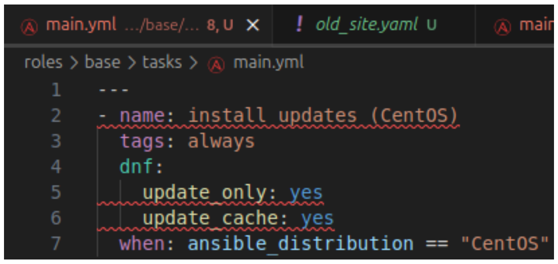
Save the file and exit.

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.

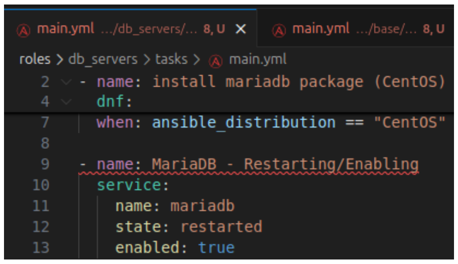


```
north@workstation:~/CPE212_Potestades/roles$ ls
base  db_servers  file_servers  web_servers  workstations
north@workstation:~/CPE212_Potestades/roles$ cd base
north@workstation:~/CPE212_Potestades/roles/base$ ls
tasks
```

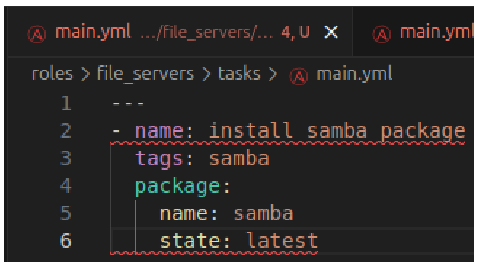
3. Go to tasks for all directory and create a file. Name it main.yml. In each of the tasks for all directories, copy and paste the code from the old site.yml file. Show all contents of main.yml files for all tasks.



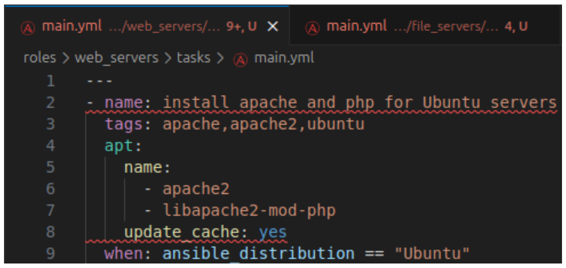
```
---
- name: install updates (CentOS)
  tags: always
  dnf:
    update_only: yes
    update_cache: yes
  when: ansible_distribution == "CentOS"
```


main.yml .../db_servers/... 8, U
main.yml .../base/... 8, U
roles > db_servers > tasks > main.yml

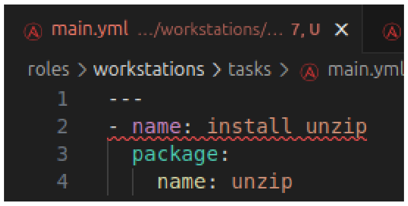
```
- name: install mariadb package (CentOS)
  dnf:
  when: ansible_distribution == "CentOS"

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


main.yml .../file_servers/... 4, U
main.yml
roles > file_servers > tasks > main.yml

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


main.yml .../web_servers/... 9+, U
main.yml .../file_servers/... 4, U
roles > web_servers > tasks > main.yml

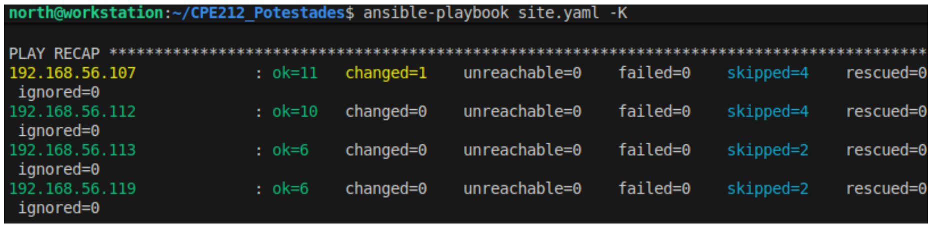
```
---
- name: install apache and php for Ubuntu servers
  tags: apache,apache2,ubuntu
  apt:
    name:
      - apache2
      - libapache2-mod-php
    update_cache: yes
  when: ansible_distribution == "Ubuntu"
```


main.yml .../workstations/... 7, U
roles > workstations > tasks > main.yml

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

The contents of each file only contains the code which needs to be run on that specified role, removing the hosts section from the main.yml files.

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



```
north@workstation:~/CPE212_Potestades$ ansible-playbook site.yaml -K

PLAY RECAP ****************************************************************************************
192.168.56.107             : ok=11   changed=1    unreachable=0    failed=0    skipped=4    rescued=0
 ignored=0
192.168.56.112             : ok=10   changed=0    unreachable=0    failed=0    skipped=4    rescued=0
 ignored=0
192.168.56.113             : ok=6    changed=0    unreachable=0    failed=0    skipped=2    rescued=0
 ignored=0
192.168.56.119             : ok=6    changed=0    unreachable=0    failed=0    skipped=2    rescued=0
 ignored=0
```

The output is the same as the old_site.yaml file, as this is the same playbook but distributed in different roles so that they may be called individually instead of all at once.

Reflections:

Answer the following:

1. What is the importance of creating roles?

Roles are important because they allow for the organization of specific tasks to the roles which you want them to be performed on. It organizes them into clusters of playbooks instead of one long playbook which is hard to navigate, as I was having trouble doing so myself.

2. What is the importance of managing files?

Managing files is important because this includes the basics such as creating, removing, and copying files. This allows you to copy files that are needed to each remote server that needs it without needing to download it from Google Drive or another online file-sharing system. It's run locally and much faster than downloading.

Updated Repository: northpotestades/CPE212_Potestades