Name: Escosia, Jerico James	**Date Performed: 11/10/24**
Course/Section: CPE212- CPE31S21	**Date Submitted: 11/10/24**
Instructor: Engr. 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.

```
workstation@workstation:~/Act7$ mkdir files
workstation@workstation:~/Act7$ cd files
workstation@workstation:~/Act7/files$ nano default_site/html
```

```
File  Edit  View  Search  Terminal  Help
  GNU nano 2.9.3                    default_site.html

<!DOCTYPE html>
<html>
<head>
<title> Test Website </title>
</head>
<body>

<h1> Jerico Escosia</h1>
<p> Hello World</p>

</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

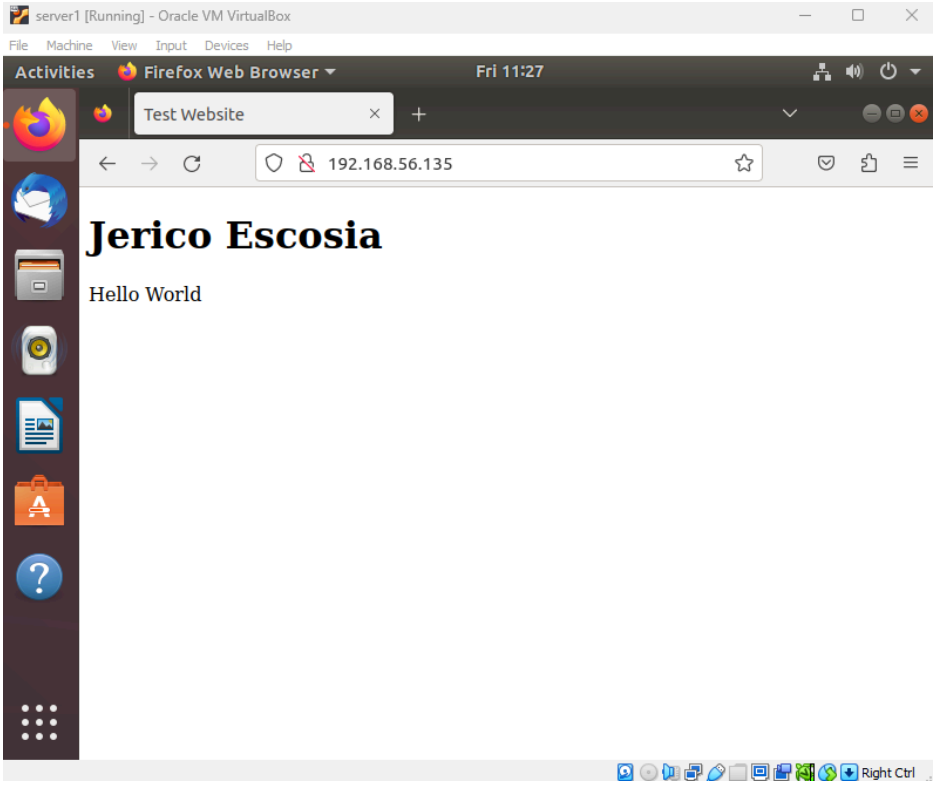
```
- hosts: db_server
  become: true
  tasks:

  - 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
  - name: install mariadb package (CentOS)
    yum:
      name: mariadb-server
      state: latest
    when: ansible_distribution is defined and ansible_distribution == "CentOS"
```

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

```
PLAY RECAP *********************************************************************
*
centos                     : ok=4    changed=0    unreachable=0    failed=0
server1                    : ok=6    changed=0    unreachable=0    failed=0
server3                    : ok=6    changed=0    unreachable=0    failed=0
```

Upon running the playbook, the ansible copies the html file inside the files directory on to the remote servers.

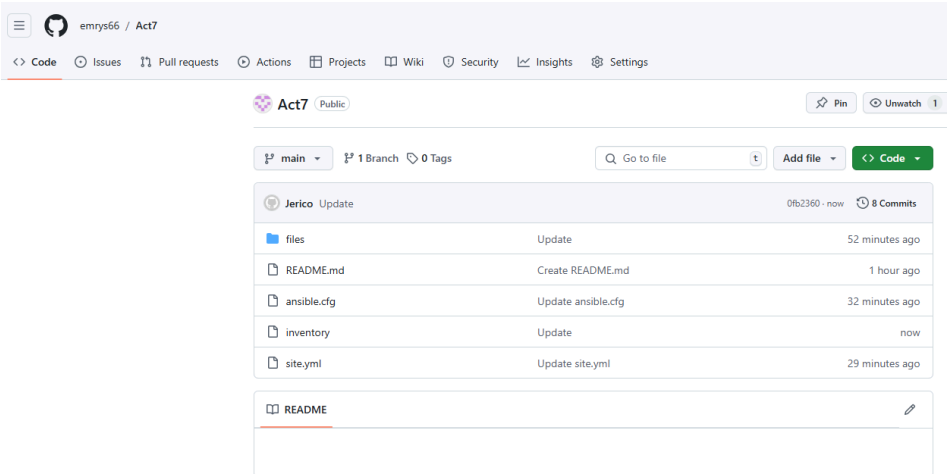
4. Go to the remote servers (*web_servers*) listed in your inventory. Use cat command to check if the index.html is the same as the local repository file (*default_site.html*). Do both for Ubuntu and CentOS servers. On the CentOS server, go to the browser and type its IP address. Describe the output.



When you type in the ip addresses of the remote servers on their browsers, the html file that was created in the directory will appear.

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

```
workstation@workstation:~/Act7$ git push origin main
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 293 bytes | 293.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To github.com:emrys66/Act7.git
   16442a7..0fb2360  main -> main
```



The changes I made on the workstation is now saved on my 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
```

```
File  Edit  View  Search  Terminal  Help
  GNU nano 2.9.3                    site.yml                    Modified

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

- hosts: fileserver
  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_
      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.

```
File  Edit  View  Search  Terminal  Help
  GNU nano 2.9.3                    inventory

[servers]
server1 ansible_host=192.168.56.135
server3 ansible_host=192.168.56.139

[db_server]
centos ansible_host=192.168.56.137 ansible_user=centos

[fileservers]
server1 ansible_host=192.168.56.135
```

3. Run the playbook. Describe the output.

```
PLAY [fileservers] ************************************************************
*

TASK [Gathering Facts] ********************************************************
*
ok: [server1]

TASK [install unzip] **********************************************************
*
ok: [server1]

TASK [install terraform] ******************************************************
*
ok: [server1]
```

When you run the play the ansible installed the zip file and installed the terraform file inside the ubuntu server.

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

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

Your version of Terraform is out of date! The latest version
is 1.9.7. You can update by downloading from https://www.terraform.io/downloads
.html
```

Upon checking on the terminal in server 1, the terraform is installed but it's out of date, it gives out a link with the updated version of terraform.

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.

```
workstation@workstation: ~/Act7
File  Edit  View  Search  Terminal  Help
  GNU nano 2.9.3                    site2.yml

- 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: servers
  become: true
  roles:
    - servers

- hosts: server_centOS
                                  [ Read 31 lines ]
```

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.

```
workstation@workstation:~/Act7$ mkdir -p roles/{servers,db_servers,fileservers}
/tasks
workstation@workstation:~/Act7$ ls
ansible.cfg  inventory  roles      site.retry
files        README.md  site2.yml  site.yml
workstation@workstation:~/Act7$ cd rolse
bash: cd: rolse: No such file or directory
workstation@workstation:~/Act7$ cd roles
workstation@workstation:~/Act7/roles$ ls
db_servers  fileservers  servers
```

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.

```
workstation@workstation:~/Act7$ cd roles
workstation@workstation:~/Act7/roles$ ls
db_servers  fileservers  servers
workstation@workstation:~/Act7/roles$ cd db_servers
workstation@workstation:~/Act7/roles/db_servers$ ls
tasks
workstation@workstation:~/Act7/roles/db_servers$ cd tasks
workstation@workstation:~/Act7/roles/db_servers/tasks$ nano main.yml
workstation@workstation:~/Act7/roles/db_servers/tasks$
```

workstation@workstation: ~/Act7/roles/db_servers/tasks

File Edit View Search Terminal Help

```
  GNU nano 2.9.3                    main.yml

---
- hosts: all
  become: true
  pre_tasks:

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

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

- hosts: fileservers
  become: true
  tasks:
```

```
workstation@workstation:~/Act7/roles$ cd servers
workstation@workstation:~/Act7/roles/servers$ cd tasks
workstation@workstation:~/Act7/roles/servers/tasks$ nano main.yml
```

```
workstation@workstation: ~/Act7/roles/servers/tasks
File  Edit  View  Search  Terminal  Help
  GNU nano 2.9.3                    main.yml

---
- hosts: all
  become: true
  pre_tasks:

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

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

- hosts: fileservers
  become: true
  tasks:
```

```
workstation@workstation:~/Act7/roles$ cd fileservers
workstation@workstation:~/Act7/roles/fileservers$ cd tasks
workstation@workstation:~/Act7/roles/fileservers/tasks$ nano main.yml
workstation@workstation:~/Act7/roles/fileservers/tasks$
```

```
workstation@workstation: ~/Act7/roles/fileservers/tasks
File  Edit  View  Search  Terminal  Help
  GNU nano 2.9.3                    main.yml

---
- hosts: all
  become: true
  pre_tasks:

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

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

- hosts: fileservers
  become: true
  tasks:
```

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

```
TASK [install mariadb package (CentOS)] ************************************
*
skipping: [centos]

PLAY [fileservers] *********************************************************
*

TASK [Gathering Facts] *****************************************************
*
ok: [server1]

TASK [install samba package] ***********************************************
*
ok: [server1]

PLAY RECAP *****************************************************************
*
centos                     : ok=4    changed=0    unreachable=0    failed=0
server1                    : ok=10   changed=0    unreachable=0    failed=0
server3                    : ok=5    changed=0    unreachable=0    failed=0

workstation@workstation:~/Act7$
```

When running the new playbook, the output was the same as the one on the original playbook earlier.

Reflections:

Answer the following:

1. What is the importance of creating roles?

   Creating roles in ansible is important in a way that it makes it easier to manage or point out which tasks have an error, it makes the playbook more organized, it also allows you to maintain and update individual roles independently.

2. What is the importance of managing files?

   Managing files in system administration is crucial for maintaining integrity, security, and performance of computer systems in a network. Properly managing these files using ansible makes the taks much more easier and efficient.