Name: San Jose, Kier Justin F.	Date Performed: 11/10/2024
Course/Section: CPE212 - CPE31S21	Date Submitted: 11/10/2024
Instructor: Engr. Robin Valenzuela	Semester and SY: 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.

```
kier@hostname:~/hoa7.1$ cd files
kier@hostname:~/hoa7.1/files$
```

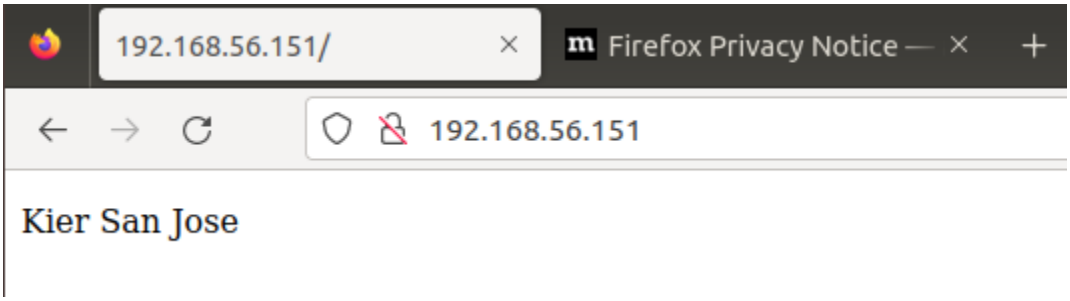
```
<!DOCTYPE html>
<html>
<body>
        <p>Kier San Jose</p>
</body>
</html>
```

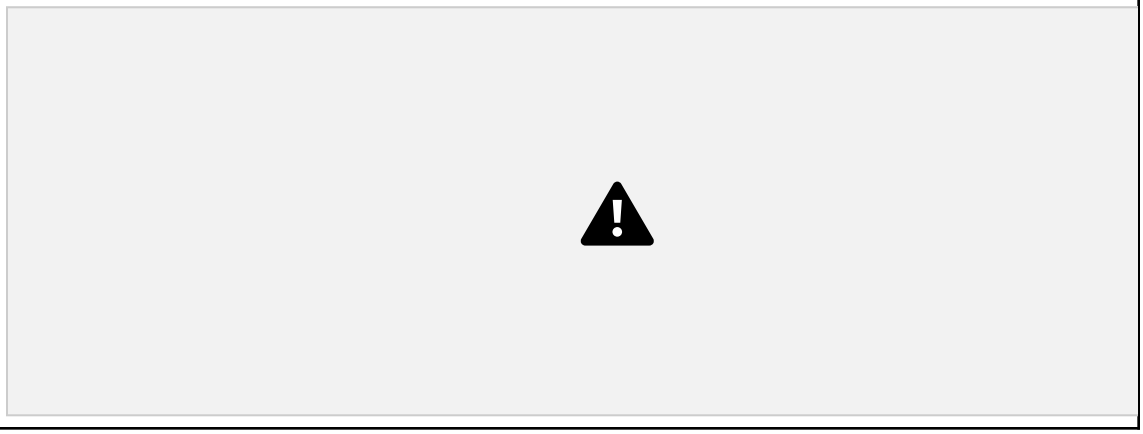
2. Edit the *site.yml* file and just below the *web_servers* play, create a new file to copy the default html file for site:
   - name: copy default html file for site

     tags: apache, apache2, httpd
     copy:
       src: default_site.html
       dest: /var/www/html/index.html
       owner: root
       group: root
       mode: 0644
3. Run the playbook *site.yml*. Describe the changes.

```
TASK [copy default html file for site] **
*
changed: [server1]
changed: [test3]
changed: [centos]
```

   - It displays the design you created in the html file and it needs to be in the correct ip for that to work.
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.





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

```
    package:
      name: unzip

  - name: install terraform
    unarchive:
      src: https://releases.hashicorp.com/terraform/0.12.28/terraform_0
      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.

```
[servers]
server1 ansible_host=192.168.56.151 ansible_user=kier
test3 ansible_host=192.168.56.155 ansible_user=kier

[server_centos]
centos ansible_host=192.168.56.154 ansible_user=kiersanjose

[fileservers]
server1 ansible_host=192.168.56.151 ansible_user=kier
```

3. Run the playbook. Describe the output.

```
TASK [copy default html file for site] ********************************
*
ok: [server1]
ok: [test3]

PLAY [db_server] ******************************************************
*
skipping: no hosts matched

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

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

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

PLAY RECAP ************************************************************
*
centos                     : ok=1    changed=0    unreachable=0    faile
server1                    : ok=10   changed=2    unreachable=0    faile
test3                      : ok=5    changed=0    unreachable=0    faile
```

- It installs the terraform as we did in the playbook.

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

```
kier@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/down
```

- It successfully installs the terraform and it shows the version of it.

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.

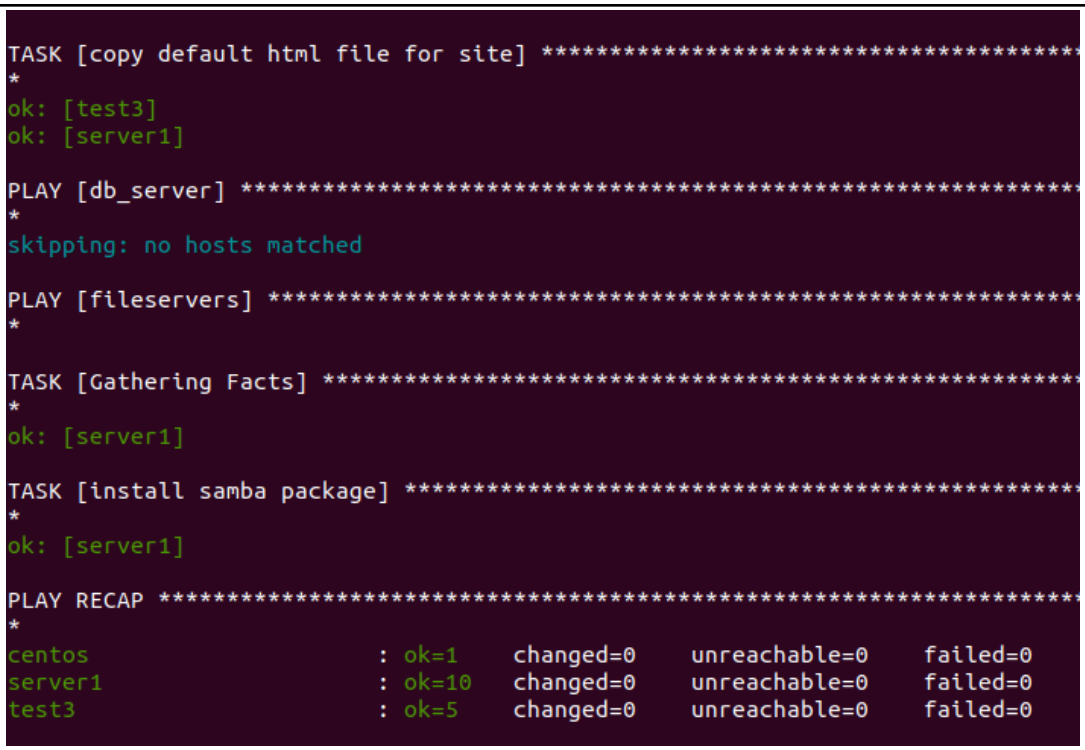
2. Under the same directory, create a new directory and name it roles. Enter the roles directory and create new directories: base, web_servers, file_servers, db_servers and workstations. For each directory, create a directory and name it tasks.

```
kier@hostname:~/hoa7.1$ mkdir -p roles/{servers,db_servers,fileservers}/tas
kier@hostname:~/hoa7.1$ ls
ansible.cfg  files      README.md  site.retry
apache.yml   inventory  roles      site.yml
kier@hostname:~/hoa7.1$ cd roles
kier@hostname:~/hoa7.1/roles$ ls
db_servers  fileservers  servers
kier@hostname:~/hoa7.1/roles$ cd db_servers
kier@hostname:~/hoa7.1/roles/db_servers$ nano main.yml
kier@hostname:~/hoa7.1/roles/db_servers$ cd
kier@hostname:~$ cd hoa7.1
kier@hostname:~/hoa7.1$ cd roles
kier@hostname:~/hoa7.1/roles$ cd fileservers
kier@hostname:~/hoa7.1/roles/fileservers$ nano main.yml
kier@hostname:~/hoa7.1/roles/fileservers$ cd ..
kier@hostname:~/hoa7.1/roles$ cd servers
kier@hostname:~/hoa7.1/roles/servers$ nano main.yml
```

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.

```
kier@hostname:~/hoa7.1$ mkdir -p roles/{servers,db_servers,fileservers}/tas
kier@hostname:~/hoa7.1$ ls
ansible.cfg  files      README.md  site.retry
apache.yml   inventory  roles      site.yml
kier@hostname:~/hoa7.1$ cd roles
kier@hostname:~/hoa7.1/roles$ ls
db_servers  fileservers  servers
kier@hostname:~/hoa7.1/roles$ cd db_servers
kier@hostname:~/hoa7.1/roles/db_servers$ nano main.yml
kier@hostname:~/hoa7.1/roles/db_servers$ cd
kier@hostname:~$ cd hoa7.1
kier@hostname:~/hoa7.1$ cd roles
kier@hostname:~/hoa7.1/roles$ cd fileservers
kier@hostname:~/hoa7.1/roles/fileservers$ nano main.yml
kier@hostname:~/hoa7.1/roles/fileservers$ cd ..
kier@hostname:~/hoa7.1/roles$ cd servers
kier@hostname:~/hoa7.1/roles/servers$ nano main.yml
```

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



- It is the same output as the previous playbook.

Reflections:

Answer the following:

1. What is the importance of creating roles?
   - The Roles and permissions that users can have are determined by roles, which are preset user groups or access levels. Administrators with roles can have management powers assigned to them for each access group.
2. What is the importance of managing files?
   - File management has several advantages, such as keeping things organized, facilitating sharing, lowering the possibility of losing crucial information, and offering a backup in case something goes wrong.