Activity 7: Managing Files and Creating Roles in Ansible	
Name: Buduan, Christian Aaron C.	**Date Performed:** 07/10/24
Course/Section: CPE31S2	**Date Submitted:** 09/10/24
Instructor: Engr. Robin Valenzuela	**Semester and SY:** 1st sem 2024-2025

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.

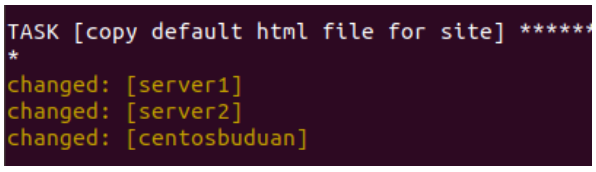
Task 1: Create a file and copy it to remote servers

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
```

3. Run the playbook *site.yml*. Describe the changes. **running the site.yml with the added html file causes it to copy the default html file to the assigned servers under group web_servers.**

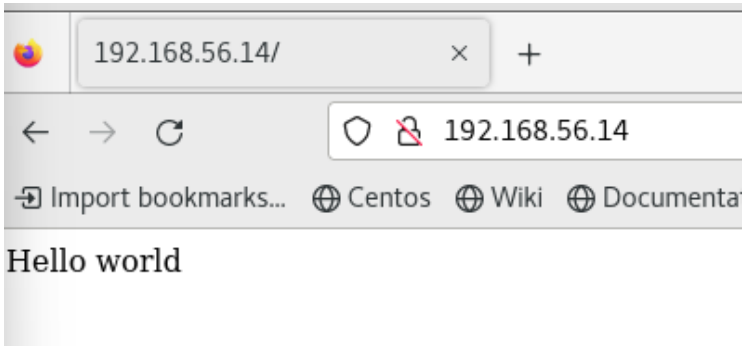


```
PLAY RECAP *********************************************************************
*
centosbuduan               : ok=11   changed=2    unreachable=0    failed=0
skipped=3    rescued=0    ignored=0
server1                    : ok=5    changed=1    unreachable=0    failed=0
skipped=3    rescued=0    ignored=0
server2                    : ok=5    changed=1    unreachable=0    failed=0
skipped=3    rescued=0    ignored=0
server3                    : ok=5    changed=1    unreachable=0    failed=0
skipped=2    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. **the contents of the default_site.html is copied to the index.html in the remote servers under the group web_servers, you can verify this by using cat command on index.html or you can enter the IP address of centOS in a browser and it runs the index.html file**

```
qcacbuduan@server1:~$ cat /var/www/html/index.html
<html>
        <body>
                <header>Hello world</header>
        </body>
<html>
qcacbuduan@server1:~$
```

```
qcacbuduan@server2:~$ cat /var/www/html/index.html
<html>
        <body>
                <header>Hello world</header>
        </body>
<html>
```



5. Sync your local repository with GitHub and describe the changes.
   I pushed the files from my local repo to the github repo to upload and update my playbooks and directories. I learned that you can also add the contents of the directory by just uploading the directory itself.

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

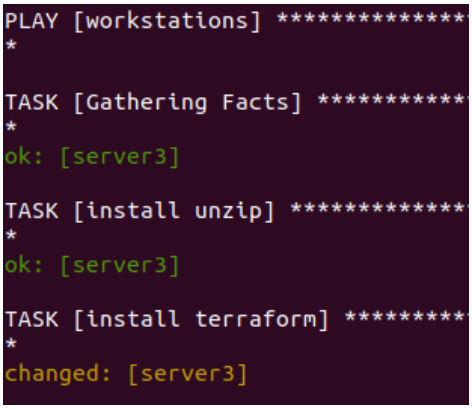
1. Edit the site.yml. Just before the web_servers play, create a new play:

```
- 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. **When running the playbook, it performs installation of the unzip package and downloads terraform using the official download website of terraform on the workstations group which I assigned server 3 into.**



```
centosbuduan               : ok=11   changed=1    unreachable=0    failed=0
skipped=3    rescued=0    ignored=0
server1                    : ok=5    changed=0    unreachable=0    failed=0
skipped=3    rescued=0    ignored=0
server2                    : ok=5    changed=0    unreachable=0    failed=0
skipped=3    rescued=0    ignored=0
server3                    : ok=8    changed=2    unreachable=0    failed=0
skipped=2    rescued=0    ignored=0
```

4. On the Ubuntu remote workstation, type terraform to verify installation of terraform. Describe the output. **when typing terraform in the terminal of server 3, it shows the usage, options, and commands we can use.**

```
qcacbuduan@server3:~$ 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
```

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.

```
qcacbuduan@Workstation:~/CPE-212-Activity7/roles$ ls
base  db_servers  file_servers  web_servers  workstations
```

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.

```
qcacbuduan@Workstation:~/CPE-212-Activity7/roles/base/tasks$ cat main.yml
---
- 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"
```

base role

```
qcacbuduan@Workstation:~/CPE-212-Activity7/roles/workstations/tasks$ cat main.y
ml
---
- name: install unzip
  package:
    name: unzip

- name: install terraform
  unarchive:
    src: https://releases.hashicorp.com/terraform/0.12.28/terraform_0.12.28_lin
ux_amd64.zip
    dest: /usr/local/bin
    remote_src: yes
    mode: 0755
    owner: root
    group: root
```

workstations role

```
qcacbuduan@Workstation:~/CPE-212-Activity7/roles/web_servers/tasks$ cat main.ym
l
---
- 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 apache and php for ubuntu servers
  tags: apache,apache2,ubuntu
  apt:
    name:
      - apache2
      - libapache2-mod-php
    state: latest
    update_cache: yes
  when: ansible_distribution == "Ubuntu"

- name: install apache and php for CentOS servers
  tags: apache,apache2,ubuntu
  dnf:
```

web_servers role

```
qcacbuduan@Workstation:~/CPE-212-Activity7/roles/db_servers/tasks$ cat main.yml
- name: install mariadb package (CentOS)
  tags: centos, db,mariadb
  yum:
    name: mariadb-server
    state: latest
  when: ansible_distribution == "CentOS"

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

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

db_servers role

```
qcacbuduan@Workstation:~/CPE-212-Activity7/roles/file_servers/tasks$ cat main.y
ml
- name: install samba package
  tags: samba
  package:
    name: samba
    state: latest
```

file_servers role

4. Run the site.yml playbook and describe the output. **In the new site.yml file that was created earlier, we assigned roles to each group of the remote servers and created a task directory for each role, inside the tasks role is the main.yml playbook that executes specific tasks.**

```
PLAY RECAP *********************************************************************
*
centosbuduan               : ok=13   changed=1    unreachable=0    failed=0
skipped=4    rescued=0    ignored=0
server1                    : ok=7    changed=0    unreachable=0    failed=0
skipped=4    rescued=0    ignored=0
server2                    : ok=7    changed=0    unreachable=0    failed=0
skipped=4    rescued=0    ignored=0
server3                    : ok=10   changed=1    unreachable=0    failed=0
skipped=3    rescued=0    ignored=0
```

Github repo: https://github.com/buduman/CPE-212-Activity7.git

Reflections:

Answer the following:

1. What is the importance of creating roles?

   The importance of creating roles in managing servers is that it allows you to organize tasks, handlers, and configurations into self-contained pieces, making playbooks easier to understand and manageable. as we are automating servers, it makes it easier to maintain the servers.

2. What is the importance of managing files?

   The importance of managing files especially when controlling multiple remote servers is that it keeps our files in a structured manner. Good file management

makes it easier to manage and locate file information and it also has good security when it comes to securing essential files. A good file management also helps when it comes to backing up and recovering your files, making your data be restored quickly whenever you lose those files. It improves overall efficiency and effectiveness and reduces clutter and confusion in managing files, making it easier for the user to manage files especially when managing files in multiple servers.