

<b>Name:</b> Erebete, Jan Kenneth F.	<b>Date Performed:</b> 10/14/2025
<b>Course/Section:</b> CPE31S4 - CPE212	<b>Date Submitted:</b> 10/16/2025
<b>Instructor:</b> Engr. Robin Valenzuela	<b>Semester and SY:</b> 2025-2026 1st Sem

### **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
erebete@Workstation: ~/CPE232_Erebete/files
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>helluwo</title>
</head>
<body>
    <h1> hello po</h1>
    <p> Sir tapos na po</p>
</body>
</html>
```

[ Read 12 lines ]

^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location  
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line

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

```
- 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.

```
skipping: [192.168.56.104]
ok: [192.168.56.102]

PLAY [file_servers] ****
*****
TASK [Gathering Facts] ****
*****
ok: [192.168.56.102]

TASK [install samba packages] ****
*****
ok: [192.168.56.102]

PLAY RECAP ****
*****
192.168.56.102 : ok=10    changed=2    unreachable=0    failed=0
                  skipped=4    rescued=0    ignored=0
192.168.56.104 : ok=9     changed=2    unreachable=0    failed=0
                  skipped=3    rescued=0    ignored=0
```

- it works on running it

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.

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

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

- 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
- host: 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
```

- 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.

```
PLAY [workstations] ****
*****
TASK [Gathering Facts] ****
*****
ok: [192.168.56.101]

TASK [install unzip] ****
*****
ok: [192.168.56.101]

TASK [install terraform] ****
*****
changed: [192.168.56.101]
```

- it works on running it

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

```
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 configuratio
n
  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 ho
st
```

- terraform successfully is installed

### 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

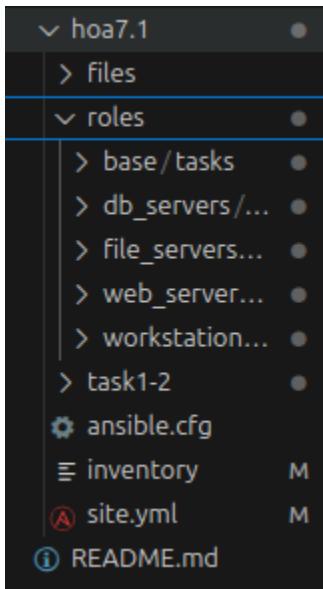
- hosts: db_servers
  become: true
  roles:
    - db_servers

- 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.



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.

```
main.yml .../base... u x
main.yml .../db_servers/... u
main.yml .../file_servers/... u
main.yml .../web_server/... u
main.yml .../workstation/... u

hoa7.1 > roles > base > tasks > main.yml

1  ---
2
3 - name: install updates (CentOS)
4   tags: always
5   dnf:
6     update_only: yes
7     update_cache: yes
8     when: ansible_distribution == "CentOS"
9
10 - name: install updates (Ubuntu)
11   tags: always
12   apt:
13     upgrade: dist
14     update_cache: yes
15     when: ansible_distribution == "Ubuntu"
```

EXPLORER

CPE23... .vscode assignment6.1 hoa4.1 hoa5.1 hoa6.1 hoa7.1 files roles base/tasks main.yml db\_servers/... main.yml file\_servers... main.yml web\_server... main.yml workstation... main.yml task1-2 ansible.cfg inventory site.yml README.md

main.yml .../base/... U main.yml .../db\_servers/... U X

hoa7.1 > roles > db\_servers > tasks > main.yml

```
1 ---  
2 - name: install mariadb package (CentOS)  
3   tags: centos,db,mariadb  
4     yum:  
5       name: mariadb-server  
6       state: latest  
7       when: ansible_distribution == "CentOS"  
8 - name: "Mariadb - Restarting/Enabling"  
9   service:  
10    name: mariadb  
11    state: restarted  
12    enabled: true  
13 - name: install mariadb package (Ubuntu)  
14   tags: db, mariadb,ubuntu  
15     apt:  
16      name: mariadb-server  
17      state: latest  
18      when: ansible_distribution == "Ubuntu"
```

The screenshot shows the Visual Studio Code interface with the following details:

**EXPLORER** pane (left):

- Root folder: CPE23...
- Subfolders: .vscode, assignment6.1, hoa4.1, hoa5.1, hoa6.1, hoa7.1 (selected)
- Files under hoa7.1:
  - files
  - roles
    - base/tasks
      - main.yml
    - db\_servers...
      - main.yml
    - file\_servers...
      - main.yml (selected)
      - web\_server...
        - main.yml
      - workstation...
        - main.yml
      - task1-2
  - ansible.cfg
  - inventory
  - site.yml
  - README.md

**EDITOR** pane (right):

File path: hoa7.1 > roles > file\_servers... > tasks > main.yml

```
1  ---
2
3 - name: install samba packages
4   tags: samba
5   package:
6     name: samba
7     state: latest
8
```

EXPLORER ... main.yml .../file\_servers/... U main.yml .../web\_servers/... U X ⌂ ⌂ ...

ho7.1 > roles > web\_servers > tasks > main.yml

```
1   ---
2
3   - name: install apache and php for Ubuntu server
4     tags: apache,apache2,ubuntu
5     apt:
6       name:
7         - apache2
8         - libapache2-mod-php
9       state: latest
10      when: ansible_distribution == "Ubuntu"
11
12  - name: install apache and php for CentOS server
13    tags: apache,centos,httpd
14    dnf:
15      name:
16        - httpd
17        - php
18      state: latest
19      when: ansible_distribution == "CentOS"
20
21  - name: start httpd (CentOS)
22    tags: apache,centos,httpd
23    service:
24      name: httpd
25      state: started
26      when: ansible_distribution == "CentOS"
27
```

The screenshot shows a VS Code interface with the following details:

- EXPLORER**: Shows a tree view of the project structure under "CPE23...".
- ANSIBLE**: Shows files: `ansible.cfg`, `main.yml`, `task1-2`, `inventory`, `site.yml`, and `README.md`.
- EDITOR**: Displays the content of `main.yml` in the "workstations/tasks" directory.

```
hoat7.1 > roles > workstations > tasks > main.yml
1 ---  
2  
3 - name: install unzip  
4   tags: workstation  
5     package:  
6       name: unzip  
7       state: present  
8  
9 - name: install terraform  
10  tags: workstation  
11    unarchive:  
12      src: https://releases.hashicorp.com/terrafa  
13      dest: /usr/local/bin  
14      remote_src: yes  
15      mode: '0755'  
16      owner: root  
17      group: root  
18
```

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

```

1$ ansible-playbook site.yml -K
skipping: [192.168.56.102]
ok: [192.168.56.104]

TASK [db_servers : Mariadb - Restarting/Enabling] ****
*****
changed: [192.168.56.102]
changed: [192.168.56.104]

TASK [db_servers : install mariadb package (Ubuntu)] *
*****
skipping: [192.168.56.104]
ok: [192.168.56.102]

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

TASK [Gathering Facts] ****
*****
ok: [192.168.56.102]

TASK [file_servers : install samba packages] ****
*****
ok: [192.168.56.102]

PLAY RECAP ****
*****
192.168.56.101      : ok=7    changed=0    unre
achable=0  failed=0  skipped=2    rescued=0    ign
ored=0
192.168.56.102      : ok=11   changed=1    unre
achable=0  failed=0  skipped=5    rescued=0    ign
ored=0
192.168.56.104      : ok=10   changed=1    unre
achable=0  failed=0  skipped=4    rescued=0    ign
ored=0

```

- Omitted the rest, but as shown, the roles helped consolidate plays to only 'hit' the aforementioned IPs stated in the inventory file.

#### Reflections:

Answer the following:

- 1. What is the importance of creating roles?**

- Defining roles in Ansible is essential for building clean, scalable automation workflows. Rather than cramming all tasks into a single bulky playbook, roles break things down into organized components like tasks, handlers, templates, variables, and files each housed in its own directory. This structure improves readability, simplifies maintenance, and makes it easier to expand your automation over time. Roles also encourage reuse: you can apply the same role across different hosts or projects without duplicating code. Ultimately, roles introduce modularity, streamline development, and boost efficiency in managing infrastructure.

## **2. What is the importance of managing files?**

- Managing files in Ansible is key to keeping systems consistent and secure. It makes sure the right versions of config files, scripts, and binaries are deployed across all servers. This avoids mismatches, sets correct permissions, and saves time by automating file setup. It also helps keep all systems aligned with your intended setup.