Name:Tracey Dee Bringuela	**Date Performed:9/10/24**
Course/Section:CPE31S2	**Date Submitted:9/10/24**
Instructor: Robin 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**:

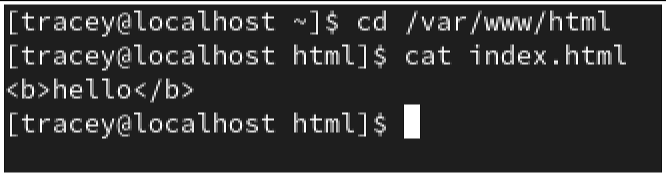
In this activity, we look at the concept of copying a file to a server. We are going to create a file into our git repository and use Ansible to grab that file and put it into a particular place so that we could do things like customize a default website, or maybe install a default configuration file. 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.
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:

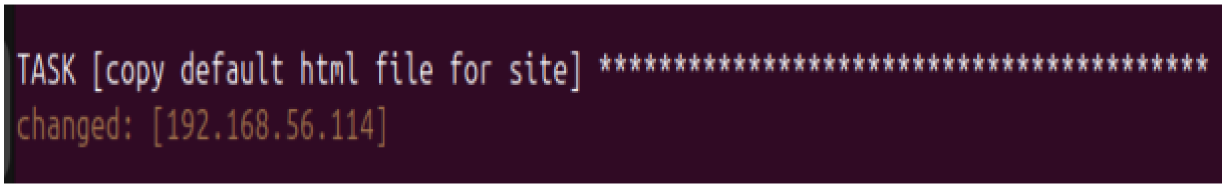
```
- name: copy default html file for site

  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.
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.

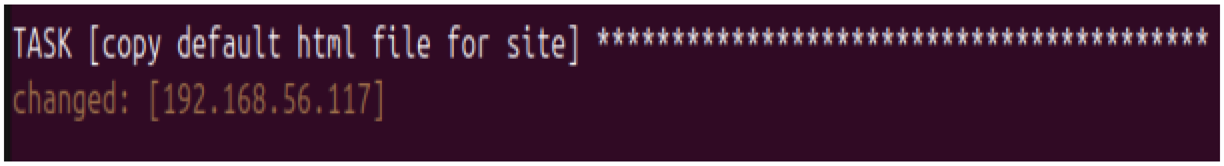


```
[tracey@localhost ~]$ cd /var/www/html
[tracey@localhost html]$ cat index.html
<b>hello</b>
[tracey@localhost html]$
```

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



```
TASK [copy default html file for site] *******************************************
changed: [192.168.56.114]
```



```
TASK [copy default html file for site] *******************************************
changed: [192.168.56.117]
```

it has copied the default html file

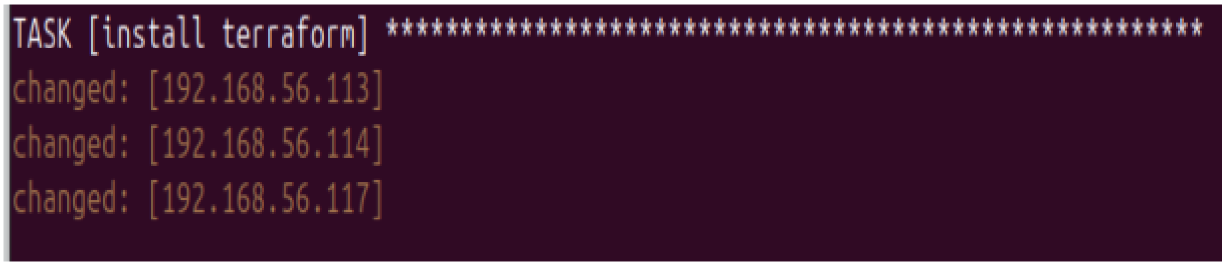
Task 2: Download a file and extract it to a remote server

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

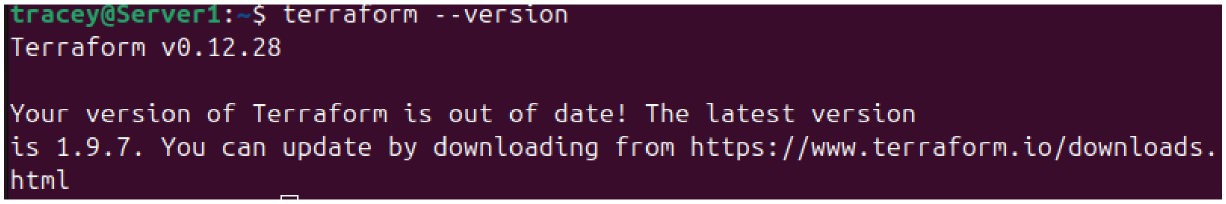
```
- hosts: workstations
  become: true
  tasks:

  - 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.

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



```
TASK [install terraform] ******************************************************
changed: [192.168.56.113]
changed: [192.168.56.114]
changed: [192.168.56.117]
```

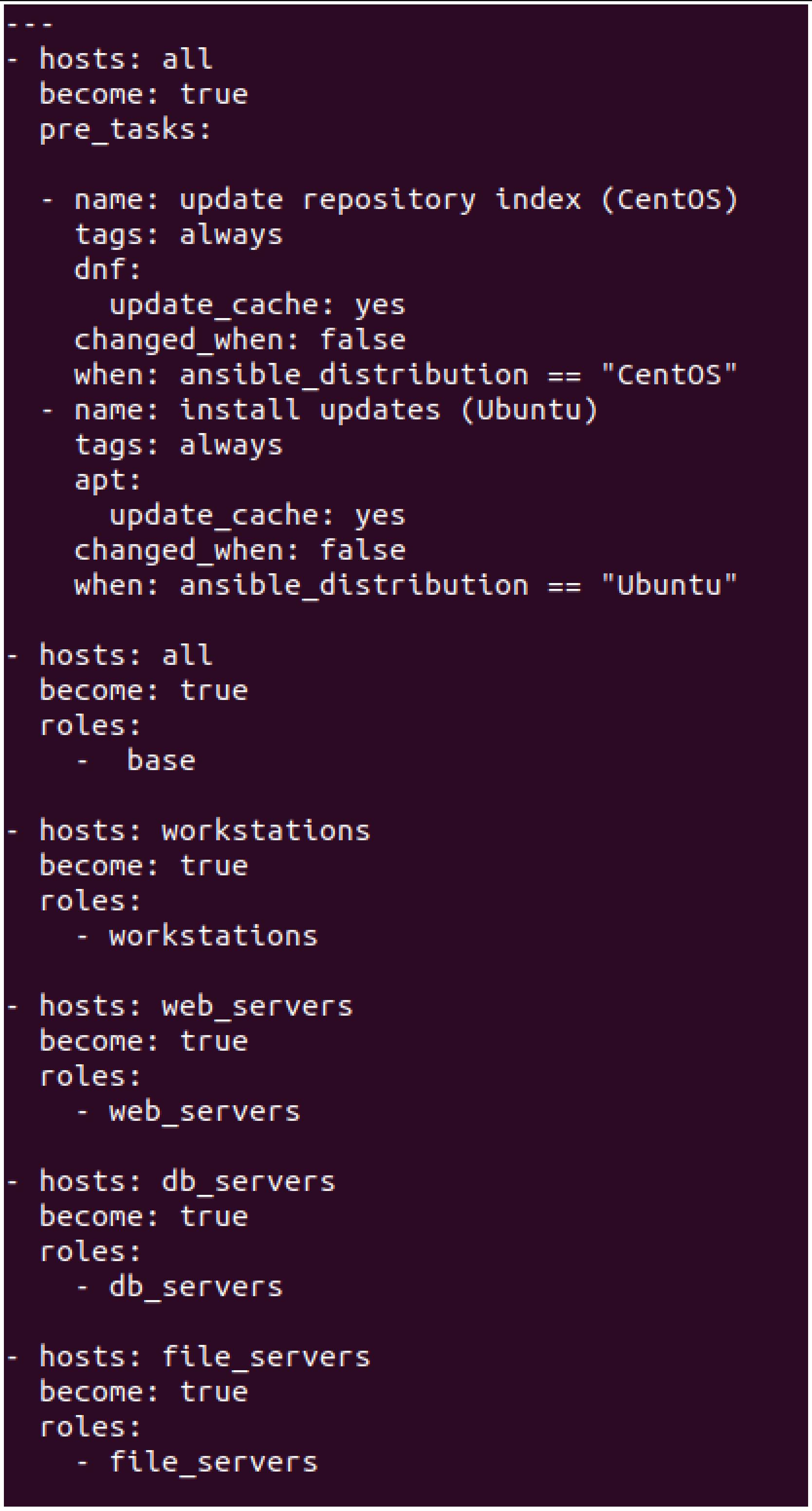


```
tracey@Server1:~$ terraform --version
Terraform v0.12.28

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
```

this shows that the terraform is now installed in my ubuntu servers

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)

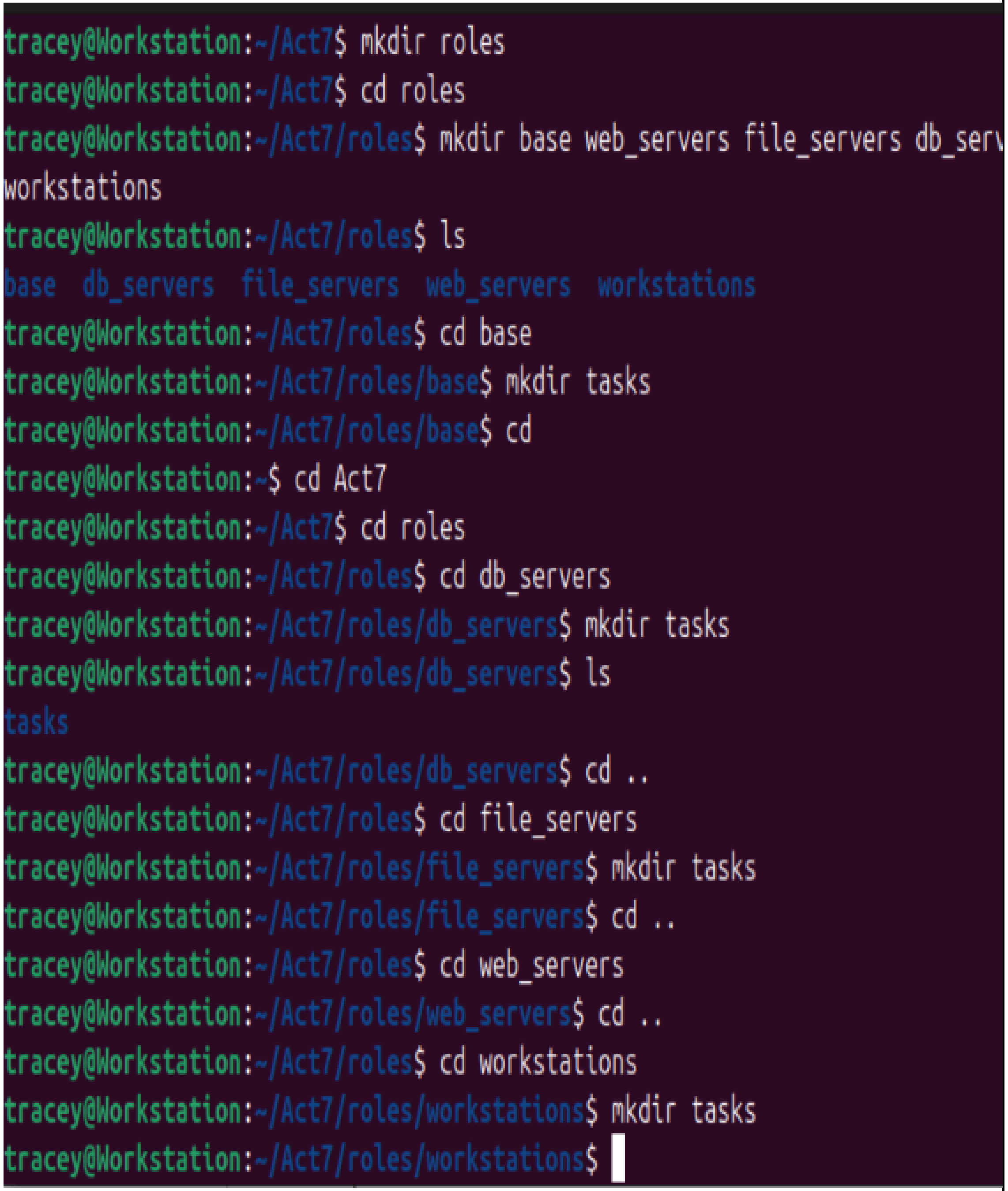


```
---
- hosts: all
  become: true
  pre_tasks:

  - 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,

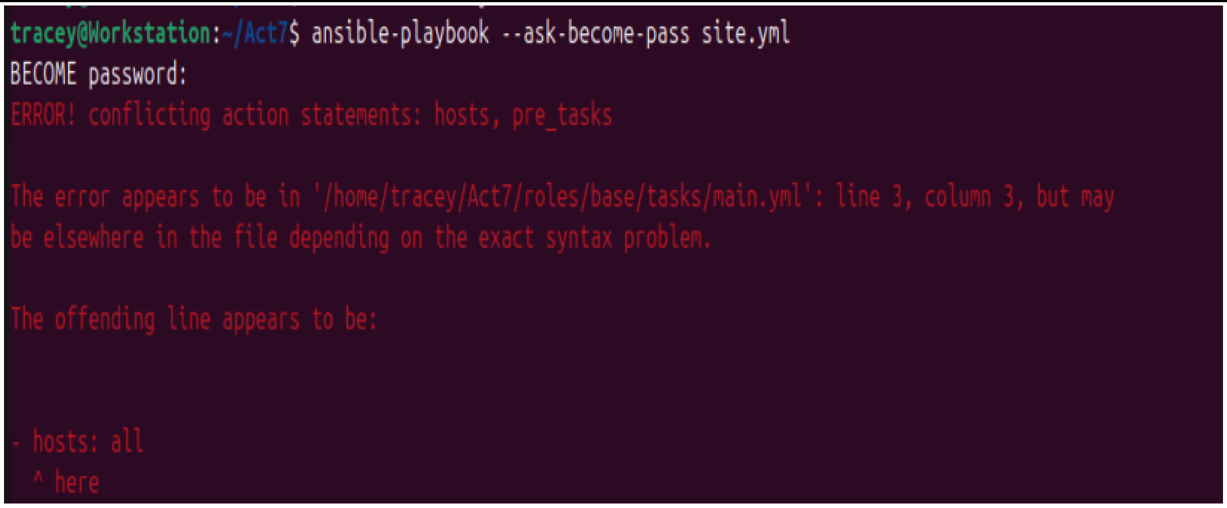
db_servers and workstations. For each directory, create a directory and name it tasks.



```
tracey@Workstation:~/Act7$ mkdir roles
tracey@Workstation:~/Act7$ cd roles
tracey@Workstation:~/Act7/roles$ mkdir base web_servers file_servers db_serv
workstations
tracey@Workstation:~/Act7/roles$ ls
base  db_servers  file_servers  web_servers  workstations
tracey@Workstation:~/Act7/roles$ cd base
tracey@Workstation:~/Act7/roles/base$ mkdir tasks
tracey@Workstation:~/Act7/roles/base$ cd
tracey@Workstation:~$ cd Act7
tracey@Workstation:~/Act7$ cd roles
tracey@Workstation:~/Act7/roles$ cd db_servers
tracey@Workstation:~/Act7/roles/db_servers$ mkdir tasks
tracey@Workstation:~/Act7/roles/db_servers$ ls
tasks
tracey@Workstation:~/Act7/roles/db_servers$ cd ..
tracey@Workstation:~/Act7/roles$ cd file_servers
tracey@Workstation:~/Act7/roles/file_servers$ mkdir tasks
tracey@Workstation:~/Act7/roles/file_servers$ cd ..
tracey@Workstation:~/Act7/roles$ cd web_servers
tracey@Workstation:~/Act7/roles/web_servers$ cd ..
tracey@Workstation:~/Act7/roles$ cd workstations
tracey@Workstation:~/Act7/roles/workstations$ mkdir tasks
tracey@Workstation:~/Act7/roles/workstations$ 
```

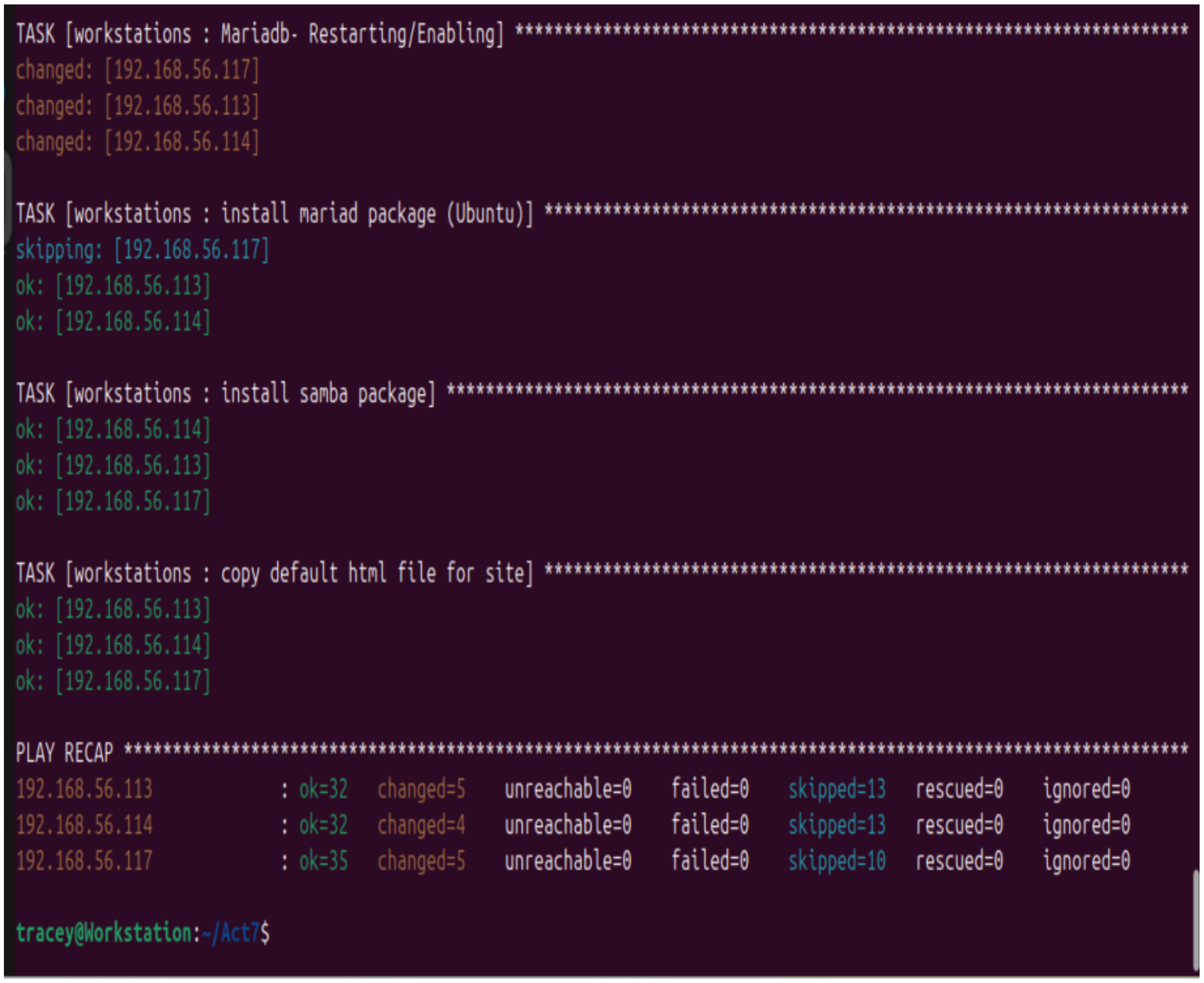
3. Go to tasks for all directory and create a file. Name it main.yml. In each of the tasks for all directories, copy and paste the code from the old site.yml file. Show all contents of main.yml files for all tasks.

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



```
tracey@Workstation:~/Act7$ ansible-playbook --ask-become-pass site.yml
BECOME password:
ERROR! conflicting action statements: hosts, pre_tasks

The error appears to be in '/home/tracey/Act7/roles/base/tasks/main.yml': line 3, column 3, but may
be elsewhere in the file depending on the exact syntax problem.

The offending line appears to be:


- hosts: all
  ^ here
```

this is the result when you did not delete the - hosts syntax in the main.yml



```
TASK [workstations : Mariadb- Restarting/Enabling] ****************************************************************
changed: [192.168.56.117]
changed: [192.168.56.113]
changed: [192.168.56.114]

TASK [workstations : install mariad package (Ubuntu)] *************************************************************
skipping: [192.168.56.117]
ok: [192.168.56.113]
ok: [192.168.56.114]

TASK [workstations : install samba package] ***********************************************************************
ok: [192.168.56.114]
ok: [192.168.56.113]
ok: [192.168.56.117]

TASK [workstations : copy default html file for site] *************************************************************
ok: [192.168.56.113]
ok: [192.168.56.114]
ok: [192.168.56.117]

PLAY RECAP ********************************************************************************************************
192.168.56.113             : ok=32   changed=5    unreachable=0    failed=0    skipped=13   rescued=0    ignored=0
192.168.56.114             : ok=32   changed=4    unreachable=0    failed=0    skipped=13   rescued=0    ignored=0
192.168.56.117             : ok=35   changed=5    unreachable=0    failed=0    skipped=10   rescued=0    ignored=0

tracey@Workstation:~/Act7$
```

this is the output when you delete the - hosts syntax in all the main.yml

Reflections:

Answer the following:

1. What is the importance of creating roles?

   Creating roles in Ansible enhances modularity and reusability, allowing for the encapsulation of specific functionalities in a structured way. This organization simplifies maintenance and promotes collaboration among team members by enabling parallel development. Ultimately, roles lead to more consistent and predictable configurations, adhering to best practices in automation.

2. What is the importance of managing files?

   Effective file management ensures data integrity and security by organizing files systematically, which minimizes the risk of loss or corruption. It facilitates efficient version control, allowing teams to track changes and collaborate more effectively. Additionally, proper management supports compliance with regulations and enhances recovery processes in case of data loss.