Name: Castillo, Joshua L.	Date Performed: 3/21/24
Course/Section: CPE31S1	Date Submitted: 3/24/2024
Instructor: Dr. Jonathan Tylar	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

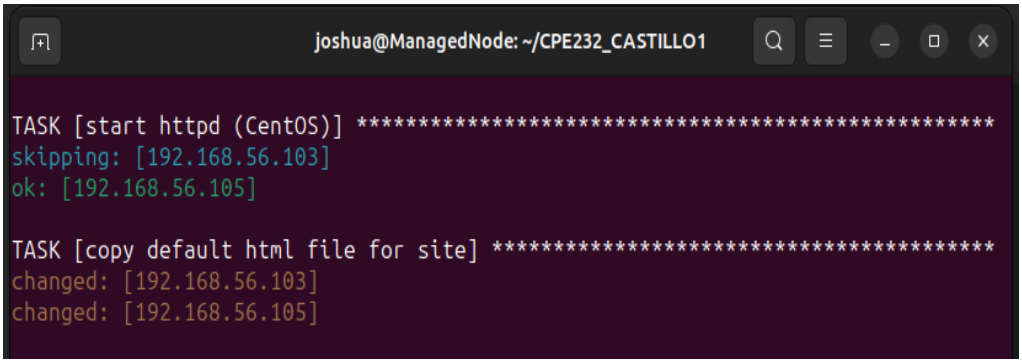
2. **Discussion**:

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.

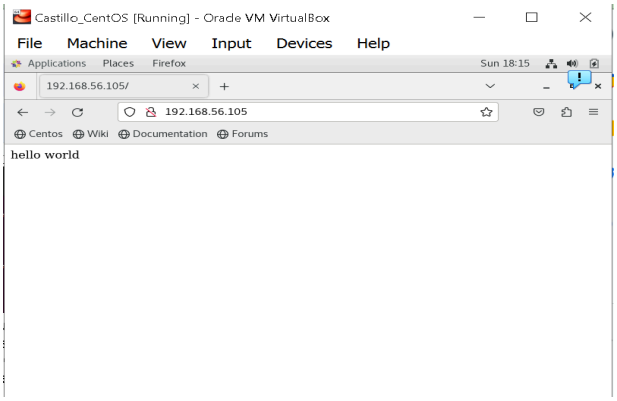
Task 1: Create a file and copy it to remote servers

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

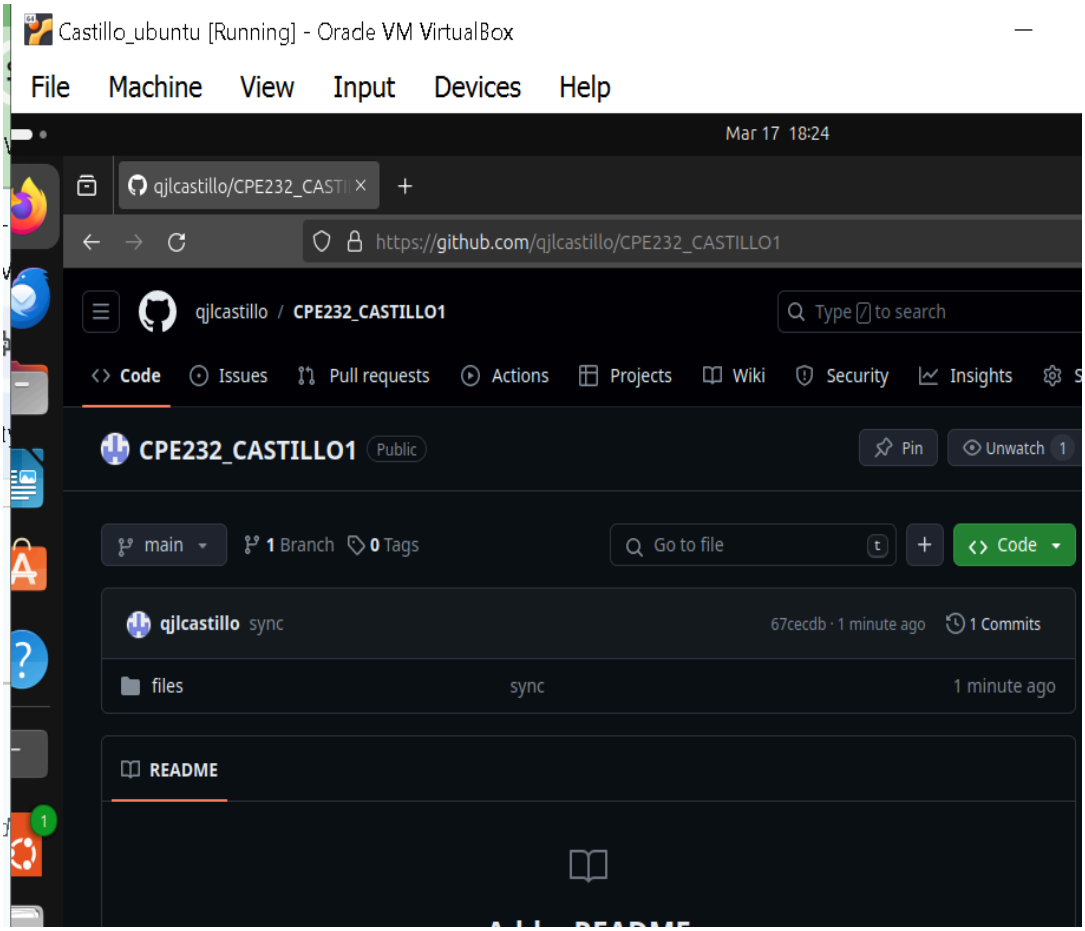
     tags: apache, apache2, httpd
     copy:
       src: default_site.html
       dest: /var/www/html/index.html
       owner: root
       group: root
       mode: 0644
3. Run the playbook *site.yml*. Describe the changes.



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.



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:

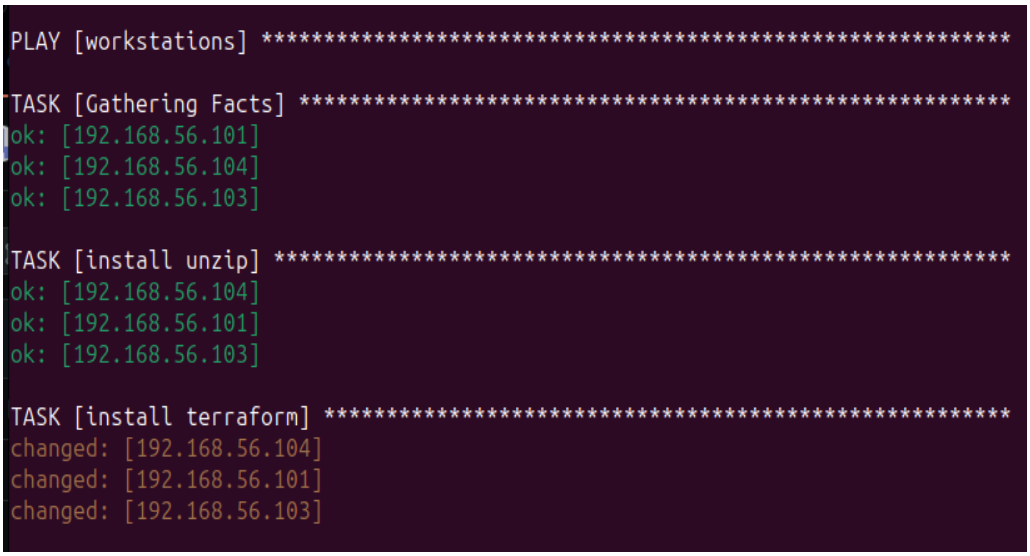
```
- hosts: workstations
  become: true
  tasks:

  - 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

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.

```
joshua@ControlNode1:~$ 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
    destroy            Destroy Terraform-managed infrastructure
    env                Workspace management
    fmt                Rewrites config files to canonical format
    get                Download and install modules for the configuration
    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 host
    output             Read an output from a state file
    plan               Generate and show an execution plan
    providers          Prints a tree of the providers used in the configuration
```

```
joshua@ControlNode2:~$ 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
    destroy            Destroy Terraform-managed infrastructure
    env                Workspace management
    fmt                Rewrites config files to canonical format
    get                Download and install modules for the configuration
    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 host
    output             Read an output from a state file
    plan               Generate and show an execution plan
    providers          Prints a tree of the providers used in the configuration
```

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.

```
joshua@ManagedNode:~/CPE232_CASTILLO1$ cd roles
joshua@ManagedNode:~/CPE232_CASTILLO1/roles$ tree
.
├── base
│   └── tasks
│       └── main.yml
├── db_servers
│   └── tasks
│       └── main.yml
├── file_servers
│   └── tasks
│       └── main.yml
├── web_servers
│   └── tasks
│       └── main.yml
└── workstations
    └── tasks
        └── main.yml

11 directories, 5 files
joshua@ManagedNode:~/CPE232_CASTILLO1/roles$
```

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 main.yml

Open

main.yml
~/CPE232_CASTILLO1/roles/base/tasks

```
---
  - 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"
```

workstations Main.yml

Open

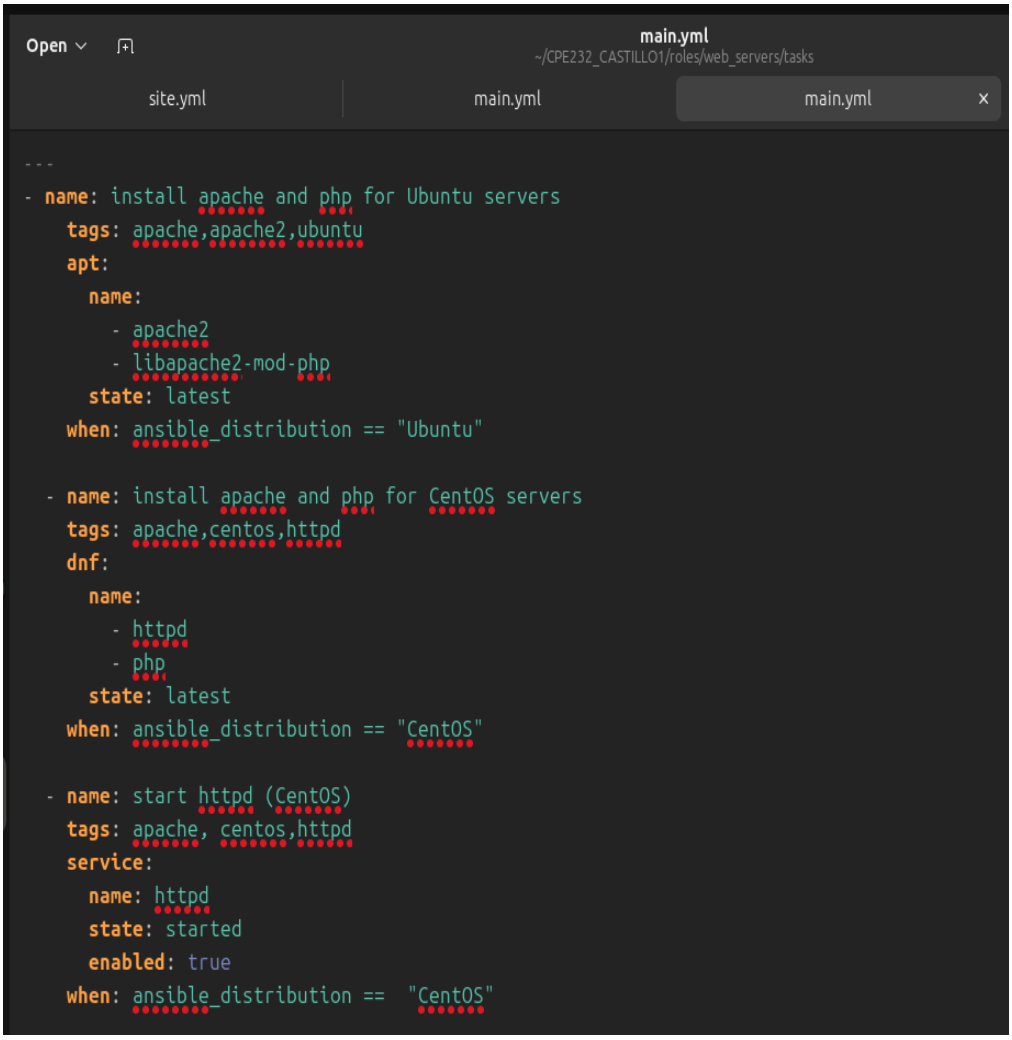
main.yml
~/CPE232_CASTILLO1/roles/workstations/tasks

site.yml | main.yml × | main.yml

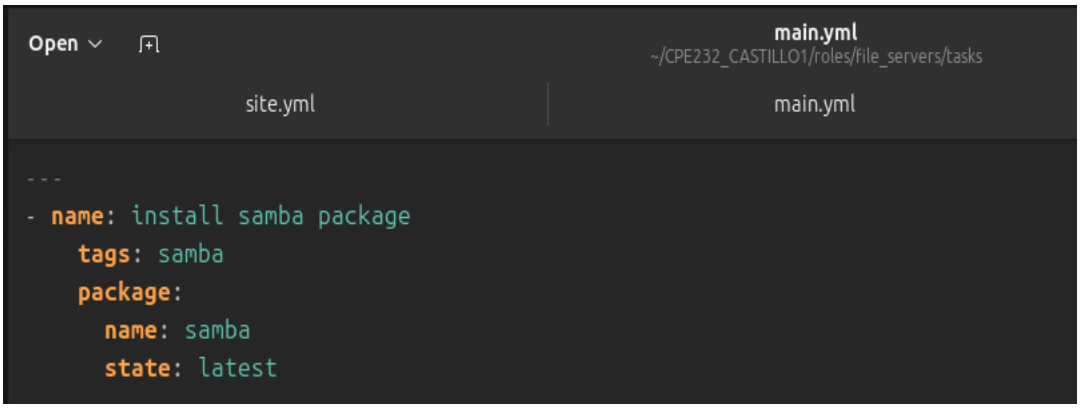
```
---
- name: install unzip
     package:
       name: unzip

- 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
```

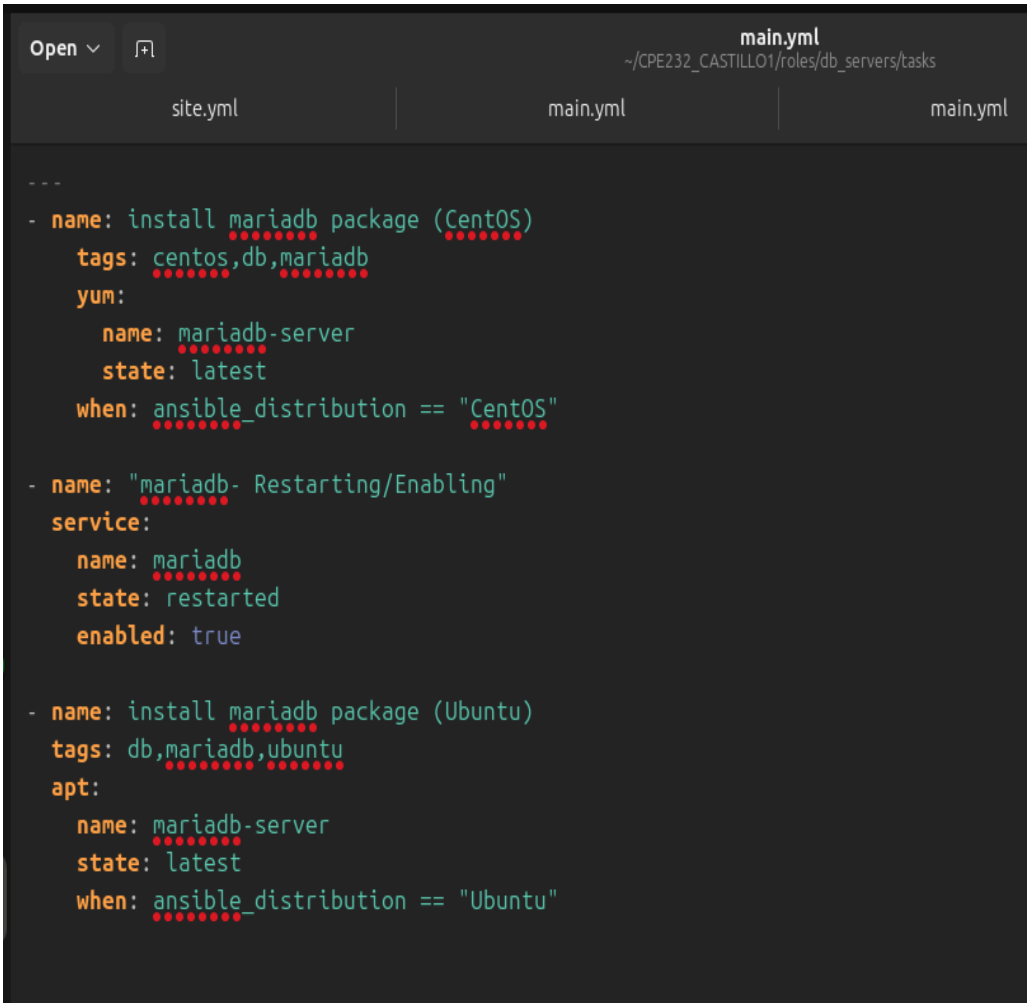
web_servers Main.yml



file_servers Main.yml



db_servers Main.yml



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

```
TASK [update repository index (CentOS)] ************************************
skipping: [192.168.56.103]
skipping: [192.168.56.101]
ok: [192.168.56.105]

TASK [install updates (Ubuntu)] ********************************************
skipping: [192.168.56.105]
ok: [192.168.56.103]
ok: [192.168.56.101]

PLAY [all] *****************************************************************

TASK [Gathering Facts] *****************************************************
ok: [192.168.56.101]
ok: [192.168.56.103]
ok: [192.168.56.105]

TASK [base : install updates  (CentOS)] ************************************
skipping: [192.168.56.103]
skipping: [192.168.56.101]
ok: [192.168.56.105]

TASK [base : install updates (Ubuntu)] *************************************
skipping: [192.168.56.105]
ok: [192.168.56.101]
ok: [192.168.56.103]
```

```
PLAY [workstations] ********************************************************************************

TASK [Gathering Facts] *****************************************************************************
ok: [192.168.56.103]
ok: [192.168.56.101]

TASK [workstations : install unzip] ****************************************************************
ok: [192.168.56.101]
ok: [192.168.56.103]

TASK [workstations : install terraform] ************************************************************
ok: [192.168.56.103]
ok: [192.168.56.101]

PLAY [web_servers] *********************************************************************************

TASK [Gathering Facts] *****************************************************************************
ok: [192.168.56.103]
ok: [192.168.56.105]

TASK [web_servers : copy default html file for site] ***********************************************
ok: [192.168.56.103]
ok: [192.168.56.105]

TASK [web_servers : install apache and php for Ubuntu servers] *************************************
skipping: [192.168.56.105]
ok: [192.168.56.103]

TASK [web_servers : install apache and php for CentOS servers] *************************************
skipping: [192.168.56.103]
ok: [192.168.56.105]
```

```
TASK [db_servers : install mariadb package (CentOS)] ***************************************************
skipping: [192.168.56.101]

TASK [db_servers : mariadb- Restarting/Enabling] *******************************************************
changed: [192.168.56.101]

TASK [db_servers : install mariadb package (Ubuntu)] ***************************************************
ok: [192.168.56.101]

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

PLAY RECAP *********************************************************************************************
192.168.56.101             : ok=10   changed=1    unreachable=0    failed=0    skipped=3    rescued=0    ig
192.168.56.103             : ok=10   changed=0    unreachable=0    failed=0    skipped=4    rescued=0    ig
192.168.56.104             : ok=0    changed=0    unreachable=1    failed=0    skipped=0    rescued=0    ig
192.168.56.105             : ok=8    changed=0    unreachable=0    failed=0    skipped=3    rescued=0    ig

joshua@ManagedNode:~/CPE232_CASTILLO1$
```

file_servers main.yml output

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

TASK [Gathering Facts] *********************************************************************************
ok: [192.168.56.104]

TASK [file_servers : install samba package] ************************************************************
ok: [192.168.56.104]

PLAY RECAP *********************************************************************************************
192.168.56.101             : ok=0    changed=0    unreachable=1    failed=0    skipped=0    rescued=0    ig
192.168.56.103             : ok=10   changed=0    unreachable=0    failed=0    skipped=4    rescued=0    ig
192.168.56.104             : ok=9    changed=0    unreachable=0    failed=0    skipped=2    rescued=0    ig
192.168.56.105             : ok=8    changed=0    unreachable=0    failed=0    skipped=3    rescued=0    ig

joshua@ManagedNode:~/CPE232_CASTILLO1$
```

In this output we break down the tasks into their respective components.

After the tasks, I added the changes in my github repository.

note: i only used 3 vm's at a time because of the slow responsiveness of the device, i switch back to 1 vm(db_servers) once i'm done configuring its components.

Reflections:

Answer the following:

1. What is the importance of creating roles?
   Creating roles in Ansible organizes playbook management by distributing the tasks into their respective roles making it a reusable component, it also allows its user to reuse a playbook for other tasks which saves time.

2. What is the importance of managing files?

   file management is important for accurate configurations and security against unauthorized access, lastly it optimizes system efficiency.