Name: John Renzo L. Mendoza	**Date Performed:** October 17, 2023
Course/Section: CPE31S5	**Date Submitted:** October 18, 2023
Instructor: Engr. Roman Richard	**Semester and SY:** 1st Semester 2023-2024

Activity 7: Managing Files and Creating Roles in Ansible

1. **Objectives:**

1.1 Manage files in remote servers

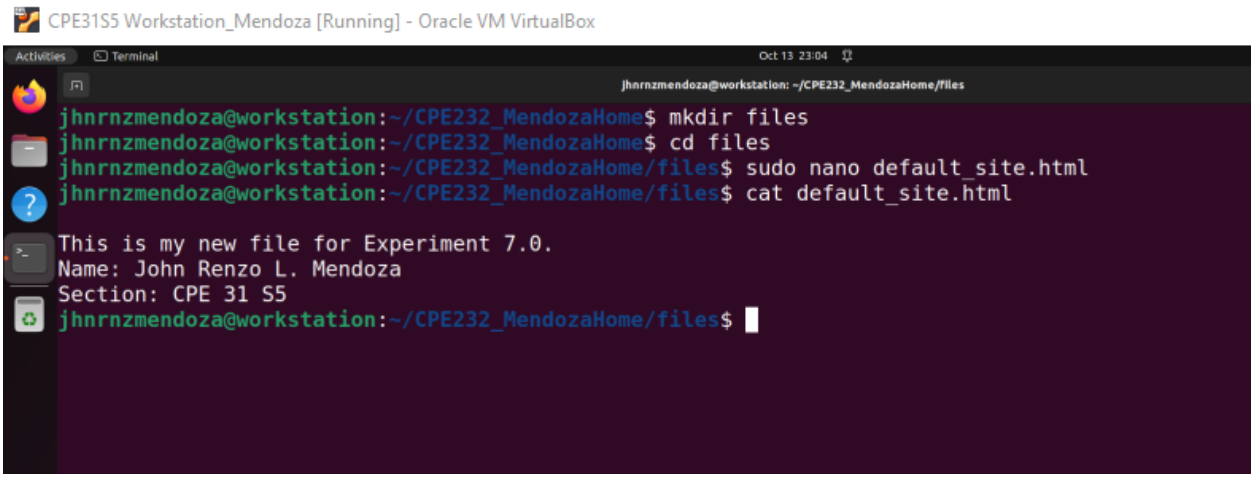
1.2 Implement roles in ansible

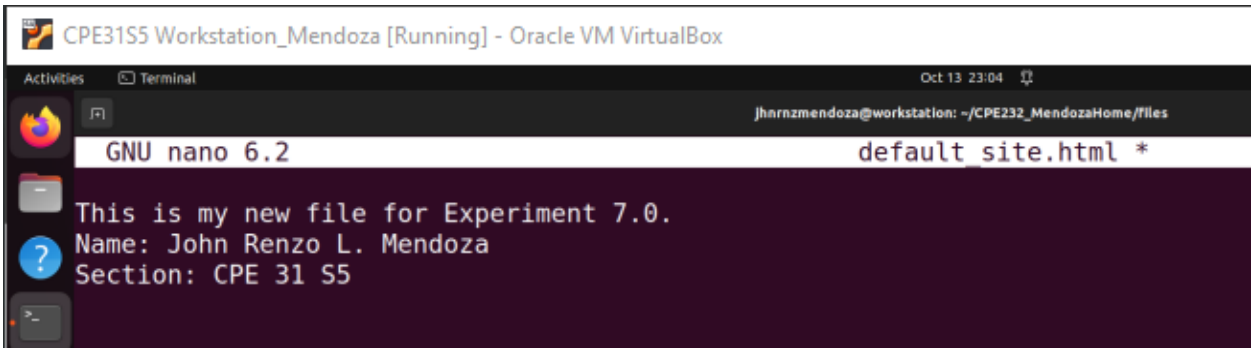
2. **Discussion**:

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

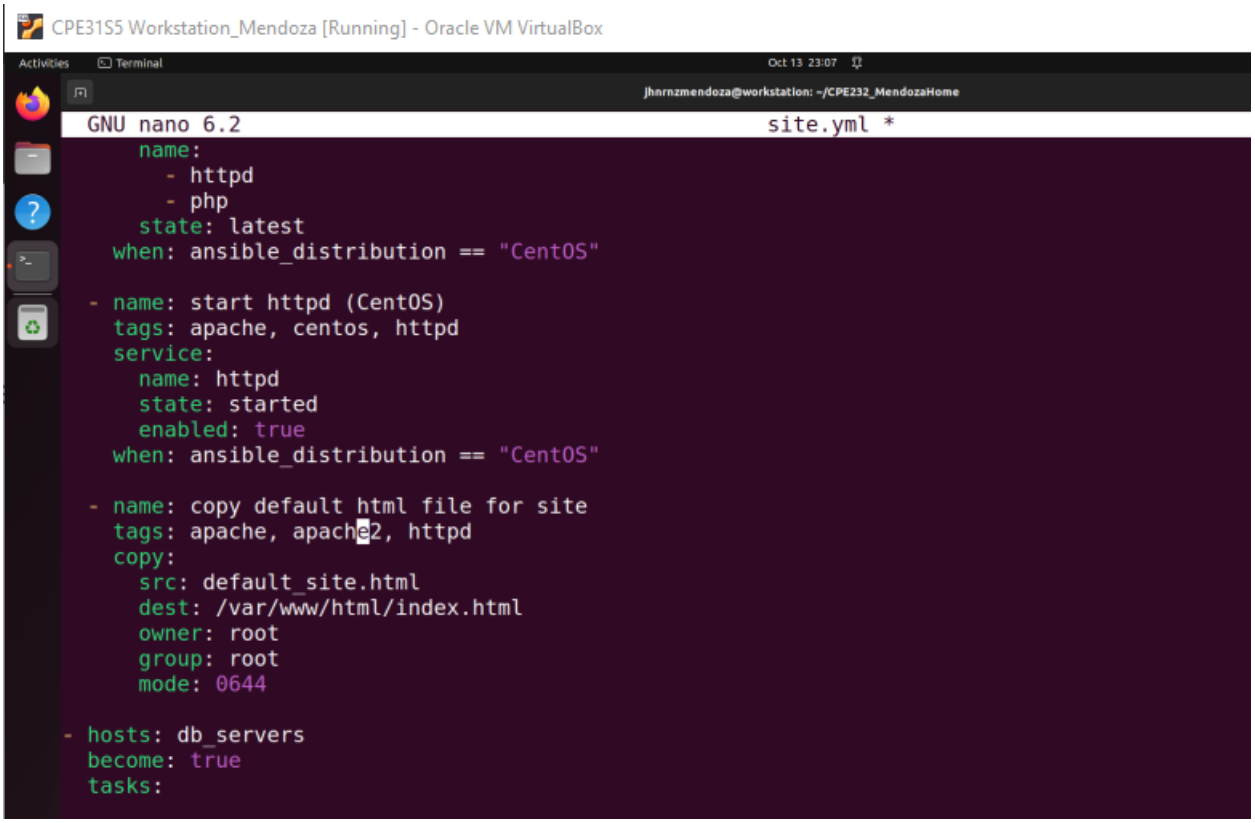
1. Using the previous directory we created, created a directory, and named it "*files*." Create a file inside that directory and name it "*default_site.html*." Edit the file and put basic HTML syntax. Any content will do, as long as it will display text later. Save the file and exit.





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

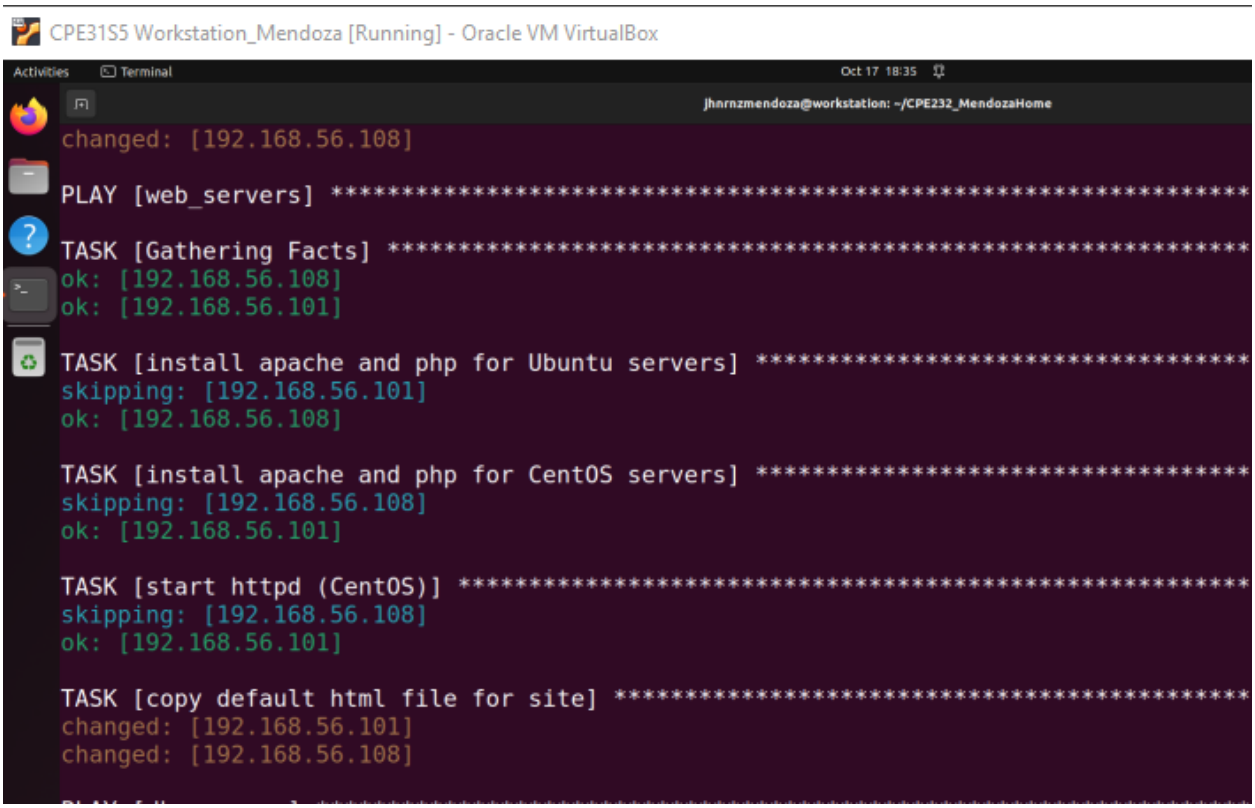
     tags: apache, apache2, httpd
     copy:
       src: default_site.html
       dest: /var/www/html/index.html
       owner: root
       group: root
       mode: 0644



Observation:

- The given new task was added under the web_servers play. This the task would basically copy the file defined in src, and send it to the absolute path defined by the dst on the remote servers.

3. Run the playbook *site.yml*. Describe the changes.

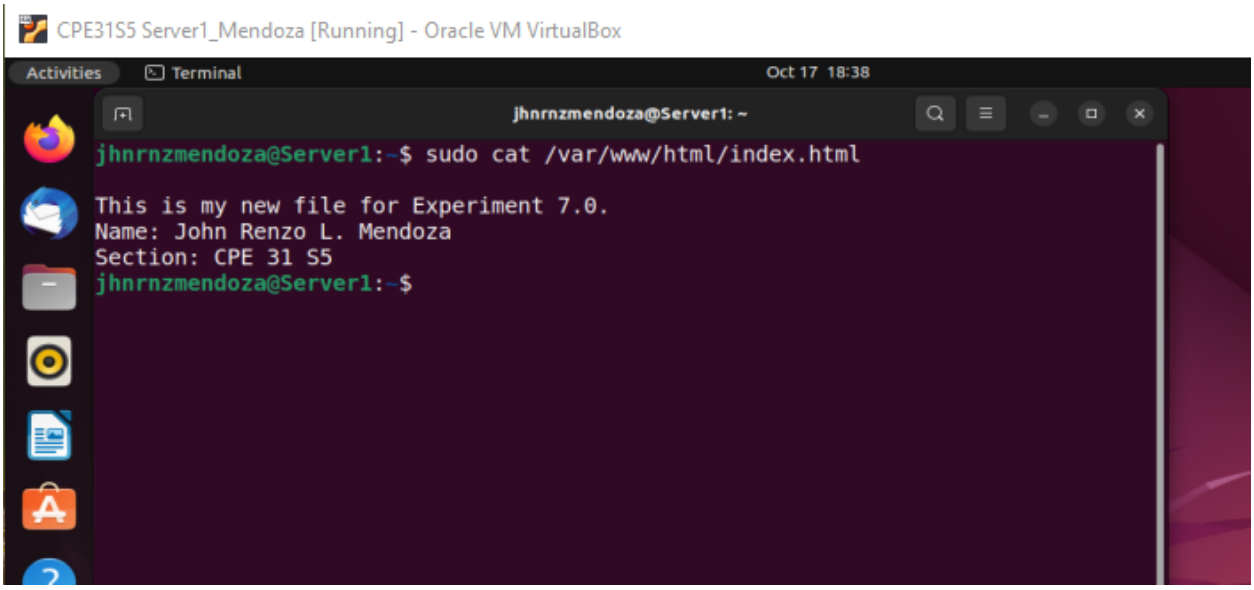


Observation:

- On this play, we can observe that the new task was successfully performed on the target remote nodes which are the ubuntu server (Server 1) and centos server. we can assume that the file index.html has been created on the remote servers specifically on the dst absolute path.

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.

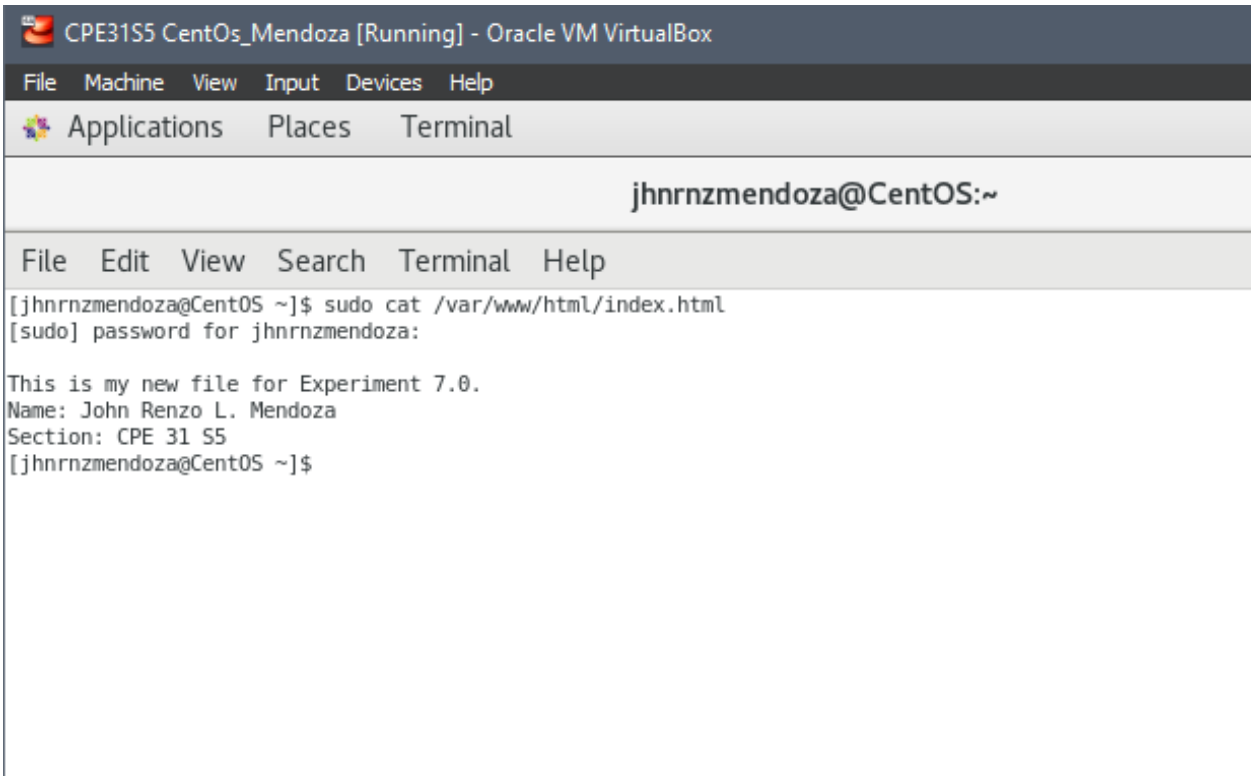
   Server 1: Web Server



   Observation:

   - In server 1, I have concatenated the contents of the file using the cat command and the absolute path defined earlier. As observed we can see the contents of the file that we have created earlier.

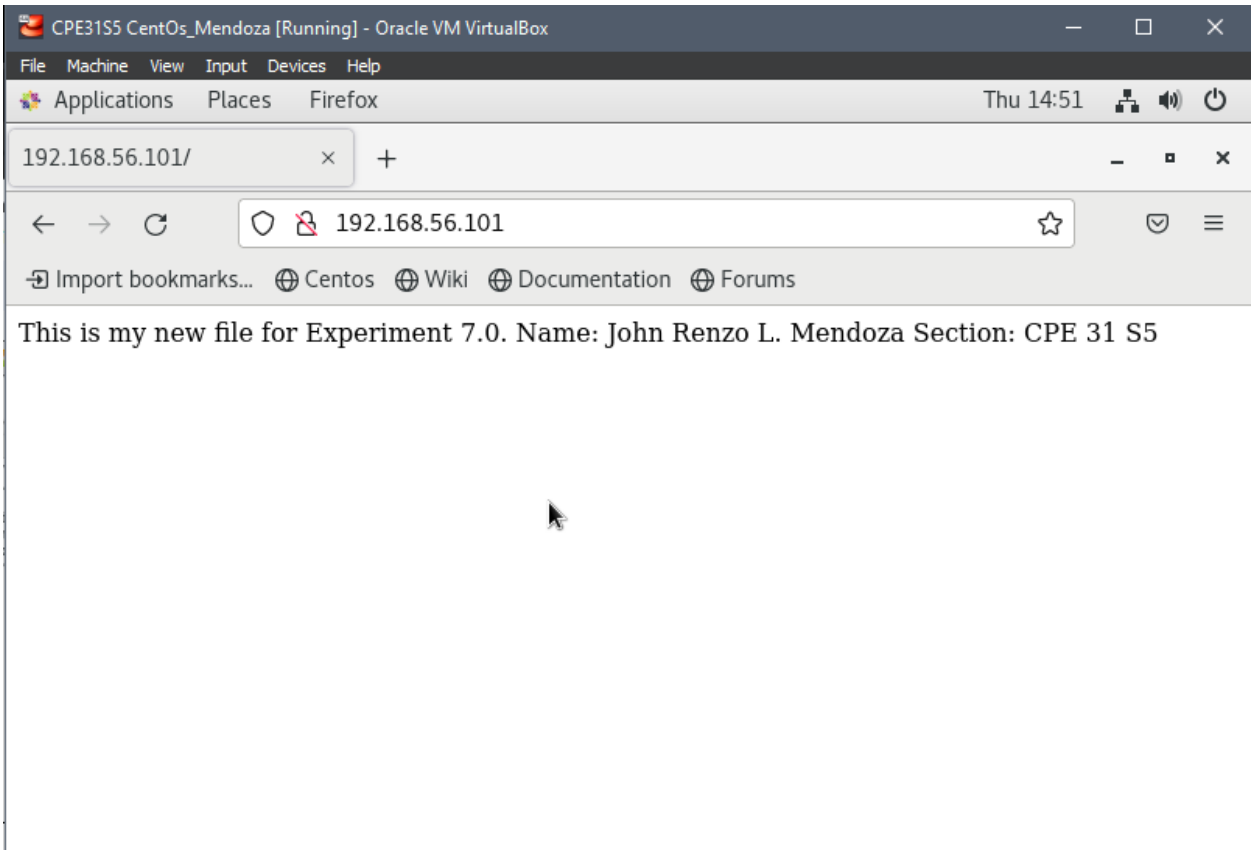
CentOS: Web_Server



Observation:

- In server 1, I have concatenated the contents of the file using the cat command and the absolute path defined earlier. As observed we can see the contents of the file that we have created earlier.

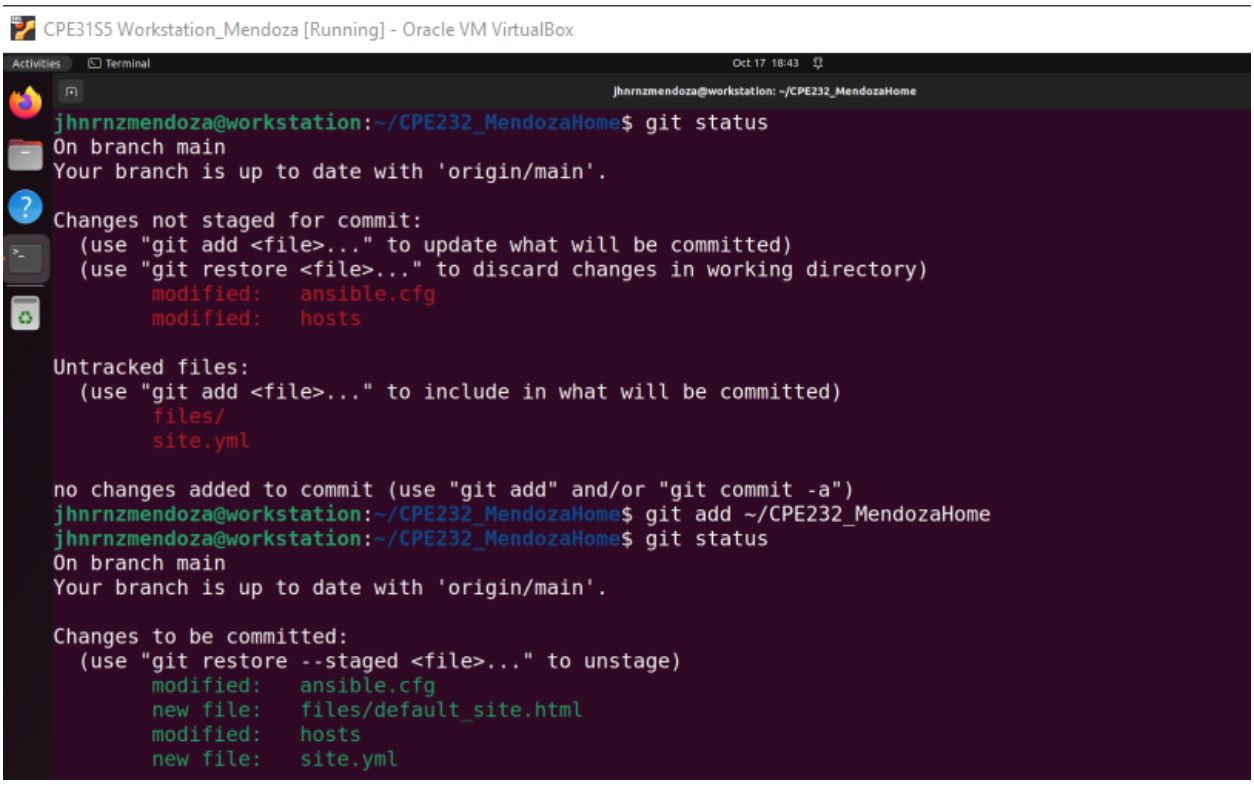
Checking of IP Address

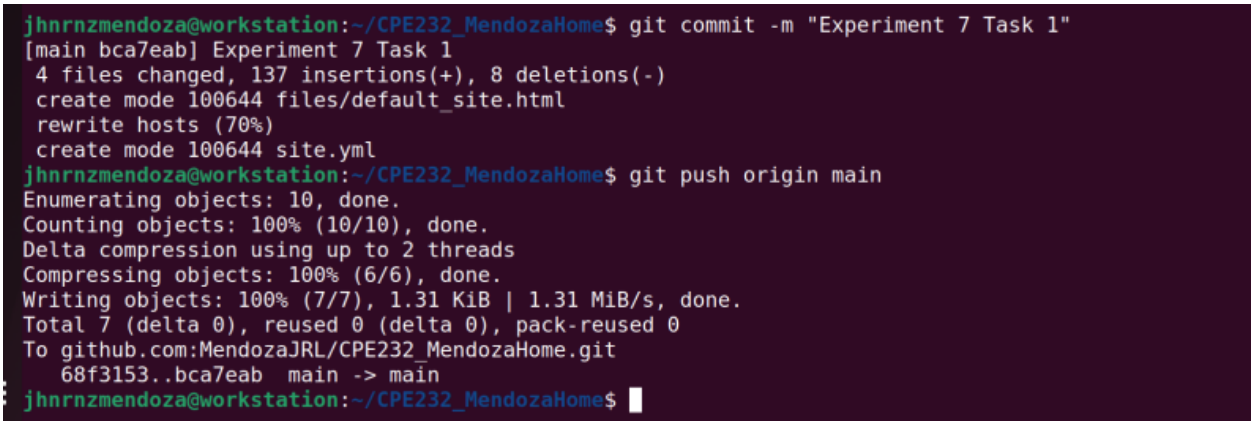


Observation:

- The contents of the file was also seen on the browser by entering the IP address of the db server. I believe this was possible since the absolute path defined is for the internet browsers /var/www/html.

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

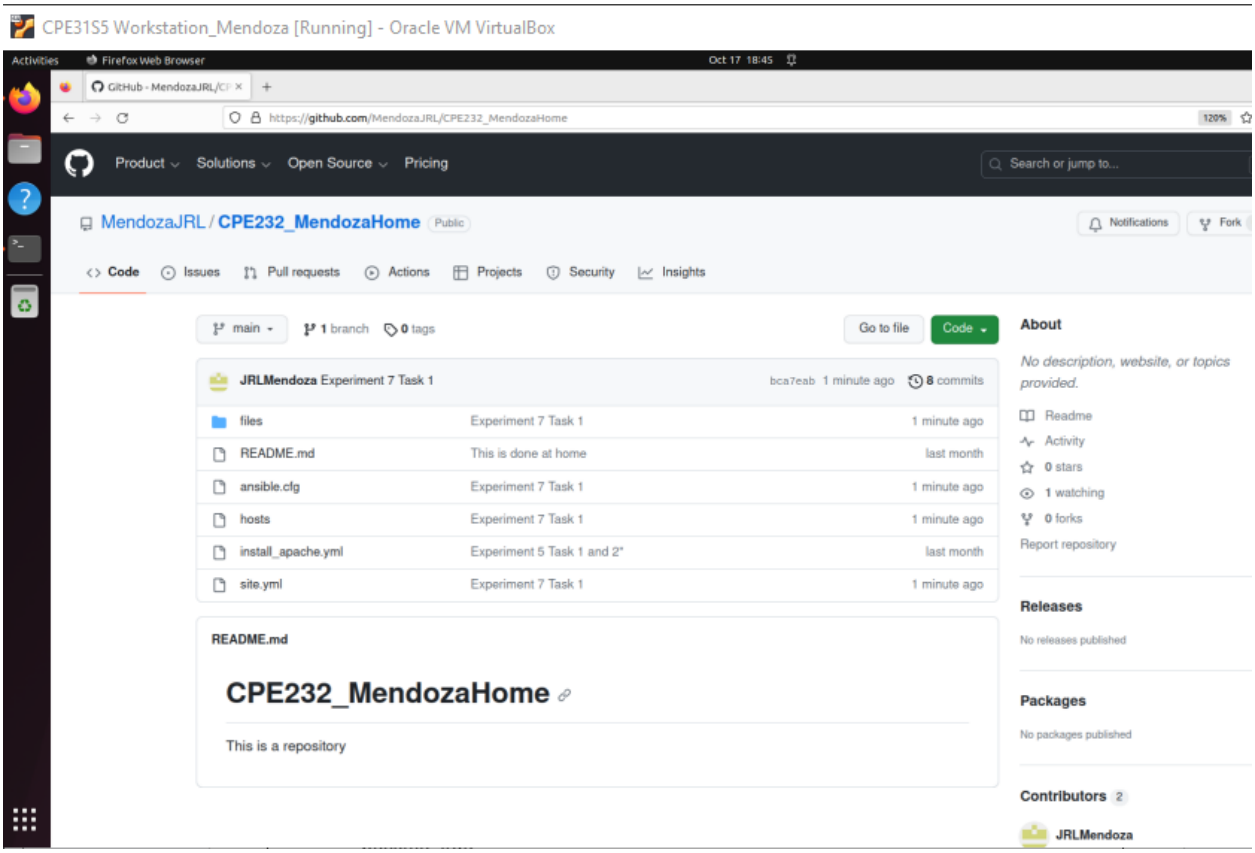




Observation:

- Using the previously learned git commands, I have successfully committed and pushed the changes to the GitHub cloud repository.

GitHub Repository



Observation:

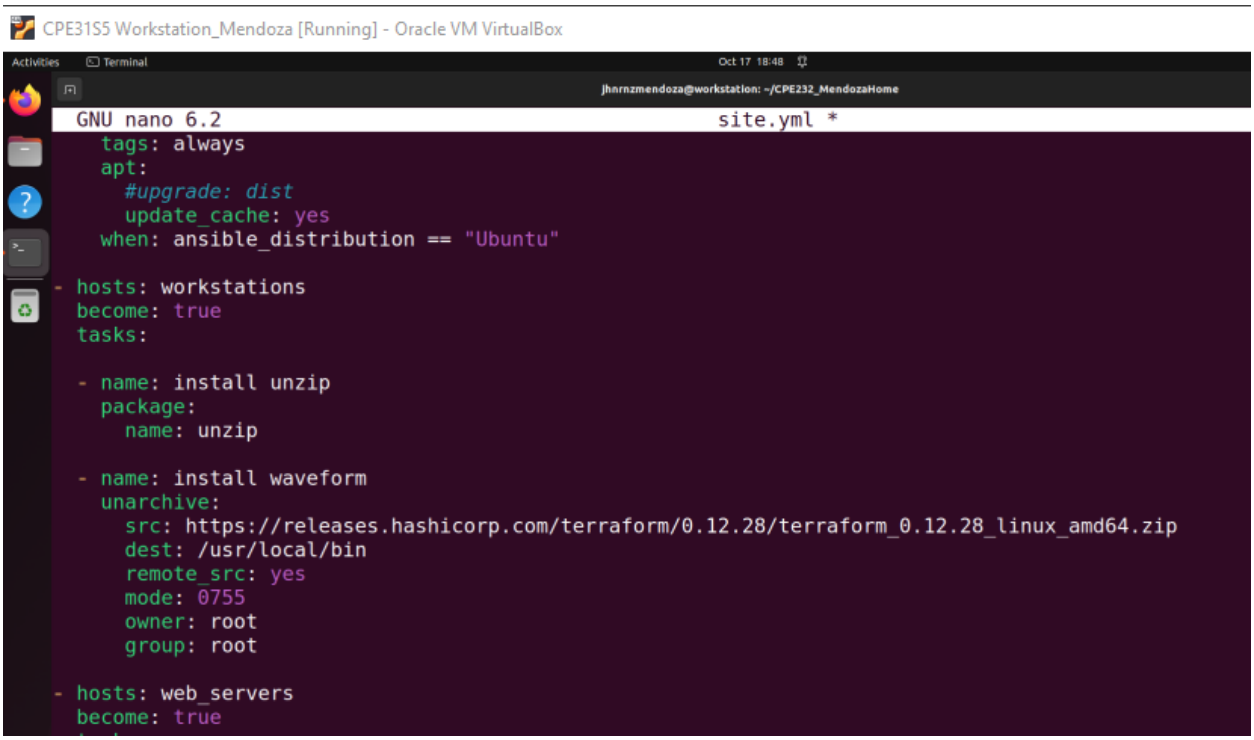
- In the GitHub repository, we can observe that the new files site.yml is added and the other related files are also updated.

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

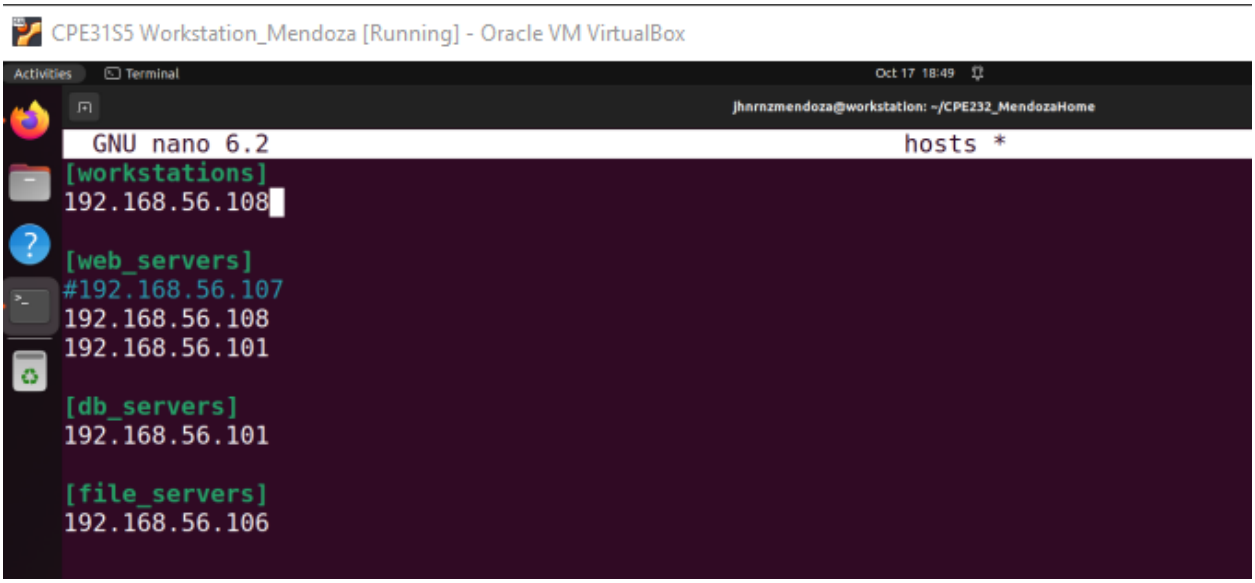
1. Edit the site.yml. Just before the web_servers play, create a new play:
   - hosts: workstations
     become: true
     tasks:

     - name: install unzip
       package:
         name: unzip

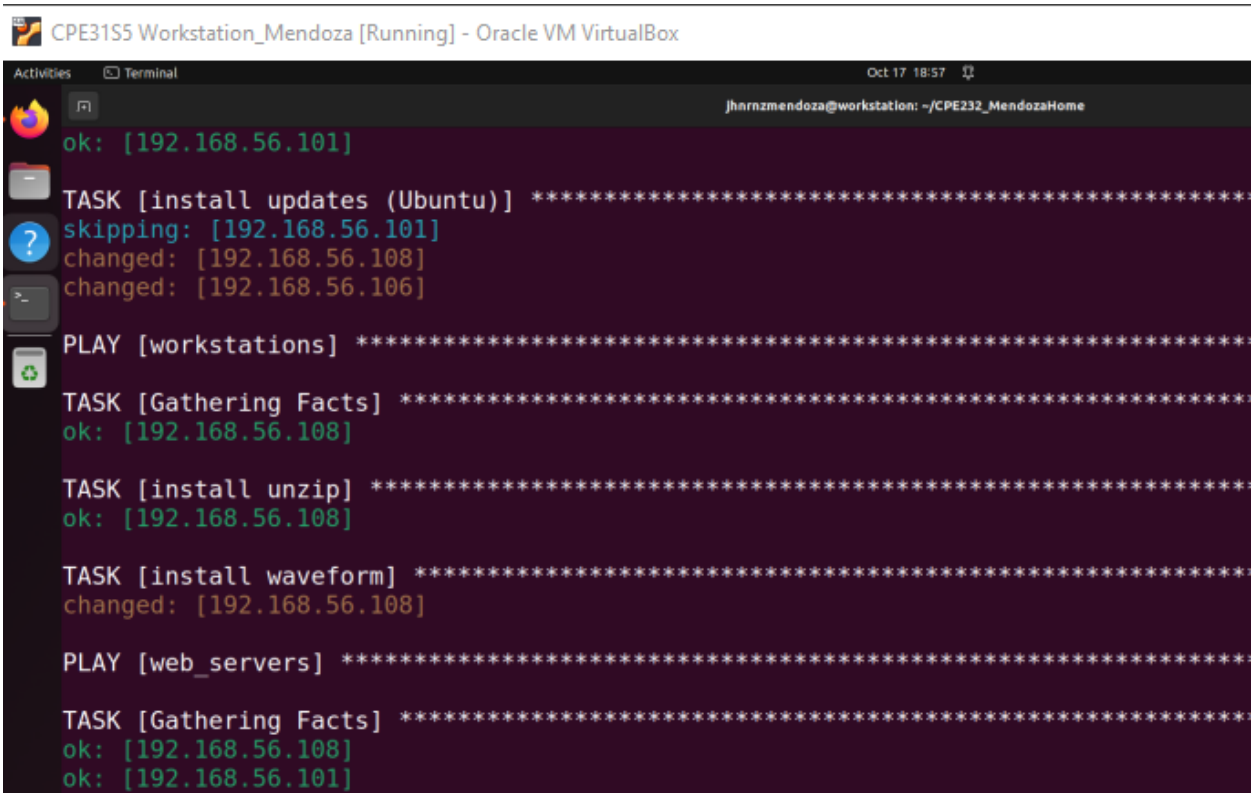
     - name: install terraform
       unarchive:
         src: https://releases.hashicorp.com/terraform/0.12.28/terraform_0.12.28_linux_amd64.zip
         dest: /usr/local/bin
         remote_src: yes
         mode: 0755
         owner: root
         group: root



2. Edit the inventory file and add a workstation group. Add any Ubuntu remote server. Make sure to remember the IP address.



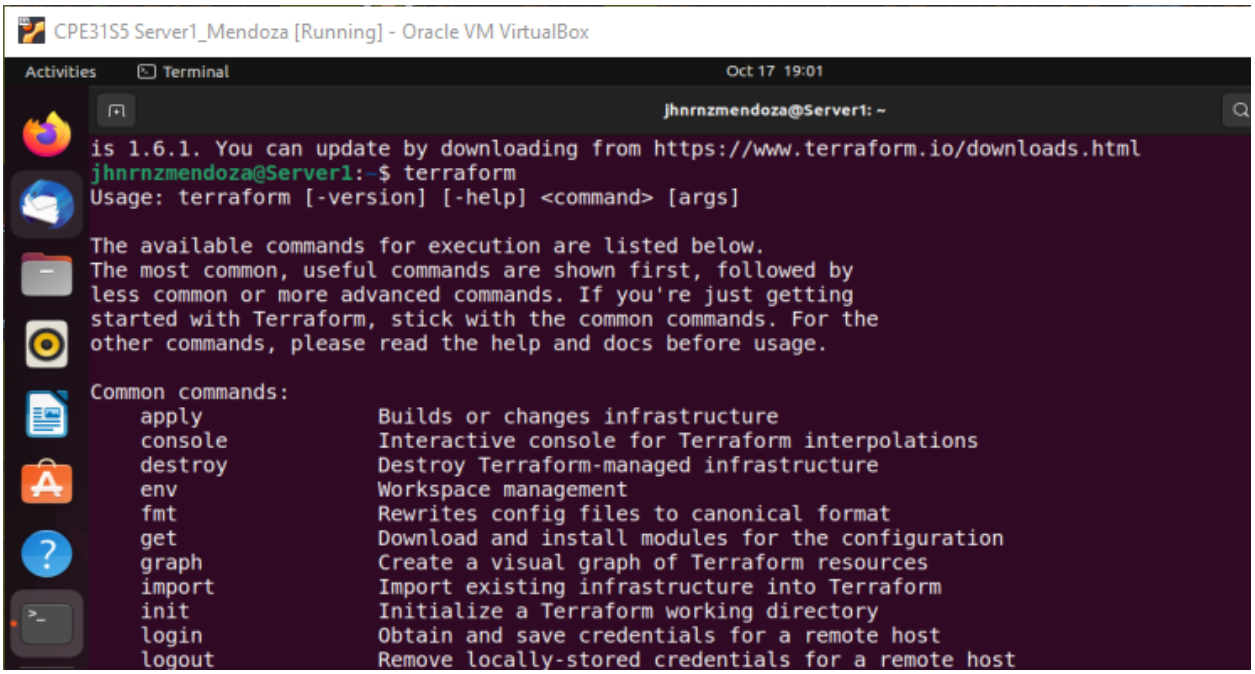
3. Run the playbook. Describe the output.

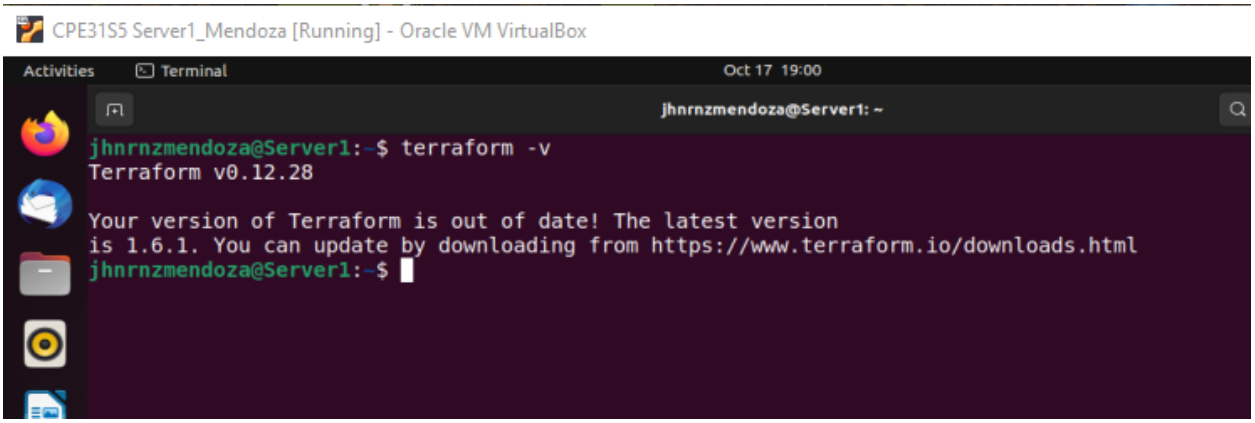


Observation:

- By following the instructions, the new play for [workstations] has been successfully implemented and was able to perform the defined task to the remote ubuntu server which is server 1.

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





Observation:

- On server 1, we can observe that the terraform was successfully installed by the executed playbook from the control node. This was proven by running the terraform command on server 1. The option -v displays the version of the command.

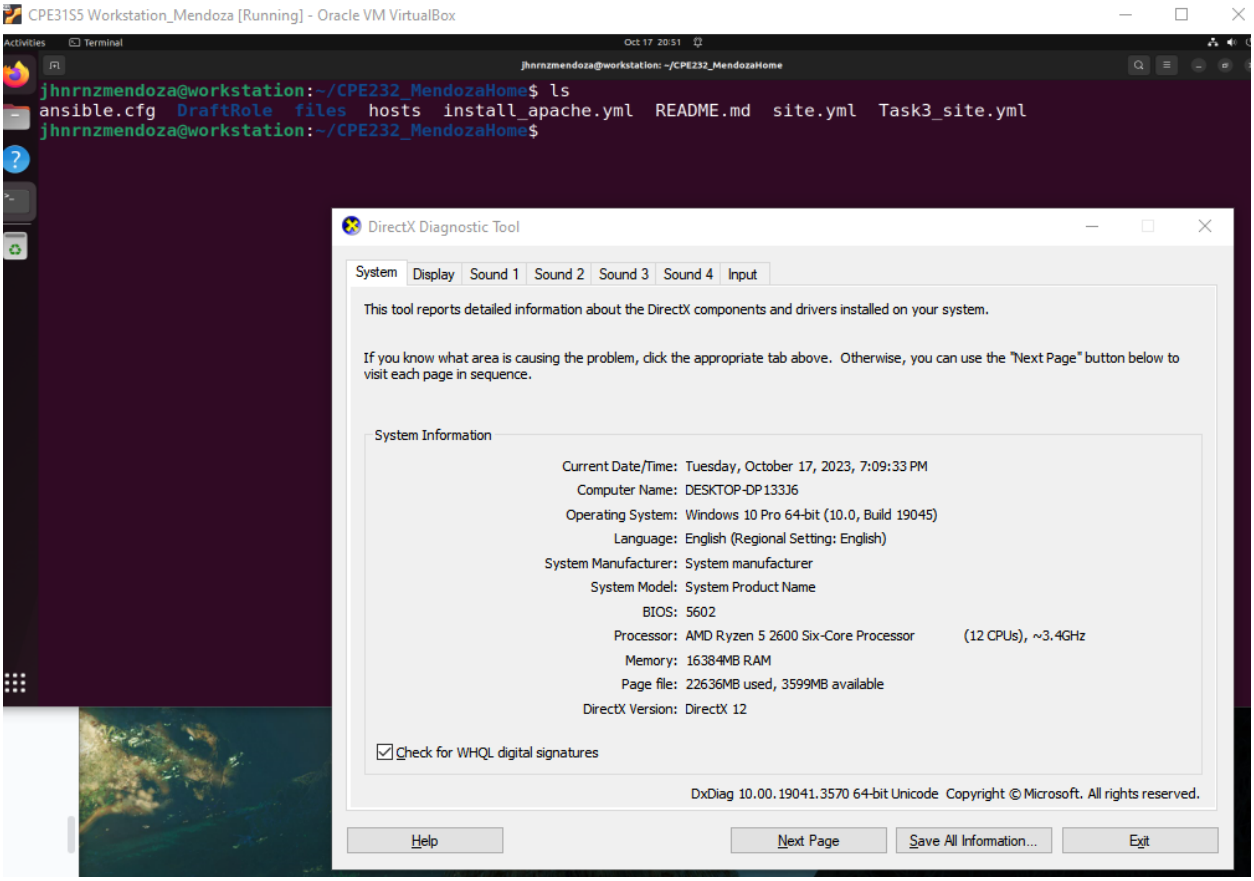
Task 3: Create roles

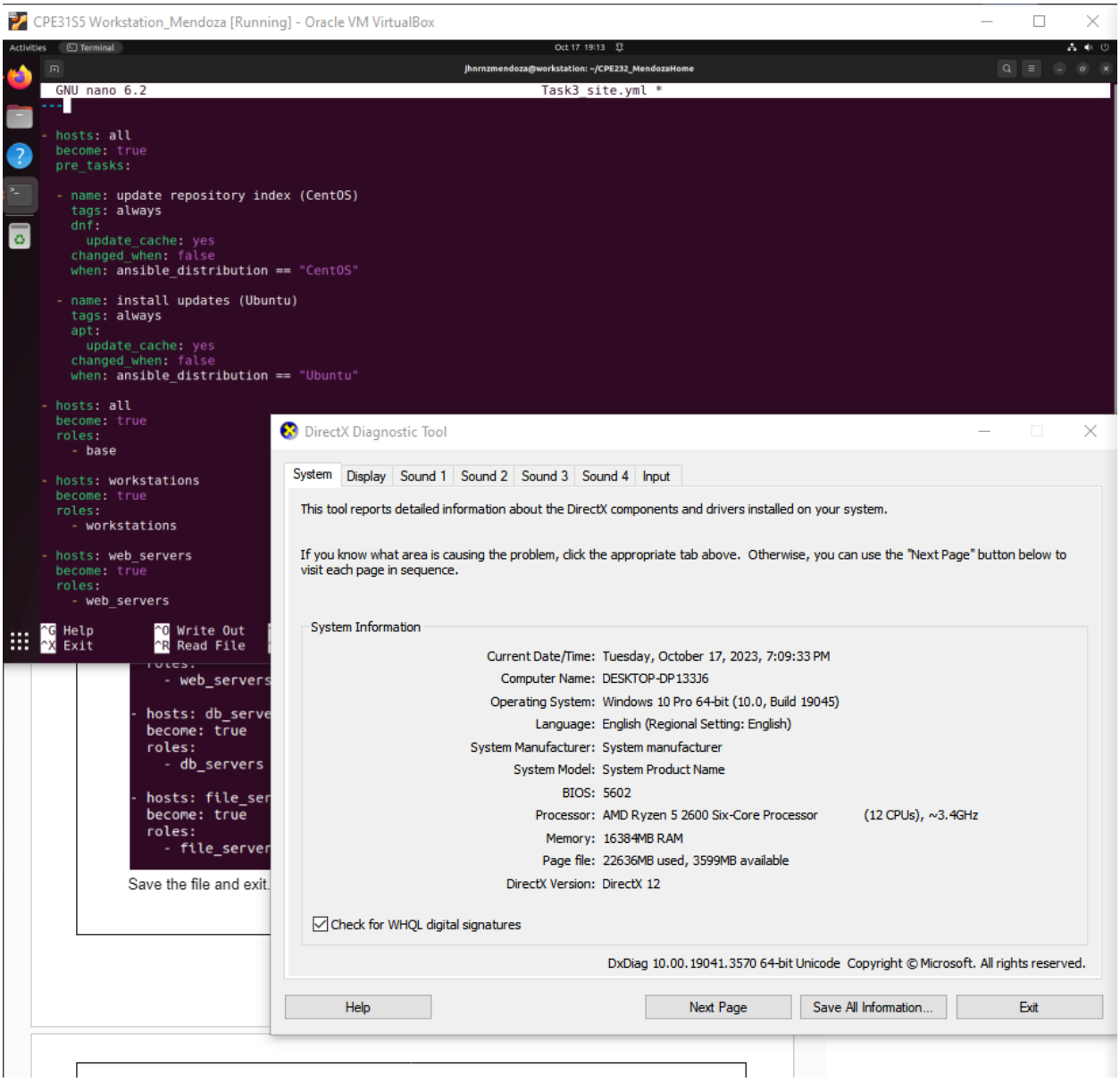
1. Edit the site.yml. Configure roles as follows: (make sure to create a copy of the old site.yml file because you will be copying the specific plays for all groups)

```
---
- 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.

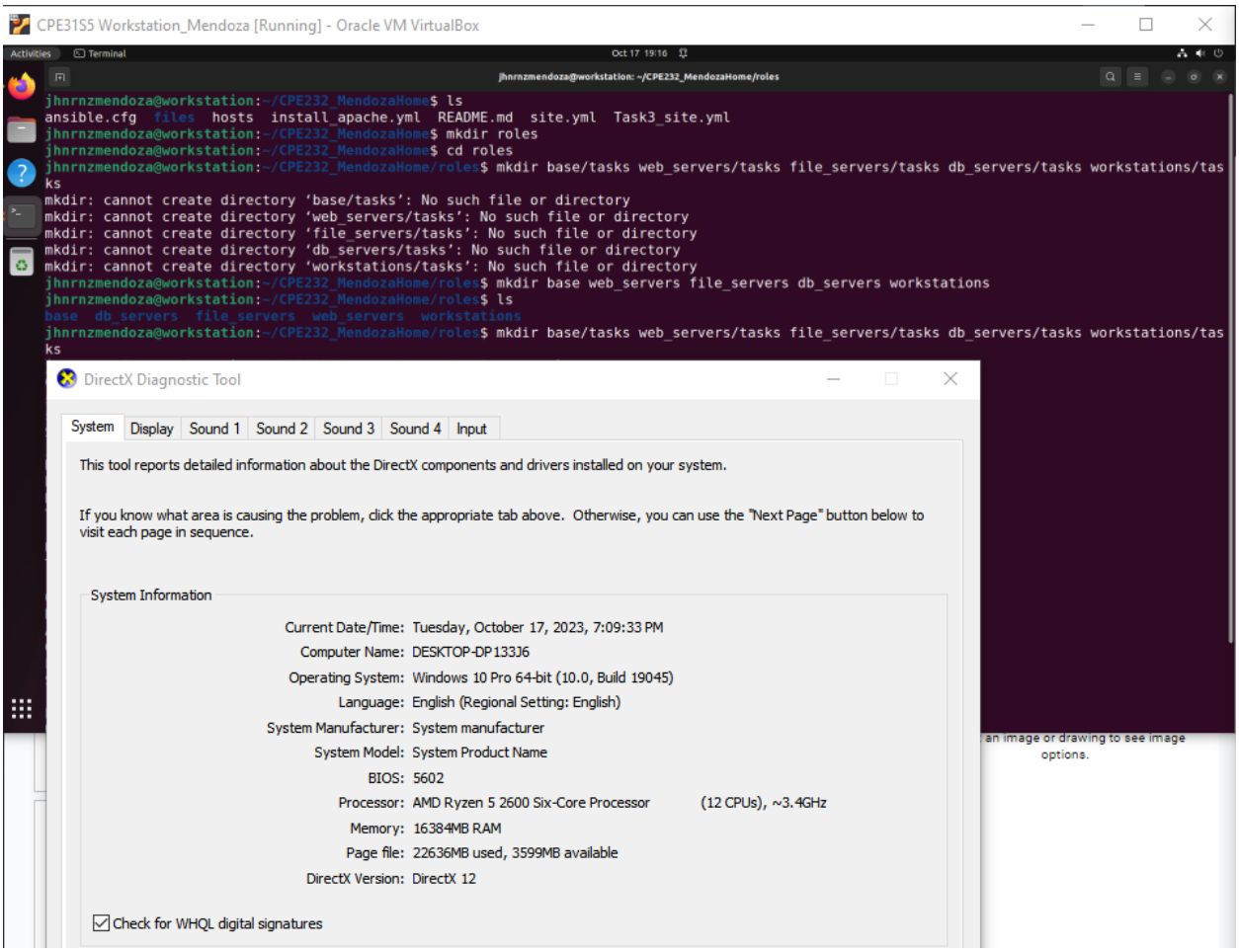
Copying the contents of site.yml to a new file for backup.

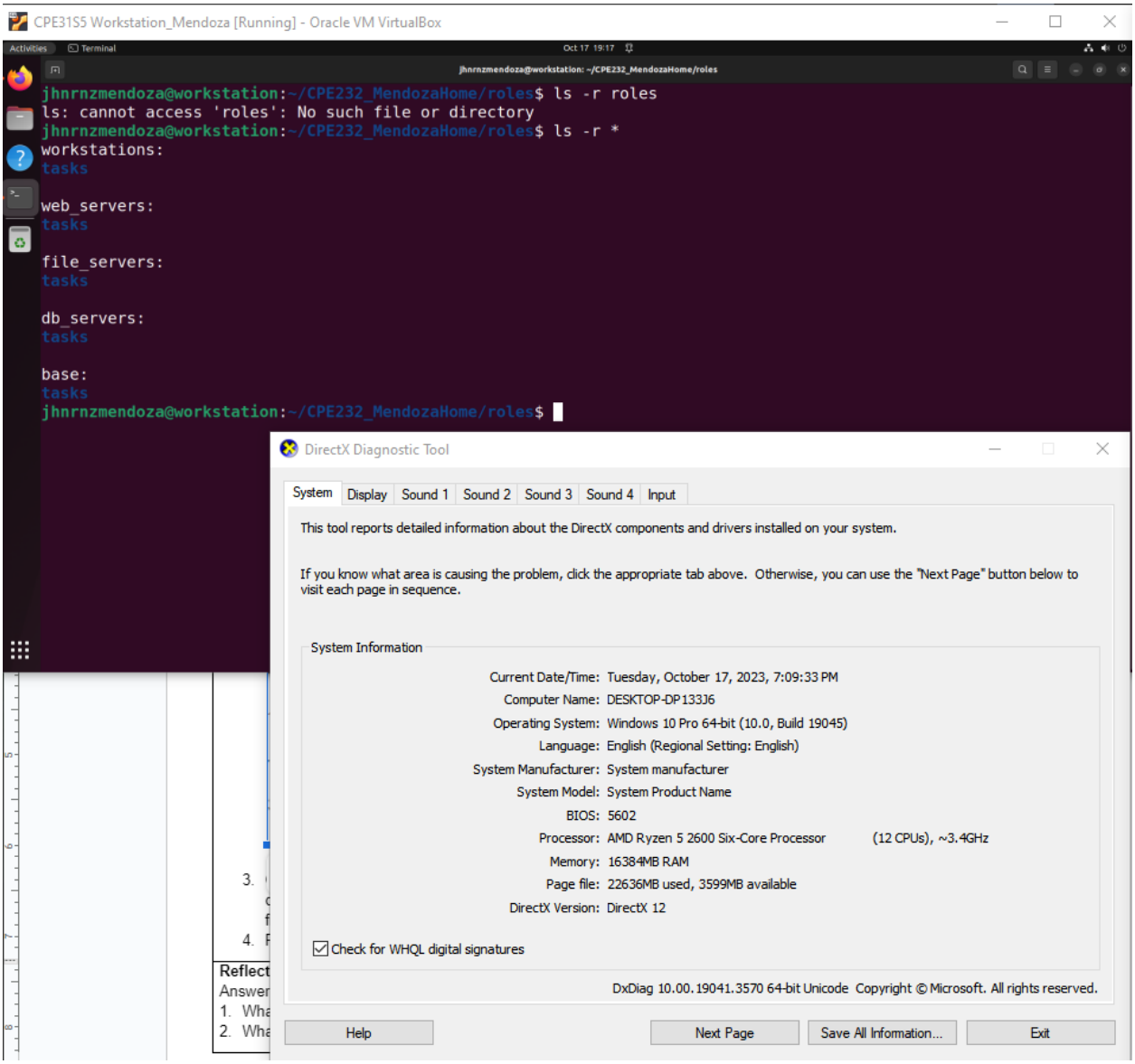



CPE31S5 Workstation_Mendoza [Running] - Oracle VM VirtualBox
Activities
Terminal
Oct 17 19:13
jhnrnzmendoza@workstation: ~/CPE232_MendozaHome
GNU nano 6.2
Task3_site.yml *

- 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
^G Help
^O Write Out
^X Exit
^R Read File
- web_servers
- hosts: db_serve
become: true
roles:
- db_servers
- hosts: file_ser
become: true
roles:
- file_server
Save the file and exit
DirectX Diagnostic Tool
System
Display
Sound 1
Sound 2
Sound 3
Sound 4
Input
This tool reports detailed information about the DirectX components and drivers installed on your system.
If you know what area is causing the problem, click the appropriate tab above. Otherwise, you can use the "Next Page" button below to visit each page in sequence.
System Information
Current Date/Time: Tuesday, October 17, 2023, 7:09:33 PM
Computer Name: DESKTOP-DP133J6
Operating System: Windows 10 Pro 64-bit (10.0, Build 19045)
Language: English (Regional Setting: English)
System Manufacturer: System manufacturer
System Model: System Product Name
BIOS: 5602
Processor: AMD Ryzen 5 2600 Six-Core Processor (12 CPUs), ~3.4GHz
Memory: 16384MB RAM
Page file: 22636MB used, 3599MB available
DirectX Version: DirectX 12
Check for WHQL digital signatures
DxDiag 10.00.19041.3570 64-bit Unicode Copyright © Microsoft. All rights reserved.
Help
Next Page
Save All Information...
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.

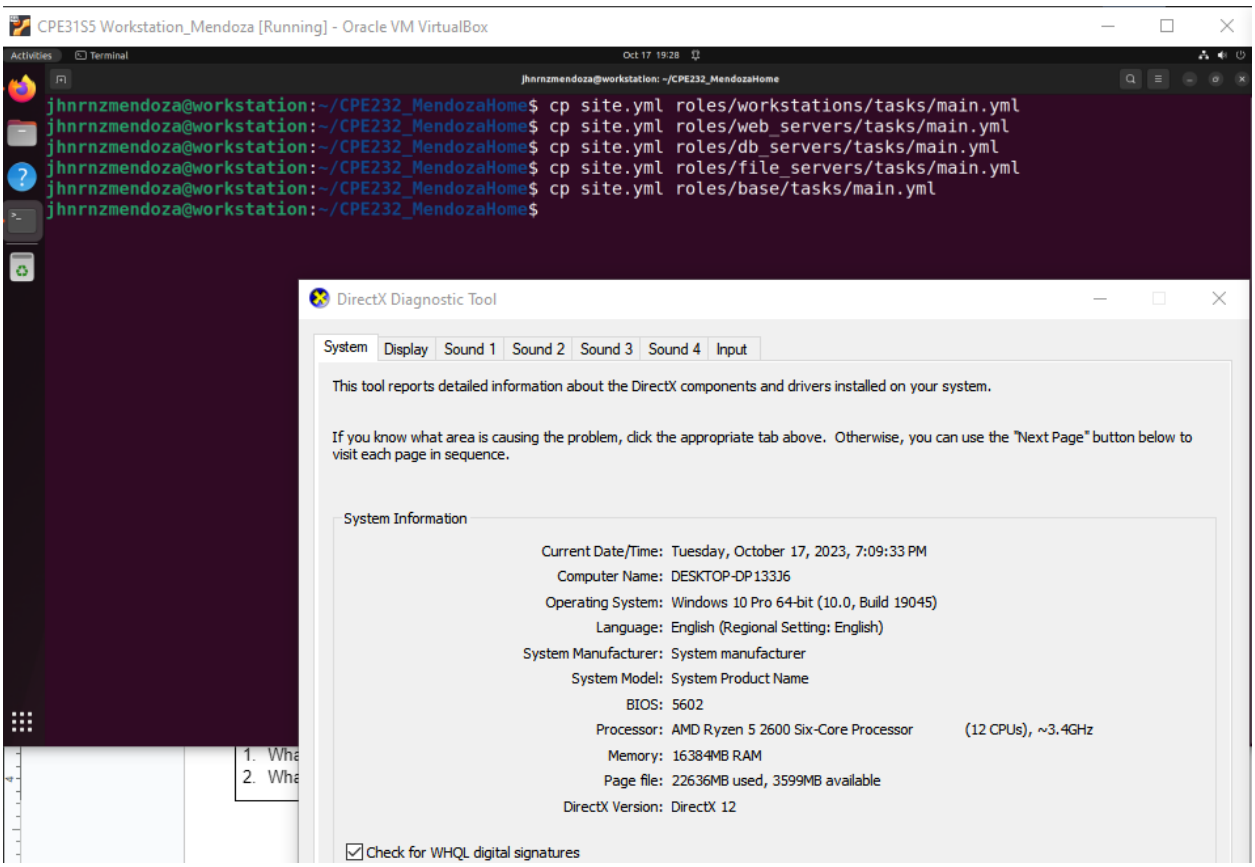



CPE31S5 Workstation_Mendoza [Running] - Oracle VM VirtualBox
Activities
Terminal
Oct 17 19:17
jhnrnzmendoza@workstation: ~/CPE232_MendozaHome/roles
jhnrnzmendoza@workstation:~/CPE232_MendozaHome/roles$ ls -r roles
ls: cannot access 'roles': No such file or directory
jhnrnzmendoza@workstation:~/CPE232_MendozaHome/roles$ ls -r *
workstations:
tasks
web_servers:
tasks
file_servers:
tasks
db_servers:
tasks
base:
tasks
jhnrnzmendoza@workstation:~/CPE232_MendozaHome/roles$
DirectX Diagnostic Tool
System
Display
Sound 1
Sound 2
Sound 3
Sound 4
Input
This tool reports detailed information about the DirectX components and drivers installed on your system.
If you know what area is causing the problem, click the appropriate tab above. Otherwise, you can use the "Next Page" button below to visit each page in sequence.
System Information
Current Date/Time: Tuesday, October 17, 2023, 7:09:33 PM
Computer Name: DESKTOP-DP133J6
Operating System: Windows 10 Pro 64-bit (10.0, Build 19045)
Language: English (Regional Setting: English)
System Manufacturer: System manufacturer
System Model: System Product Name
BIOS: 5602
Processor: AMD Ryzen 5 2600 Six-Core Processor (12 CPUs), ~3.4GHz
Memory: 16384MB RAM
Page file: 22636MB used, 3599MB available
DirectX Version: DirectX 12
Check for WHQL digital signatures
DxDiag 10.00.19041.3570 64-bit Unicode Copyright © Microsoft. All rights reserved.
Help
Next Page
Save All Information...
Exit

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.

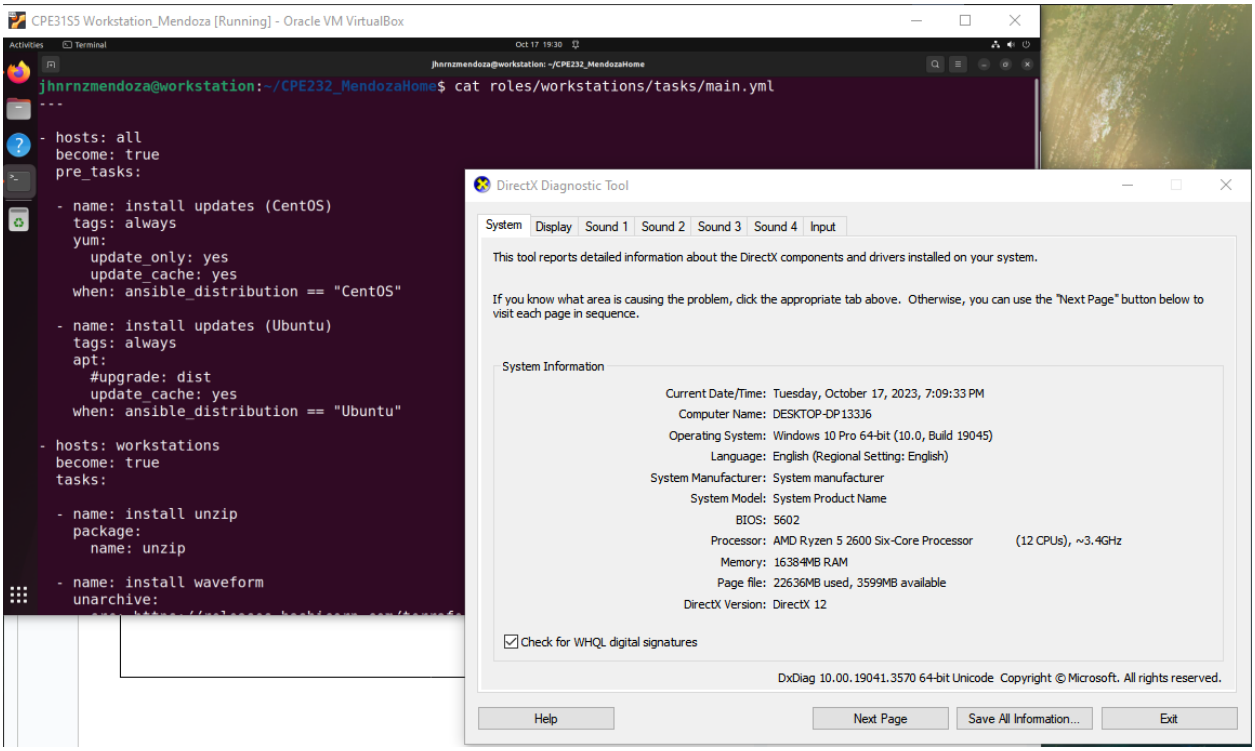
   Manually going to each directory and copy and pasting the contents of the site.yml playbook is hassle and inefficient. Therefore, I have used the cp command to copy the old site.yml to main.yml on each directory/tasks.

   Here is the command that I have used:

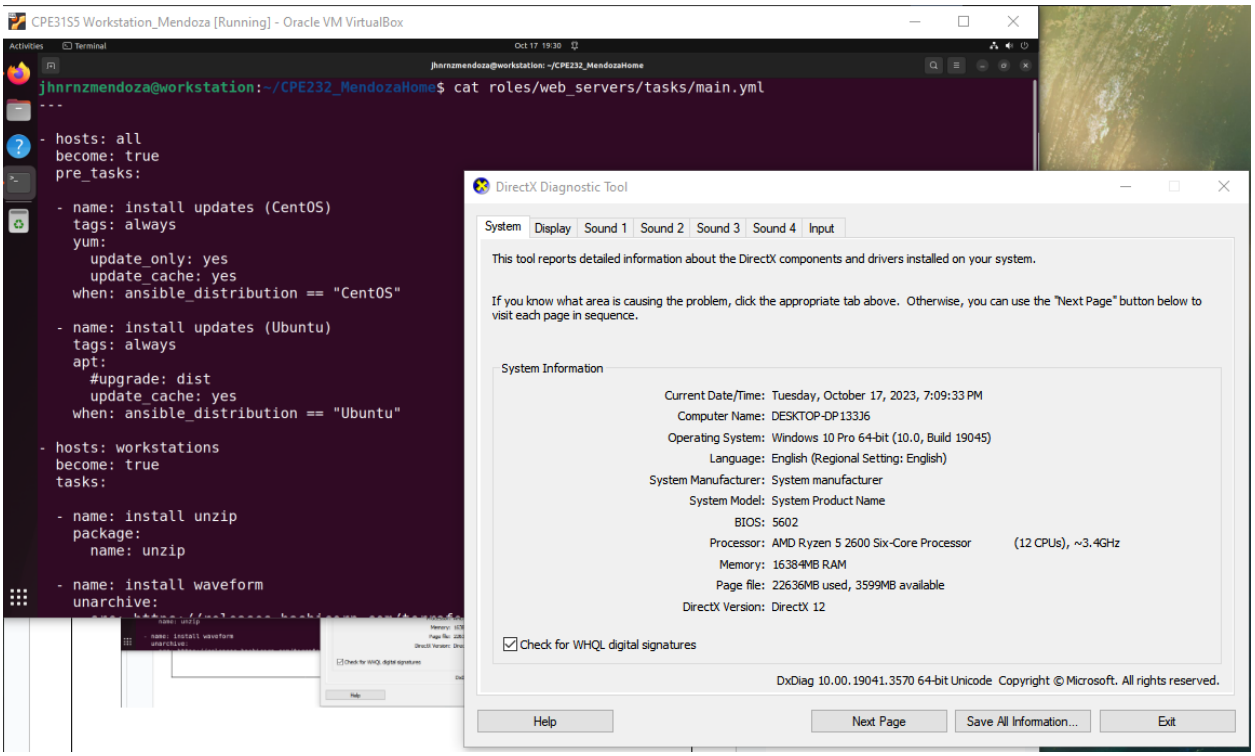


Checking if the new main.yml files are added with the appropriate contents.

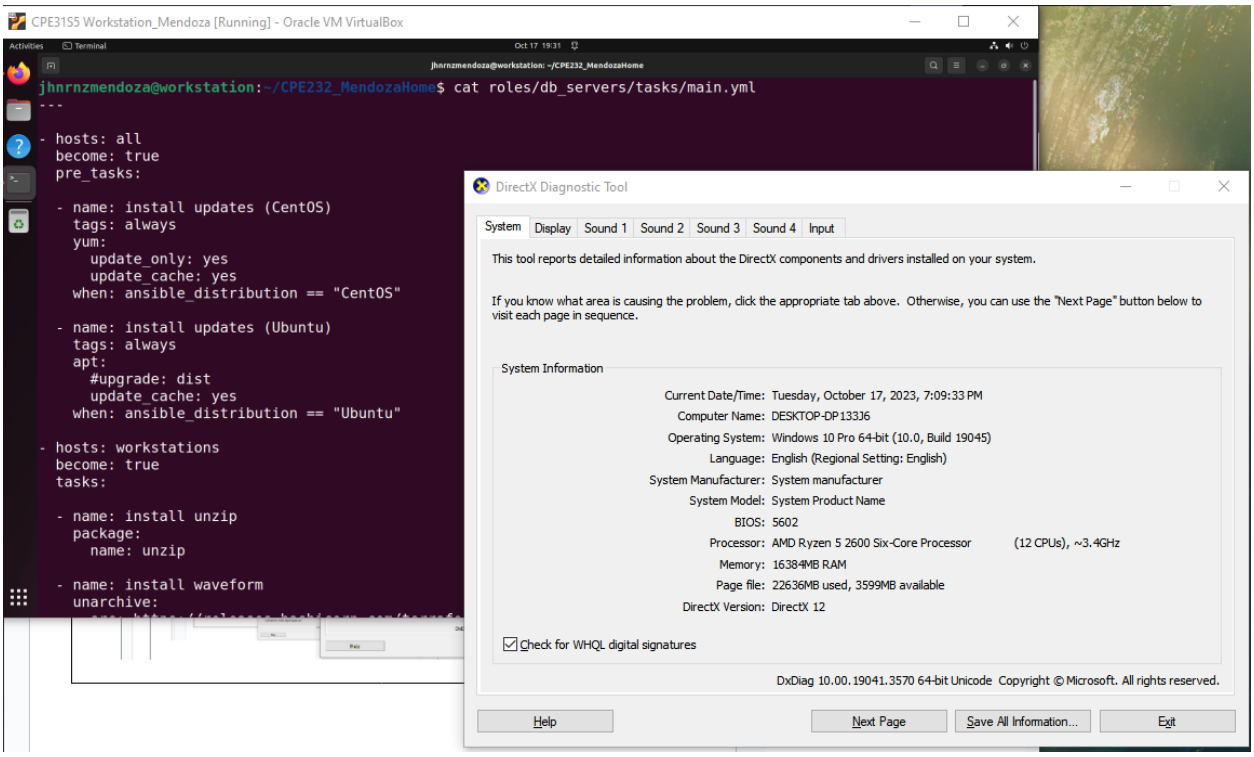
Directory: roles/workstations/tasks



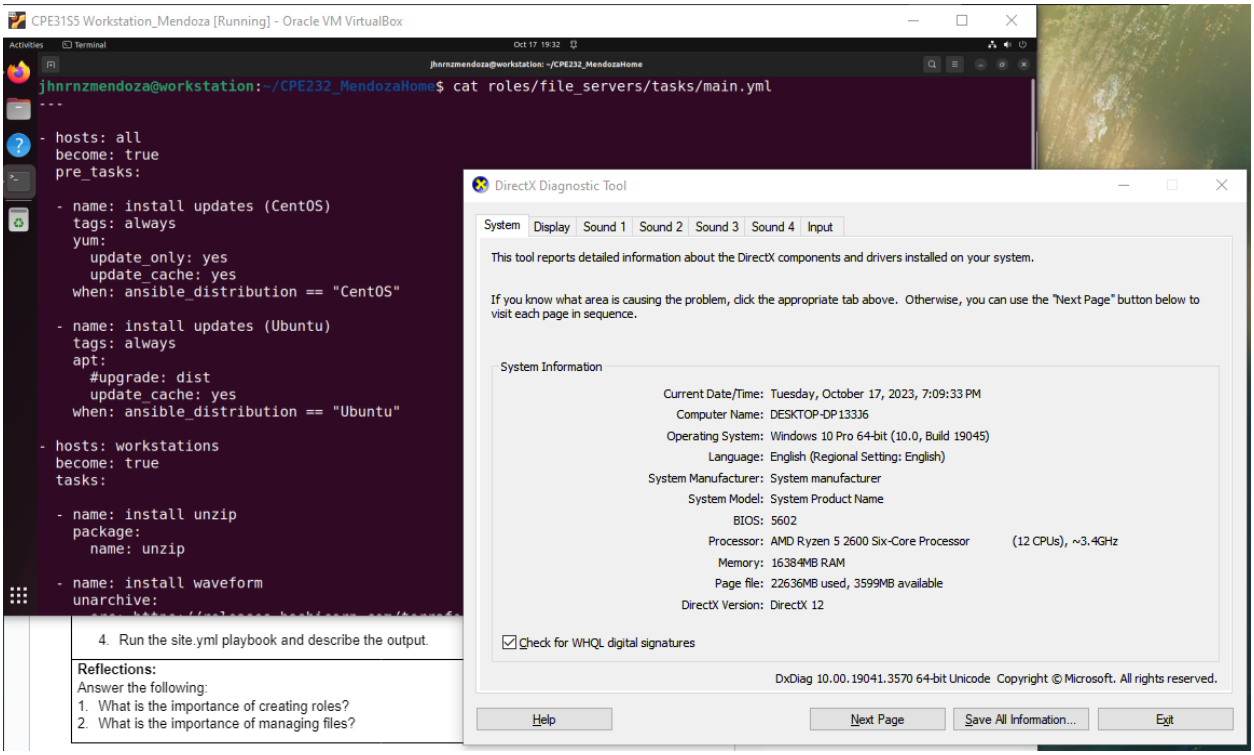
Directory: roles/web_servers/tasks



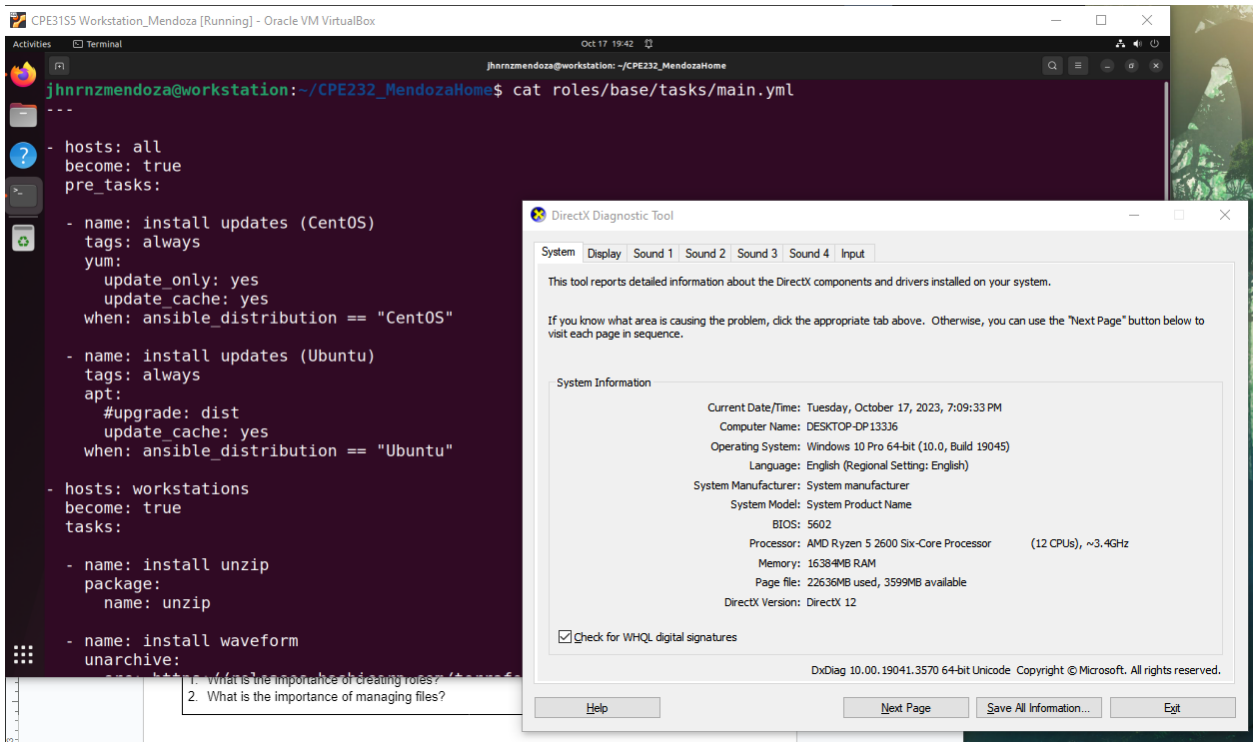
Directory: roles/db_servers/tasks



Directory: roles/file_servers/tasks



Directory: roles/base/tasks



Observation:

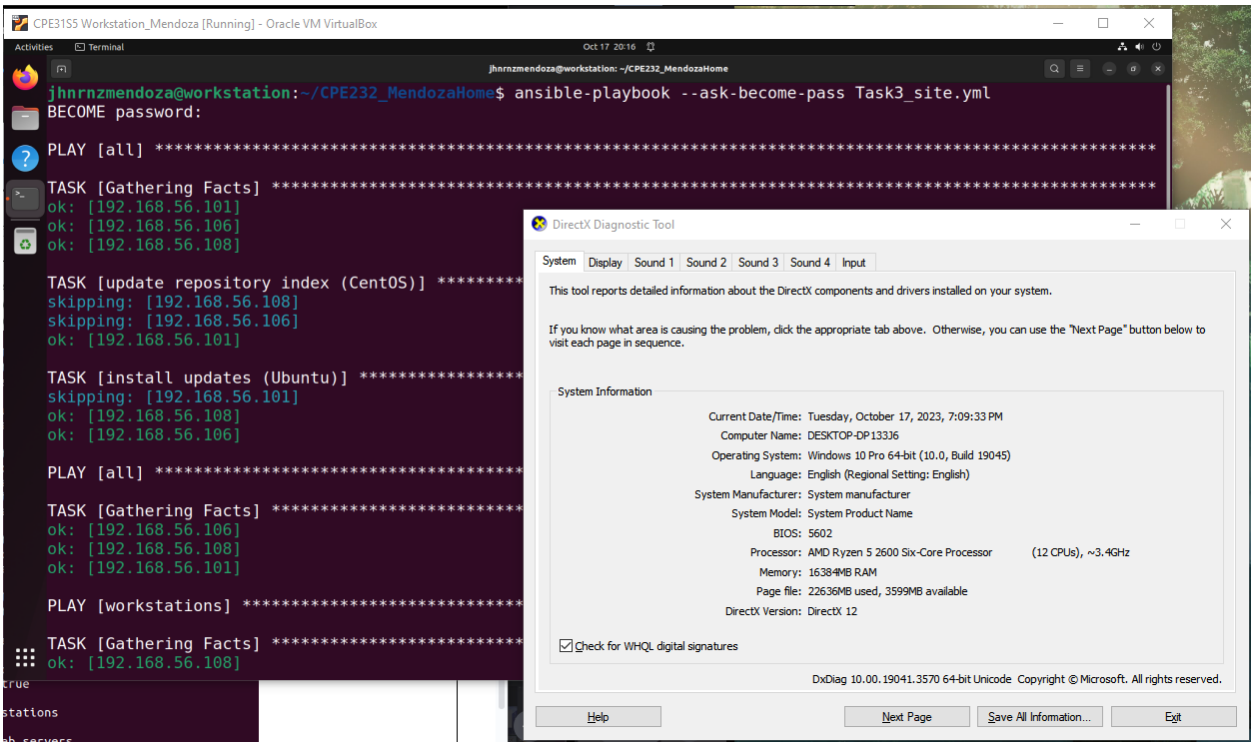
- In this step, I have used the cp command in order to copy the old site.yml to be the contents of the main.yml. In my case, the old site.yml is stored on the Task3_site.yml. I have concatenated the contents of each main.yml under each roles/directory.
- As observed on each query, the old site.yml was successfully copied to each main.yml.

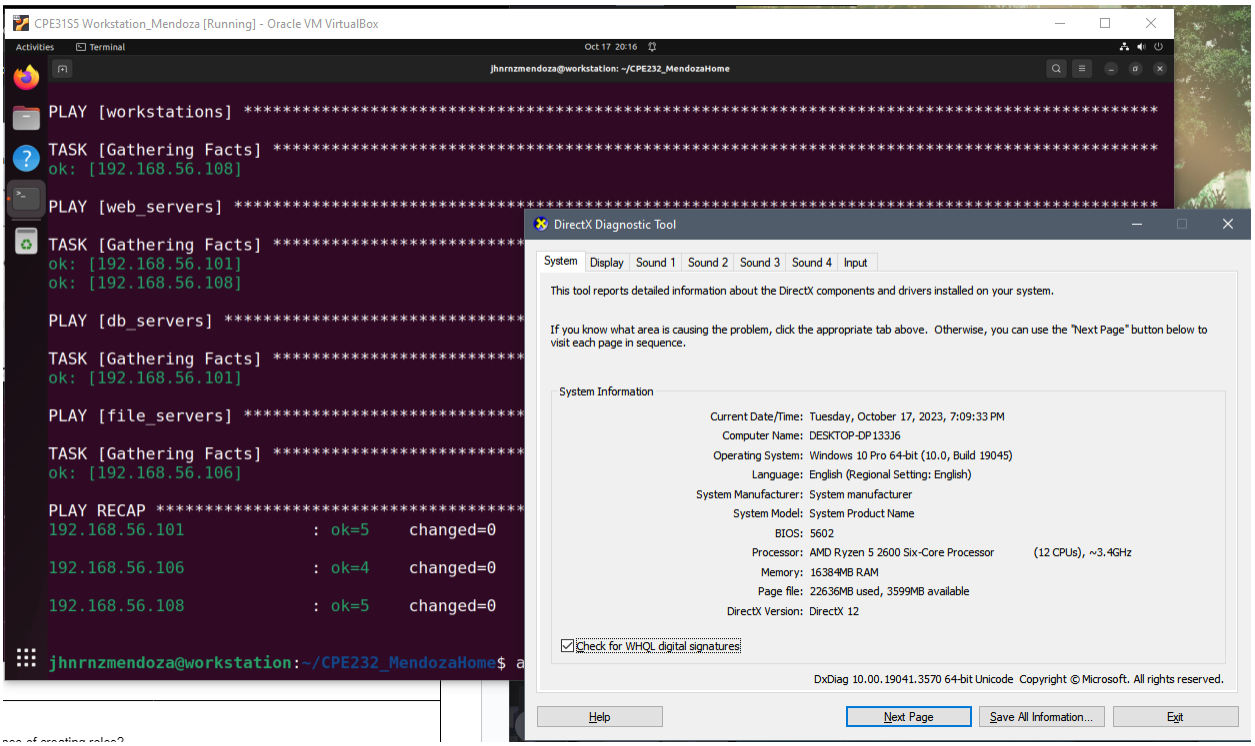
4. Run the site.yml playbook and describe the output.

   The original code has errors when executed, therefore I have changed some of the syntaxes such as:

   - Changing the package manager from `yum` from `dnf`
   - Changing the syntax on defining the roles.

   Output:





Observation:

- After changing some of the syntax, the playbook was successfully executed. We can observe that each of the directory under the roles directory has been executed along the run playbook.
- In my opinion, the roles specifically run the playbook file within its directory. For instance, the main.yml in the directory role/workstations/tasks/, when we run the new site.yml, the workstation directory will now read the playbook main.yml and show which will be the target remote server.
- As observed, the play web_servers targeted the remote server 1 and centOS since these 2 servers will be needed in that specific play, which is why we cannot see the file server 3 to be “skipped”.

Reflections:

Answer the following:

1. What is the importance of creating roles?
    - Creating roles in ansible makes playbook run much easier to understand since it will display the target remote servers that will be used in the run. As observed on the previous playbook run, it was able to gather which remote servers will be used specifically without traversing every server and skipping them.
    - Essentially, ansible roles also help in organizing the structure of the ansible playbooks and executing them. By having roles, the playbook task only targets specific nodes. In addition, it is a way to group

2. What is the importance of managing files?
    - By managing files, we would have an organized structure of our directories. As observed in this activity, each ansible role has a defined directory having the same playbook files under it. This makes the loading of the variables and needed files efficient. We would also be able to debug the errors easily since the directories are properly organized. In addition, the playbook would be able to identify the remote nodes that is needed in each playbook task.