Name: Quizon, Nowell Gabriel C.	**Date Performed:** 10/18/2023
Course/Section: CPE31S5	**Date Submitted:** 10/18/2023
Instructor: Engr. Roman Richard	**Semester and SY:** 1st – 2023 - 2024

Activity 7: Managing Files and Creating Roles in Ansible

1. **Objectives:**

1.1 Manage files in remote servers

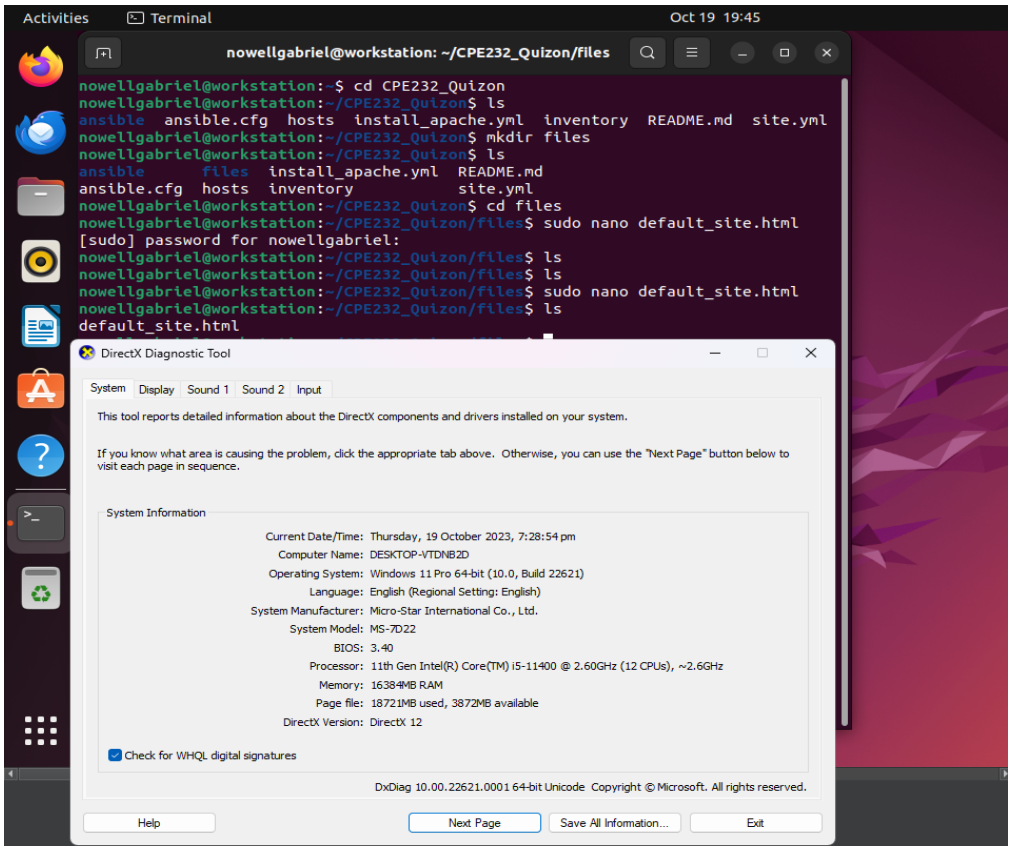
1.2 Implement roles in ansible

2. **Discussion**:

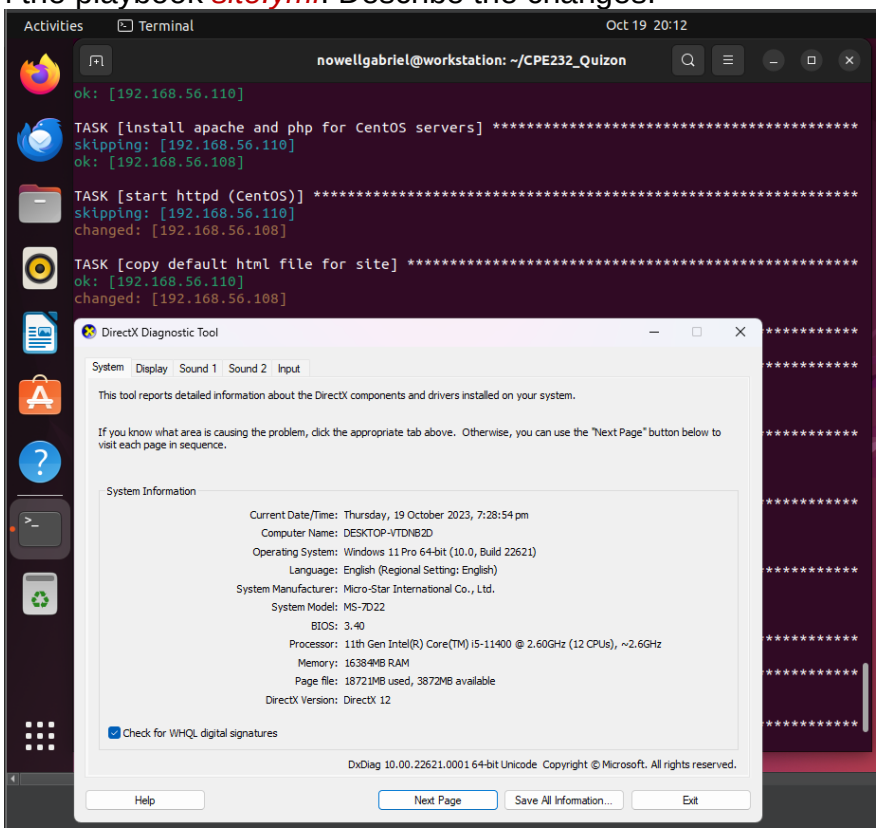
In this activity, we look at the concept of copying a file to a server. We are going to create a file into our git repository and use Ansible to grab that file and put it into a particular place so that we could do things like customize a default website, or maybe install a default configuration file.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

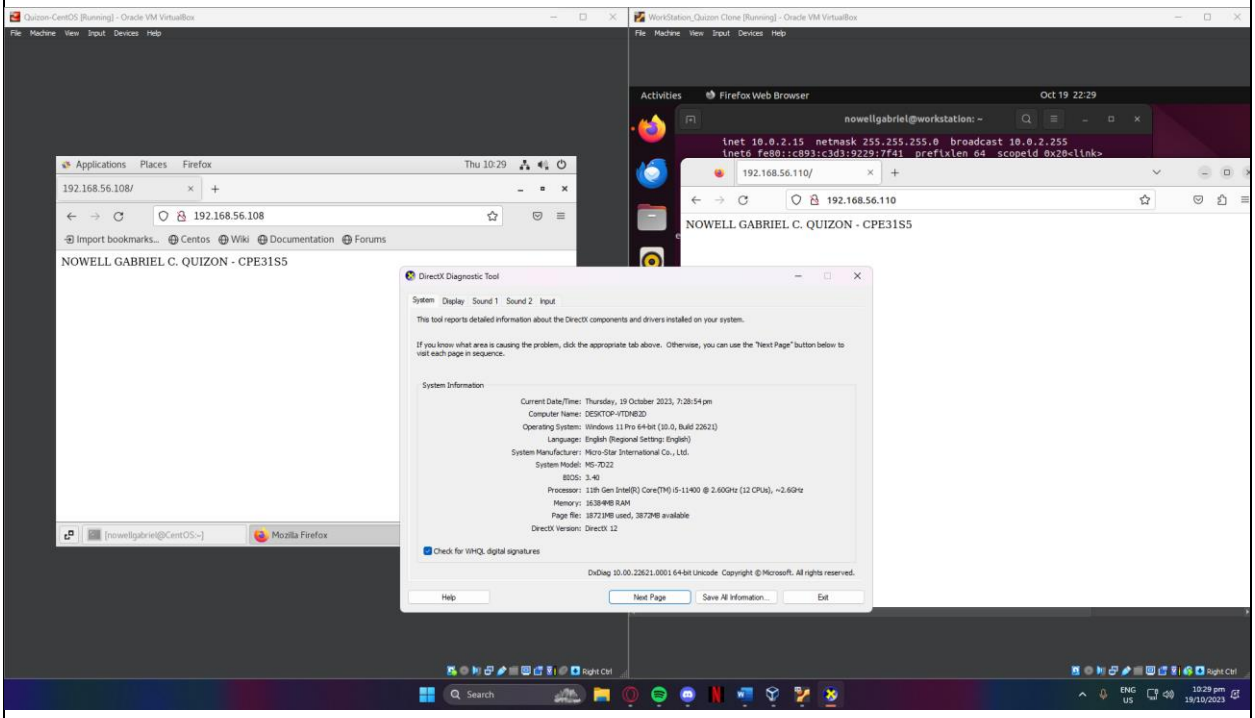


2. Edit the *site.yml* file and just below the *web_servers* play, create a new file to copy the default html file for site:
   - name: copy default html file for site
     tags: apache, apache2, httpd
     copy:
       src: default_site.html
       dest: /var/www/html/index.html
       owner: root
       group: root
       mode: 0644
3. Run the playbook *site.yml*. Describe the changes.

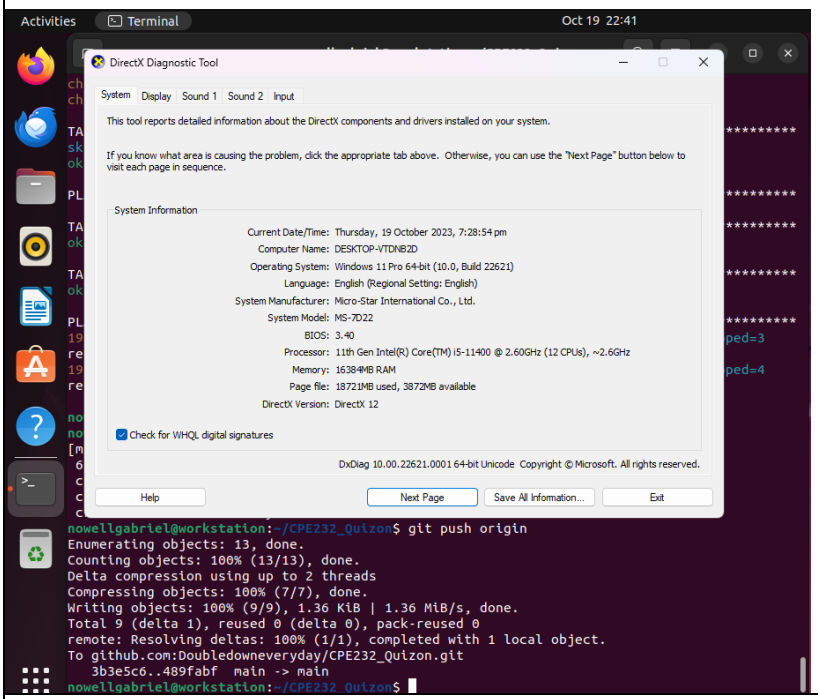


   - What it did is copy the default html to a new file and directory. The ownership of the file belongs to root users and also it belongs to the root group. The permission of the file is 0644.

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.

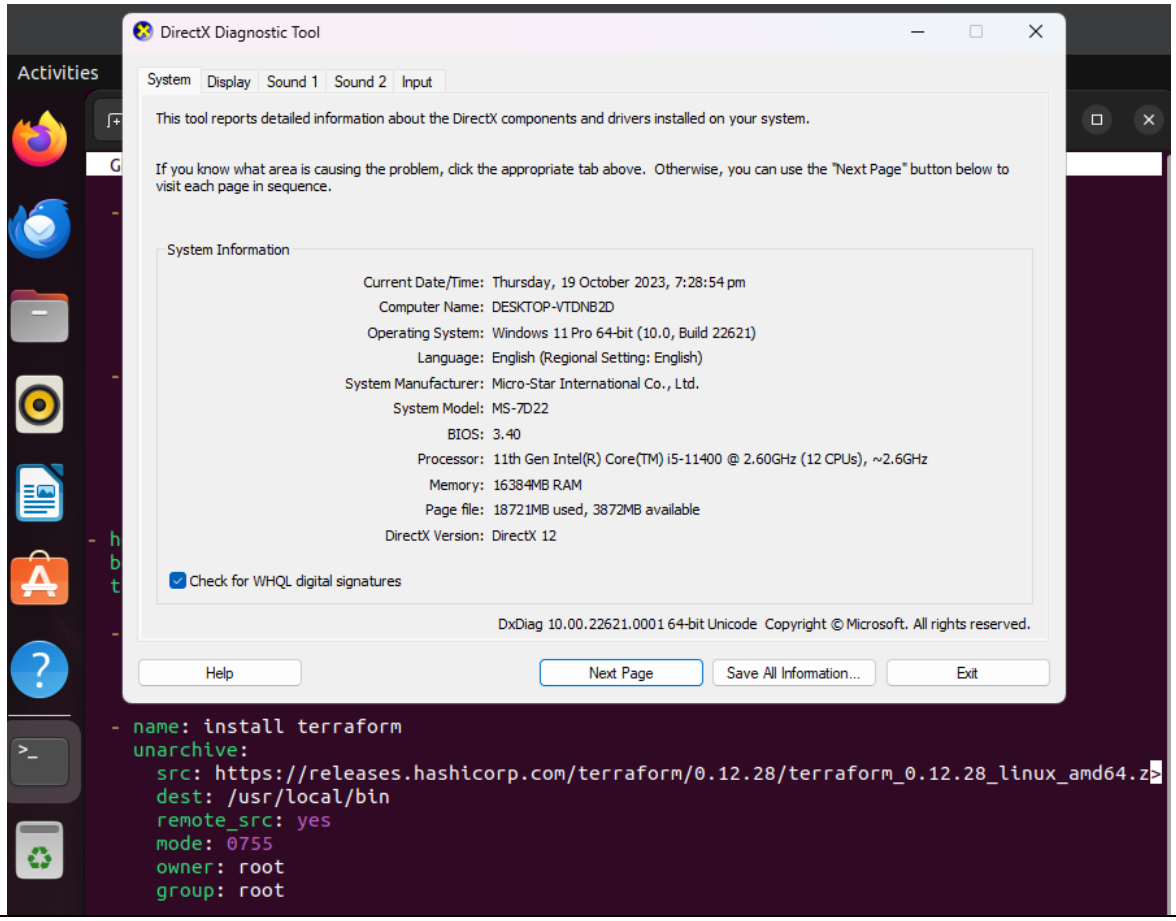


Task 2: Download a file and extract it to a remote server

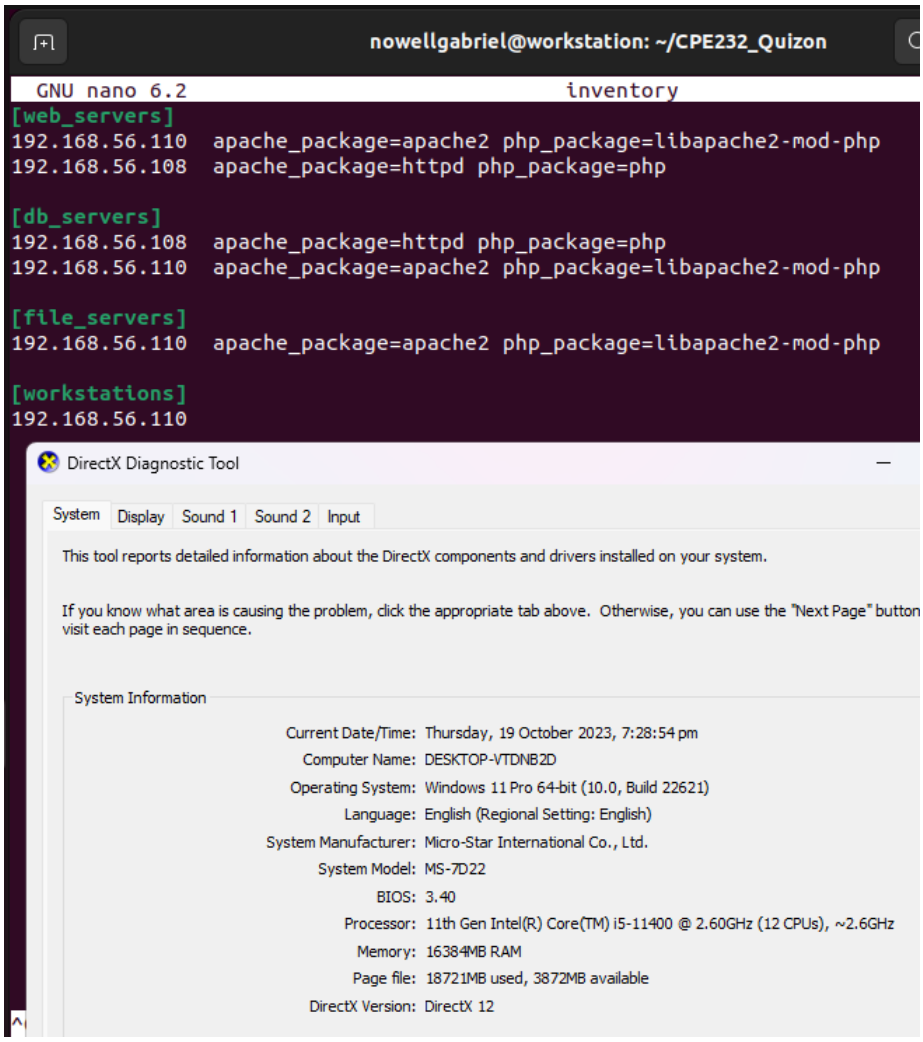
1. Edit the site.yml. Just before the web_servers play, create a new play:
   - hosts: workstations
     become: true
     tasks:

     - name: install unzip
       package:
         name: unzip

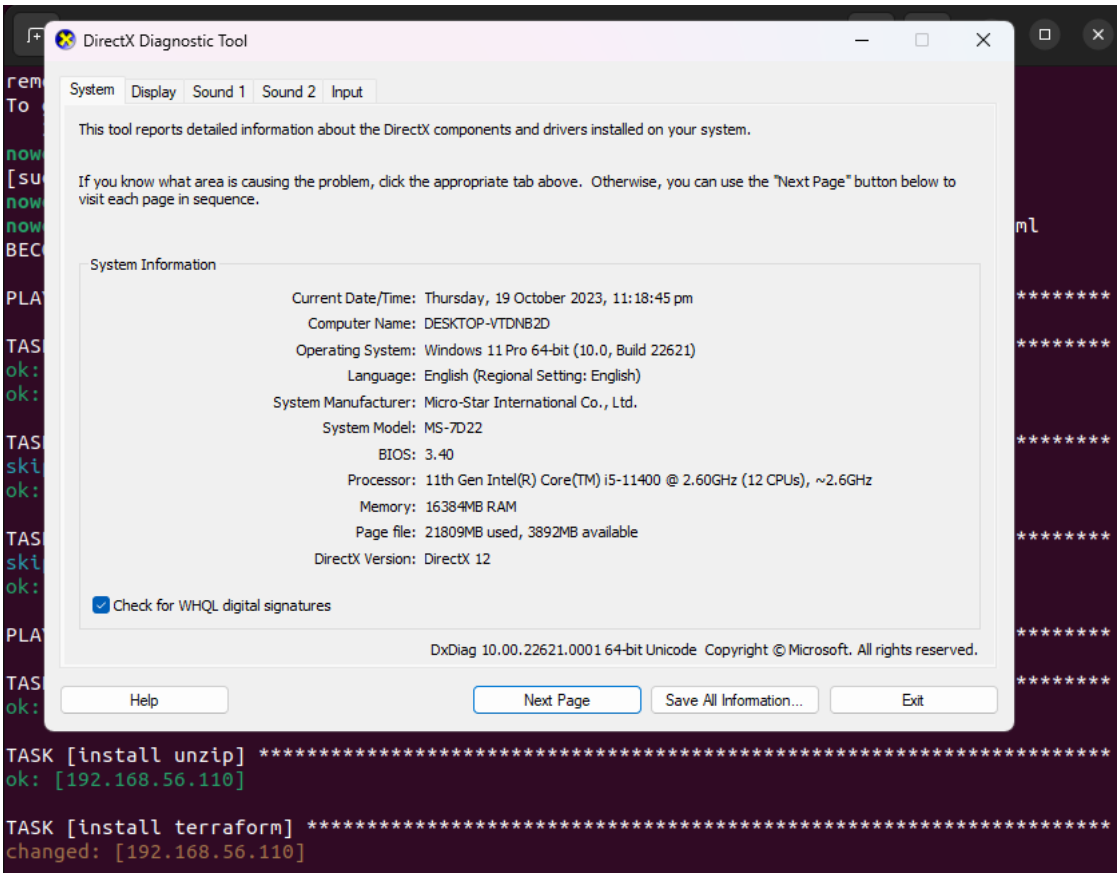
     - name: install terraform
       unarchive:
         src:
     https://releases.hashicorp.com/terraform/0.12.28/terraform_0.12.28_linux_amd64.zip
         dest: /usr/local/bin
         remote_src: yes
         mode: 0755
         owner: root
         group: root



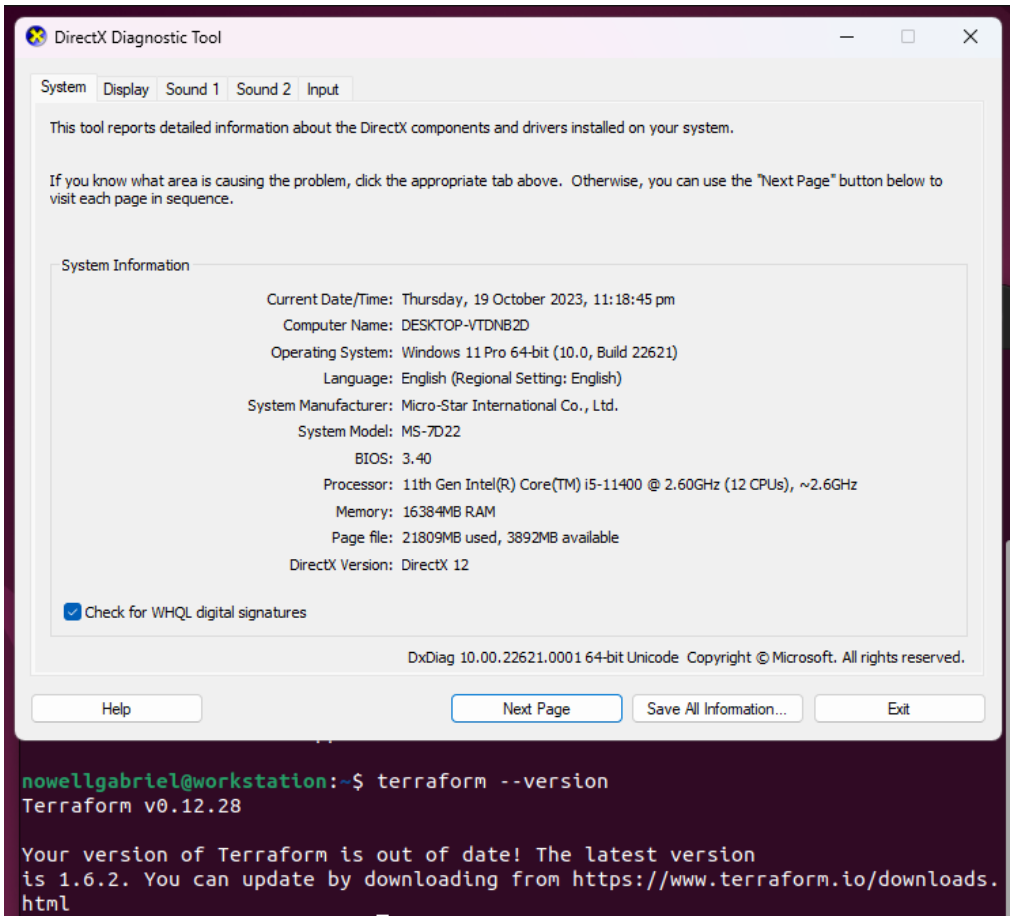
2. Edit the inventory file and add workstations group. Add any Ubuntu remote server. Make sure to remember the IP address.



3. Run the playbook. Describe the output.



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



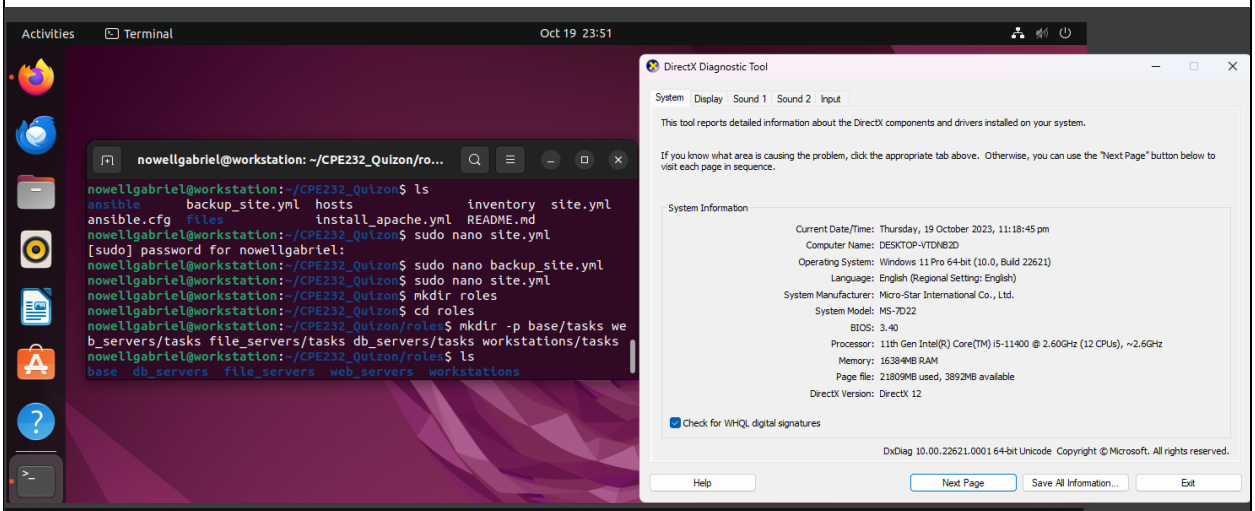
Task 3: Create roles

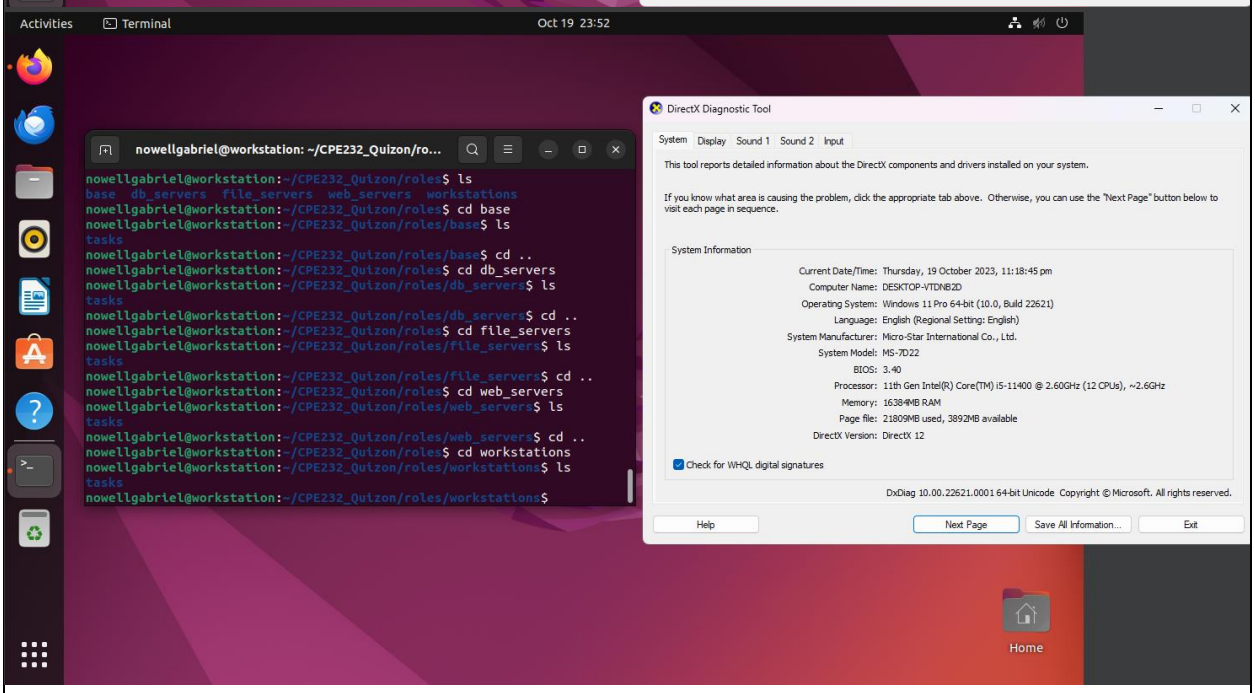
1. Edit the site.yml. Configure roles as follows: (make sure to create a copy of the old site.yml file because you will be copying the specific plays for all groups)

```
---
- 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
```

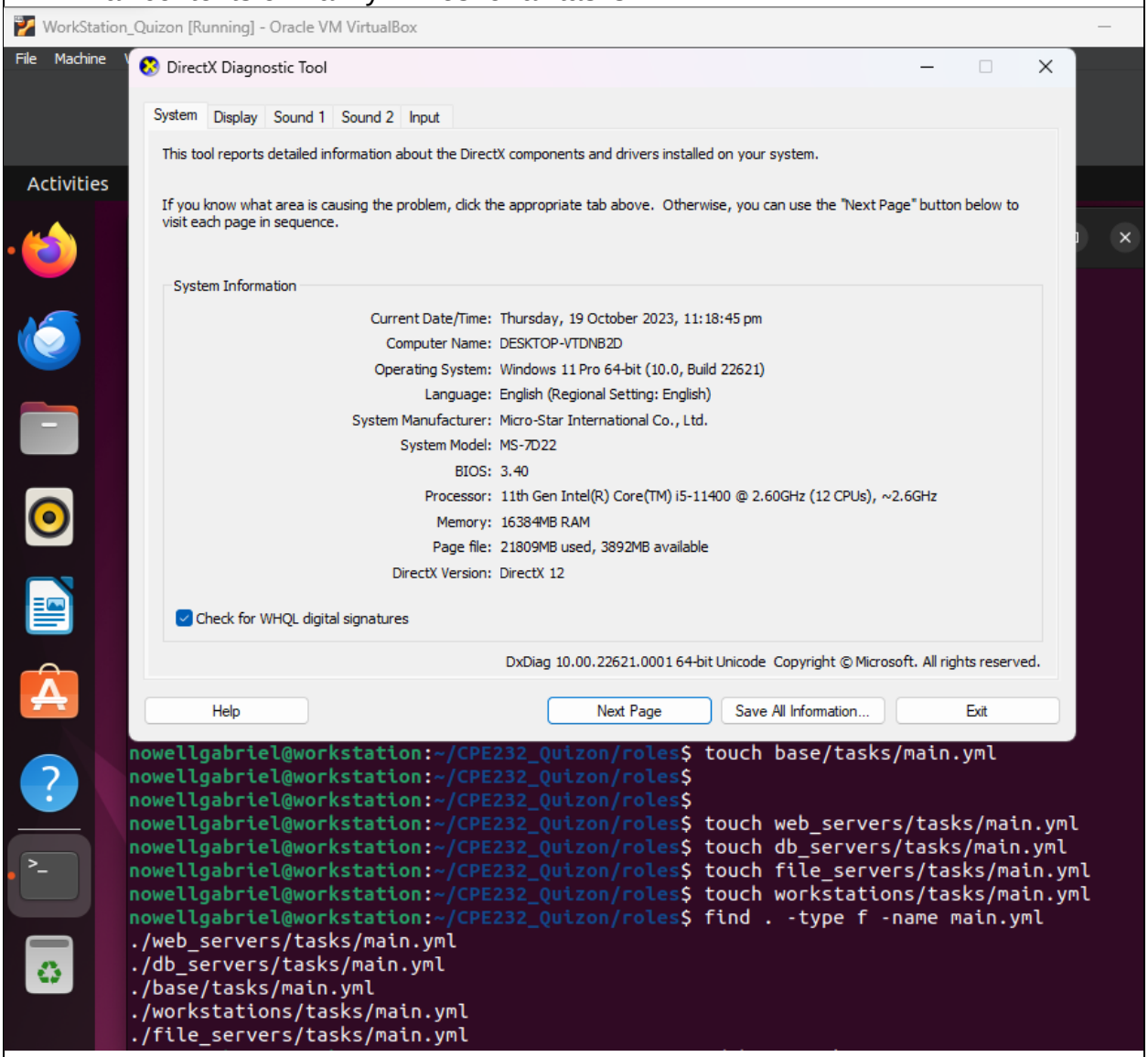
Save the file and exit.

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.

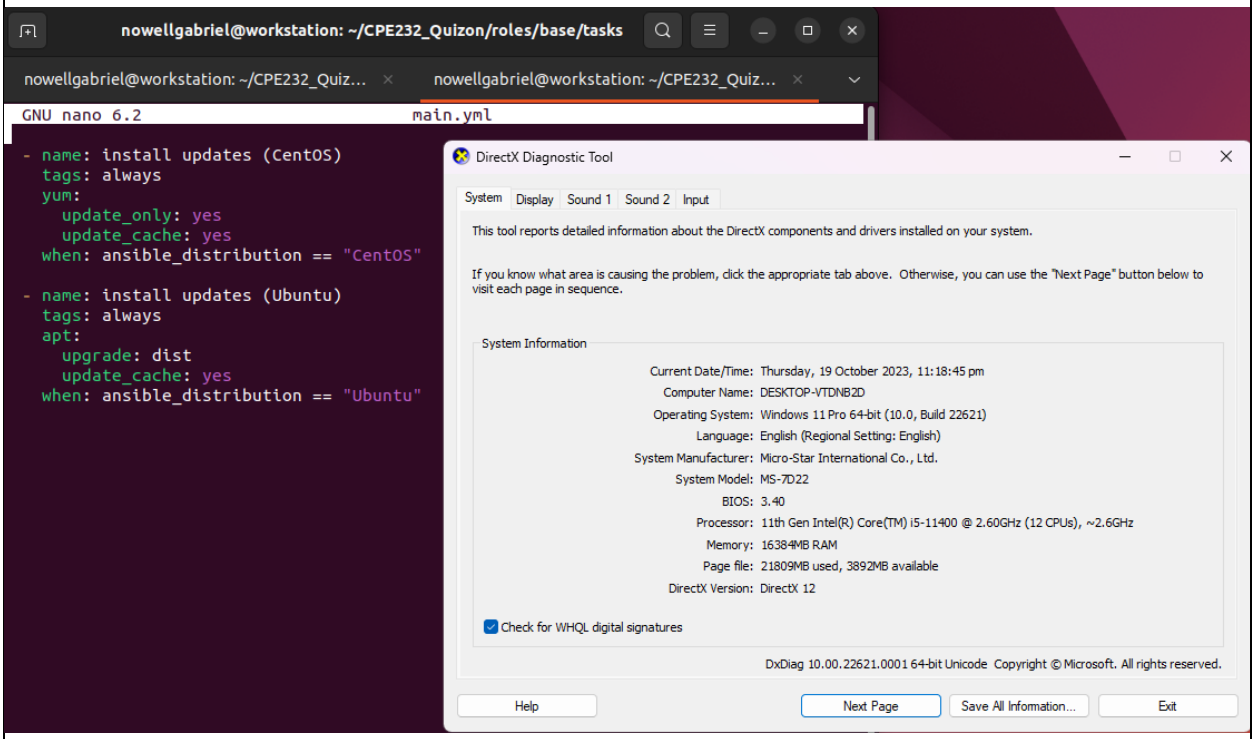




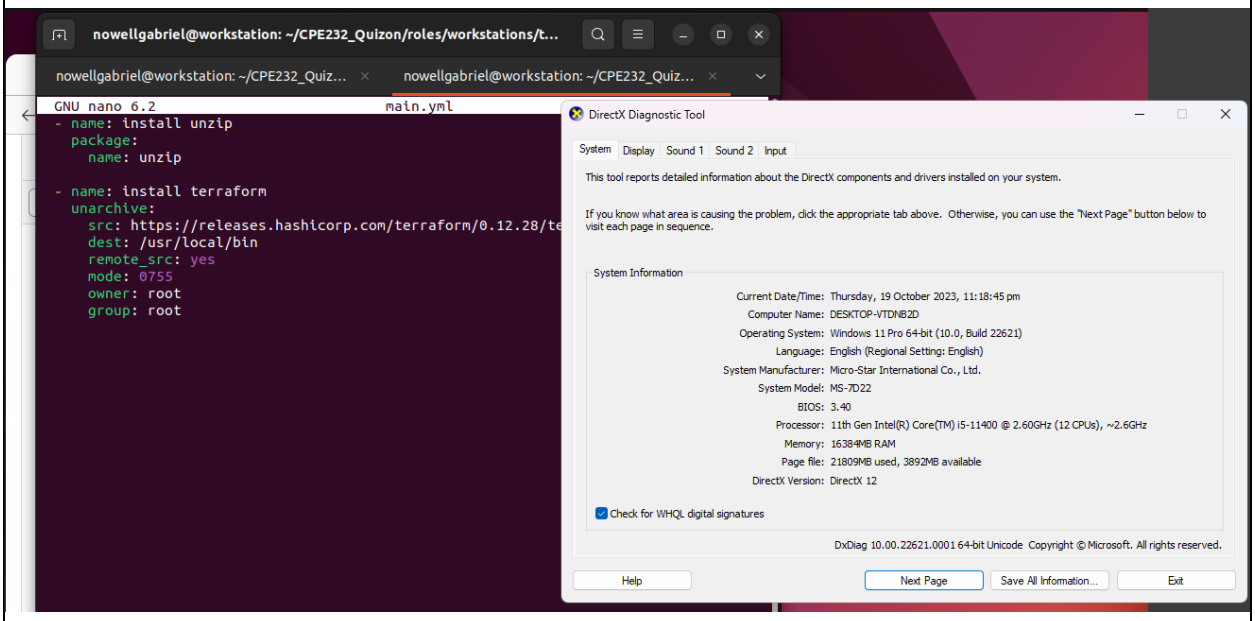
3. Go to tasks for all directory and create a file. Name it main.yml. In each of the tasks for all directories, copy and paste the code from the old site.yml file. Show all contents of main.yml files for all tasks.



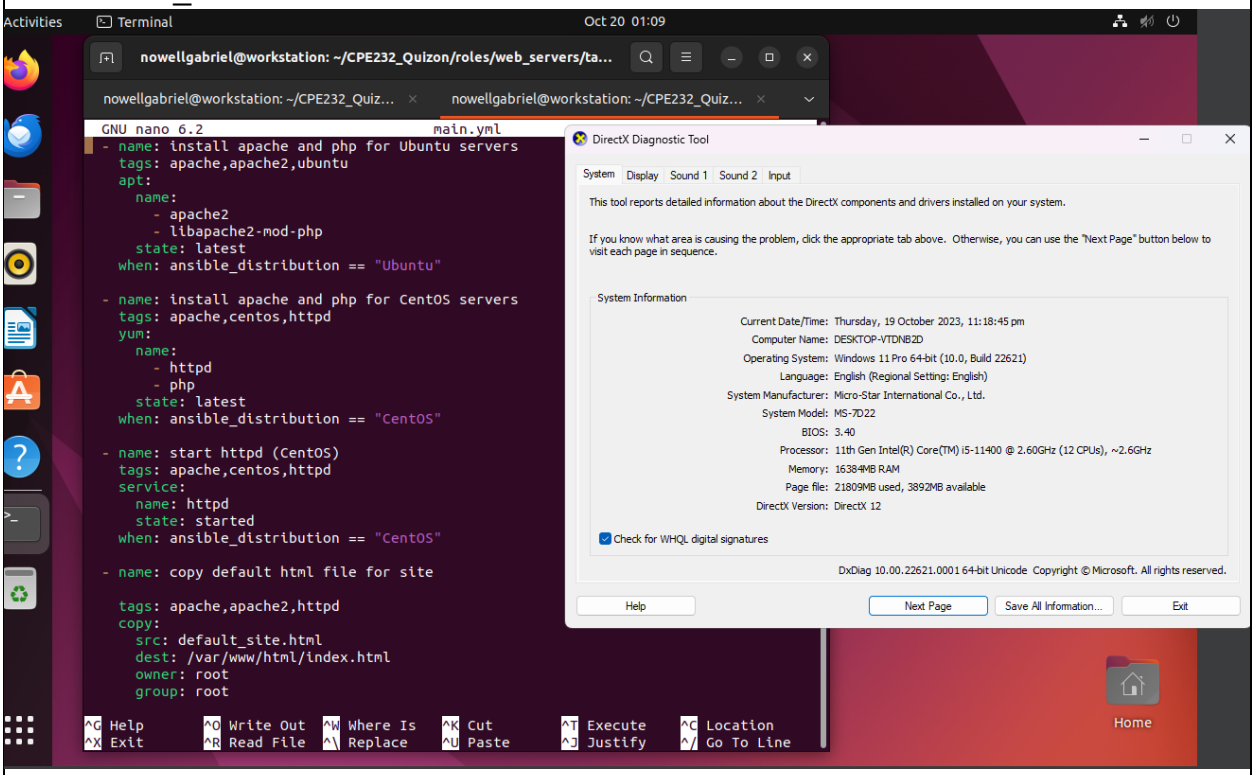
Base:



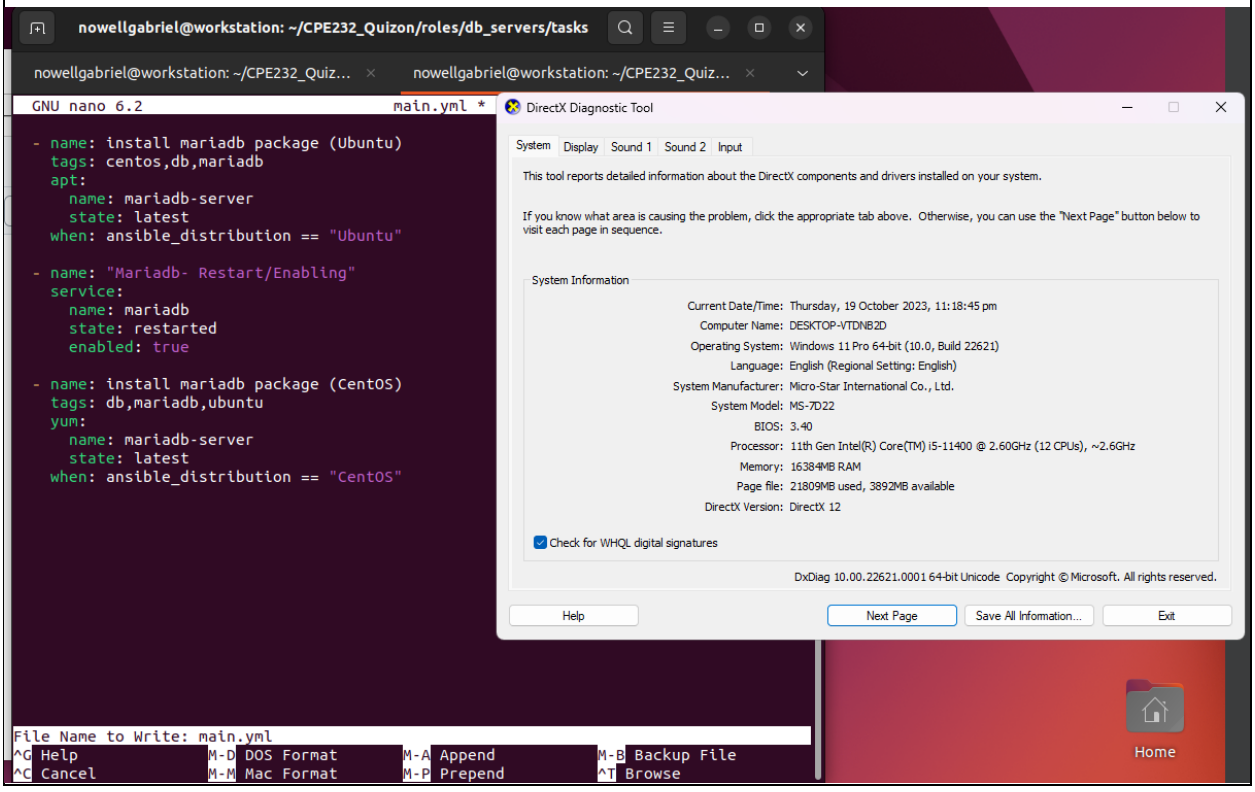
Workstations:



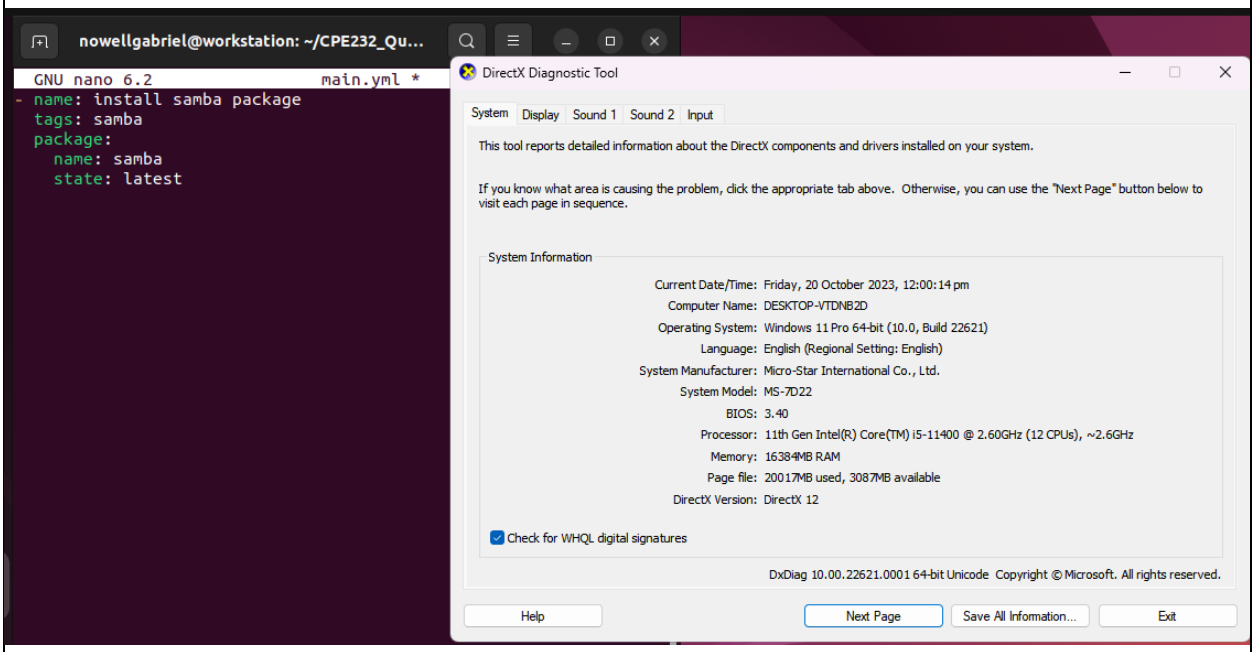
web_servers:



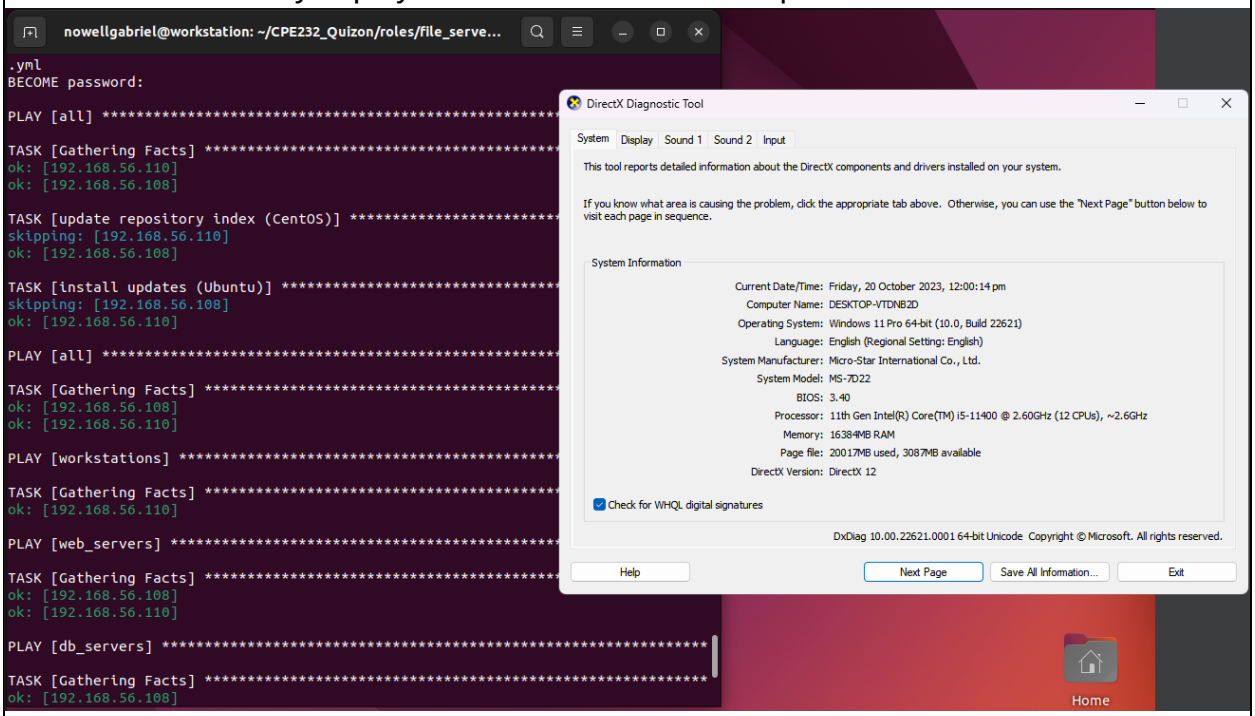
db_servers:



file_servers:



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



- From what I observed from the output, when it ran the playbook, because of the roles assigned to each group, it is not confusing anymore because it's more efficient.

Reflections:

Answer the following:

1. What is the importance of creating roles?
   The important of creating roles is to lessen the work that is being done because you only call on specific roles, so it will not do unnecessary tasks anymore compared to the previous configuration we made.
2. What is the importance of managing files?
   Managing files is vital if you want security and efficiency in organizing important data. It is easier to access and to distinguish if you manage your files properly.