

Name: Rustom C. Cariño	Date Performed: 10/13/2023
Course/Section: CPE31S5	Date Submitted:10/14/2023
Instructor: Engr. Roman Richard	Semester and SY: 1st sem / 2023-2024
Activity 6: Targeting Specific Nodes and Managing Services	
1. Objectives: 1.1 Individualize hosts 1.2 Apply tags in selecting plays to run 1.3 Managing Services from remote servers using playbooks	
2. Discussion: <p>In this activity, we try to individualize hosts. For example, we don't want apache on all our servers, or maybe only one of our servers is a web server, or maybe we have different servers like database or file servers running different things on different categories of servers and that is what we are going to take a look at in this activity.</p> <p>We also try to manage services that do not automatically run using the automations in playbook. For example, when we install web servers or httpd for CentOS, we notice that the service did not start automatically.</p> <p>Requirement: In this activity, you will need to create another Ubuntu VM and name it Server 3. Likewise, you need to activate the second adapter to a host-only adapter after the installations. Take note of the IP address of the Server 3. Make sure to use the command <i>ssh-copy-id</i> to copy the public key to Server 3. Verify if you can successfully SSH to Server 3.</p>	
Task 1: Targeting Specific Nodes 1. Create a new playbook and named it site.yml. Follow the commands as shown in the image below. Make sure to save the file and exit.	

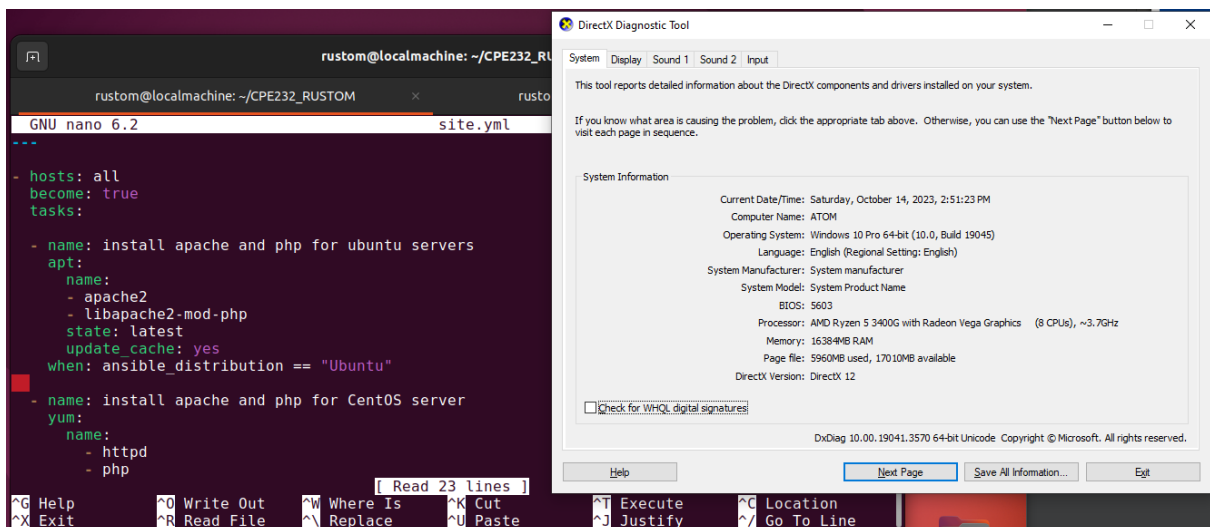
```

---
- hosts: all
  become: true
  tasks:

  - name: install apache and php for Ubuntu servers
    apt:
      name:
        - apache2
        - libapache2-mod-php
      state: latest
      update_cache: yes
      when: ansible_distribution == "Ubuntu"

  - name: install apache and php for CentOS servers
    dnf:
      name:
        - httpd
        - php
      state: latest
      when: ansible_distribution == "CentOS"

```



2. Edit the inventory file. Remove the variables we put in our last activity and group according to the image shown below:

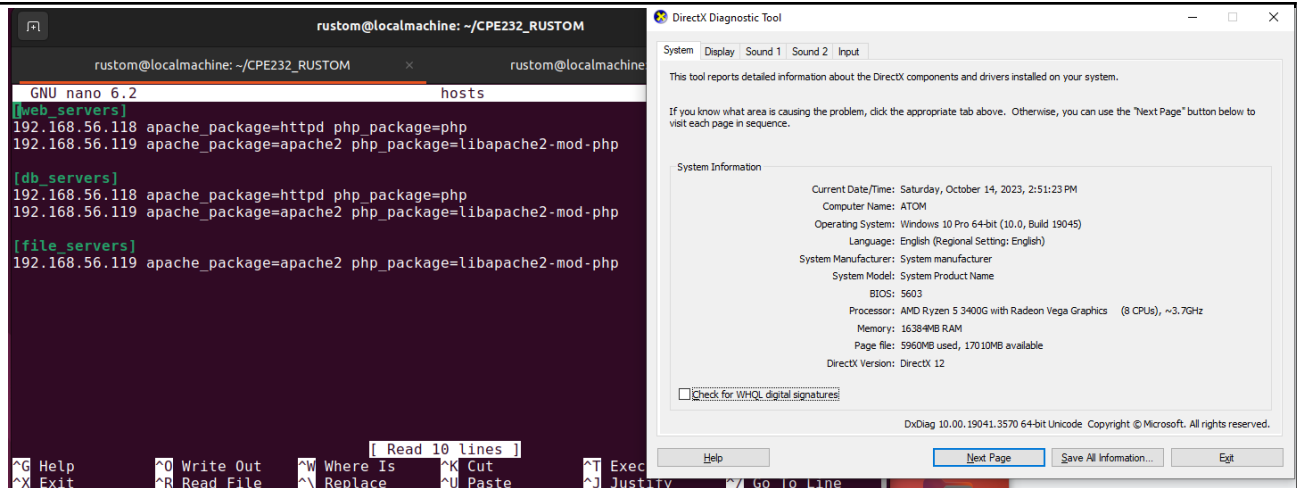
```

[web_servers]
192.168.56.120
192.168.56.121

[db_servers]
192.168.56.122

[file_servers]
192.168.56.123

```



Make sure to save the file and exit.

Right now, we have created groups in our inventory file and put each server in its own group. In other cases, you can have a server be a member of multiple groups, for example you have a test server that is also a web server.

3. Edit the *site.yml* by following the image below:

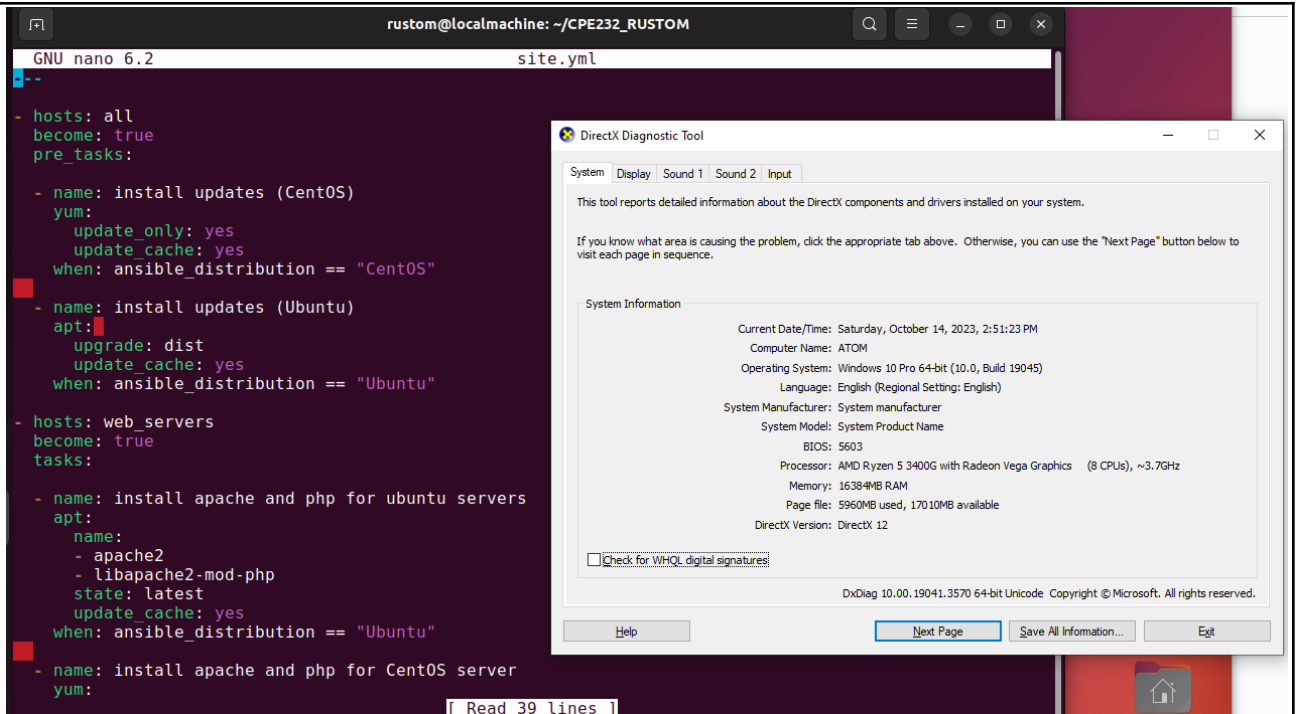
```

- -
- hosts: all
  become: true
  pre_tasks:
    - name: install updates (CentOS)
      dnf:
        update_only: yes
        update_cache: yes
        when: ansible_distribution == "CentOS"
    - name: install updates (Ubuntu)
      apt:
        upgrade: dist
        update_cache: yes
        when: ansible_distribution == "Ubuntu"

- hosts: web_servers
  become: true
  tasks:
    - name: install apache and php for Ubuntu servers
      apt:
        name:
          - apache2
          - libapache2-mod-php
        state: latest
        when: ansible_distribution == "Ubuntu"
    - name: install apache and php for CentOS servers
      dnf:
        name:
          - httpd
          - php
        state: latest
        when: ansible_distribution == "CentOS"

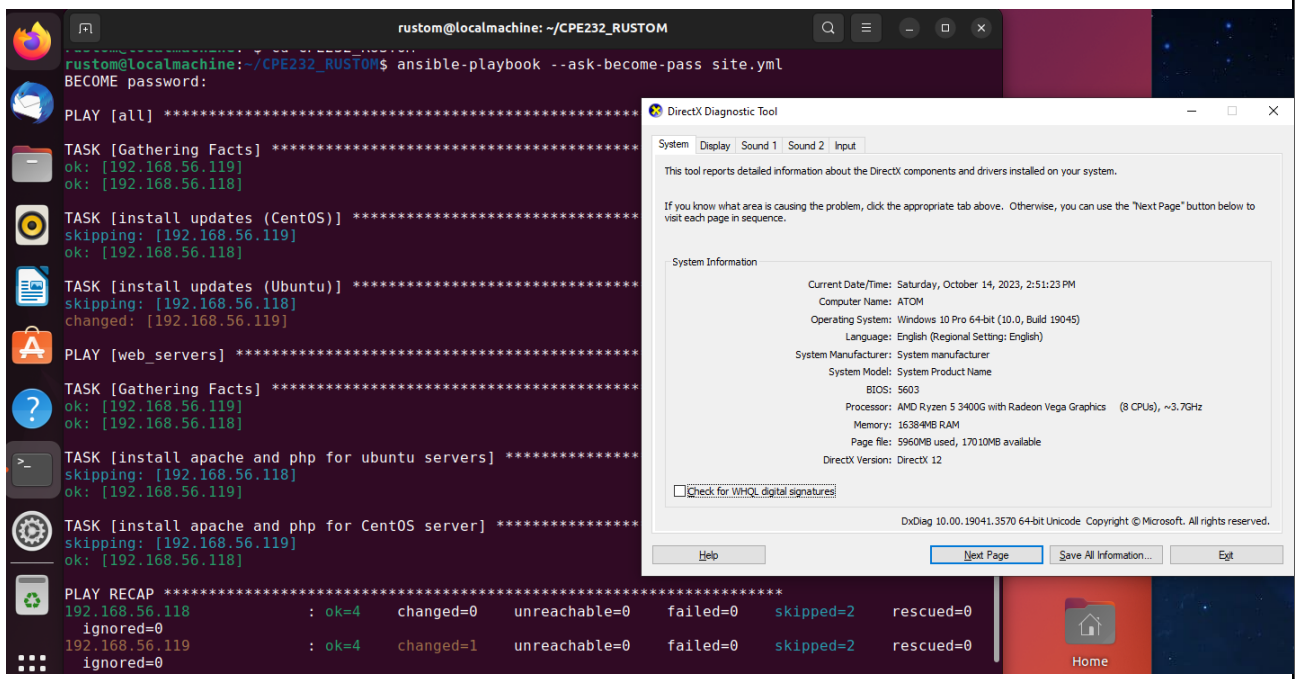
```

Make sure to save the file and exit.



The ***pre-tasks*** command tells the ansible to run it before any other thing. In the ***pre-tasks***, CentOS will install updates while Ubuntu will upgrade its distribution package. This will run before running the second play, which is targeted at ***web_servers***. In the second play, apache and php will be installed on both Ubuntu servers and CentOS servers.

Run the ***site.yml*** file and describe the result.



4. Let's try to edit again the *site.yml* file. This time, we are going to add plays targeting the other servers. This time we target the *db_servers* by adding it on the current *site.yml*. Below is an example: (Note add this at the end of the playbooks from task 1.3).

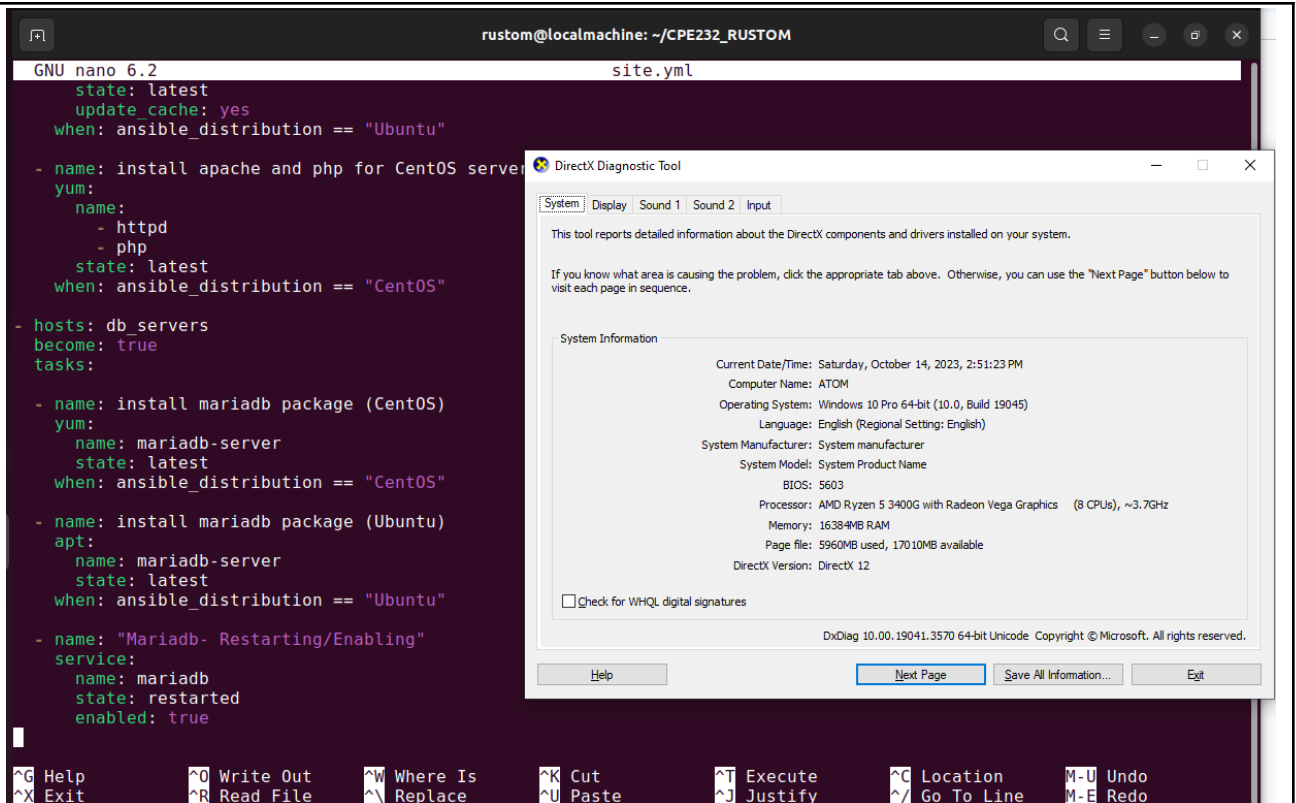
```
- hosts: db_servers
  become: true
  tasks:

    - name: install mariadb package (CentOS)
      yum:
        name: mariadb-server
        state: latest
      when: ansible_distribution == "CentOS"

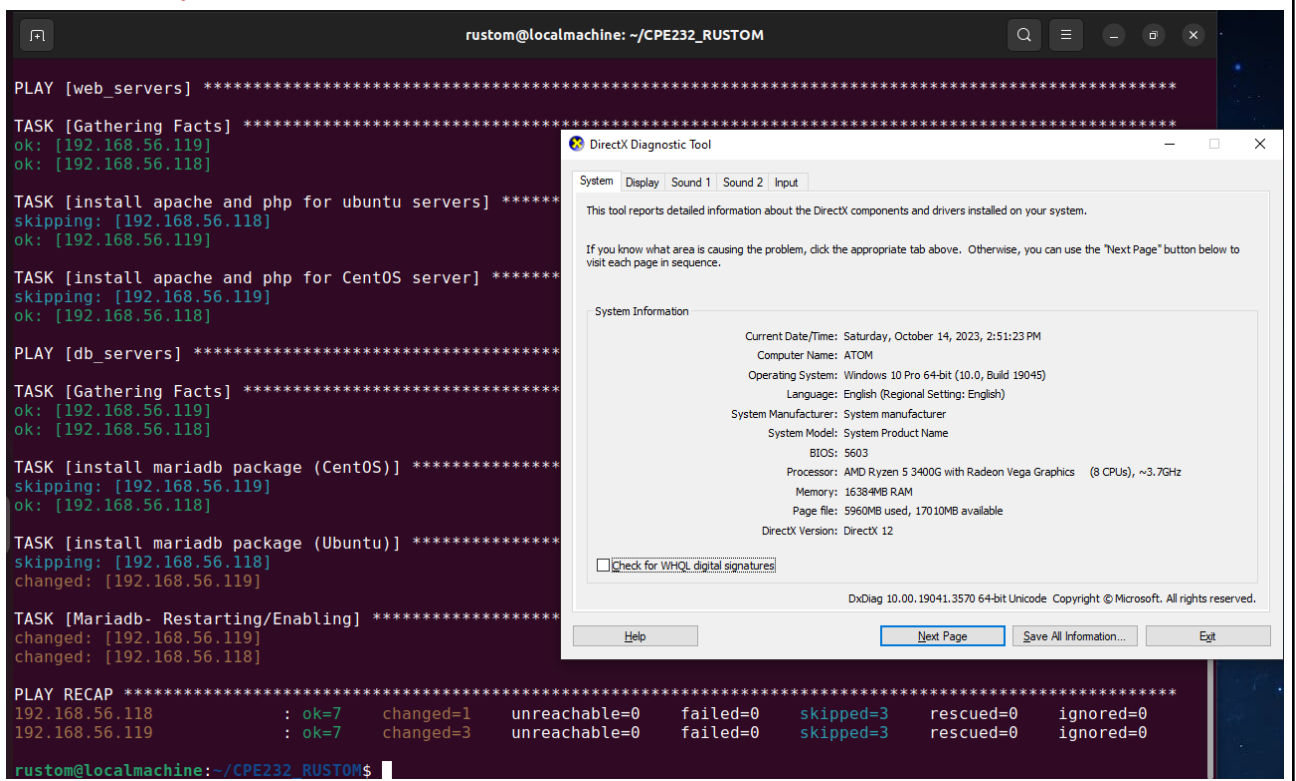
    - name: "Mariadb- Restarting/Enabling"
      service:
        name: mariadb
        state: restarted
        enabled: true

    - name: install mariadb package (Ubuntu)
      apt:
        name: mariadb-server
        state: latest
      when: ansible_distribution == "Ubuntu"
```

Make sure to save the file and exit.



Run the *site.yml* file and describe the result.



5. Go to the remote server (Ubuntu) terminal that belongs to the db_servers group and check the status for mariadb installation using the command: **systemctl status mariadb**. Do this on the CentOS server also.

The image displays two terminal windows and a DirectX Diagnostic Tool window. The top terminal window shows the output of 'systemctl status mariadb' on an Ubuntu server (rustom@server3: ~). The output indicates that the mariadb.service is active (running) and has been loaded since Saturday, October 14, 2023, at 15:28:06 CST. The bottom terminal window shows the output of 'systemctl status mariadb' on a CentOS server (rustom@centos:~). The output indicates that the mariadb.service is active (running) and has been loaded since Saturday, October 14, 2023, at 16:22:50 PST. The DirectX Diagnostic Tool window is open on the right side of the image, showing system information such as the current date/time, computer name, operating system, and hardware details.

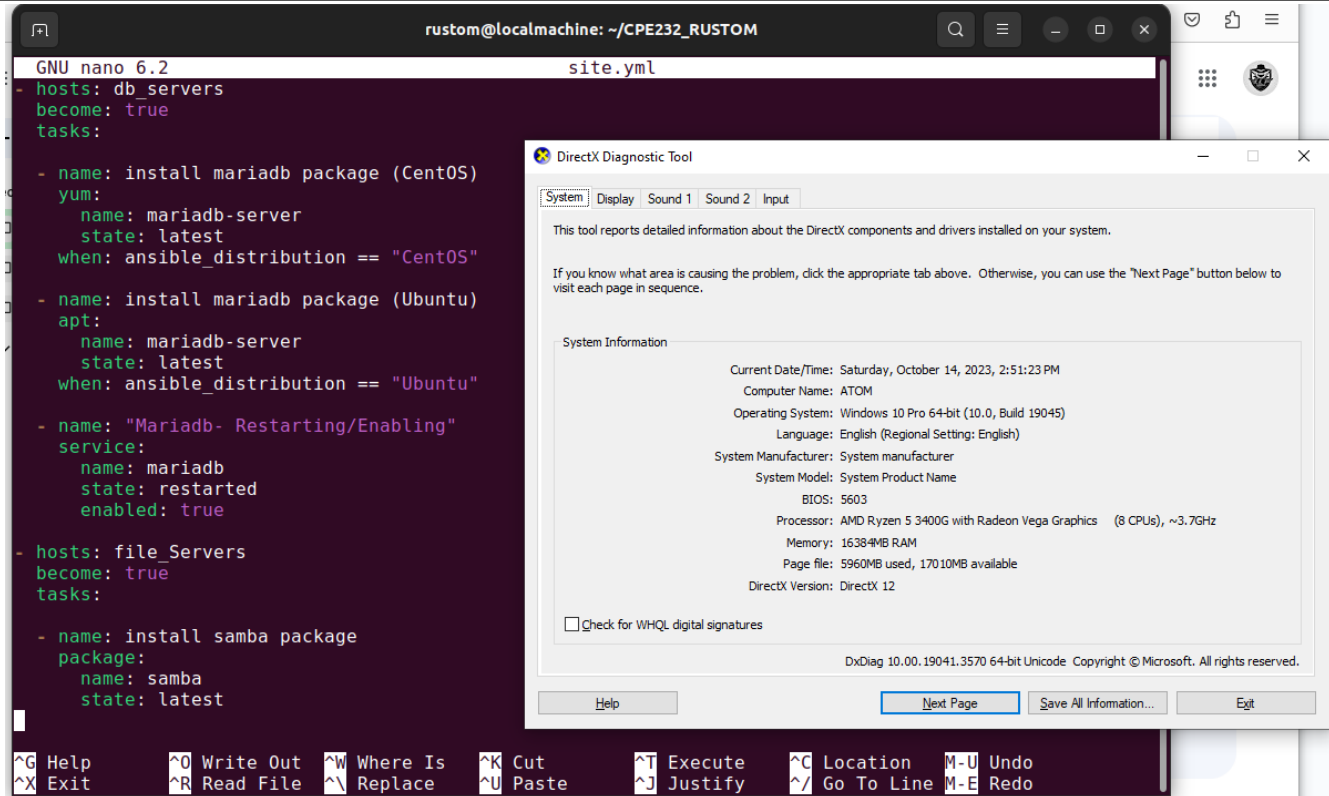
```
rustom@server3: ~  
RX packets 135 bytes 12784 (12.7 KB)  
RX errors 0 dropped 0 overruns 0 frame 0  
TX packets 135 bytes 12784 (12.7 KB)  
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
rustom@server3:~$ systemctl status mariadb  
● mariadb.service - MariaDB 10.6.12 database server  
   Loaded: loaded (/lib/systemd/system/mariadb.service; enabled; vendor preset: enabled)  
   Active: active (running) since Sat 2023-10-14 15:28:06 CST; 3min 57s ago  
     Docs: man:mariadb(8)  
           https://mariadb.com/kb/en/library/systemd/  
   Process: 34725 ExecStartPre=/usr/bin/install -m 755 -o mysql -g root -d /var/lib/mysql (code=exited, status=0/SUCCESS)  
   Process: 34726 ExecStartPre=/bin/sh -c systemctl unset-environment _WSREP_SSR (code=exited, status=0/SUCCESS)  
   Process: 34728 ExecStartPre=/bin/sh -c [ ! -e /usr/bin/galera_recovery ] && (code=exited, status=0/SUCCESS)  
   Process: 34770 ExecStartPost=/bin/sh -c systemctl unset-environment _WSREP_SSR (code=exited, status=0/SUCCESS)  
   Process: 34772 ExecStartPost=/etc/mysql/debian-start (code=exited, status=0/SUCCESS)  
 Main PID: 34757 (mariadb)  
   Status: "Taking your SQL requests now..."  
    Tasks: 9 (limit: 2261)  
  Memory: 63.5M  
     CPU: 350ms  
   CGroup: /system.slice/mariadb.service  
           └─34757 /usr/sbin/mariabdd  
lines 1-17/17 (END)
```

```
rustom@centos:~$ systemctl status mariadb  
● mariadb.service - MariaDB database server  
   Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled; vendor preset: disabled)  
   Active: active (running) since Sat 2023-09-30 16:22:50 PST; 3min 8s ago  
     Process: 17730 ExecStartPost=/usr/libexec/mariadb-wait-ready $MAINPID (code=exited, status=0/SUCCESS)  
     Process: 17694 ExecStartPre=/usr/libexec/mariadb-prepare-db-dir %n (code=exited, status=0/SUCCESS)  
 Main PID: 17729 (mysqld_safe)  
    Tasks: 20  
   CGroup: /system.slice/mariadb.service  
           └─17729 /bin/sh /usr/bin/mysqld_safe --basedir=/usr  
             └─17894 /usr/libexec/mysqld --basedir=/usr --datadir=/var/lib/mysql --plu...  
  
Sep 30 16:22:48 centos systemd[1]: Starting MariaDB database server...  
Sep 30 16:22:48 centos mariadb-prepare-db-dir[17694]: Database MariaDB is probably ...  
Sep 30 16:22:48 centos mariadb-prepare-db-dir[17694]: If this is not the case, make...  
Sep 30 16:22:48 centos mysqld_safe[17729]: 230930 16:22:48 mysqld_safe Logging to ...  
Sep 30 16:22:48 centos mysqld_safe[17729]: 230930 16:22:48 mysqld_safe Starting my...ql  
Sep 30 16:22:50 centos systemd[1]: Started MariaDB database server.  
Hint: Some lines were ellipsized, use -l to show in full.  
[rustom@centos ~]$
```

Describe the output.

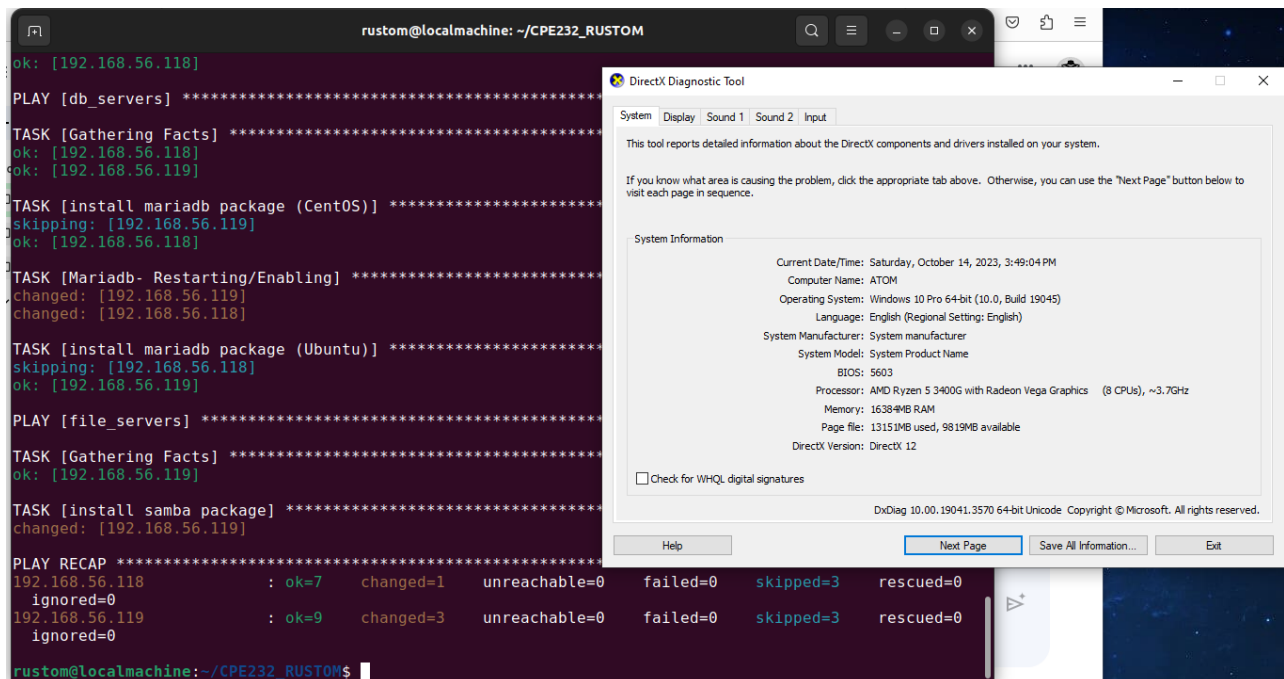
6. Edit the **site.yml** again. This time we will append the code to configure installation on the **file_servers** group. We can add the following on our file.

```
- hosts: file_servers  
  become: true  
  tasks:  
  
  - name: install samba package  
    package:  
      name: samba  
      state: latest
```

Make sure to save the file and exit.

Run the *site.yml* file and describe the result.



The testing of the *file_servers* is beyond the scope of this activity, and as well as our topics and objectives. However, in this activity we were able to show that we can target hosts or servers using grouping in ansible playbooks.

Task 2: Using Tags in running playbooks

In this task, our goal is to add metadata to our plays so that we can only run the plays that we want to run, and not all the plays in our playbook.

1. Edit the *site.yml* file. Add tags to the playbook. After the name, we can place the tags: *name_of_tag*. This is an arbitrary command, which means you can use any name for a tag.

```
---
- hosts: all
  become: true
  pre_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"
```

```
- hosts: web_servers
  become: true
  tasks:

  - name: install apache and php for Ubuntu servers
    tags: apache,apache2,ubuntu
    apt:
      name:
        - apache2
        - libapache2-mod-php
      state: latest
    when: ansible_distribution == "Ubuntu"

  - name: install apache and php for CentOS servers
    tags: apache,centos,httpd
    dnf:
      name:
        - httpd
        - php
      state: latest
    when: ansible_distribution == "CentOS"
```

```
- hosts: db_servers
  become: true
  tasks:

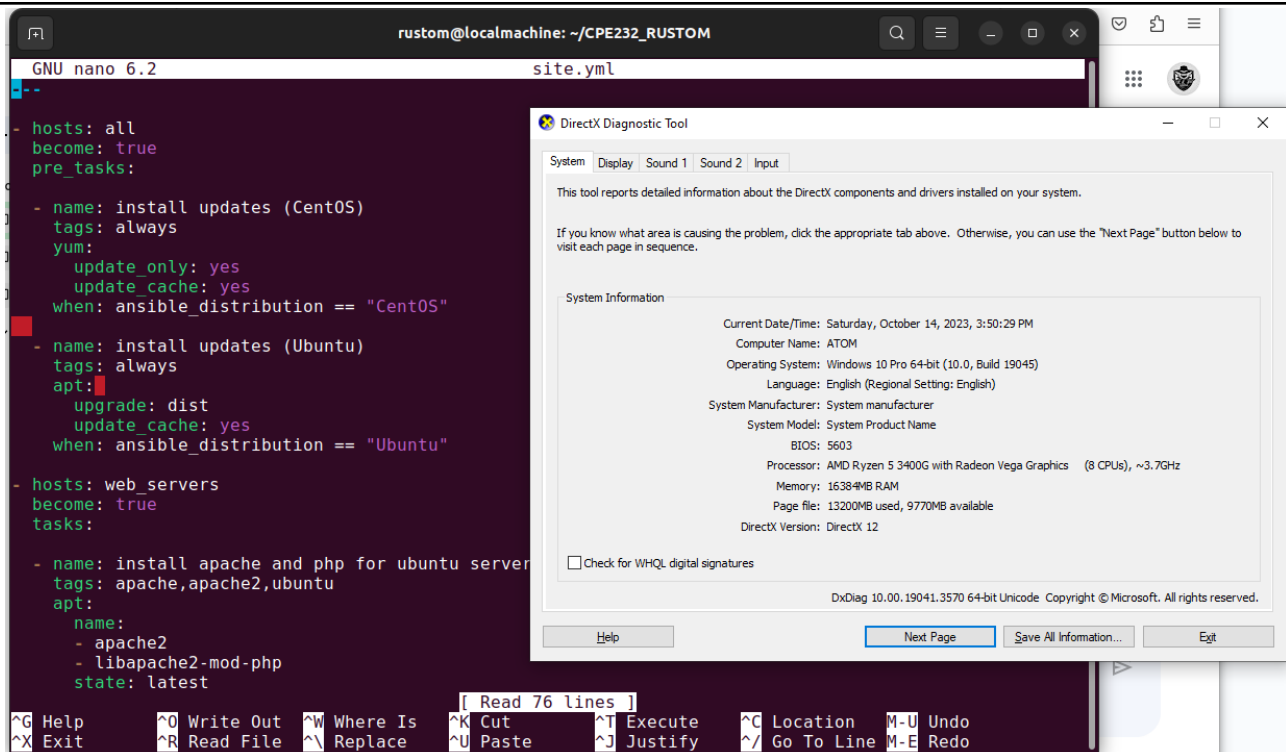
  - name: install mariadb package (CentOS)
    tags: centos, db,mariadb
    dnf:
      name: mariadb-server
      state: latest
    when: ansible_distribution == "CentOS"

  - name: "Mariadb- Restarting/Enabling"
    service:
      name: mariadb
      state: restarted
      enabled: true

  - name: install mariadb package (Ubuntu)
    tags: db, mariadb,ubuntu
    apt:
      name: mariadb-server
      state: latest
    when: ansible_distribution == "Ubuntu"

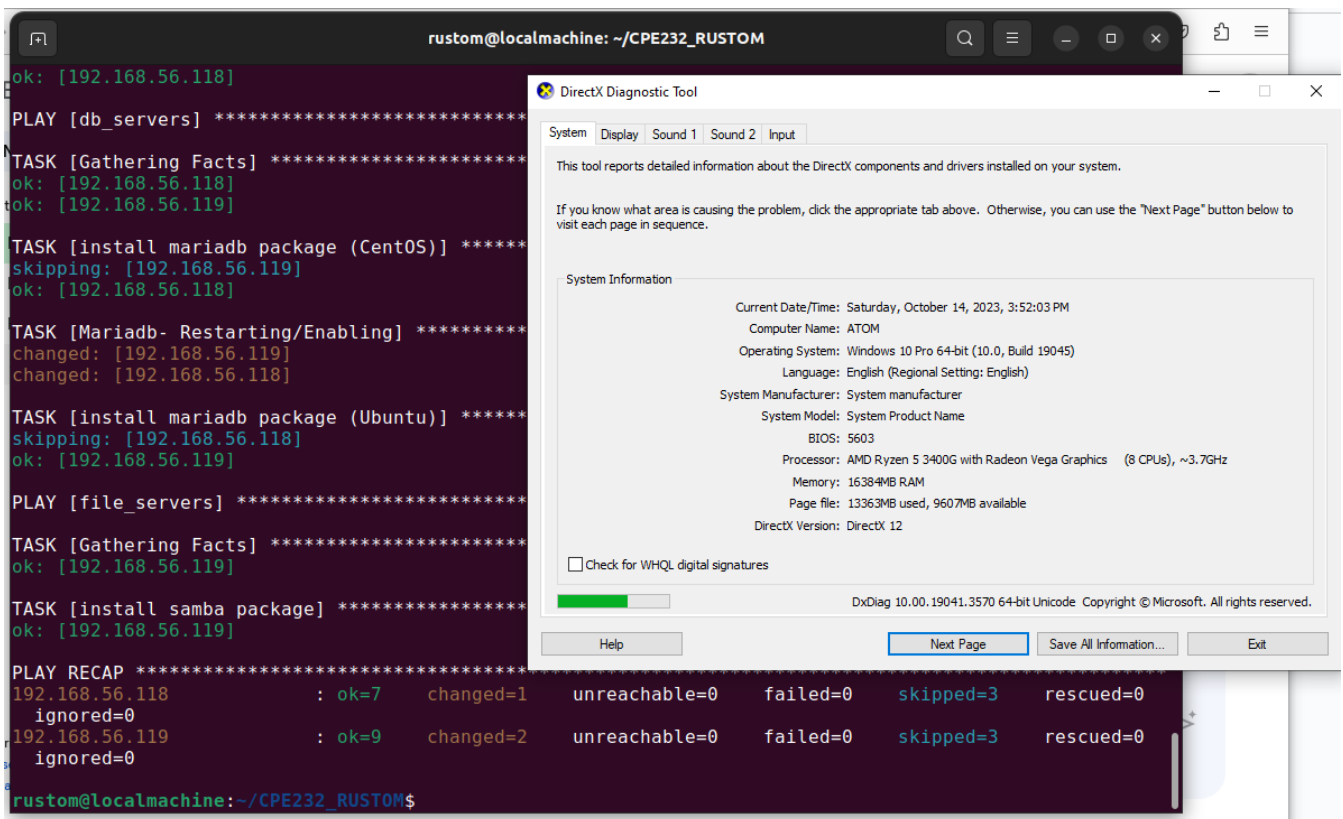
- hosts: file_servers
  become: true
  tasks:

  - name: install samba package
    tags: samba
    package:
      name: samba
      state: latest
```



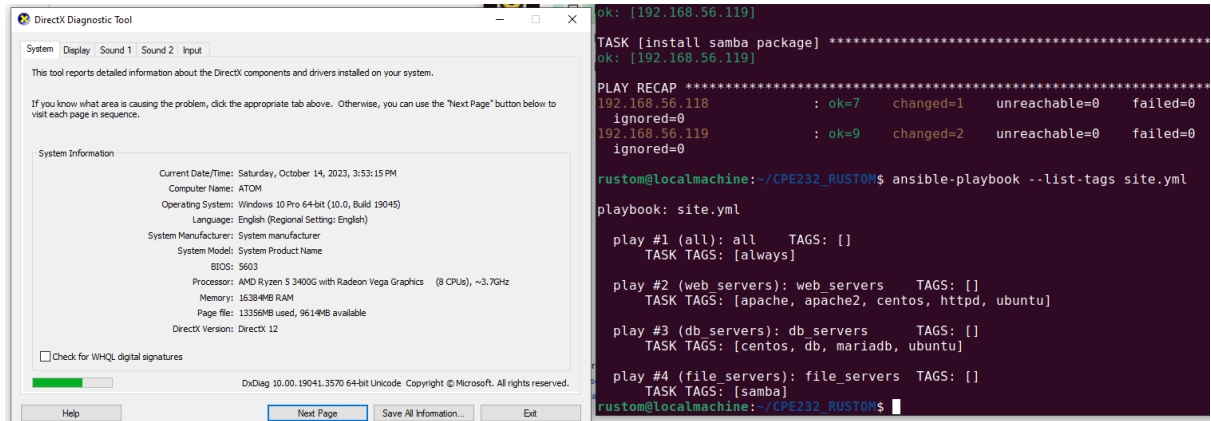
Make sure to save the file and exit.

Run the *site.yml* file and describe the result.

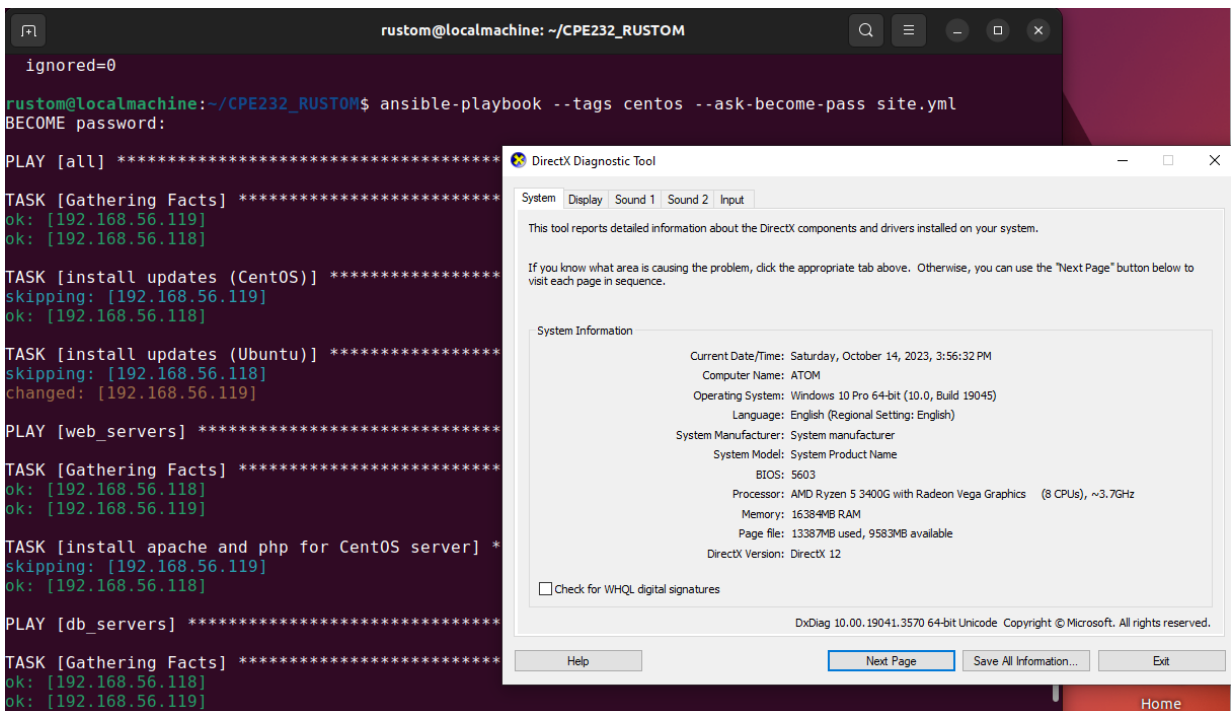


2. On the local machine, try to issue the following commands and describe each result:

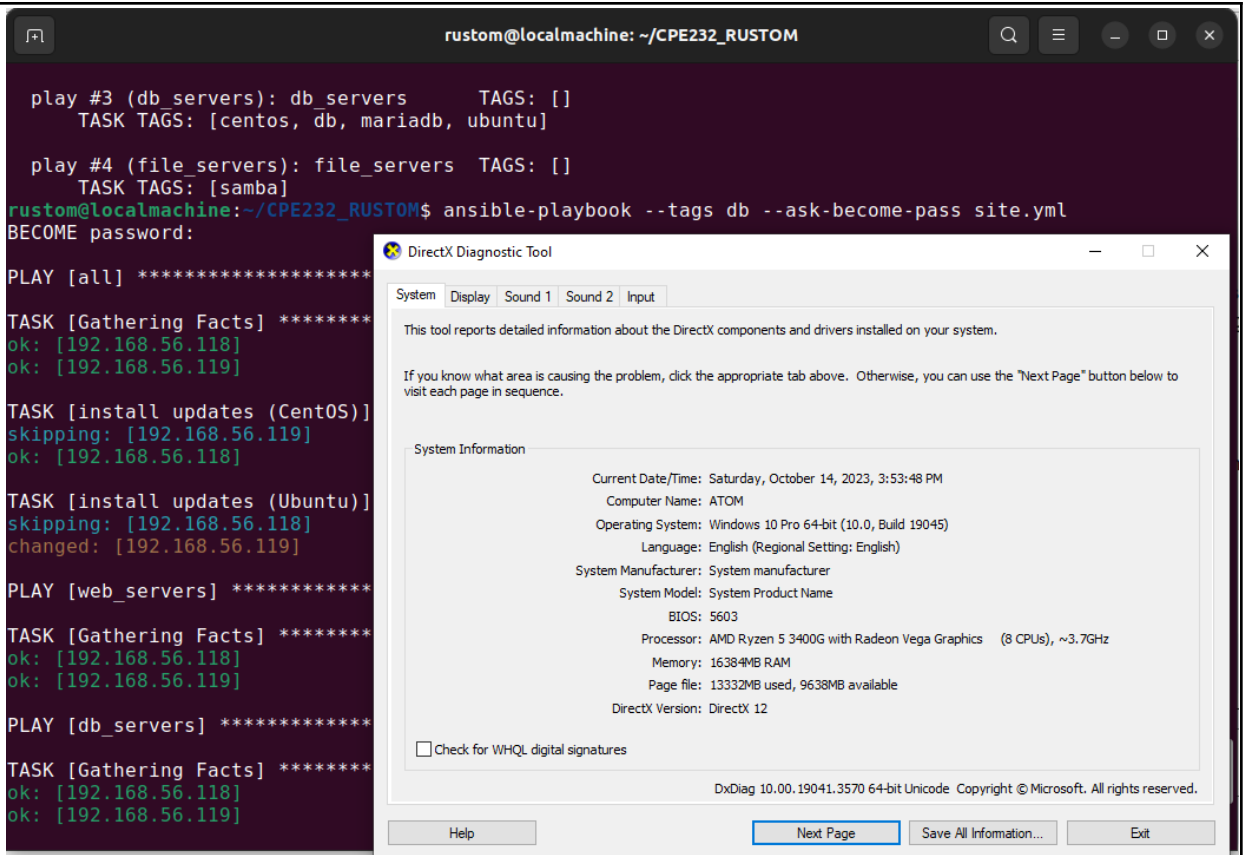
2.1 *ansible-playbook --list-tags site.yml*



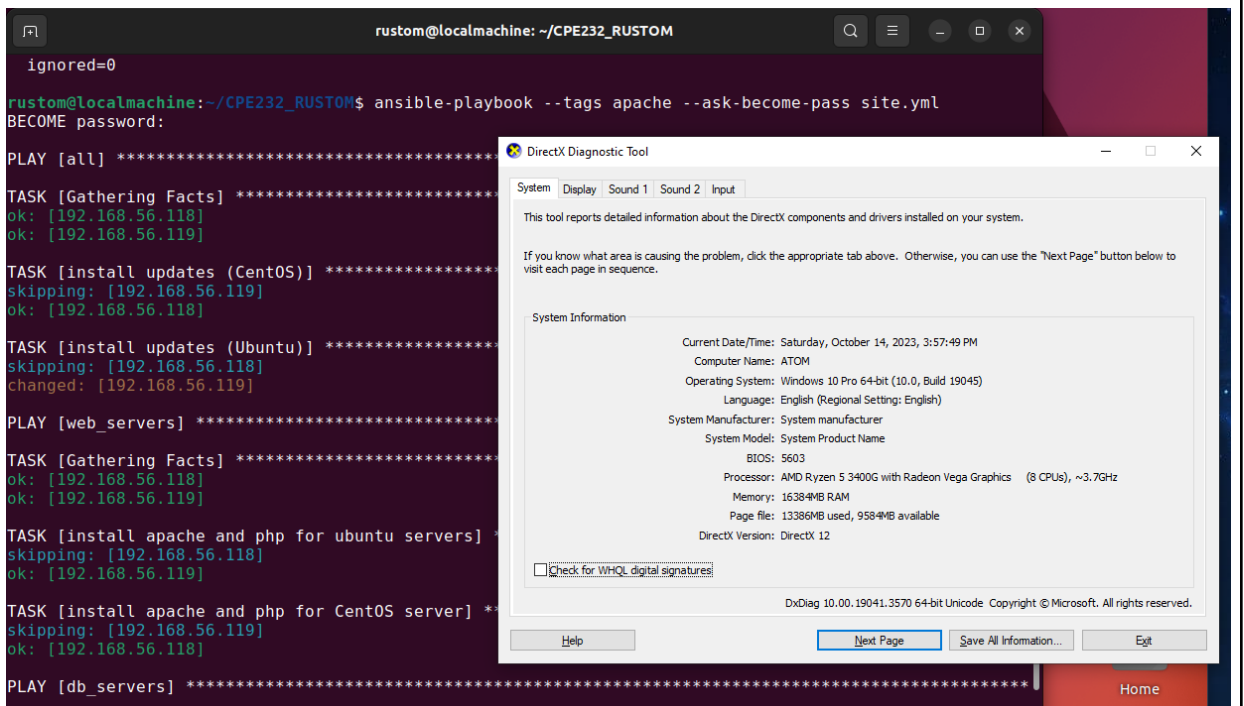
2.2 *ansible-playbook --tags centos --ask-become-pass site.yml*



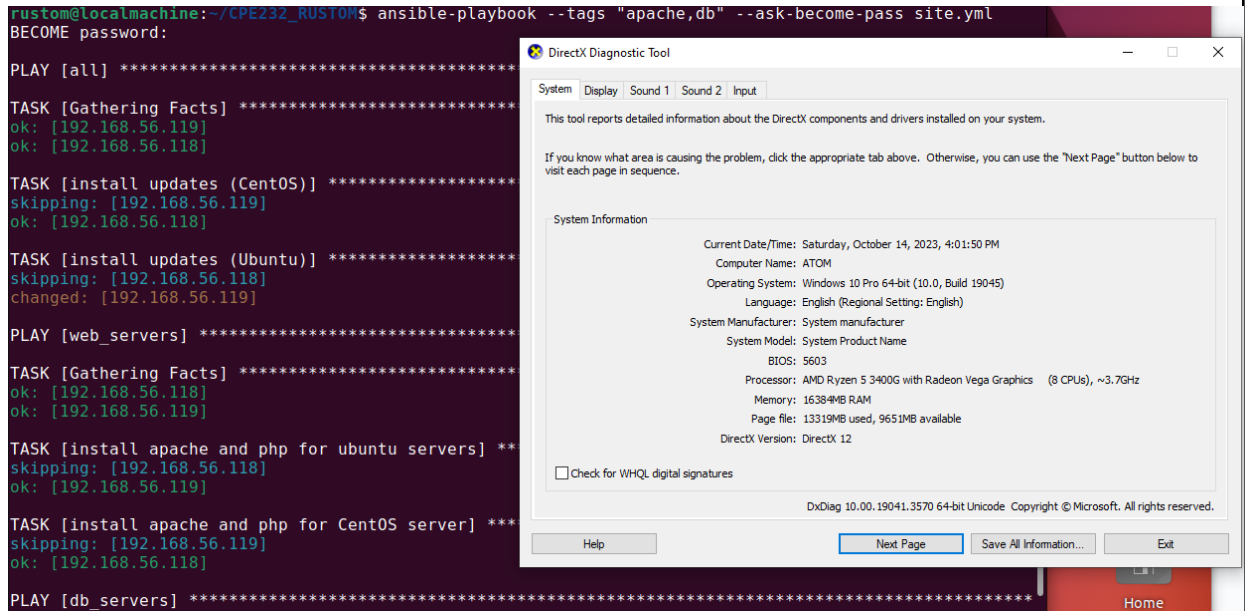
2.3 *ansible-playbook --tags db --ask-become-pass site.yml*



2.4 *ansible-playbook --tags apache --ask-become-pass site.yml*



2.5 *ansible-playbook --tags "apache,db" --ask-become-pass site.yml*



The screenshot shows a terminal window on the left and a DirectX Diagnostic Tool window on the right. The terminal window displays the execution of an Ansible playbook with the following output:

```
rustom@localmachine:~/CPE232_RUSTOM$ ansible-playbook --tags "apache,db" --ask-become-pass site.yml
BECOME password:

PLAY [all] *****

TASK [Gathering Facts] *****
ok: [192.168.56.119]
ok: [192.168.56.118]

TASK [install updates (CentOS)] *****
skipping: [192.168.56.119]
ok: [192.168.56.118]

TASK [install updates (Ubuntu)] *****
skipping: [192.168.56.118]
changed: [192.168.56.119]

PLAY [web_servers] *****

TASK [Gathering Facts] *****
ok: [192.168.56.118]
ok: [192.168.56.119]

TASK [install apache and php for ubuntu servers] **
skipping: [192.168.56.118]
ok: [192.168.56.119]

TASK [install apache and php for CentOS server] ***
skipping: [192.168.56.119]
ok: [192.168.56.118]

PLAY [db_servers] *****
```

The DirectX Diagnostic Tool window on the right shows system information:

- Current Date/Time: Saturday, October 14, 2023, 4:01:50 PM
- Computer Name: ATOM
- 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: 5603
- Processor: AMD Ryzen 5 3400G with Radeon Vega Graphics (8 CPUs), ~3.7GHz
- Memory: 16384MB RAM
- Page file: 13319MB used, 9651MB available
- DirectX Version: DirectX 12

At the bottom of the DirectX Diagnostic Tool window, there is a checkbox for "Check for WHQL digital signatures" and buttons for "Help", "Next Page", "Save All Information...", and "Exit".

Task 3: Managing Services

1. Edit the file site.yml and add a play that will automatically start the httpd on CentOS server.

```
- name: install apache and php for CentOS servers
  tags: apache,centos,httpd
  dnf:
    name:
      - httpd
      - php
    state: latest
    when: ansible_distribution == "CentOS"

- name: start httpd (CentOS)
  tags: apache, centos,httpd
  service:
    name: httpd
    state: started
    when: ansible_distribution == "CentOS"
```

Figure 3.1.1

Make sure to save the file and exit.

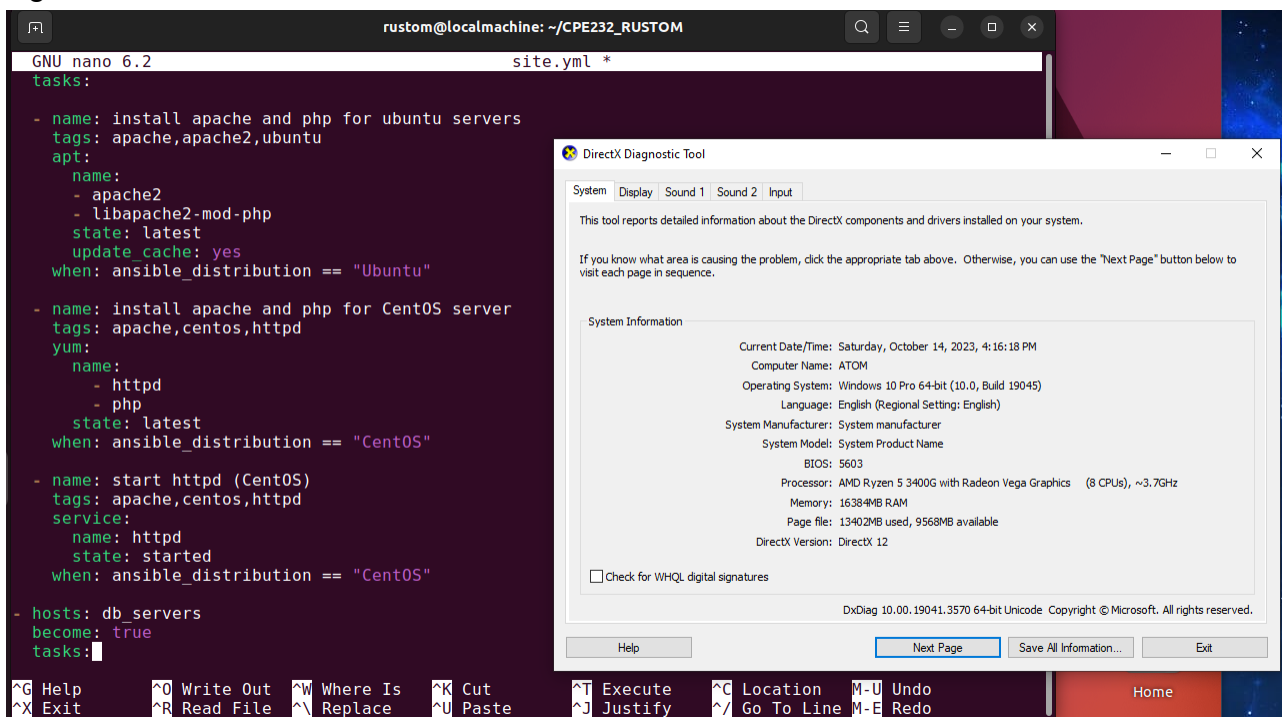
You would also notice from our previous activity that we already created a module that runs a service.

```
- hosts: db_servers
  become: true
  tasks:

  - name: install mariadb package (CentOS)
    tags: centos, db, mariadb
    dnf:
      name: mariadb-server
      state: latest
      when: ansible_distribution == "CentOS"

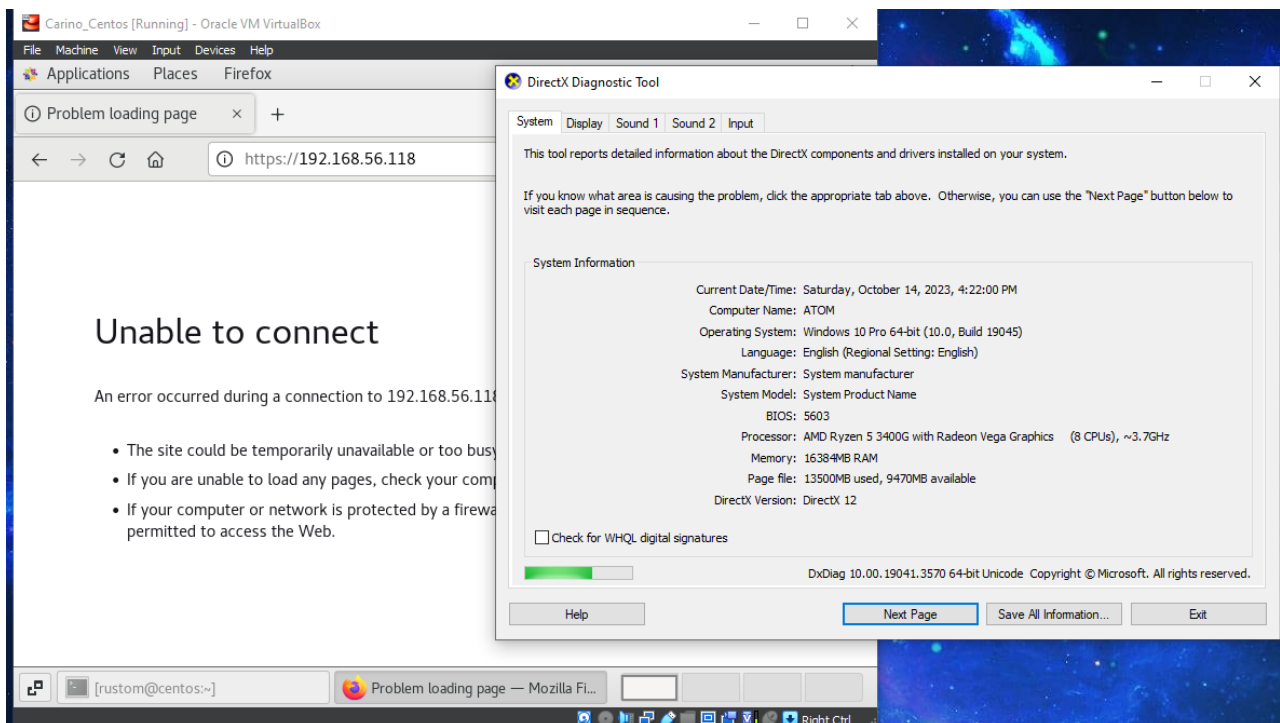
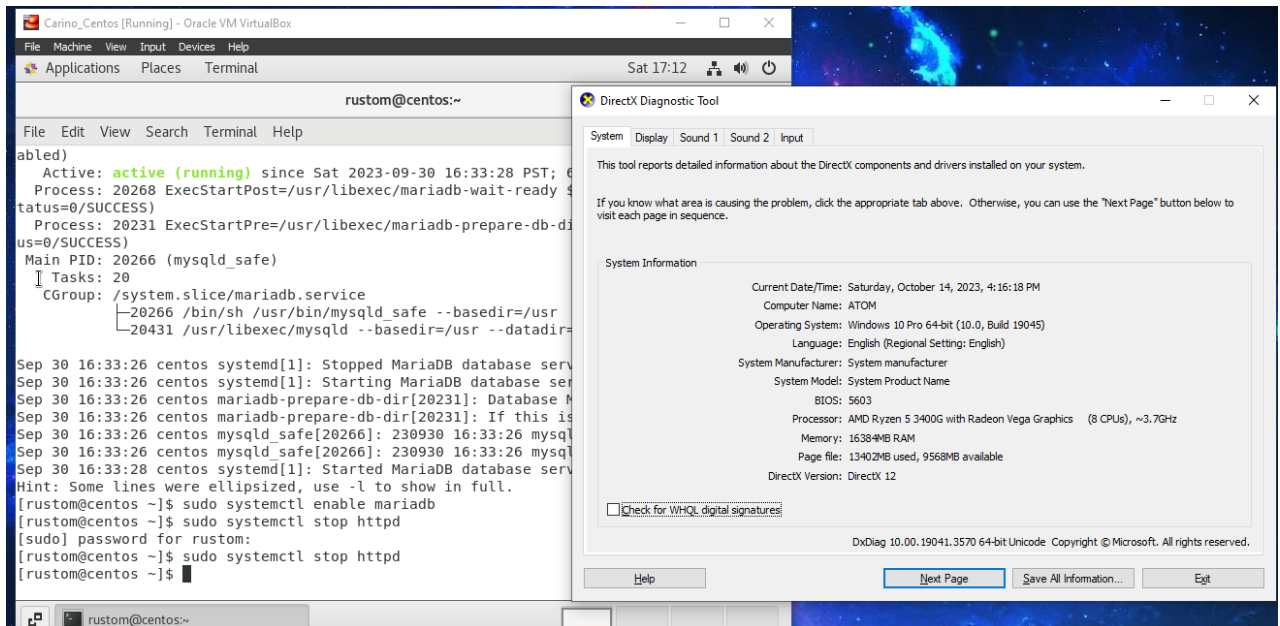
  - name: "Mariadb- Restarting/Enabling"
    service:
      name: mariadb
      state: restarted
      enabled: true
```

Figure 3.1.2

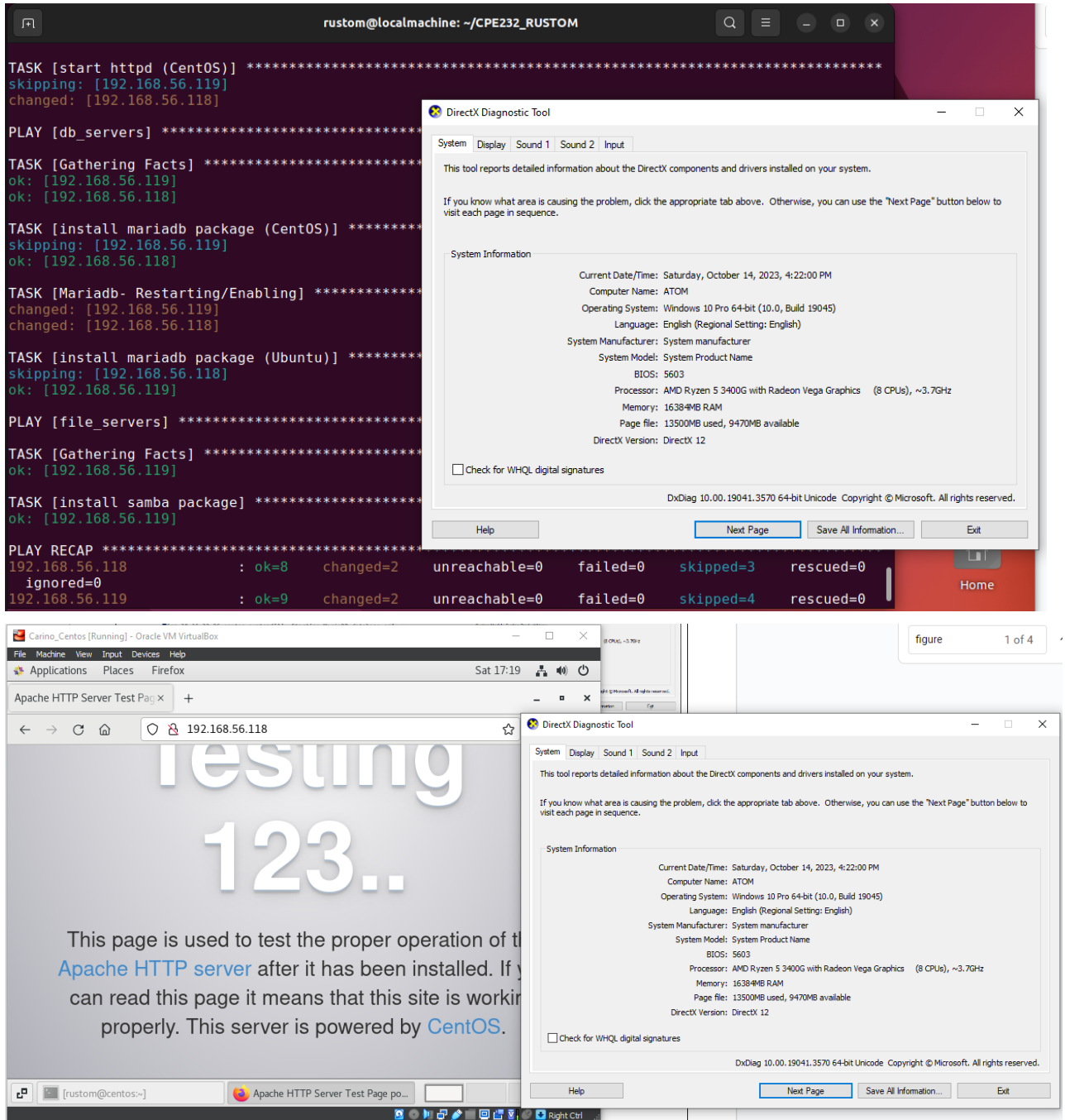


This is because in CentOS, installed packages' services are not run automatically. Thus, we need to create the module to run it automatically.

- To test it, before you run the saved playbook, go to the CentOS server and stop the currently running httpd using the command ***sudo systemctl stop httpd***. When prompted, enter the sudo password. After that, open the browser and enter the CentOS server's IP address. You should not be getting a display because we stopped the httpd service already.



3. Go to the local machine and this time, run the *site.yml* file. Then after running the file, go again to the CentOS server and enter its IP address on the browser. Describe the result. To automatically enable the service every time we run the playbook, use the command *enabled: true* similar to Figure 7.1.2 and save the playbook.



Reflections:

Answer the following:

1. What is the importance of putting our remote servers into groups?

The importance of putting the remote servers into groups because it allows for easy application of configuration changes to all servers in the group. Grouping also increases efficiency by allowing for simultaneous tasks on multiple servers using a remote management tool. Grouping also simplifies troubleshooting by comparing the configuration of the affected server to other servers in the same group, enabling quicker identification of what is the cause of the problem.

2. What is the importance of tags in playbooks?

The importance of tags in playbooks is it organizes the playbooks by function or environment making them easier to manage. They also allow for control over task execution, such as running a subset for a specific environment or all tasks for a function.

3. Why do you think some services need to be managed automatically in playbooks?

Some services need to be managed automatically in playbooks to ensure consistent configuration and management of services for security and performance. They can be executed repeatedly, saving time and effort. Playbooks reduce human error risk by automating service changes. It also improves security by automating security updates and other tasks.

Conclusion:

- After the activity I will be able to individualize the hosts and apply tags in selecting plays to run. I also manage the services from servers using the playbook. In this activity I reuse my old repositories since it already has what I need such as the hosts, ansible.cfg and the inventory, furthermore I encountered an error but I troubleshoot the problem at ease since I encounter the error since module 4 in this course and If there is a new error that i didn't encounter, I just search the solution for reference.

