

<b>Name: Pastrana, Mark Laurenz S.</b>	<b>Date Performed: Oct 14, 2023</b>
<b>Course/Section: CPE 232 - CPE31S5</b>	<b>Date Submitted: Oct 17, 2023</b>
<b>Instructor: Engr. Richard Roman</b>	<b>Semester and SY:</b>
<b>Activity 6: Targeting Specific Nodes and Managing Services</b>	
<b>1. Objectives:</b> 1.1 Individualize hosts 1.2 Apply tags in selecting plays to run 1.3 Managing Services from remote servers using playbooks	
<b>2. Discussion:</b>  <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><b>Requirement:</b>  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>	
<b>Task 1: Targeting Specific Nodes</b>	
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.	

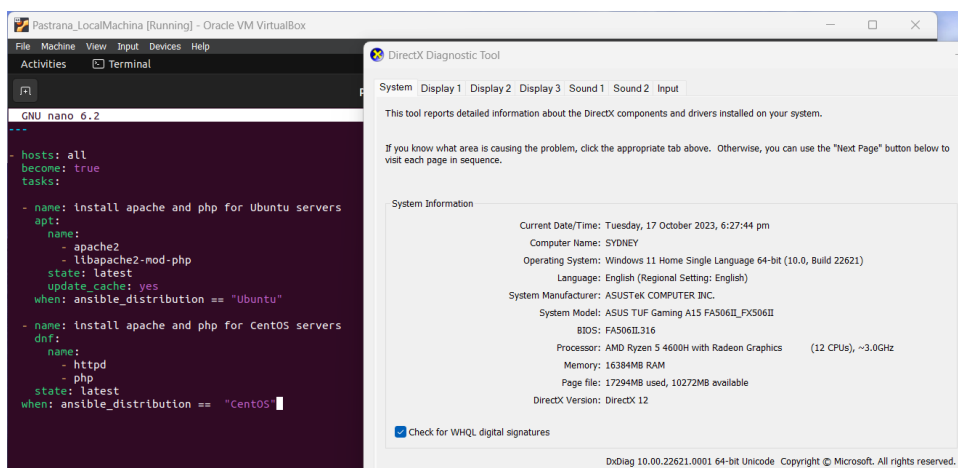
```

- 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

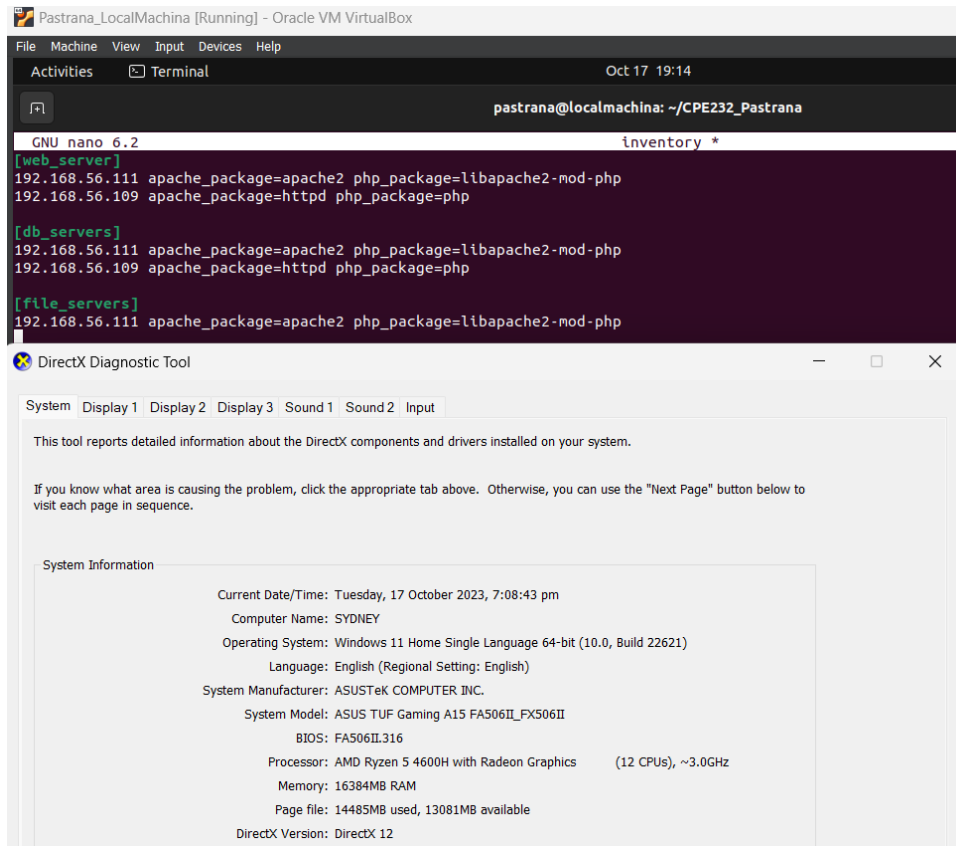
[db_servers]
192.168.56.122

[file_servers]
192.168.56.123

```

**CentOS = 192.168.56.109**

**Server3 = 192.168.56.112**



The screenshot shows a VirtualBox window titled "Pastrana\_LocalMachina [Running] - Oracle VM VirtualBox". Inside the window, there are two overlapping windows. The top window is a terminal window titled "pastrana@localmachina: ~/CPE232\_Pastrana" showing the output of a command in nano 6.2. The bottom window is a "DirectX Diagnostic Tool" window showing system information.

```
GNU nano 6.2 inventory *
[web_server]
192.168.56.111 apache_package=apache2 php_package=libapache2-mod-php
192.168.56.109 apache_package=httpd php_package=php

[db_servers]
192.168.56.111 apache_package=apache2 php_package=libapache2-mod-php
192.168.56.109 apache_package=httpd php_package=php

[file_servers]
192.168.56.111 apache_package=apache2 php_package=libapache2-mod-php
```

The DirectX Diagnostic Tool window shows the following system information:

- Current Date/Time: Tuesday, 17 October 2023, 7:08:43 pm
- Computer Name: SYDNEY
- Operating System: Windows 11 Home Single Language 64-bit (10.0, Build 22H2)
- Language: English (Regional Setting: English)
- System Manufacturer: ASUSTeK COMPUTER INC.
- System Model: ASUS TUF Gaming A15 FA506II FX506II
- BIOS: FA506II.316
- Processor: AMD Ryzen 5 4600H with Radeon Graphics (12 CPUs), ~3.0GHz
- Memory: 16384MB RAM
- Page file: 14485MB used, 13081MB available
- DirectX Version: DirectX 12

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"

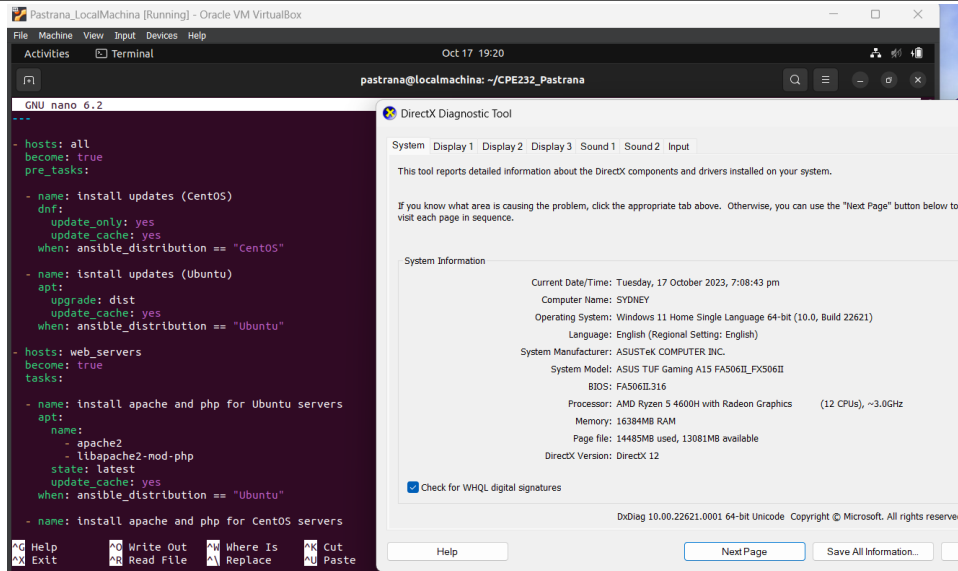
    - name: install apache and php for CentOS servers
      dnf:
        name:
          - httpd
          - php
        state: latest
        when: ansible_distribution == "CentOS"

```

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.

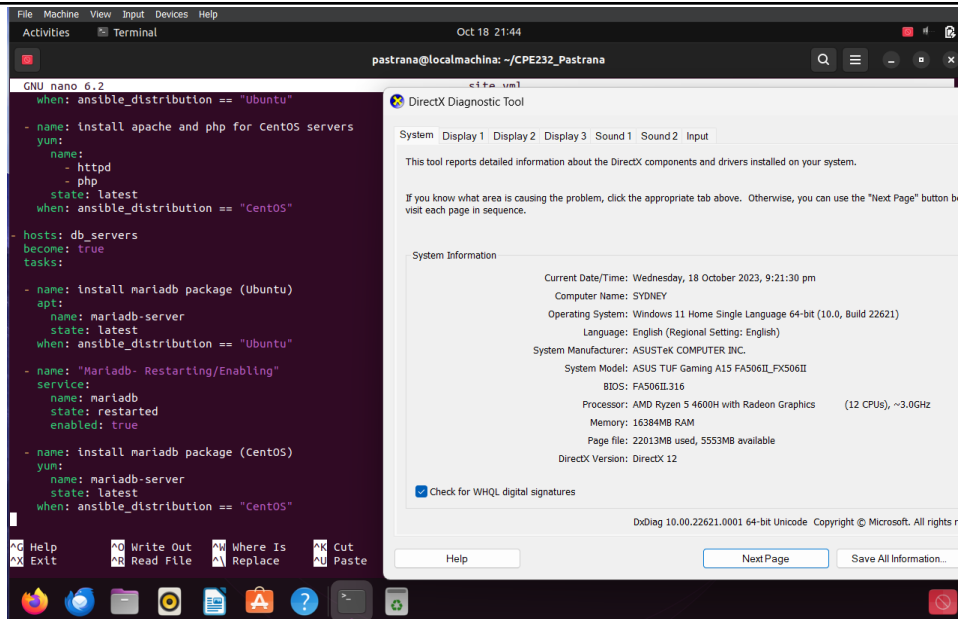


4. Let's try to edit again the *site.yml* file. This time, we are going to add plays targeting the other servers. This time we target the *db\_servers* by adding it on the current *site.yml*. Below is an example: (Note add this at the end of the playbooks from task 1.3.

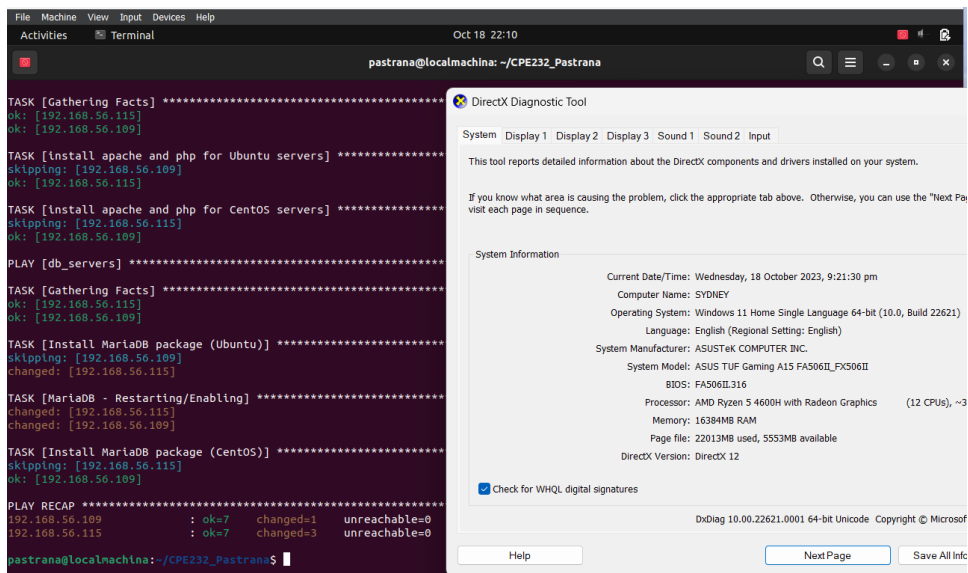
```

- 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"

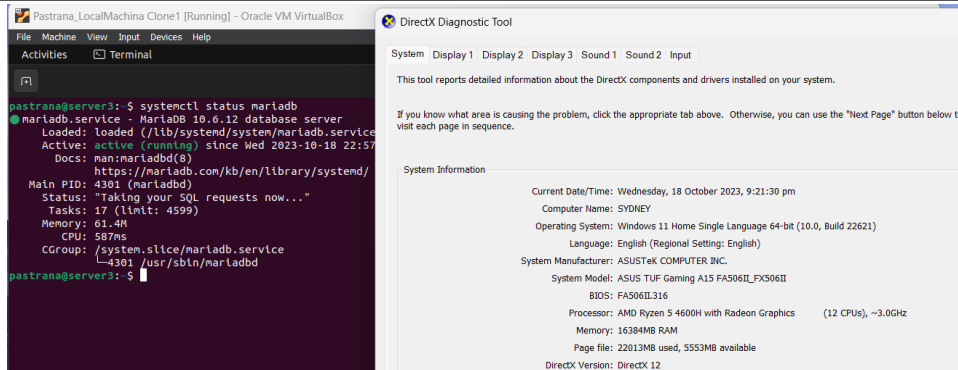
```



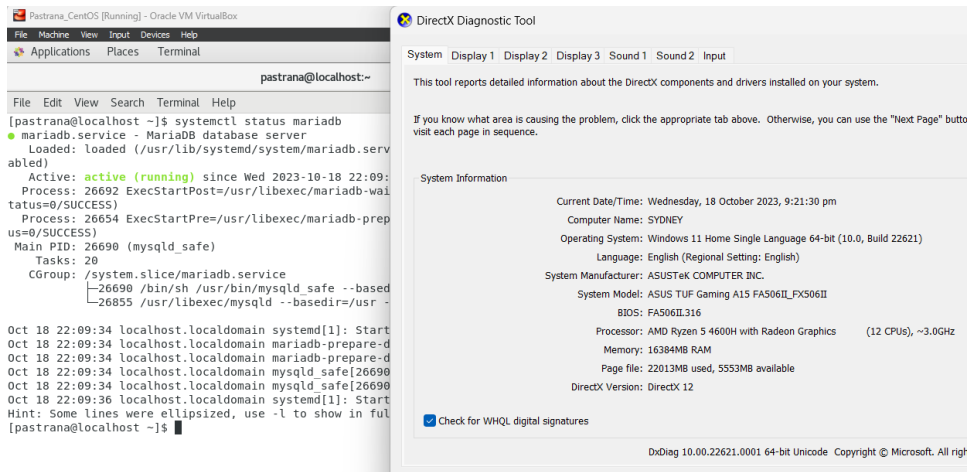
Make sure to save the file and exit.  
Run the *site.yml* file and describe the result.



- 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.  
**Remote server:**

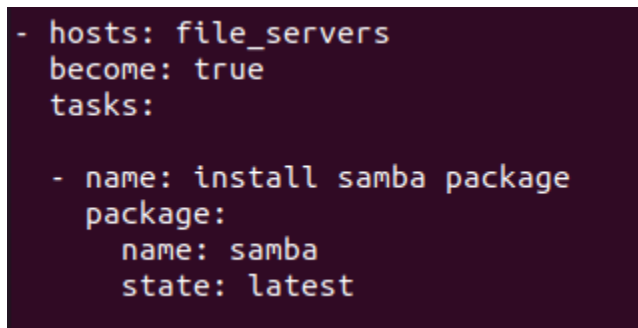


## CentOS:



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
```

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: Wednesday, 18 October 2023, 9:21:30 pm  
Computer Name: SYDNEY  
Operating System: Windows 11 Home Single Language 64-bit (10.0, Build 22621)  
Language: English (Regional Setting: English)  
System Manufacturer: ASUSTeK COMPUTER INC.  
System Model: ASUS TUF Gaming A15 FA506II-FX506II  
BIOS: FA506II.316  
Processor: AMD Ryzen 5 4600H with Radeon Graphics (12 CPUs), ~3.0GHz  
Memory: 16384MB RAM  
Page file: 22013MB used, 5553MB available  
DirectX Version: DirectX 12

☒ Check for WHQL digital signatures

Make sure to save the file and exit.  
Run the *site.yml* file and describe the result.

```
PLAY [file_servers] *****
TASK [Gathering Facts] *****
ok: [192.168.56.115]
TASK [install samba package] *****
changed: [192.168.56.115]
PLAY RECAP *****
192.168.56.109 : ok=7
192.168.56.115 : ok=9
pastrana@localmachina: ~/CPE232_Past
```

### System Information

Current Date/Time: Wednesday, 18 October 2023, 9:21:30 pm  
Computer Name: SYDNEY  
Operating System: Windows 11 Home Single Language 64-bit (10.0, Build 22621)  
Language: English (Regional Setting: English)  
System Manufacturer: ASUSTeK COMPUTER INC.  
System Model: ASUS TUF Gaming A15 FA506II-FX506II  
BIOS: FA506II.316  
Processor: AMD Ryzen 5 4600H with Radeon Graphics (12 CPUs), ~3.0GHz  
Memory: 16384MB RAM  
Page file: 22013MB used, 5553MB available  
DirectX Version: DirectX 12

☒ Check for WHQL digital signatures

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:

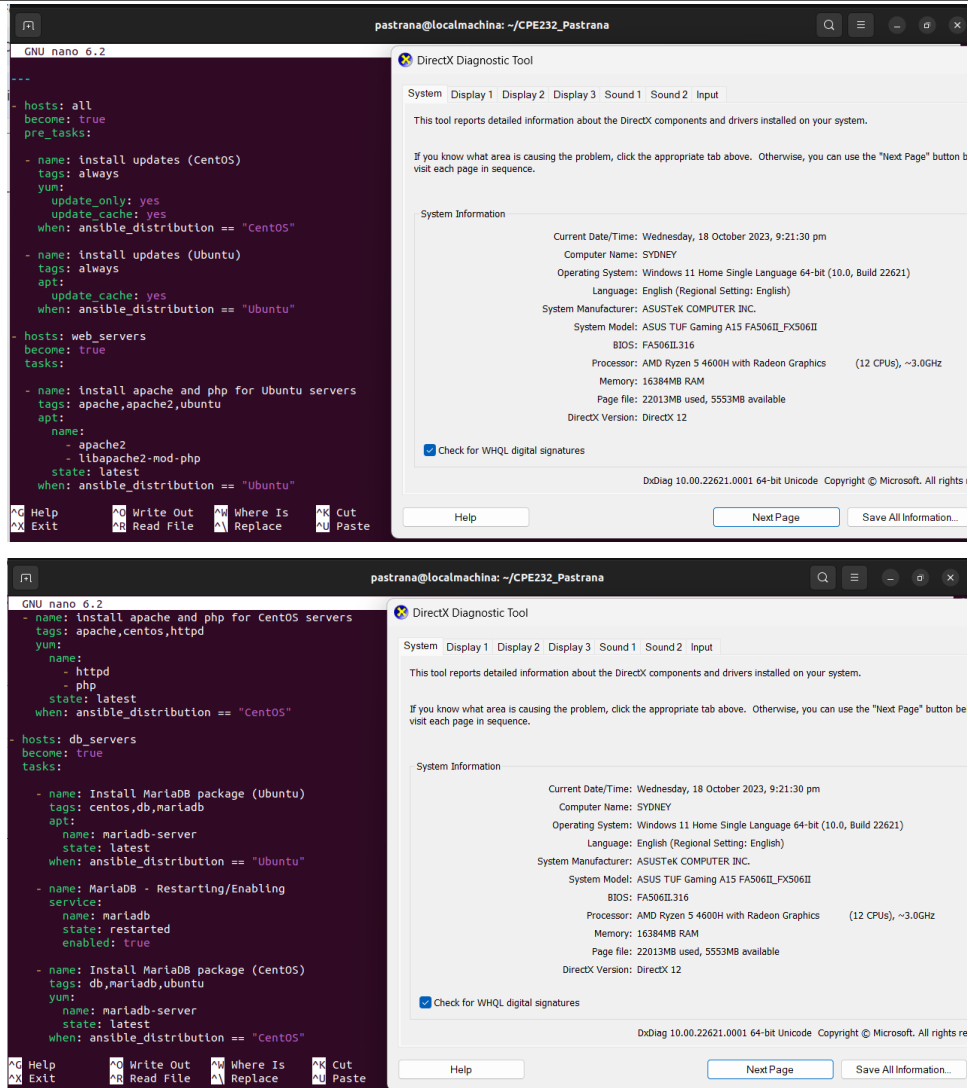
    - name: install mariadb package (CentOS)
      tags: centos, db, mariadb
      dnf:
        name: mariadb-server
        state: latest
      when: ansible_distribution == "CentOS"

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

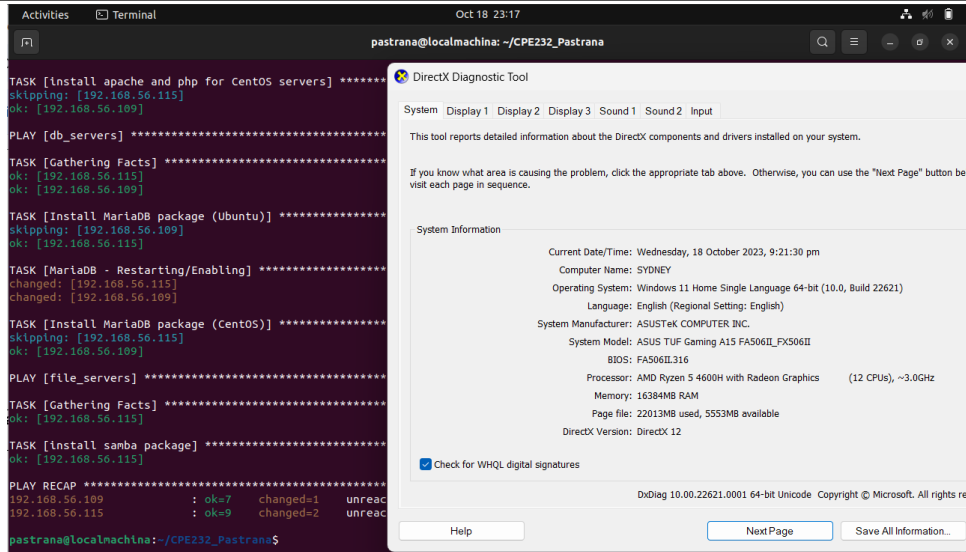
    - name: install mariadb package (Ubuntu)
      tags: db, mariadb, ubuntu
      apt:
        name: mariadb-server
        state: latest
      when: ansible_distribution == "Ubuntu"

- hosts: file_servers
  become: true
  tasks:

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

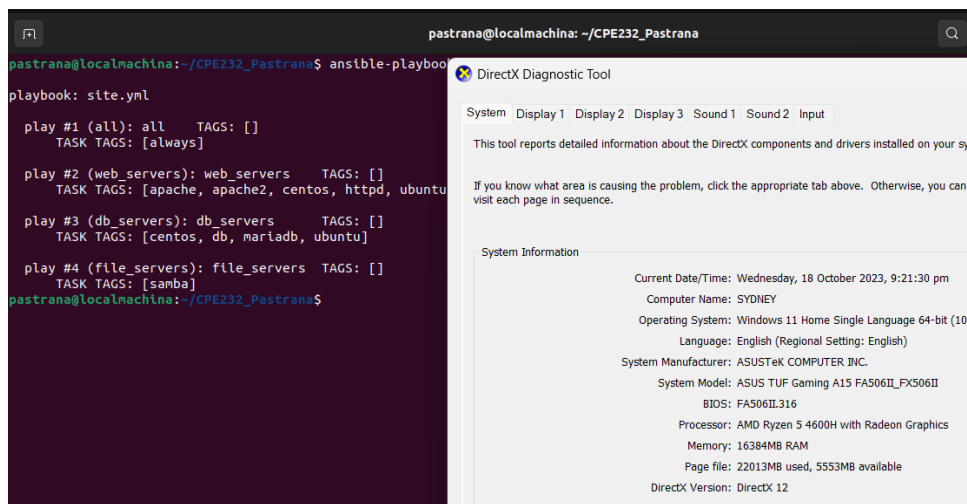


Make sure to save the file and exit.  
Run the *site.yml* file and describe the result.

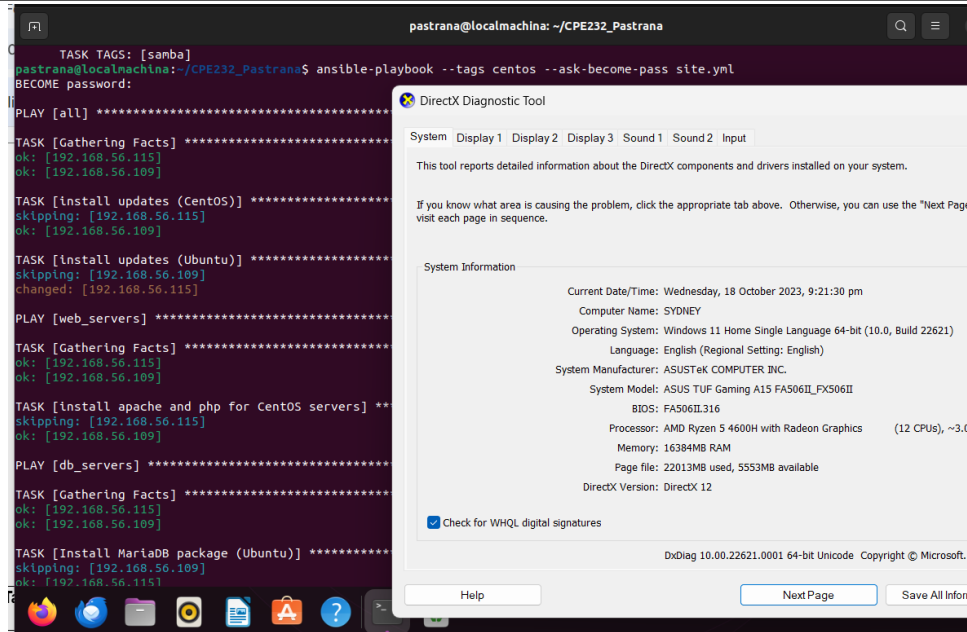


2. On the local machine, try to issue the following commands and describe each result:

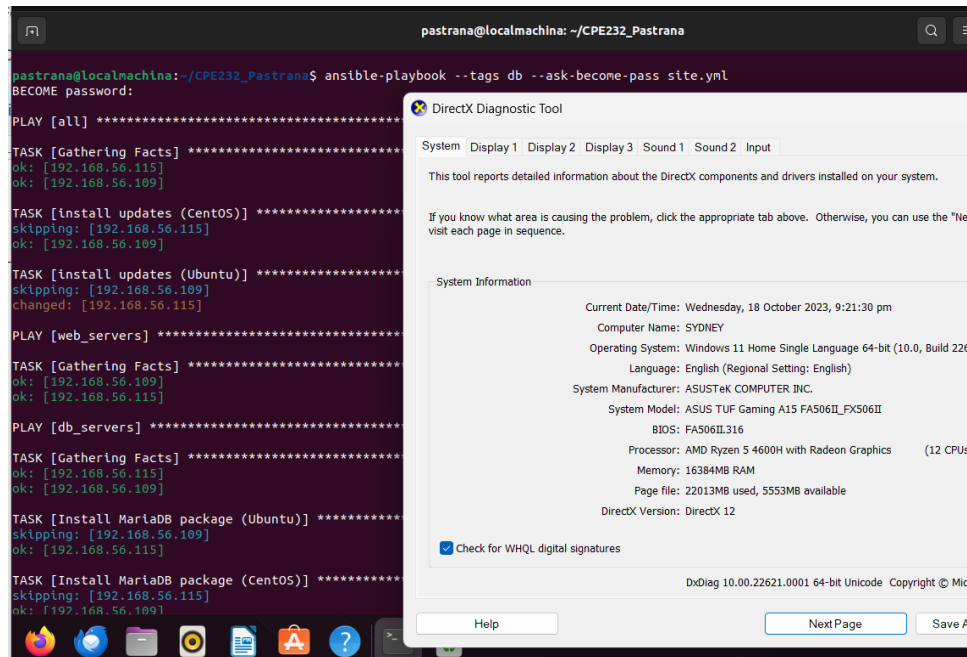
2.1 *ansible-playbook --list-tags site.yml*



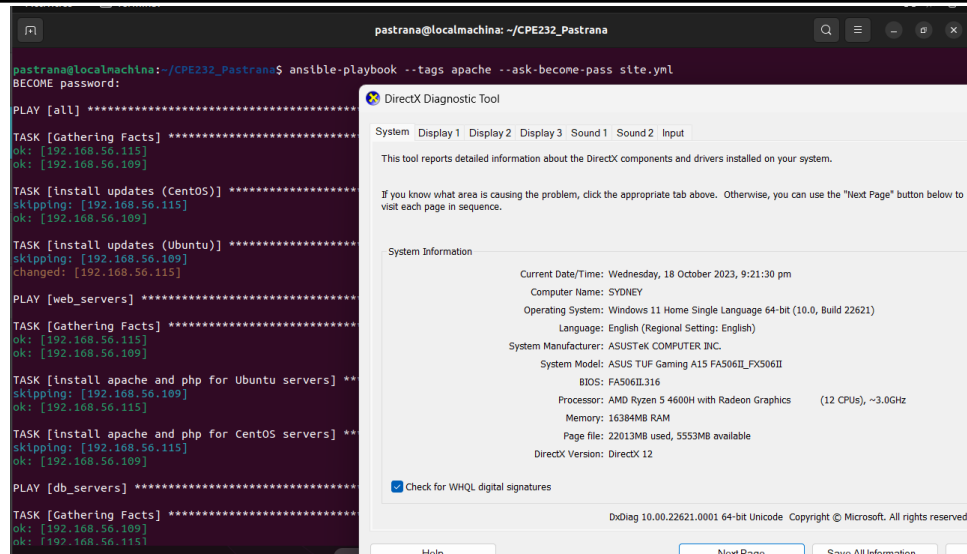
2.2 *ansible-playbook --tags centos --ask-become-pass site.yml*



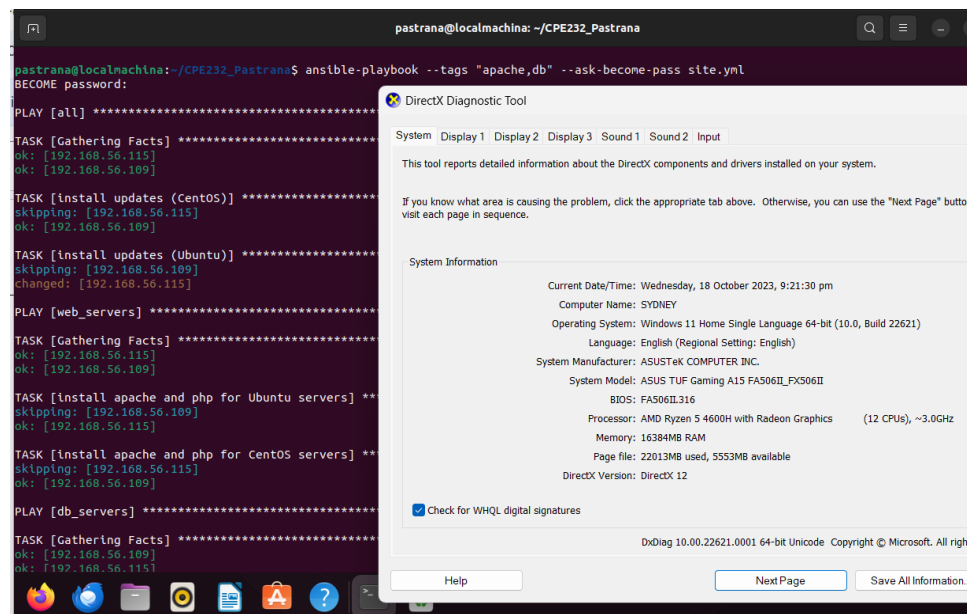
## 2.3 *ansible-playbook --tags db --ask-become-pass site.yml*



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



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



## 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"

```

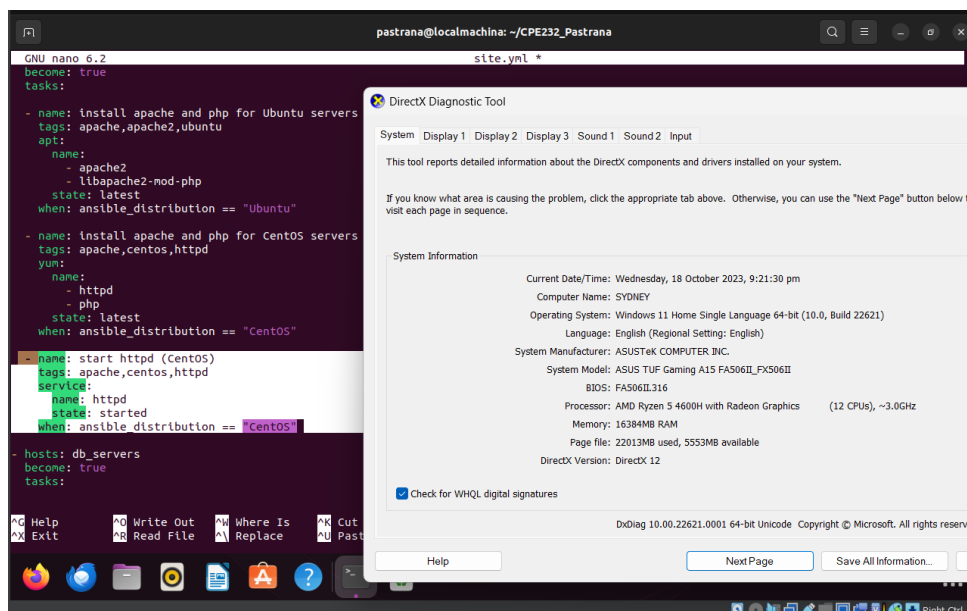


Figure 3.1.1

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"

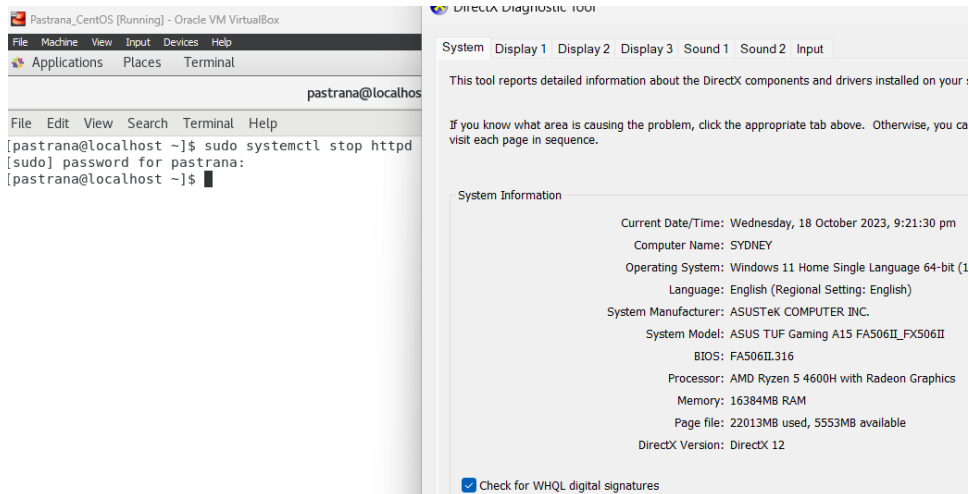
  - name: "Mariadb- Restarting/Enabling"
    service:
      name: mariadb
      state: restarted
      enabled: true

```

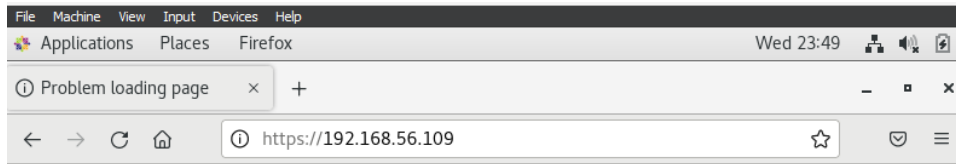
Figure 3.1.2

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

2. 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.







## Unable to connect

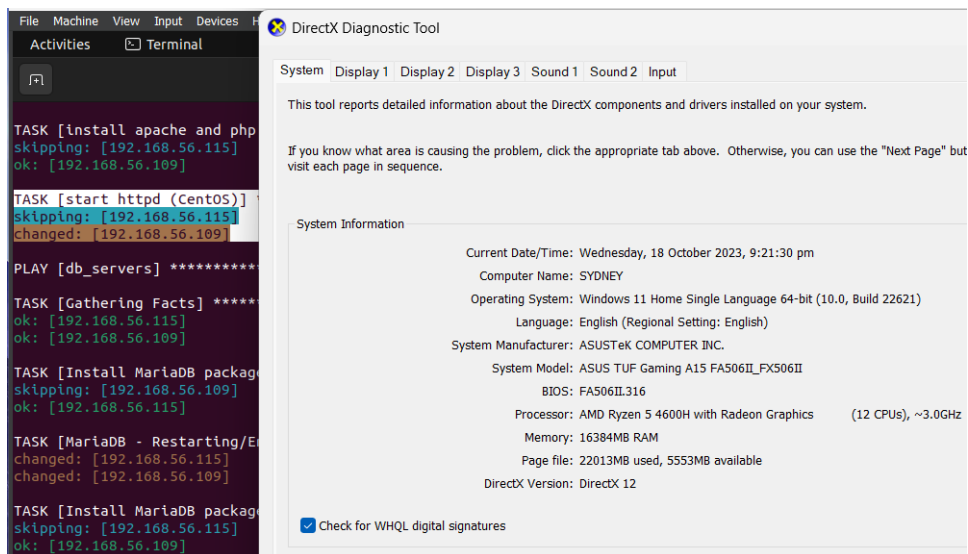
An error occurred during a connection to 192.168.56.109.

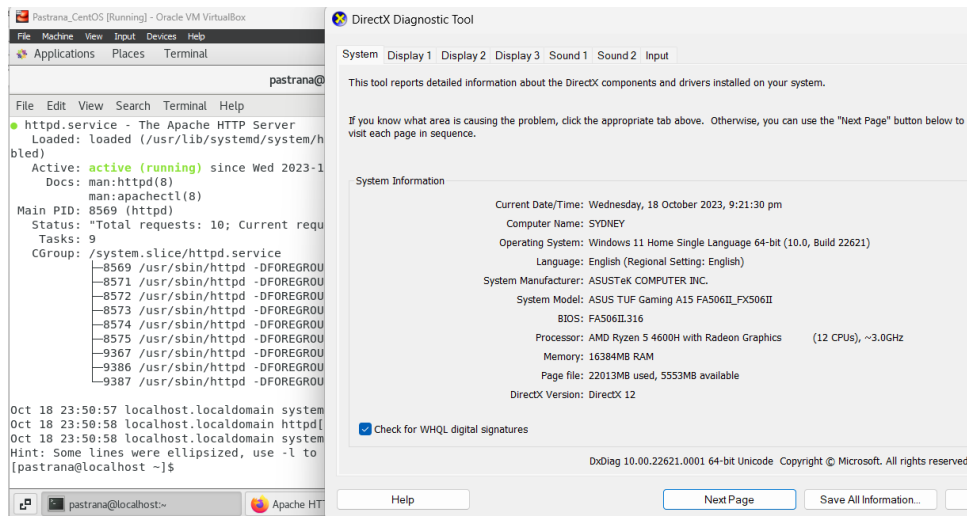
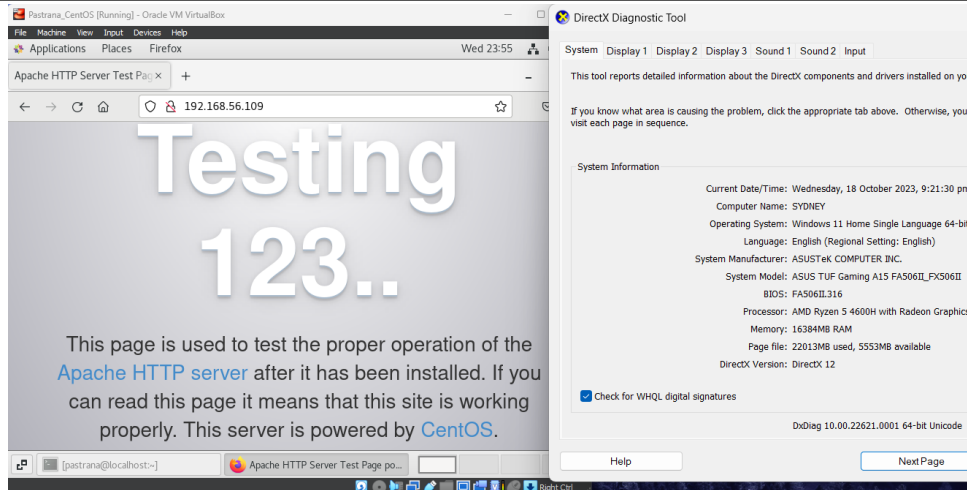
- The site could be temporarily unavailable or too busy. Try again in a few moments.
- If you are unable to load any pages, check your computer's network connection.
- If your computer or network is protected by a firewall or proxy, make sure that Firefox is permitted to access the Web.

Try Again

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?
  - Organizing remote servers into groups optimizes organization, simplifies the playbook, allows us to execute on specific nodes, and gives us the flexibility required to maintain and expand settings properly across multiple server types.
2. What is the importance of tags in playbooks?
  - Using tags in Ansible playbooks allow us to selectively execute tasks, improving efficiency and enabling targeted deployment or modification of specific tasks within the playbook.
3. Why do I think some services need to be managed automatically in playbooks?

- Managing services automatically with playbooks enables consistent, reproducible configurations and deployments, reducing human error and speeding system provisioning and maintenance.