| | |
|---|---|
| **Name: Aaron Martin P. Caro** | **Date Performed: 21/11/2023** |
| **Course/Section: CPE232/CPE31S5** | **Date Submitted: 23/11/2023** |
| **Instructor: Prof. Roman Richard** | **Semester and SY: 1st sem 2023-2024** |

**Activity 11: Containerization**

**1. Objectives**

Create a Dockerfile and form a workflow using Ansible as Infrastructure as Code (IaC) to enable Continuous Delivery process

**2. Discussion**

Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications. By taking advantage of Docker's methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.

Source: https://docs.docker.com/get-started/overview/

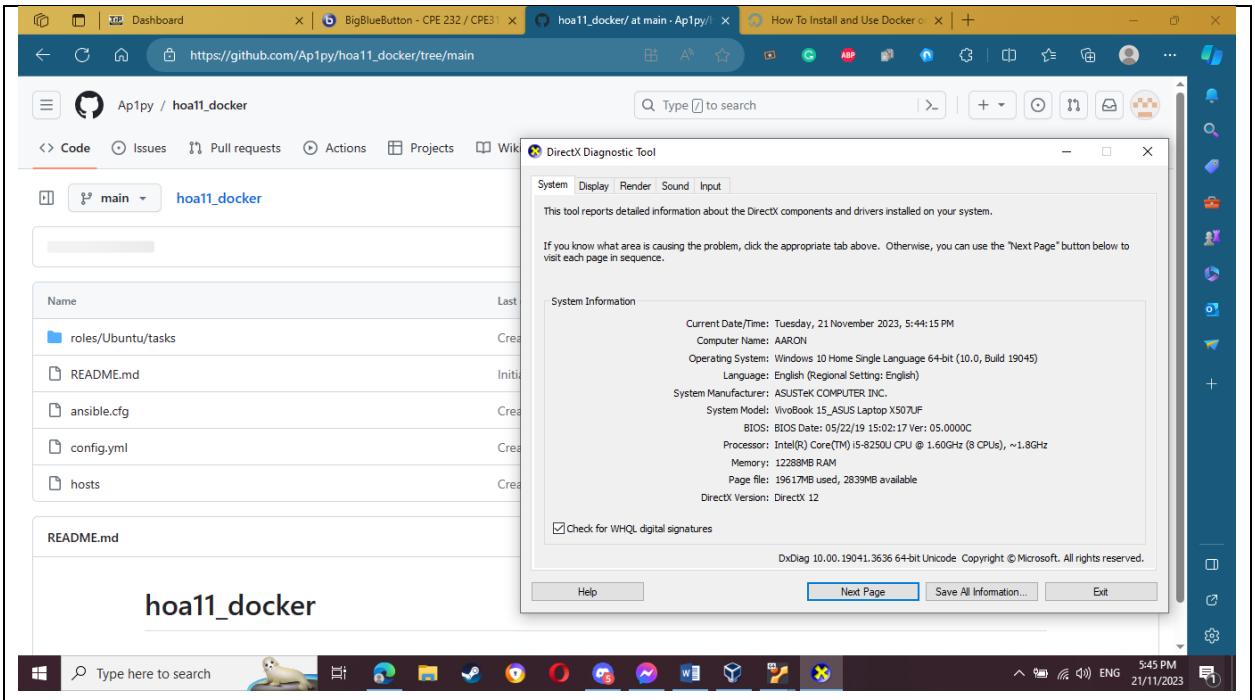
You may also check the difference between containers and virtual machines. Click the link given below.

Source: https://docs.microsoft.com/en-us/virtualization/windowscontainers/about/containers-vs-vm

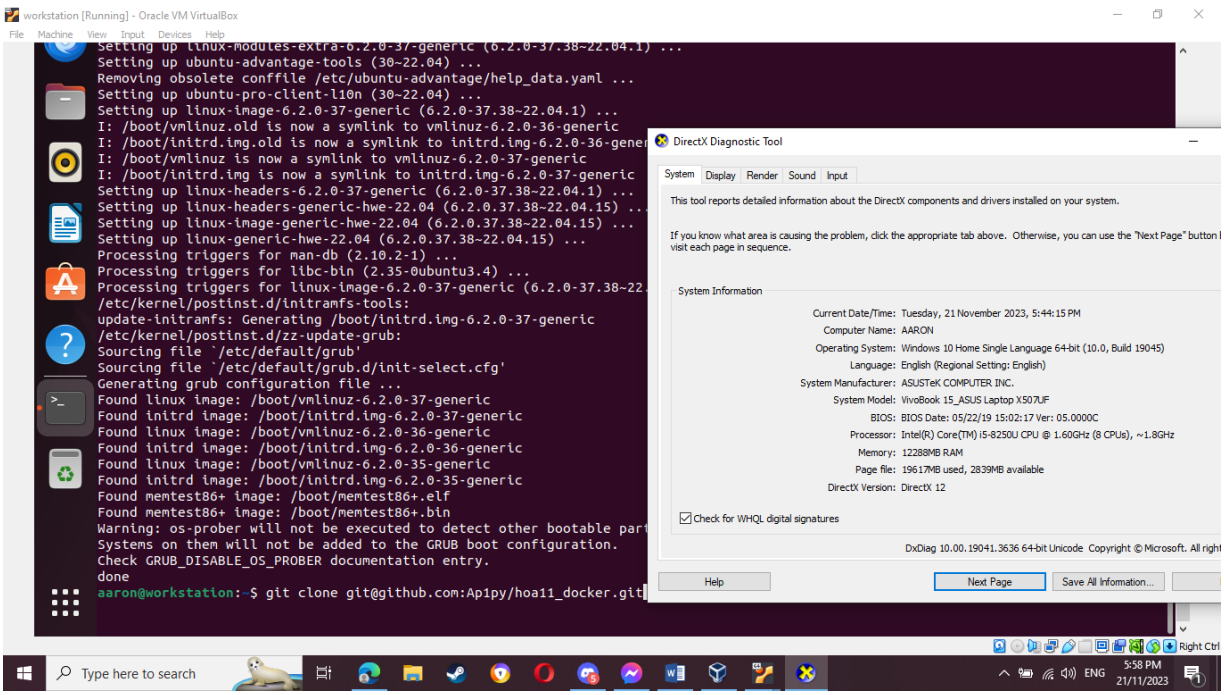
**3. Tasks**

1. Create a new repository for this activity.
2. Install Docker and enable the docker socket.
3. Add to Docker group to your current user.
4. Create a Dockerfile to install web and DB server.
5. Install and build the Dockerfile using Ansible.
6. Add, commit and push it to your repository.

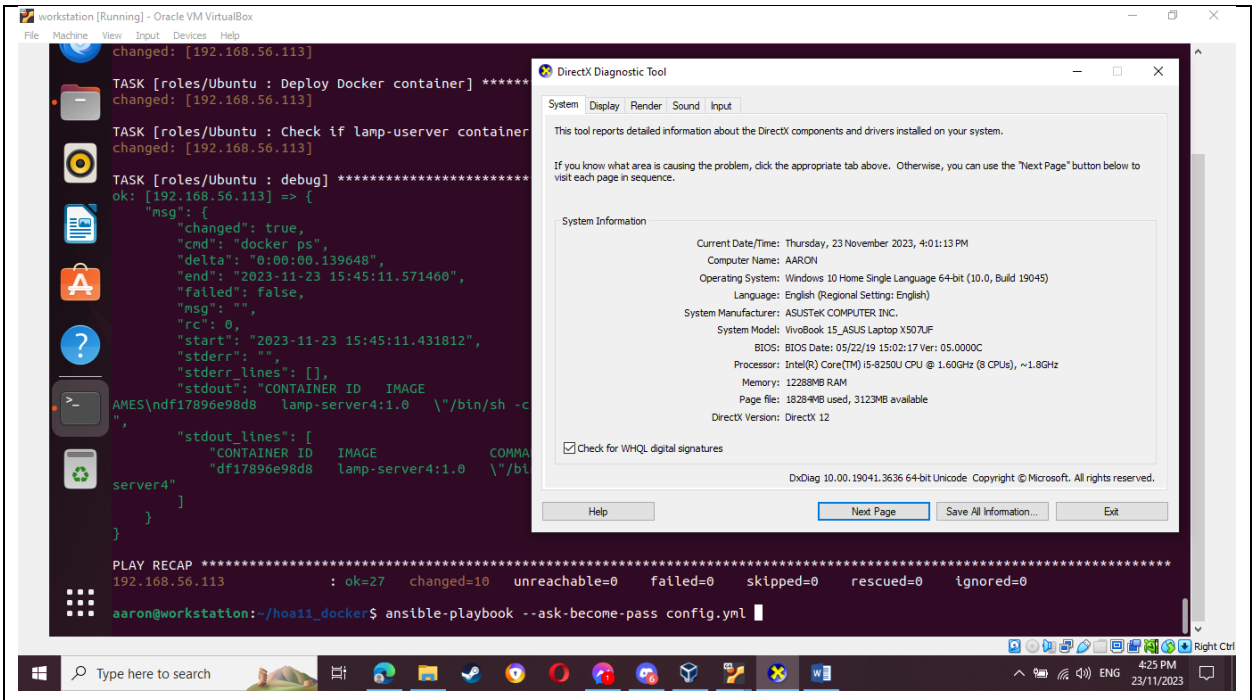
**4. Output** (screenshots and explanations)



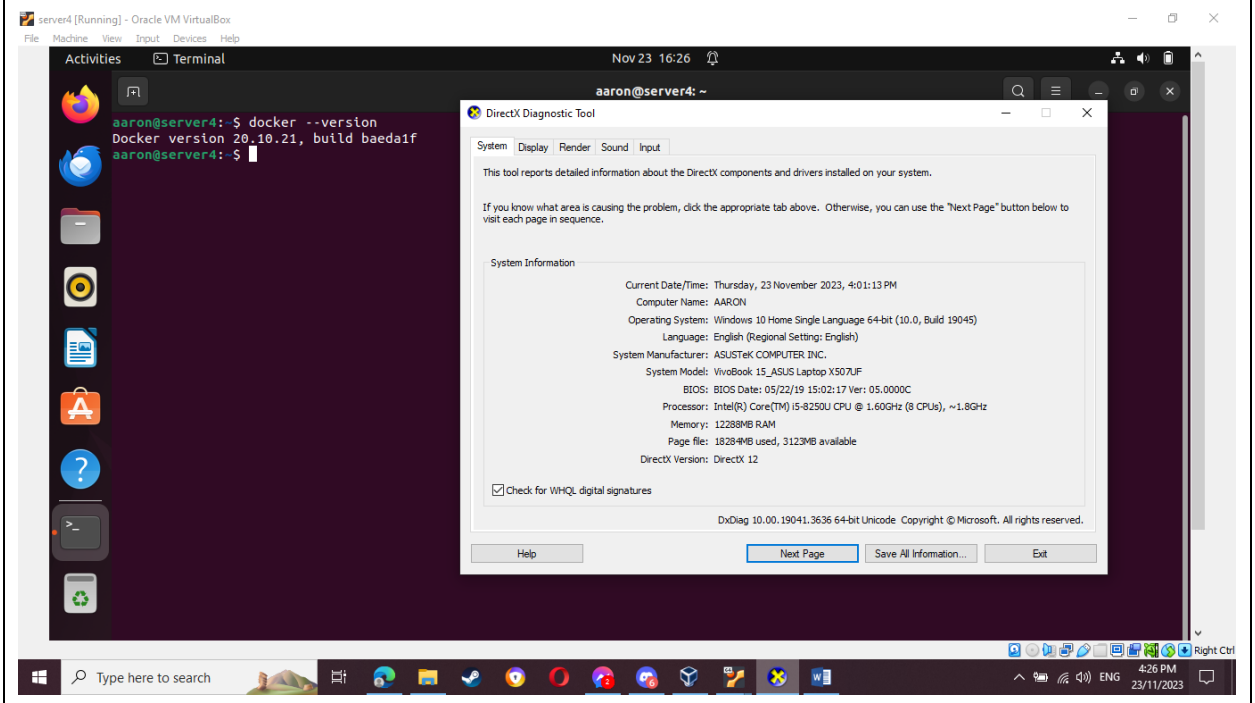

Create a new repository



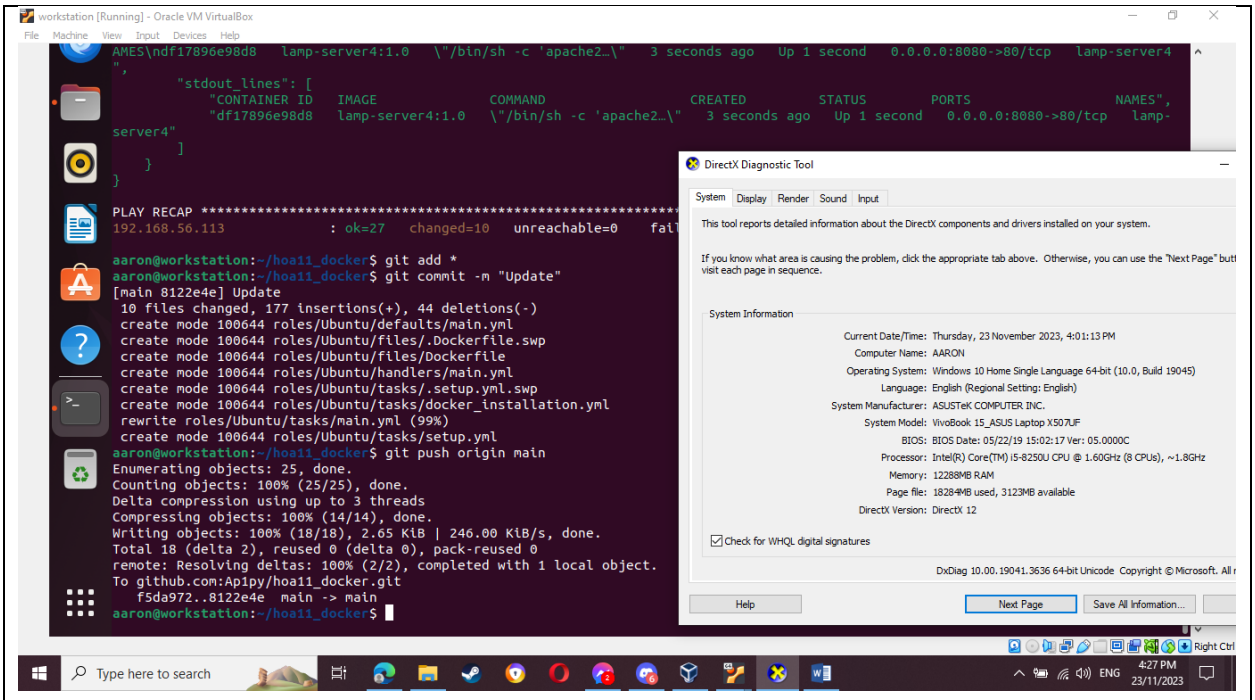

Cloning the repository



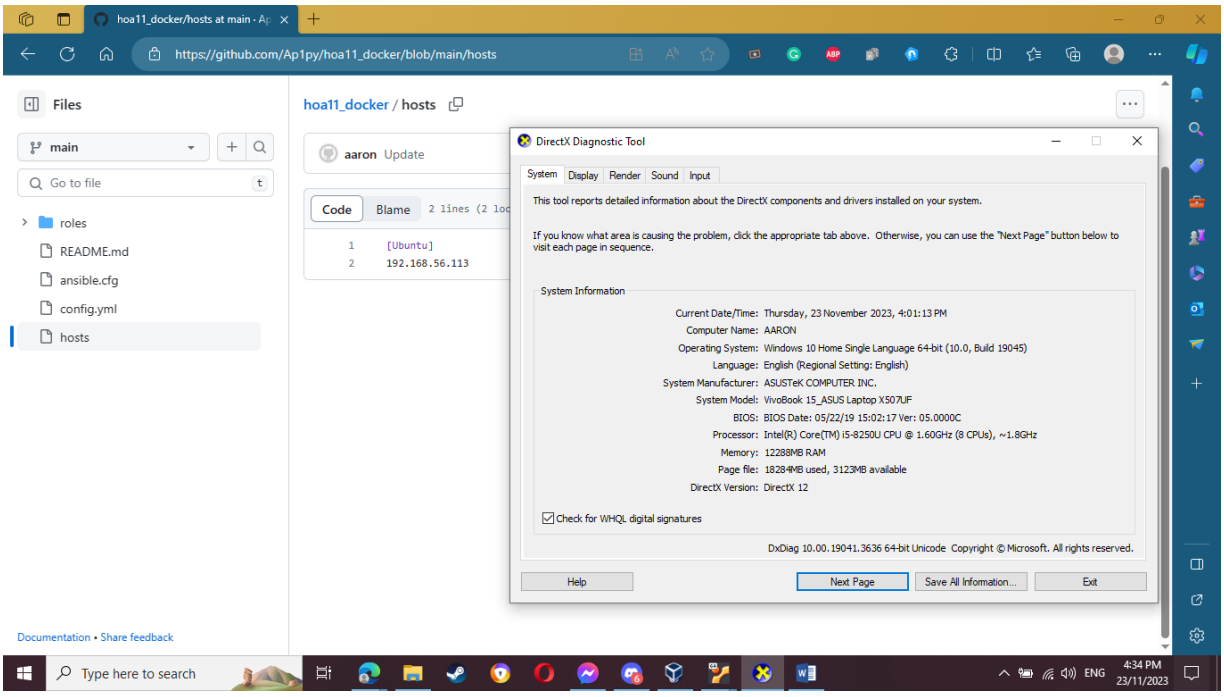

The finished installation



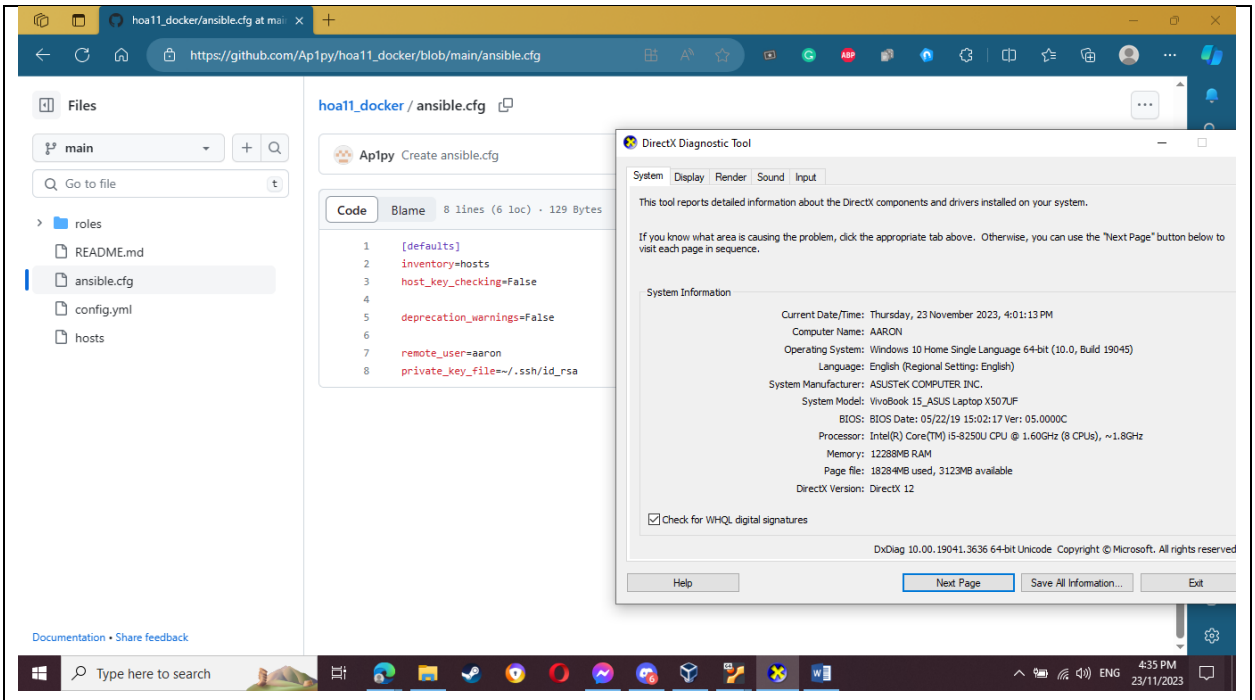

The confirmation of the version installed



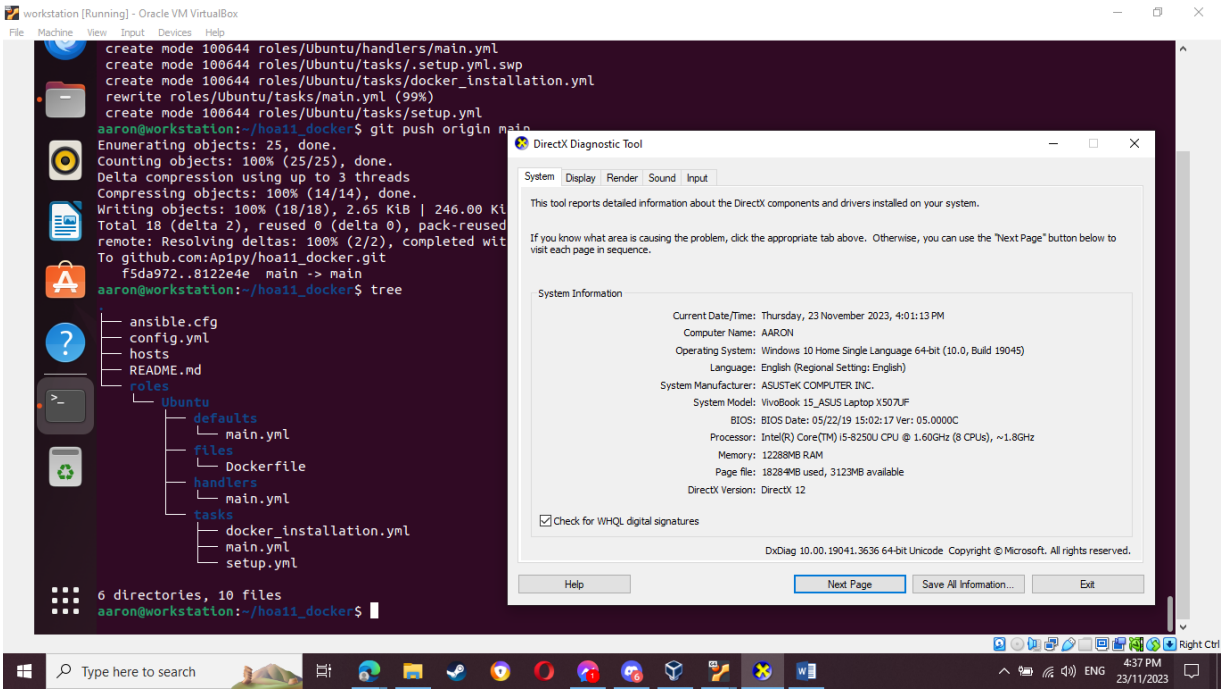

Adding the changes to the repository



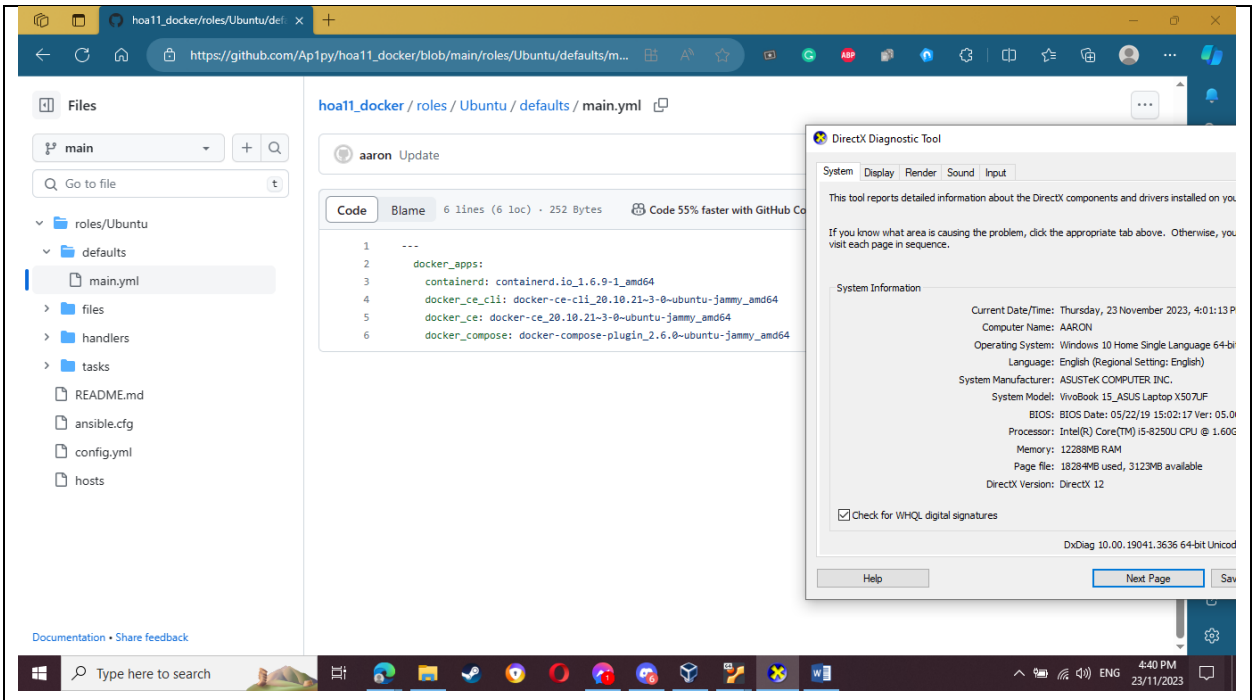

The host configuration



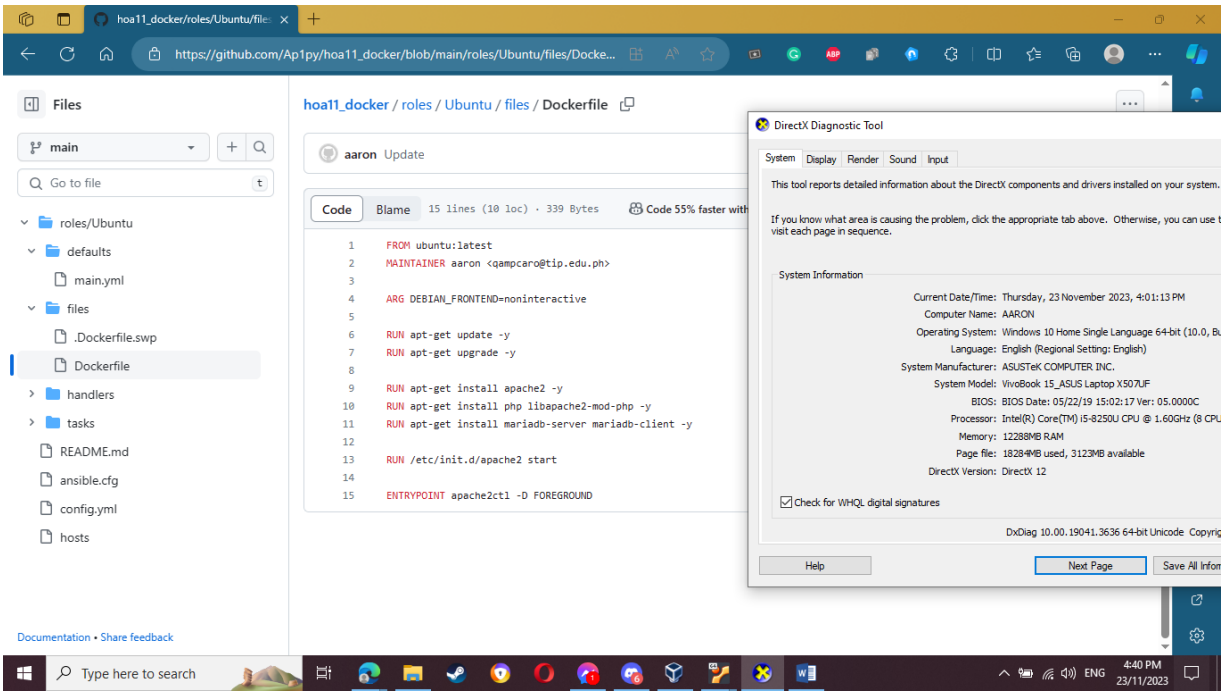

The ansible configuration copied from previous activities.



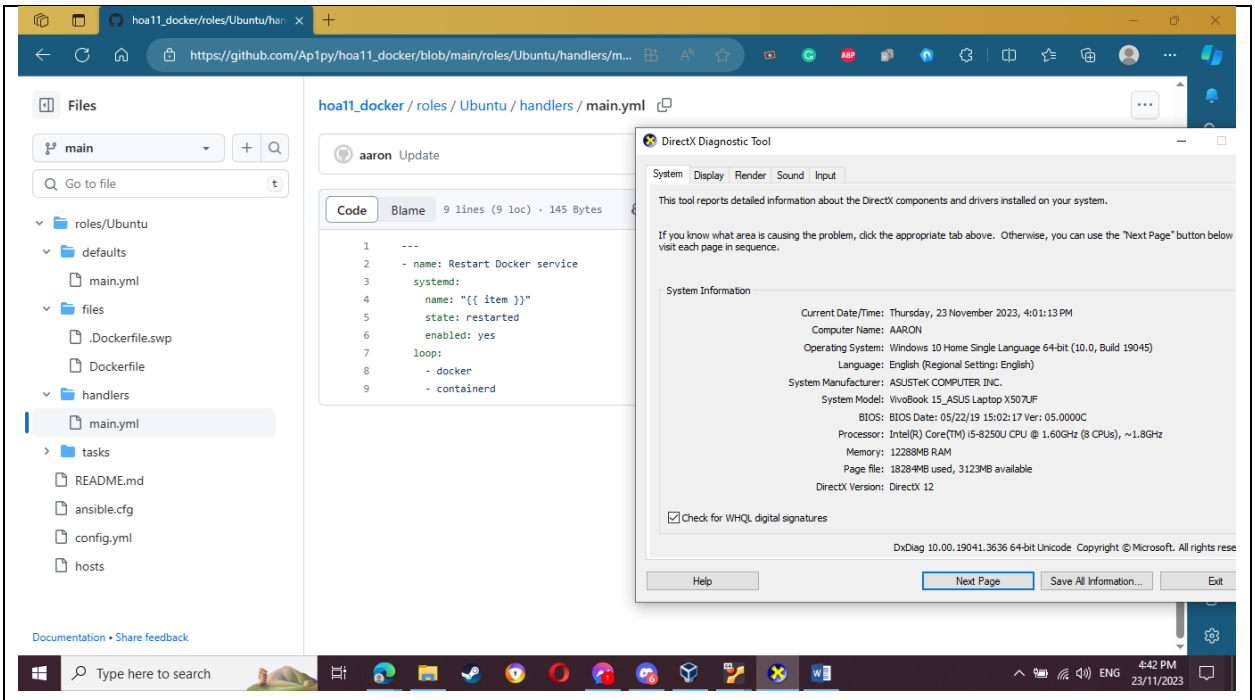

The directory tree structure



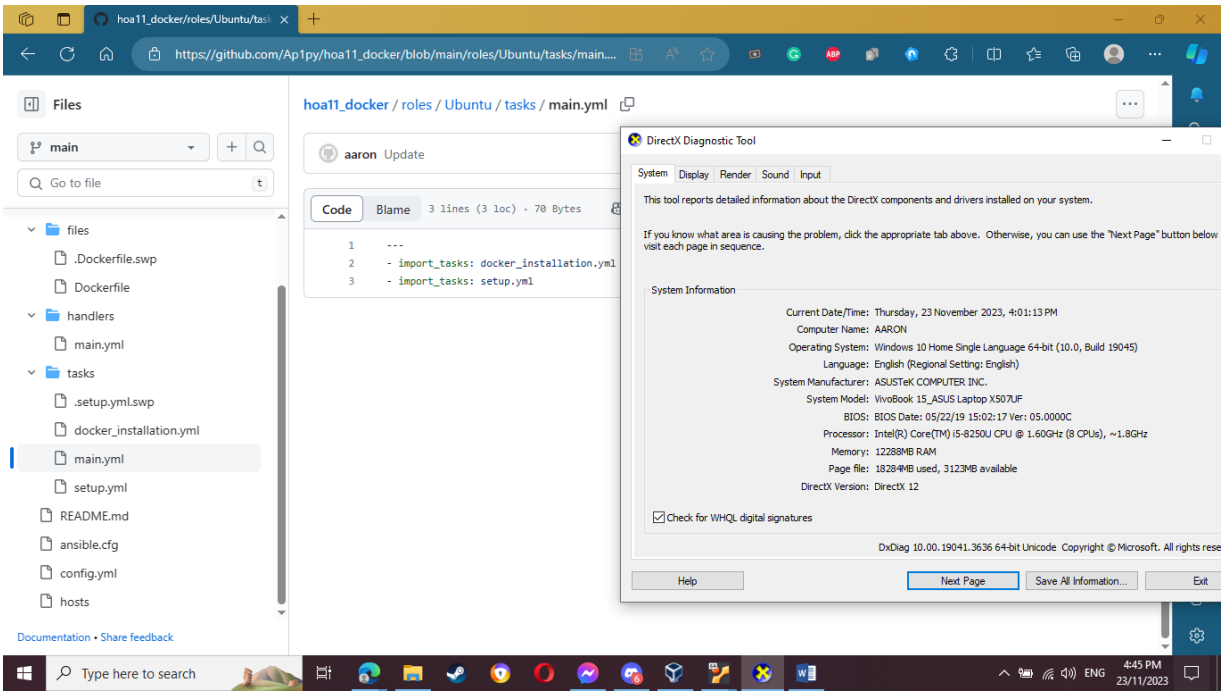

The default main.yml



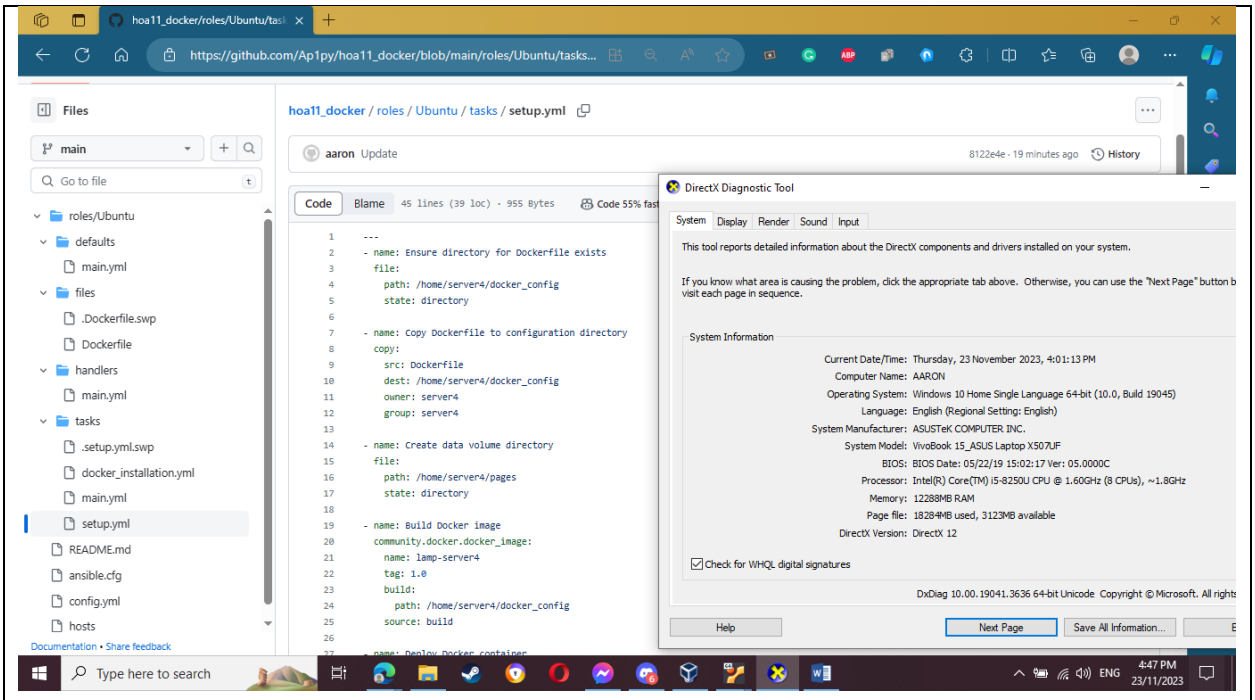

The Dockerfile configuration the maintainer section can be change but the other commands are crucial for setting up Docker



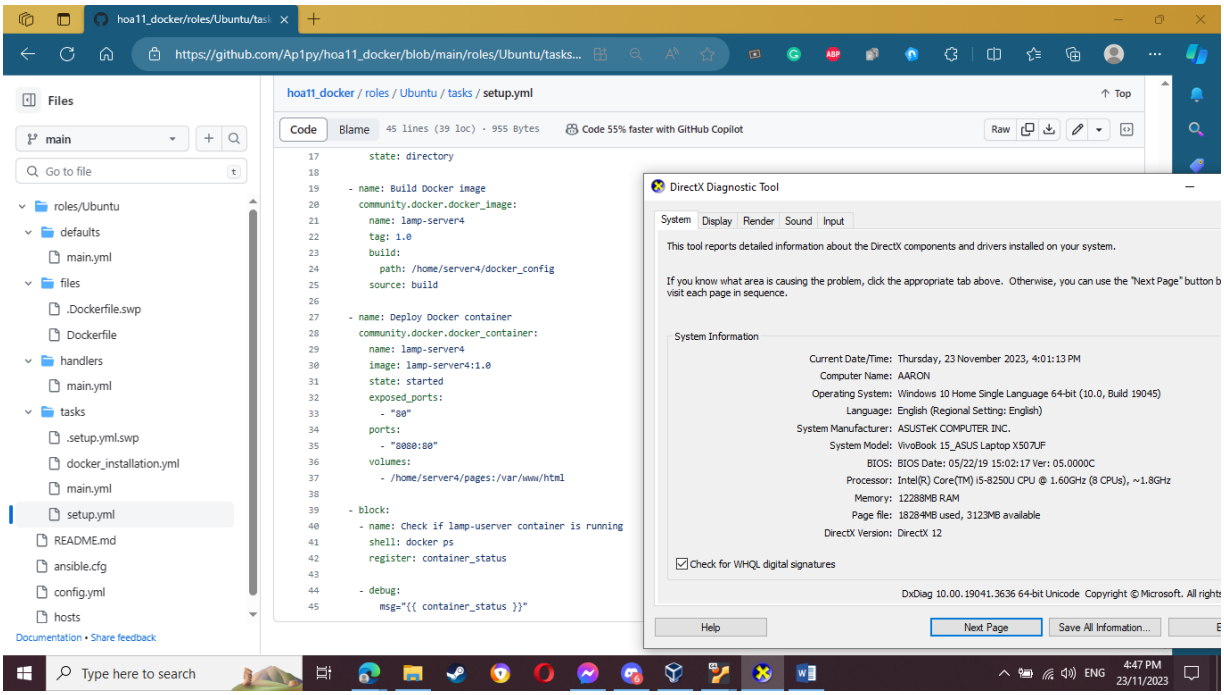

The handler file separated from the installation play so it can be easily found



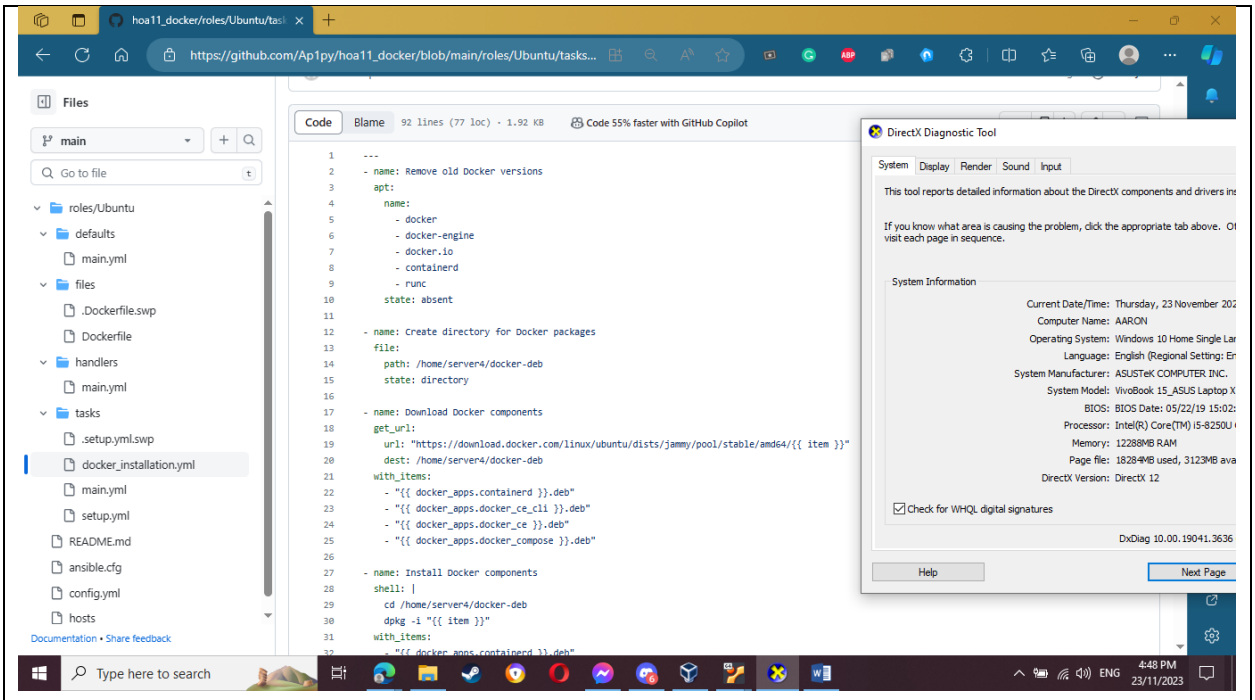

A playbook that will call out other plays within the same task directory



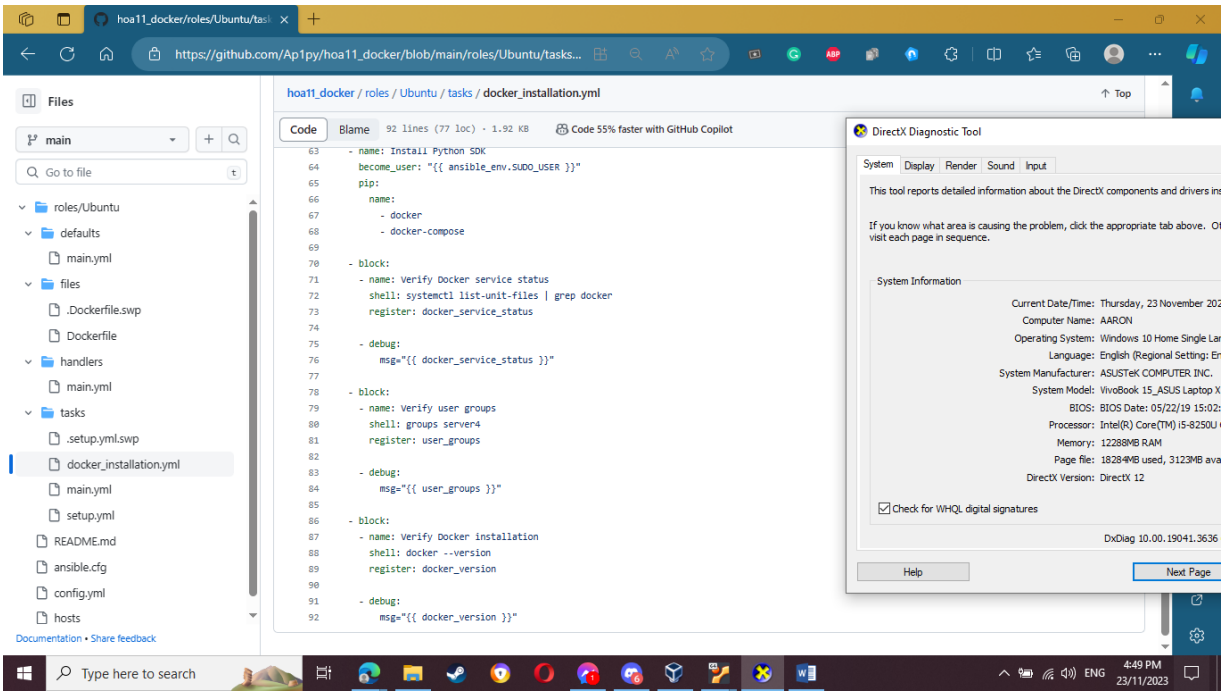

Set-up yml configuration




Setup yml configuration




The installation yml




The installation yml

**Reflections:**

Answer the following:

1. **What are the benefits of implementing containerizations?**
   Containerization, exemplified by advances like Docker, offers various advantages for current programming improvement and sending. By embodying applications and their conditions into lightweight, convenient compartments, designers can guarantee consistency across various conditions, from neighborhood improvement machines to creation servers. Holders give a normalized and disengaged climate, diminishing the "it deals with my machine" issue. They improve versatility and asset usage, taking into account proficient arrangement of microservices and empowering quick increasing or down in view of interest. Compartments smooth out the organization interaction, making it quicker and more solid, working with consistent mix and ceaseless conveyance (CI/Compact disc) pipelines. Moreover, containerization upholds a measured and effectively reasonable design, advancing cooperation and making it simpler to refresh or supplant parts without influencing the whole framework. By and large, containerization upgrades readiness, versatility, and dependability in programming advancement and organization work processes.

**Conclusions:**

All in all, containerization has arisen as a groundbreaking innovation in current programming improvement, offering a scope of convincing advantages. From guaranteeing consistency across assorted conditions to working with effective scaling and sending, compartments have become basic to nimble and DevOps rehearses. The epitome of utilizations and conditions into versatile units tends to organization challenges as well as improves coordinated effort and speeds up the advancement lifecycle. As associations progressively embrace cloud-local structures and microservices, containerization fills in as a foundation, giving an adaptable and normalized way to deal with building, delivery, and running applications. The reception of containerization, exemplified by instruments like Docker, addresses a change in perspective towards more effective, versatile, and solid programming improvement and organization processes.