Student Name and ID as per student card:

- Amit Dashgupta (20032317)

Other Group Members:

- Roshani Devdas Naik (20031751)
- Priyatharshni Jeyaratnam (20040259)
- Manoj Meram (20035407)

Course Title:

MSc Information Systems with Computing

Lecturer's Name:

Kingsley Ibomo

Module/Subject Code:

B9IS121

Module/Subject Title:

Network Systems and Administration

Assignment Title:

Automated Container deployment and Administration

Word Count: 1,135

Contents

Introduction

This report details the steps taken to deploy an Apache server using Docker and Ansible. The project involves creating a public GitHub repository, writing an Ansible playbook, configuring network settings, and verifying the deployment. The goal is to ensure a seamless and efficient setup process, demonstrating practical skills in automation and container management.

GitHub Repository Setup

GitHub link : https://github.com/20032317/Automated-Container-deployment-and-Administration

#Creating the Repository

Create a Public GitHub Repository: Initialize the repository with a README.md file that explains the purpose of the repository and provides necessary instructions.

Clone the Repository:

```
git clone https://github.com/20032317/Automated-Container-deployment-and-Administration.git
```

#Storing the Ansible Playbook

- ➢ **Introduction to Ansible Playbooks:**
  Ansible playbooks are the primary means of defining automation tasks in Ansible. Written in YAML (Yet Another Markup Language), these files specify the tasks that Ansible will perform on managed hosts, such as installing software, configuring services, or managing files.

Navigate to the Repository Directory:

```
cd /path/to/cloned/repository
```

Create the Playbook File:

```
touch docker_deploy.yml
```

Ansible Playbook Creation

- ➢ **What is an Ansible Playbook?**

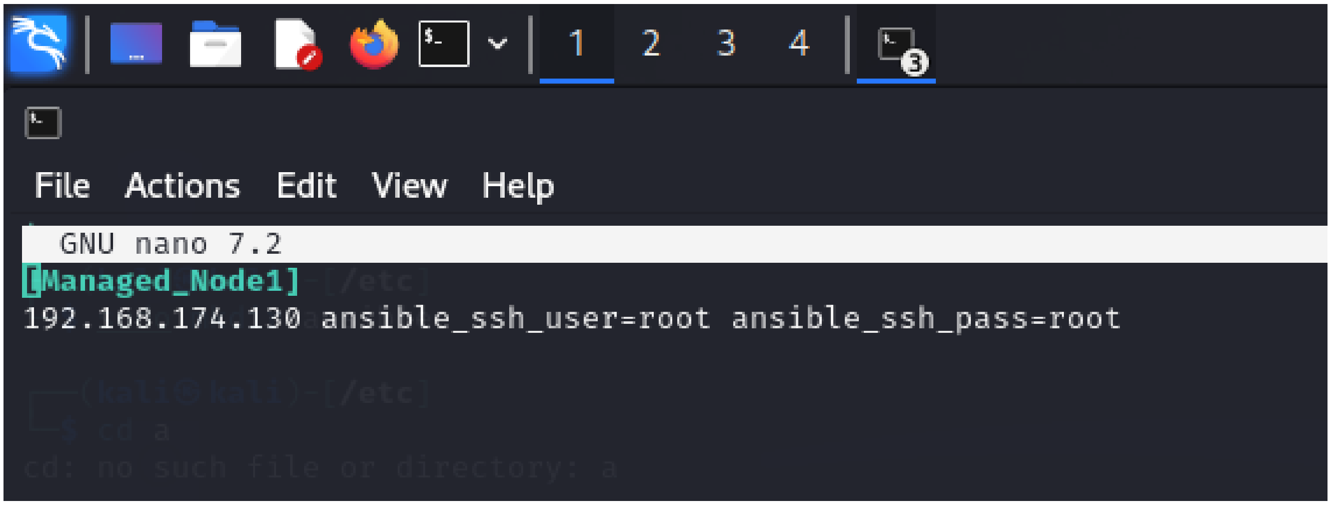
Ansible playbooks are organized configuration files that let you specify your infrastructure's ideal state in an understandable manner for humans. They specify what has to be done, including installing software, setting up services, or organizing files, on a set of hosts or groups of hosts.

Creating the Playbook

Define the Playbook Structure: The playbook includes sections for hosts, tasks, and handlers.

- **Configuring Ansible:**
    - **Inventory**

An essential part of Ansible automation is an Ansible inventory file, which is a list of the managed nodes (hosts) and the variables that go with them. This file classifies the machines into categories and specifies which ones you wish to control. There are several formats in which the inventory file can be written, such as YAML, JSON, and INI. It gives you the ability to systematically and scalable manage the setup of your infrastructure.



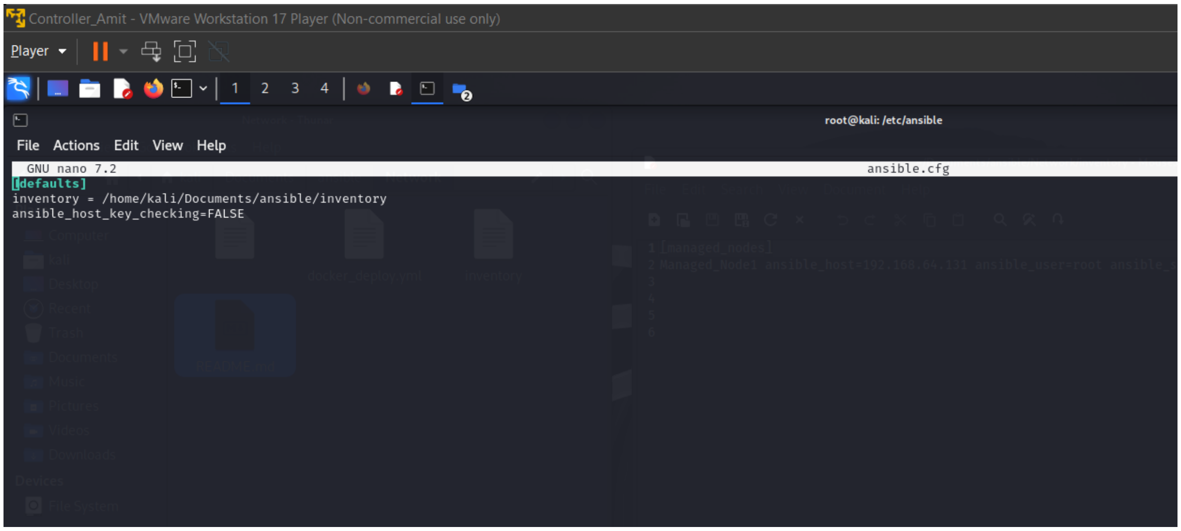
```
  GNU nano 7.2
[Managed_Node1]
192.168.174.130 ansible_ssh_user=root ansible_ssh_pass=root
```

Figure 1 Inventory file

- Configuring File
  Ansible configuration files, such as ansible.cfg, let you alter the parameters and behavior of your Ansible environment. You can save this file in several locations.

  /etc/ansible/ansible.cfg

~/.ansible.cfg

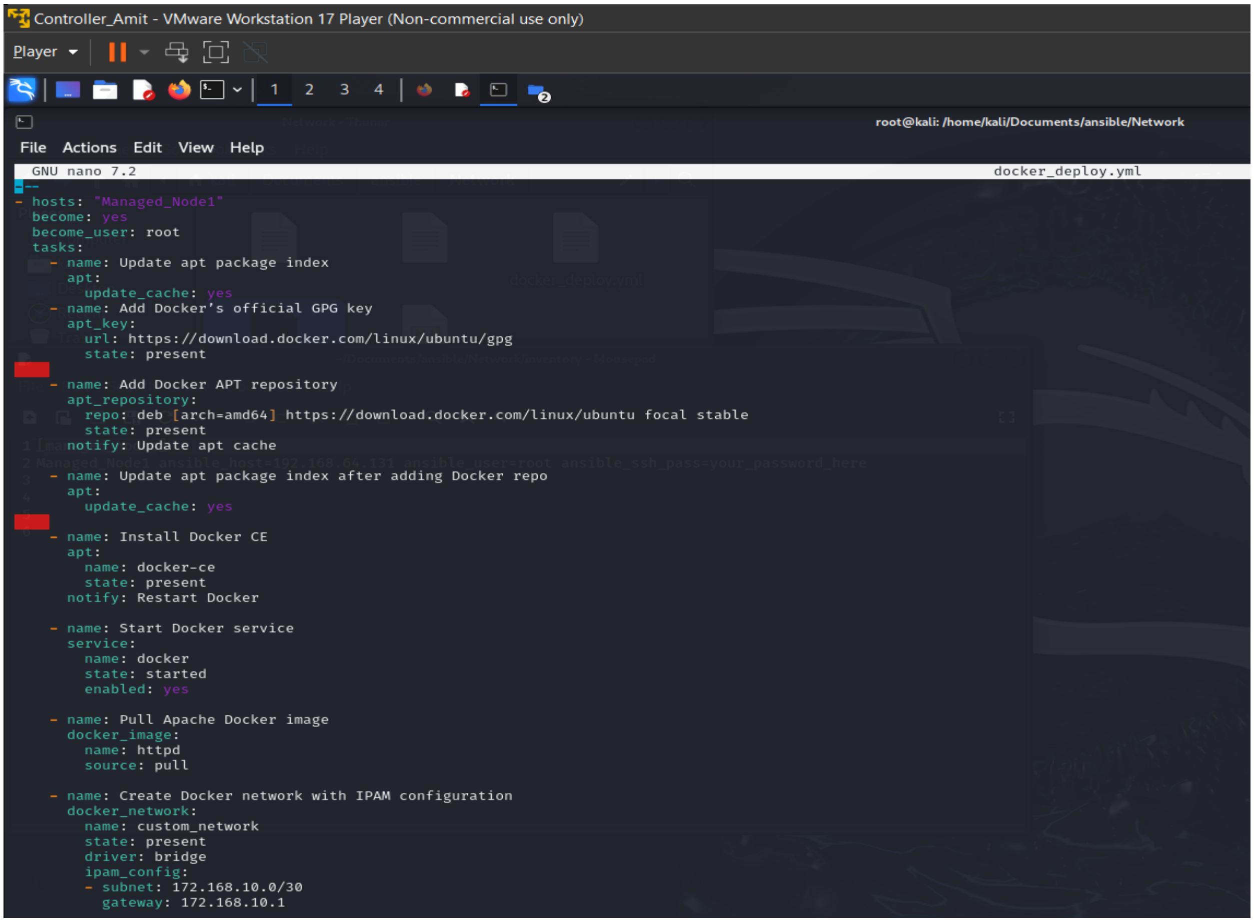

Controller_Amit - VMware Workstation 17 Player (Non-commercial use only)
Player
root@kali: /etc/ansible
File Actions Edit View Help

```
  GNU nano 7.2                                                          ansible.cfg
[defaults]
inventory = /home/kali/Documents/ansible/inventory
ansible_host_key_checking=FALSE
```

Figure 2 Config file

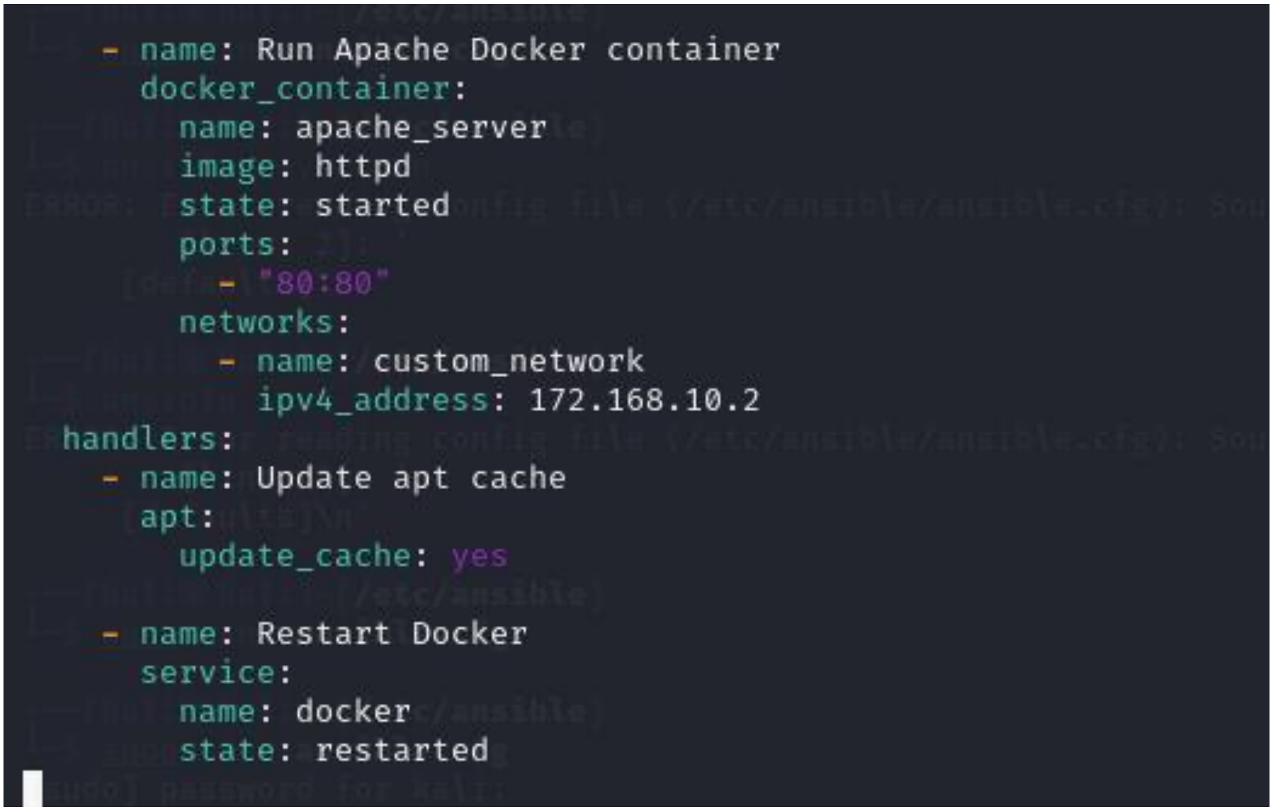
- **Writing the Playbook:**
    - Define the playbook structure in a YAML file.

YAML (YAML Aren't Markup Language) is a human-readable data serialization language used to write Ansible playbooks. Playbooks are collections of plays, and each play outlines a set of actions that must be taken on a designated list of hosts.


Controller_Amit - VMware Workstation 17 Player (Non-commercial use only)
Player
1 2 3 4
root@kali: /home/kali/Documents/ansible/Network
File Actions Edit View Help
GNU nano 7.2
docker_deploy.yml

```
---
- hosts: "Managed_Node1"
  become: yes
  become_user: root
  tasks:
    - name: Update apt package index
      apt:
        update_cache: yes
    - name: Add Docker’s official GPG key
      apt_key:
        url: https://download.docker.com/linux/ubuntu/gpg
        state: present

    - name: Add Docker APT repository
      apt_repository:
        repo: deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable
        state: present
      notify: Update apt cache

    - name: Update apt package index after adding Docker repo
      apt:
        update_cache: yes

    - name: Install Docker CE
      apt:
        name: docker-ce
        state: present
      notify: Restart Docker

    - name: Start Docker service
      service:
        name: docker
        state: started
        enabled: yes

    - name: Pull Apache Docker image
      docker_image:
        name: httpd
        source: pull

    - name: Create Docker network with IPAM configuration
      docker_network:
        name: custom_network
        state: present
        driver: bridge
        ipam_config:
          - subnet: 172.168.10.0/30
            gateway: 172.168.10.1
```

Figure 3 Playbook



```
  - name: Run Apache Docker container
    docker_container:
      name: apache_server
      image: httpd
      state: started
      ports:
        - "80:80"
      networks:
        - name: custom_network
          ipv4_address: 172.168.10.2
handlers:
  - name: Update apt cache
    apt:
      update_cache: yes

  - name: Restart Docker
    service:
      name: docker
      state: restarted
```

Figure 4 Playbook

- **Update apt package index:** This task updates the package index on the managed node using apt (a package manager for Debian-based systems).
- **Install prerequisites for Docker**: Installs necessary packages to ensure that Docker can be installed and run on your system.
- **Add Docker's official GPG key:** Adds the official Docker GPG key to verify the authenticity of Docker packages.
- **Add Docker APT repository:** Adds the Docker repository to the list of APT sources, so you can install Docker from their official repository.
- **Update apt package index after adding Docker repo:** Updates the package index again after adding the new Docker repository.
- **Install Docker CE (Community Edition):** Installs Docker CE from the newly added repo.
- **Start Docker service:** Ensures that the Docker service is started and enabled to start on boot.
- **Pull Apache Docker image:** Downloads the Apache HTTP Server image from Docker Hub.
- **Create Docker network:** Sets up a custom network for your Docker containers with specified subnet and gateway.

- **Run Apache Docker container:** Runs a container using the Apache image, binds port 80 on the host to port 80 in the container, and connects it to your custom network with a specified IP address.
- **Update apt cache:** Updates the package index if any repositories are added.
- **Restart Docker:** Restarts the Docker service if it's installed or configurations are changed.

Network Diagram

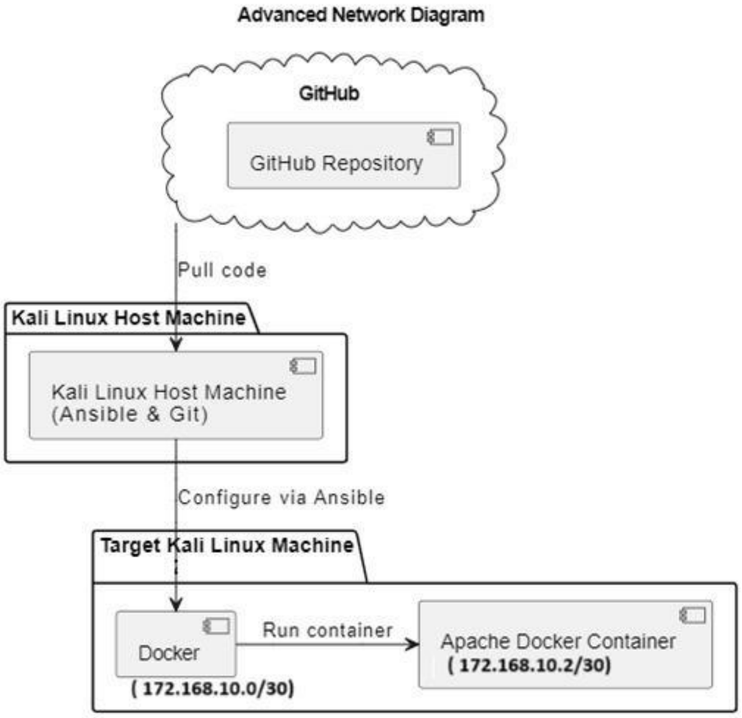


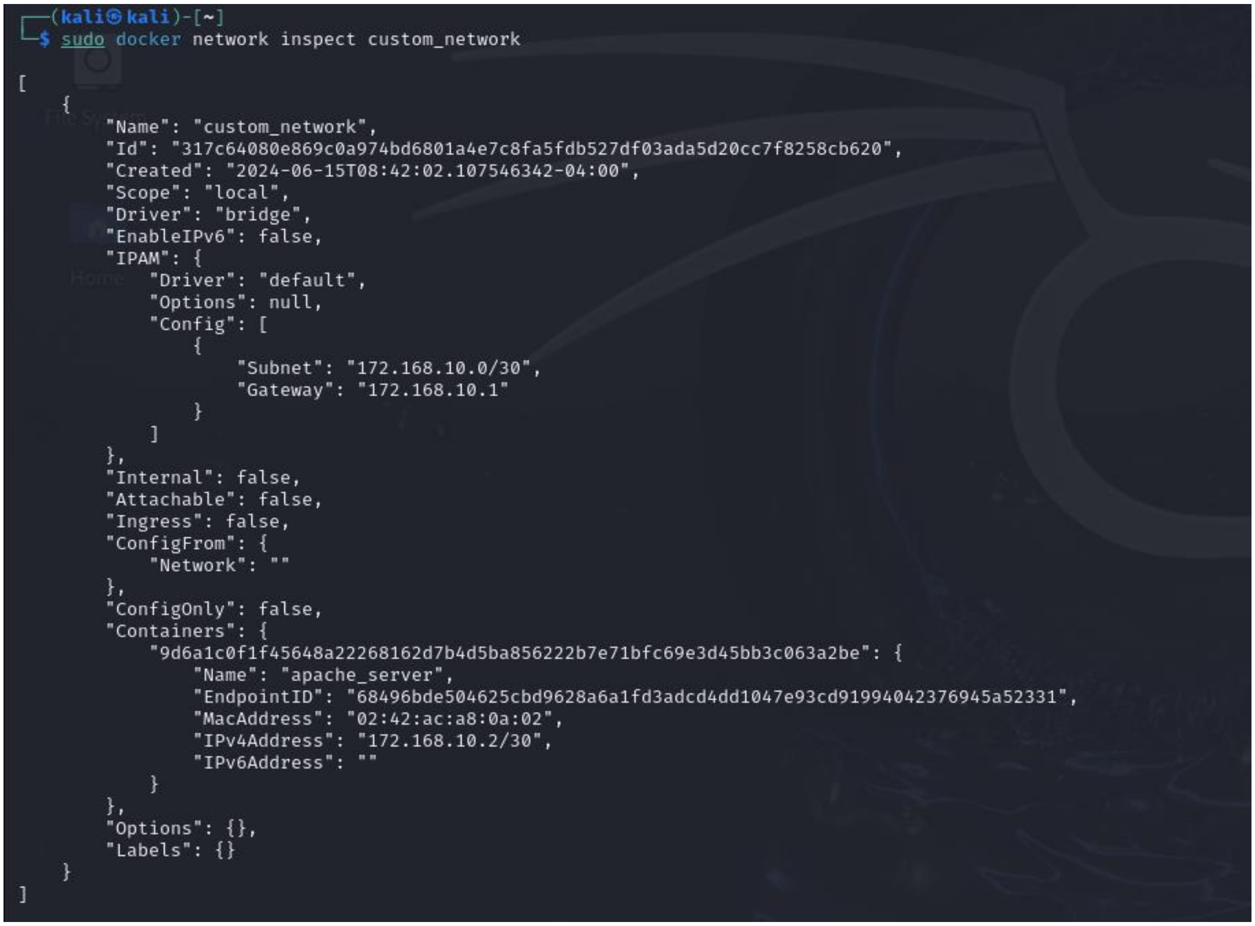
Figure 5 Network Diagram

A "Kali Linux Host Machine (Ansible & Git)" in the middle of the diagram acts as the control node for controlling Git operations and executing Ansible playbooks. Two pathways split off from here: one points directly to a "Target Kali Linux Machine (172.168.10.0/30)," which is probably the network address range for deployed machines, and the other leads to "Configure via Ansible," which suggests using Ansible for configuration management.

The use of Docker containers is indicated by the sequence on the right side that begins with "Docker." Next is "Run container > Apache Docker Container (172.168.10.2/30)," which details the deployment of an Apache server inside a Docker container that is assigned to a particular IP address within the network range.

Verification

1. **Verification Steps:**



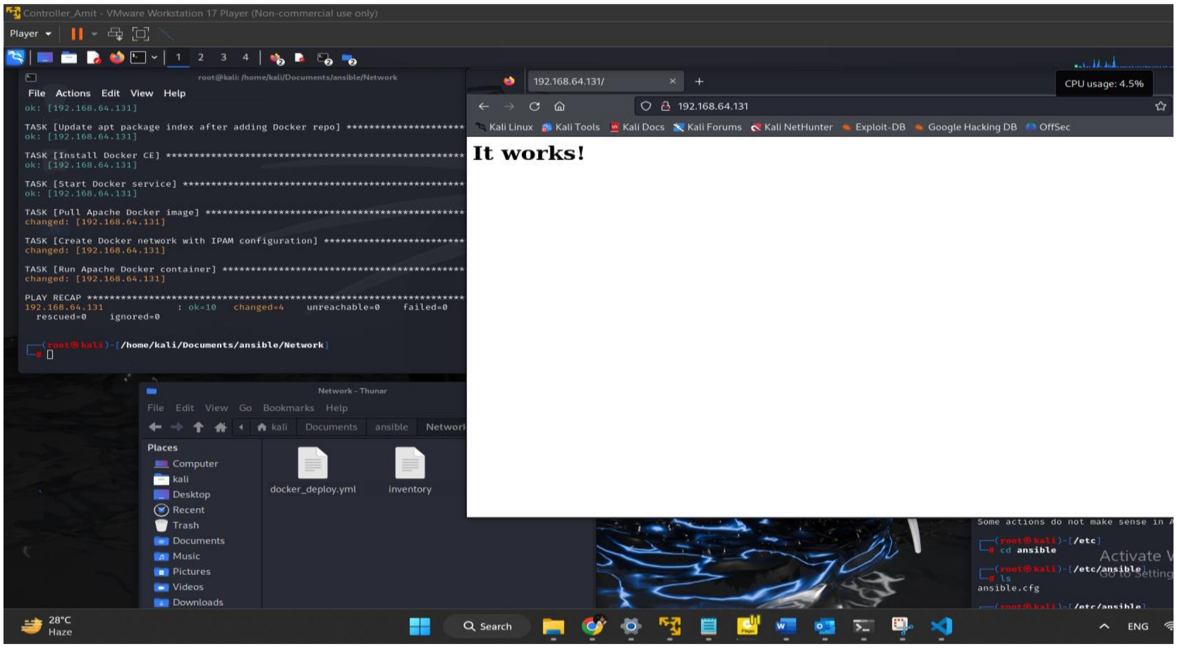
```
┌──(kali㉿kali)-[~]
└─$ sudo docker network inspect custom_network

[
    {
        "Name": "custom_network",
        "Id": "317c64080e869c0a974bd6801a4e7c8fa5fdb527df03ada5d20cc7f8258cb620",
        "Created": "2024-06-15T08:42:02.107546342-04:00",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.168.10.0/30",
                    "Gateway": "172.168.10.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "9d6a1c0f1f45648a22268162d7b4d5ba856222b7e71bfc69e3d45bb3c063a2be": {
                "Name": "apache_server",
                "EndpointID": "68496bde504625cbd9628a6a1fd3adcd4dd1047e93cd91994042376945a52331",
                "MacAddress": "02:42:ac:a8:0a:02",
                "IPv4Address": "172.168.10.2/30",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {}
    }
]
```

Figure 6 Verification

Outline the steps to verify the successful deployment of the Apache container.

2. **Testing:**

Figure 7 Webserver



```
ok: [192.168.64.131]

TASK [Update apt package index after adding Docker repo] **********************
ok: [192.168.64.131]

TASK [Install Docker CE] *******************************************************
ok: [192.168.64.131]

TASK [Start Docker service] ****************************************************
ok: [192.168.64.131]

TASK [Pull Apache Docker image] ************************************************
changed: [192.168.64.131]

TASK [Create Docker network with IPAM configuration] **************************
changed: [192.168.64.131]

TASK [Run Apache Docker container] *********************************************
changed: [192.168.64.131]

PLAY RECAP *********************************************************************
192.168.64.131             : ok=10   changed=4    unreachable=0    failed=0
  rescued=0    ignored=0


┌──(root㉿kali)-[/home/kali/Documents/ansible/Network]
└─#
```

It works!

```
Some actions do not make sense in A

┌──(root㉿kali)-[/etc]
└─# cd ansible

┌──(root㉿kali)-[/etc/ansible]
└─# ls
ansible.cfg

┌──(root㉿kali)-[/etc/ansible]
```

- ○ Access the Apache server through a web browser using the host's IP address and configured port.
- ○ Verify that the default Apache web page is displayed.

Conclusion

This project successfully demonstrates the deployment of an Apache server using Docker and Ansible. By leveraging a GitHub repository for version control and automating the process with an Ansible playbook, the setup was efficient and systematic. The network configuration ensured proper communication between the host and the container, and the verification confirmed the successful deployment. Lastly, we described the verification procedures that will guarantee the Apache container is deployed successfully. We verified that the configuration was successful by using a web browser to access the Apache server and observing the display of the default Apache web page.

Bibliography

- Ansible Documentation: https://docs.ansible.com/

- Docker Documentation: https://docs.docker.com/

- GitHub: https://github.com