REPORT FOR NETWORK AND SYSTEM ADMINISTRATION ASSESSMENT

Module Name – Network and System Administration

Module code – B9ISI21

Professor – Mr Kingsley Ibomo

Student – Coleman Okafor

Student ID - 20023341

GitHub Link - https://github.com/colejnr/Network-Project.git

SUMMARY:

The aim of my project is deploying a Docker container with an HTML page served by the Apache web server. I began with the setup of the development environment, where we installed necessary tools such as Docker, Ansible, Virtual Machine, Ubuntu and Apache. We then proceeded to configure network connectivity, ensuring seamless communication between the control machine and the target Machine.

Using Ansible, we automated various aspects of the deployment process, including the installation of Docker, creation of Docker networks, and configuration of Apache containers. This automation not only streamlined the deployment workflow but also enhanced repeatability and consistency across deployments.

Throughout the report, insights into each step of the deployment process are provided, including potential issues that may arise and corresponding solutions. By following the outlined steps, users can successfully deploy a Docker container with an HTML page using the Apache web server, leveraging Docker's flexibility and scalability for hosting web applications.

Contents

INTRODUCTION

Over the course of my project, I have embarked on a journey to deploy a Docker container with an HTML page served by the Apache web server. This endeavor has various stages, from setting up the development environment, configuring network connectivity, and deploying the containerized application. My goal is to demonstrate the seamless integration of technologies like Docker, Ansible, and Apache to streamline the deployment process and enhance the accessibility of web applications. During this project, I encountered and overcame challenges, and gained valuable insights into containerization, automation, and network management. By tools such as Ansible for automation and Docker for containerization, I have simplified the deployment process and efficiency in managing infrastructure and applications.

CONTEST

Ansible: It is an automation tool that is free and open source. It can be used for configuration management, deployment of applications and task automation. It enables users to automate the process of managing and deploying servers. In ansible, YAML files are used to define tasks, playbooks, and inventory files. YAML files are structured using indentation and provide an easy -to-read format that simplifies the process of writing and managing automation code.

Docker: It is an open-source containerization platform that allows developers to build, deploy and run applications in containers. Containers are self-contained environments that include all the necessary dependencies to run application including operating system, libraries, and system tools. Containerization platforms such as Docker, provide a standardized way of creating and managing containers. Docker images are templates that are user to create docker containers. Docker images are created using a Docker file, which is a text file that contains instructions for building the image

Homebrew: This is a package manager for macOS and Linux. It allows users to easily install, update, and manage software packages and libraries from the command line. Homebrew simplifies the process of installing software on macOS by automating tasks such as downloading, compiling, and configuring packages.

TECHNICAL DESCRIPTION OF THE PROOF OF CONCEPT

This section of the report I will provide a comprehensive account of the technical aspects and present a step-by-step explanation for deploying docker images onto virtual machines using an ansible playbook.

PROCEDURES -

Set Up VirtualBox:

- I downloaded and installed Oracle VirtualBox from the official website.
- Then Launched VirtualBox and created a new virtual machine.
- I choose Ubuntu as the guest operating system and configured the virtual machine with appropriate settings (network – bridged network).

Installation and Configuration of Ubuntu Environment:

- I downloaded Ubuntu iso file from Ubuntu's website.
- Then started the virtual machine and follow the on-screen instructions to install Ubuntu.
- I completed the installation process, including setting up user accounts and network configurations.
- Then I accessed the Ubuntu environment through VirtualBox.
- Open a terminal in Ubuntu and update the package repositories using "sudo apt update".
- Created an admin user.
- Changed network attached to bridged Adapter, and I chose the bridged adapter mode in VirtualBox to ensure that my Ubuntu virtual machine could seamlessly connect to the network and behave as if it were a standalone device on the same network as my host machine.

Installing Docker –

To download Docker you will need to go to the Docker website and download Docker Desktop for Mac. You can find it at https://www.docker.com/products/docker-desktop. After installing you can launch the application, you can also check the version by running "docker --version" on your terminal.

Install Ansible through Homebrew -

If you have not already installed Homebrew in your laptop, you can do so by running the following command in your terminal:

"/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"

When this is successful, you can easily install Ansible by running "brew install ansible" on your terminal. After the installation Ansible is now ready to use, but before that Ansible is known as an open-source automation tool that simplifies the management and configuration of IT infrastructure using playbooks, which are written in YAML format.

OpenSSH in control machine -

OpenSSH, or Open Secure Shell, is simply tools for secure remote login, file transfer, and command execution between computers over an insecure network. It provides security features such as encryption and authentication to ensure that communication between systems remains confidential and secure. In my project i used it to gain remote control of target machine for ansible to execute the task remotely. To run and Install OpenSSH, I simply did that by running “brew install openssh” on my terminal on VS Code

3.2 **Creating an ansible playbook file**

The below playbook is designed to create a web server environment that comprises sever Docker containers including Apache.

Below are the steps I took when executing my ansible playbook “docker.yaml” -

- **Installation of Necessary Tools:** The first step I took was to install the necessary tools and dependencies required for the deployment process. This includes the installation of Docker and the Docker Python module using pip. The Docker Python module is installed using the pip package manager. This module allows interaction with the Docker API from Python scripts, enabling automation of Docker-related tasks.

- **Creation of Docker Network:** The second step I took after installing Docker was to create a Docker network named "apache_network". This network provides a segregated environment for Docker containers to communicate with each other. The subnet "172.168.10.0/30" is assigned to the network to allocate IP addresses to containers within the network.

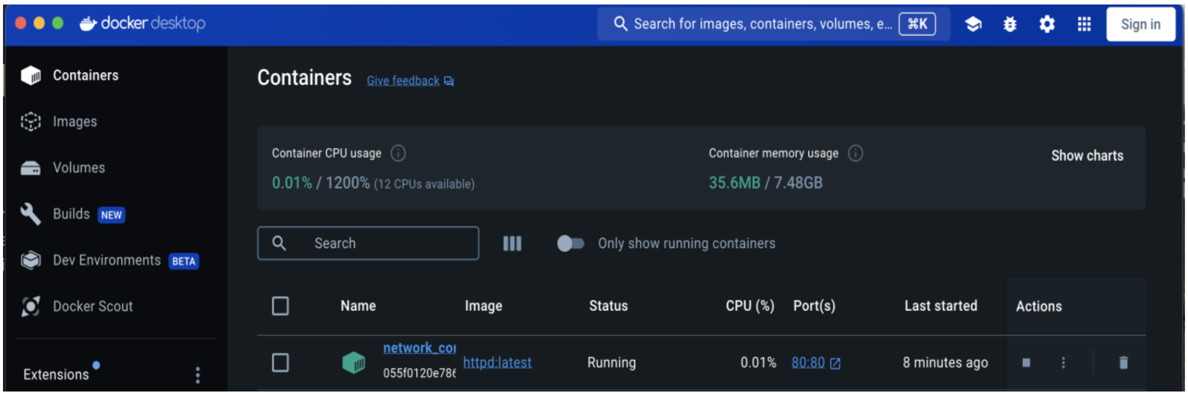
- **Configuration of the Apache Container:** With the Docker network in place, the next thing I did was to create an Apache Docker container using the httpd:latest image and I configured the container to expose port 80, the default HTTP port, allowing incoming web traffic to reach the Apache server. Additionally, I mounted a volume to the container, linking the host machine's file system to the Apache server's document root directory (/usr/local/apache2/htdocs). This allows the container to serve the HTML page stored on the host machine.

- **Copying of HTML File to the Host:** Before I will deploy the container, the HTML file that is to be served by the Apache server is copied from the local file system to a designated directory (/tmp/network) on the host machine. This HTML file contains the content that will be displayed to users accessing the website hosted by the Apache server.

- **Retrieval of Container IP Address:** Once I deployed the container, the IP address is retrieved using the Docker CLI command "docker inspect". This IP address is necessary for establishing connectivity between the Docker container and other components in the network, such as the host machine and external services.

- **Updating Inventory File for Ansible:** Finally, the retrieved container IP address is added to an inventory file named "inventory.txt" for Ansible. Ansible is a configuration management tool that allows for automated deployment, provisioning, and management of infrastructure. By updating the inventory file with the container's IP address, Ansible can remotely manage the container, enabling automated configuration tasks.

- **Running an ansible playbook file:** To run an Ansible Playbook on your terminal, you just need the code “sudo ansible-playbook docker.yaml”. When the code is ran it looks like this -

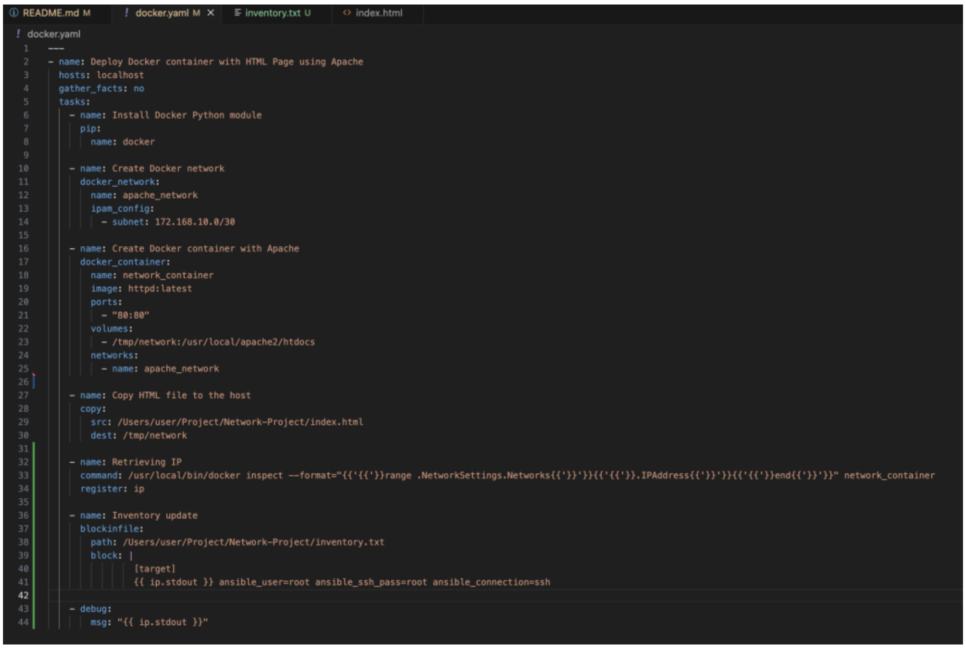


Containers

Container CPU usage 0.01% / 1200% (12 CPUs available)

Container memory usage 35.6MB / 7.48GB

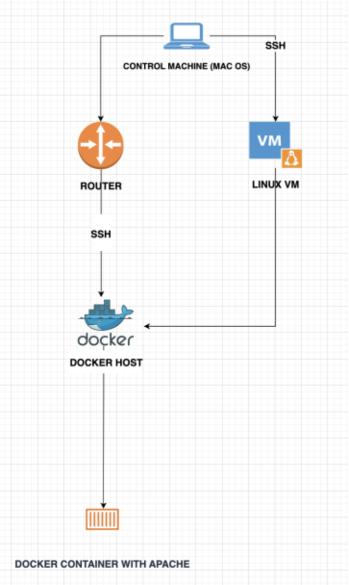
□	Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
□	network_coı 055f0120e78€	httpd:latest	Running	0.01%	80:80	8 minutes ago	



```yaml
---
- name: Deploy Docker container with HTML Page using Apache
  hosts: localhost
  gather_facts: no
  tasks:
    - name: Install Docker Python module
      pip:
        name: docker

    - name: Create Docker network
      docker_network:
        name: apache_network
        ipam_config:
          - subnet: 172.168.10.0/30

    - name: Create Docker container with Apache
      docker_container:
        name: network_container
        image: httpd:latest
        ports:
          - "80:80"
        volumes:
          - /tmp/network:/usr/local/apache2/htdocs
        networks:
          - name: apache_network

    - name: Copy HTML file to the host
      copy:
        src: /Users/user/Project/Network-Project/index.html
        dest: /tmp/network

    - name: Retrieving IP
      command: /usr/local/bin/docker inspect --format="{{'{{'}}range .NetworkSettings.Networks{{'}}'}}{{'{{'}}.IPAddress{{'}}'}}{{'{{'}}end{{'}}'}}" network_container
      register: ip

    - name: Inventory update
      blockinfile:
        path: /Users/user/Project/Network-Project/inventory.txt
        block: |
                [target]
                {{ ip.stdout }} ansible_user=root ansible_ssh_pass=root ansible_connection=ssh

    - debug:
        msg: "{{ ip.stdout }}"
```

Docker.yaml

Diagram on how it works -



CONCLUSION

In conclusion, this report has provided a detailed guide for deploying a Docker container with an HTML page served by the Apache web server. By following the outlined steps, users can effectively set up their Docker environment, configure the Apache container, and ensure seamless access to the hosted HTML page.

The deployment of Docker containers offers numerous advantages, including portability, scalability, and efficiency. Using Docker for containerization allows for the isolation of applications and their dependencies, facilitating consistent deployment across various environments. Additionally, Apache's web serving capabilities ensure reliable delivery of content to users accessing the HTML page.

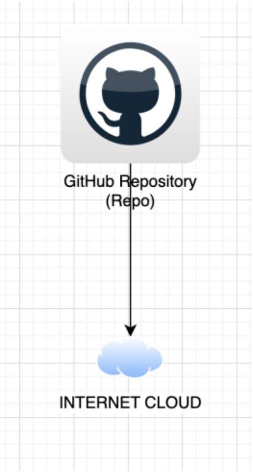
As organizations continue to embrace containerization and microservices architecture, the ability to deploy and manage Docker containers becomes increasingly important. By mastering the deployment process outlined in this report, users can gain valuable skills applicable to both development and production environments.

Furthermore, my report serves as a foundational resource for those seeking to deploy similar setups in their own projects or organizations. Whether for hosting static websites, web applications, or other content, the deployment of Docker containers with Apache web server offers a flexible and scalable solution for modern IT infrastructure needs.

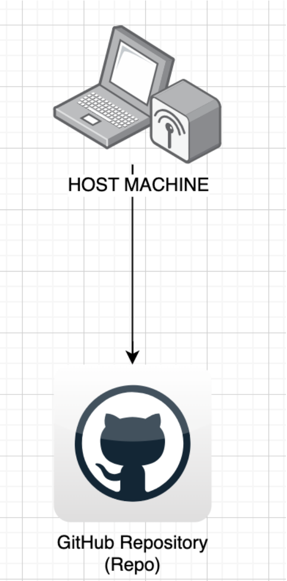
In summary, the deployment of a Docker container with an HTML page served by Apache represents a key milestone in embracing containerization technologies and modernizing IT infrastructure. By following the guidance provided in this report, users can embark on their containerization journey with confidence, empowered to deploy and manage Docker containers effectively in their environments.

NETWORK DIAGRAMS using (DRAW.IO)-

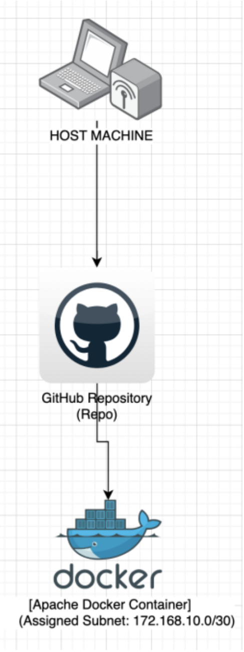
GitHub Repository Network Diagram -



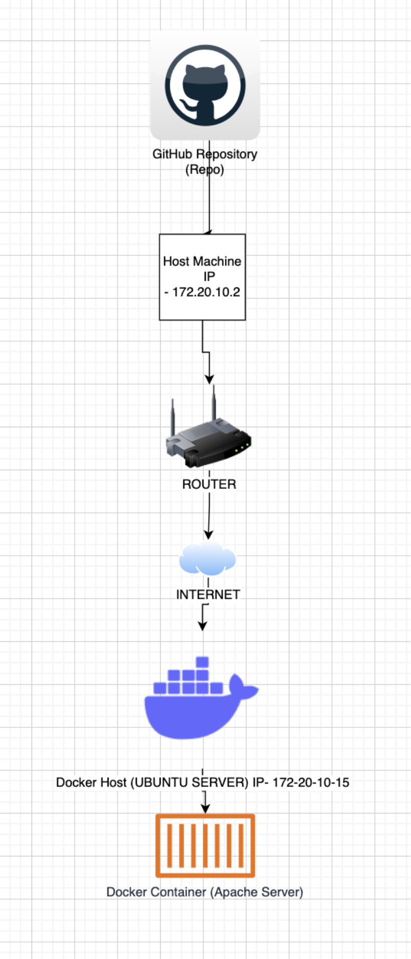
Host Machine Network Diagram -

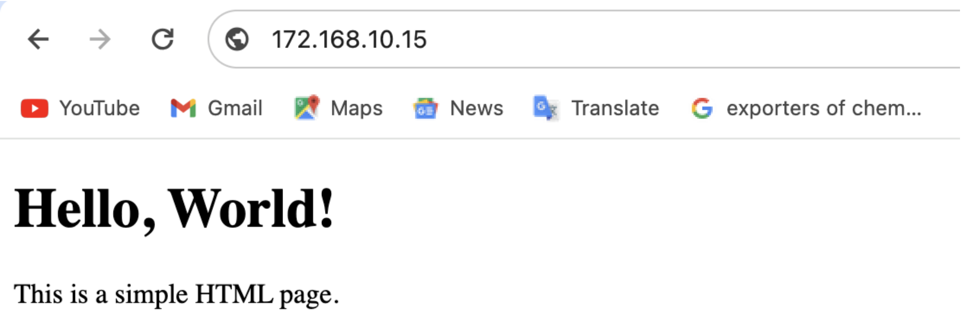


Apache Docker container with assigned subnet -



Connectivity between GitHub Repo, Host Machine and Container -


GitHub Repository
(Repo)
Host Machine
IP
- 172.20.10.2
ROUTER
INTERNET
Docker Host (UBUNTU SERVER) IP- 172-20-10-15
Docker Container (Apache Server)



Hello, World!

This is a simple HTML page.