AUTOMATED CONTAINER DEPLOYMENT AND ADMINISTRATION

Network and Systems Administration | B9IS121 (CA-ONE)

Chitra Prakash Nadig (20025270)

Lecturer: Mr. Kingsley Ibomo

Summary

The tasks listed entail utilizing Ansible and Docker to put up a reliable infrastructure deployment process. Version-controlled infrastructure-as-code (IaC) management is based on the establishment of a GitHub repository, which facilitates easy change monitoring and collaborative development. Using Ansible makes it possible to automate and replicate the deployment of Apache web servers in Docker containers in a variety of scenarios. This automation lowers the possibility of human error and guarantees consistency in server configurations. Additionally, setting up networking for the Docker containers guarantees appropriate service isolation and accessibility, improving security and usability. Ultimately, companies may improve the efficiency, scalability, and maintainability of their infrastructure management procedures by fusing the automation powers of Ansible with the lightweight and portable containerization technology of Docker. This method shortens time-to-market, optimizes deployment operations, and promotes a more agile and responsive IT infrastructure.

Contents

1. Introduction

i. Virtualization

The technique known as virtualization enables you to divide a single physical hardware system into several virtualized environments or resources. By providing a layer of abstraction between the physical hardware and the operating system, applications, or virtual machines (VMs) running on top of it, it abstracts the hardware layer and enhances the efficiency of computing resource use.

ii. Containerization

Applications and their dependencies can be packaged and isolated into independent units called containers using containerization, a lightweight type of virtualization. Containers virtualize the application runtime environment, which makes them more efficient and portable than traditional virtual machines (VMs), which virtualize the complete operating system stack.

- **A few advantages of containerization are as follows:**

1. Applications can be carried across many environments, including development, testing, and production, thanks to containers' ability to encapsulate all dependencies.

2. Lightweight process isolation is made possible by containers, which guarantee that programs operate independently of one another and the host system's underlying components.

3. Resource Economy: Compared to conventional virtual machines, containers have less overhead because they share the host operating system's kernel. Improved resource efficiency and quicker starting times are made possible by this.

4. Scalability: Perfect for microservices architectures and cloud-native applications, containers may be readily copied and scaled up or down to meet fluctuating workload demands.

5. Maintaining consistency in application environments across development, testing, and production environments is possible for developers with containerization, which lowers the possibility of deployment problems resulting from variations in runtime environments.

iii. Docker

It offers a collection of services and technologies that make containerization easier and more accessible for both operations and development teams. With the help of the well-known containerization technology Docker, developers can construct, launch, and maintain containers for their apps with ease.

- **Important Docker components consist of:**

1. The central part of the Docker platform is the Docker Engine. It oversees operating and overseeing containers on the host system. A daemon called Docker and a command-line interface called Docker CLI are both included with Docker Engine to facilitate communication with images and containers.

2. Docker Image: Read-only templates for containers are created using Docker images. They include the application code, runtime, libraries, and dependencies, among other things required to run a particular program. Docker files, which specify the environment and instructions for generating the image, are the basis from which Docker images are constructed.

3. Docker Container: Applications and their dependencies are confined within lightweight, portable, and self-contained runtime environments known as Docker containers. Because containers are isolated from the underlying host system and from one another, consistency and repeatability across many conditions are guaranteed.

4. Docker Registry: A repository for sharing and storing Docker images is Docker Registry. Developers can use Docker Hub, the default registry, to store a large selection of public images as foundation images or to share their own images with the community. Additionally, private registries can be established by organizations to house confidential or proprietary photographs.

iv. Ansible

Ansible is often described as a configuration management tool and is typically mentioned in the same breath as Chef, Puppet, and Salt. Writing down a state description for our servers and using a tool to verify that they are in that state that is, that the correct packages are installed, that configuration files have the expected values and permissions, that the correct services are operating, and so forth are the common practices associated with configuration management. Ansible provides a domain specific language (DSL) for describing the status of our servers, just like other configuration management solutions.

2. Setting up Infrastructure

i. Set up control node and manage node with Ubuntu OS in EC2 instances of AWS.

- The control node and the managed node are set upon a virtual machine, EC2 instances.

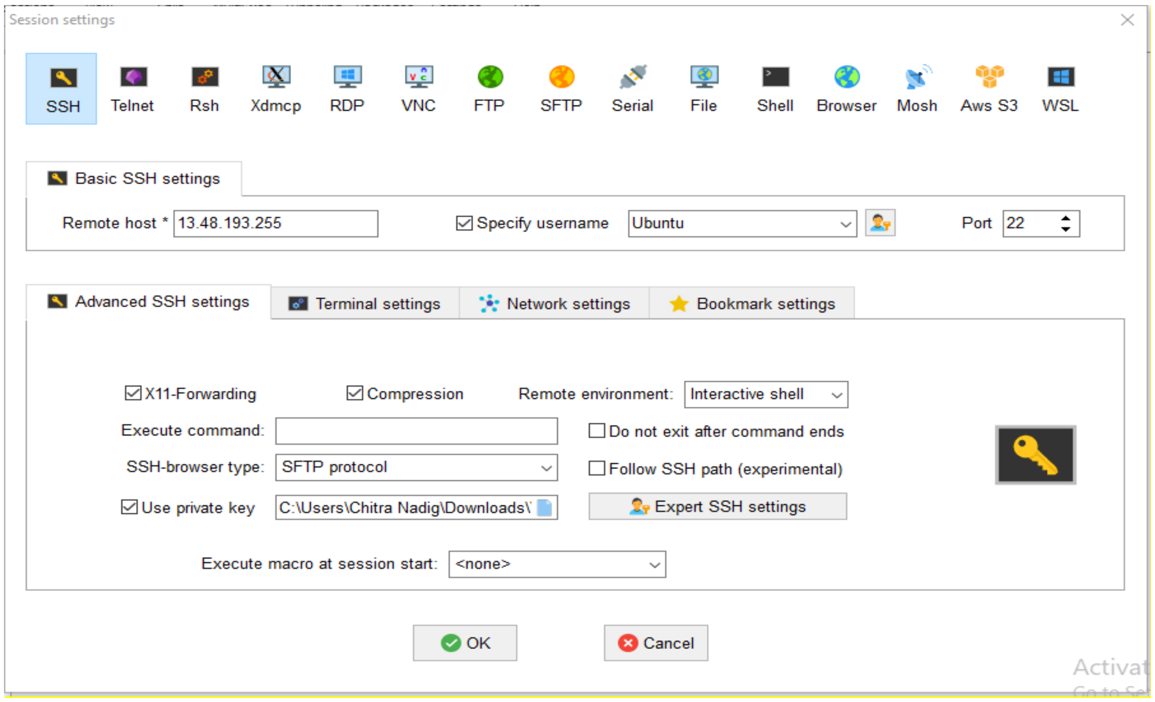
- Downloaded the mobaxterm utility and used the private pem file to log into both instances of the VMs.
- Copied the PEM file that was obtained and mapped it to the public IPv4 of the control and target node instances so that we can SSH into the VMs.

  We executed the following command from the local machine to do this.

  ssh -i C:\Users\Chitra Nadig\Downloads\ VM.pem ec2-user@13.48.193.255 for the control node

  ssh -i C:\Users\Chitra Nadig\Downloads\ VM.pem ec2-user@16.171.11.56 for Target node

- Launching the mobaxterm utility now, we ssh onto the computer by providing each node's IPV4 address and then setting the username to "ubuntu." An example of a screenshot is displayed.



Remote host * 13.48.193.255 ☑ Specify username Ubuntu Port 22

☑ X11-Forwarding ☑ Compression Remote environment: Interactive shell

Execute command: ☐ Do not exit after command ends

SSH-browser type: SFTP protocol ☐ Follow SSH path (experimental)

☑ Use private key C:\Users\Chitra Nadig\Downloads\

Execute macro at session start: <none>

Following the login process for each instance, install Ansible on the control node and a Docker container containing the Apache web service on the target node.

ii. Installation of Ansible on ‘control node’.

- Opened the terminal and executed the below commands.
- Updated the package list: sudo apt update.
- Updated the python package list list: sudo apt install python3- pip

- Installed ansible: sudo pip install ansible.

iii. Creation of ansible configuration file, inventory file, deploy_apache_container.yml file

- 'ansible.cfg' is the ansible configuration file where the inventory path and filename is mentioned. 'ansible.cfg' file path: /etc/ansible/
- 'ip.txt' is the host file where the target machine's (managenode) IP address and user credentials are mentioned. 'ip.txt' file path: /home/ubuntu/
- The ansible playbook file for every task is called "deploy_apache_container.yml." Path to "deploy_apache_container.yml" is /home/ubuntu/playbook/deploy_apache_container.yml.

3. Ansible Playbook Creation

i. Create an ansible playbook to define the tasks.

Develop a YAML file called deploy_apache_container.yml and define the tasks required to deploy containers to develop an Ansible playbook for container deployment.

The tasks included in the playbook are as follows:

- Use the Ansible Docker module to pull the Apache container image to serve a static web page on port 80.
- Using the Ansible Docker module, pull the httpd image to enable network connectivity for TCP service on port 80.
- Building a container and setting it up to operate on the 172.168.10.0/30 network.
- Building a container to operate on another subnet than the one used by Apache.

ii. Use Ansible's Docker module to pull the required container images from Docker Hub.

The necessary container pictures are pulled from the playbook using 'my_image'. The image name is specified by the name parameter, and to guarantee that the image is pulled from the Docker hub, the source parameter is set to "pull." Users can grab images from Docker Hub by replacing the placeholder names with the actual names of the images.

iii. Based on the retrieved photos, define tasks to start and generate the required containers.

The playbook contains tasks for creating and starting containers using the 'docker_container' module. The name of the container (my_apache1_container) is specified by the 'name' argument. The image to utilize is specified by the 'image' parameter and the port mappings for the container are defined by the 'ports' parameter.

iv. Establish tasks for building networks.

The playbook contains tasks for managing the Docker networks using the 'networks' module. The name of the Docker network you wish to construct (my_network) is specified by the 'name' option. In this instance, you are creating a network subnet. "The subnet range that you wish to assign to the network is specified by the "subnet" option. The "register" option is employed to record the outcome of the

4. Docker Network Configuration

i. To establish a unique network for the containers, use the Docker network commands.

Use the 'docker_network' instructions to set up a network for the containers. This is an illustration of how to set up a network called "my_network." use the command "docker network create 'my_network'" to construct a docker network. With the name "apache_network," this command will establish a brand-new Docker network.

ii. Apache image configured for the Docker container service.

Actions performed in the target node virtual machine to launch an Apache Docker container with the Apache image httpd.

- Use sudo apt install docker.io -y to install Docker.
- TO enable the docker we run this command: sudo systemctl enable docker
- Create a new directory called as docker inside /home/ubuntu to stor the Docker configured file. Below is the screenshot of the Docker configured file:

- We Use the official httpd image as the base image
  FROM httpd:latest

- Copy the static HTML file into the Apache document root directory
  COPY index.html /usr/local/apache2/htdocs/

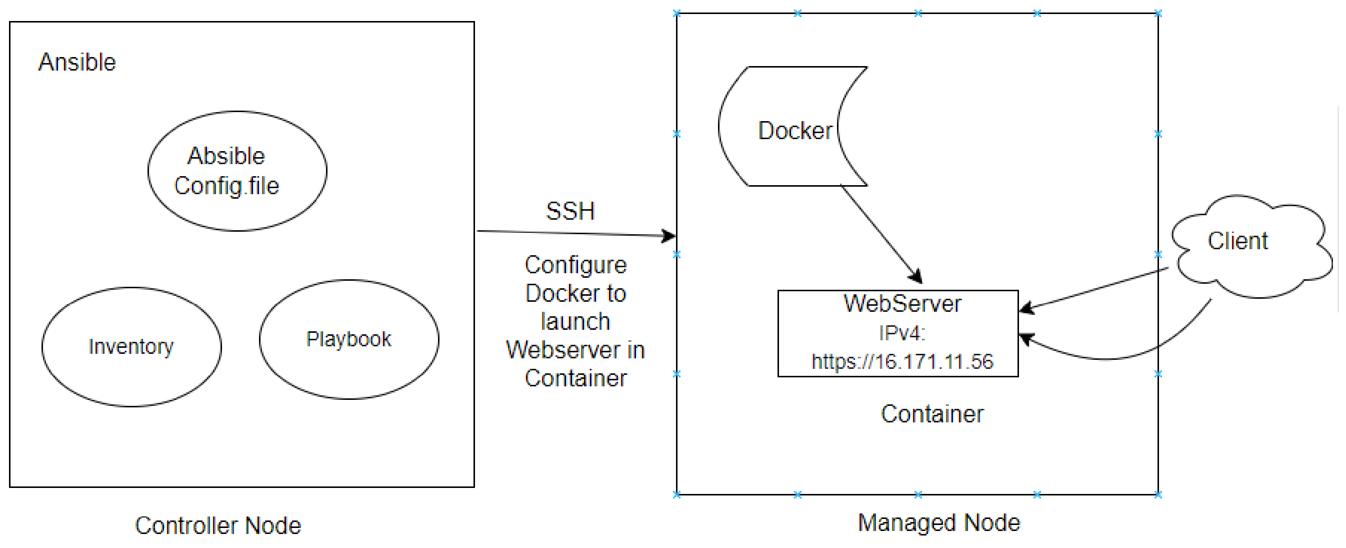
- Expose port 80 to allow external access to the Apache server
  EXPOSE 80

- Then we build a docker image to run the Apache docker container my_apache1_container using this command:
  sudo docker build -t my_image .

- We map this container to the port 80:80 and run it using the command:
  sudo docker run -d -p 80:80 --name my_apache1_container my_image

- We can see in the below screenshot of the Apache container running on the port 80

5. Network Diagram



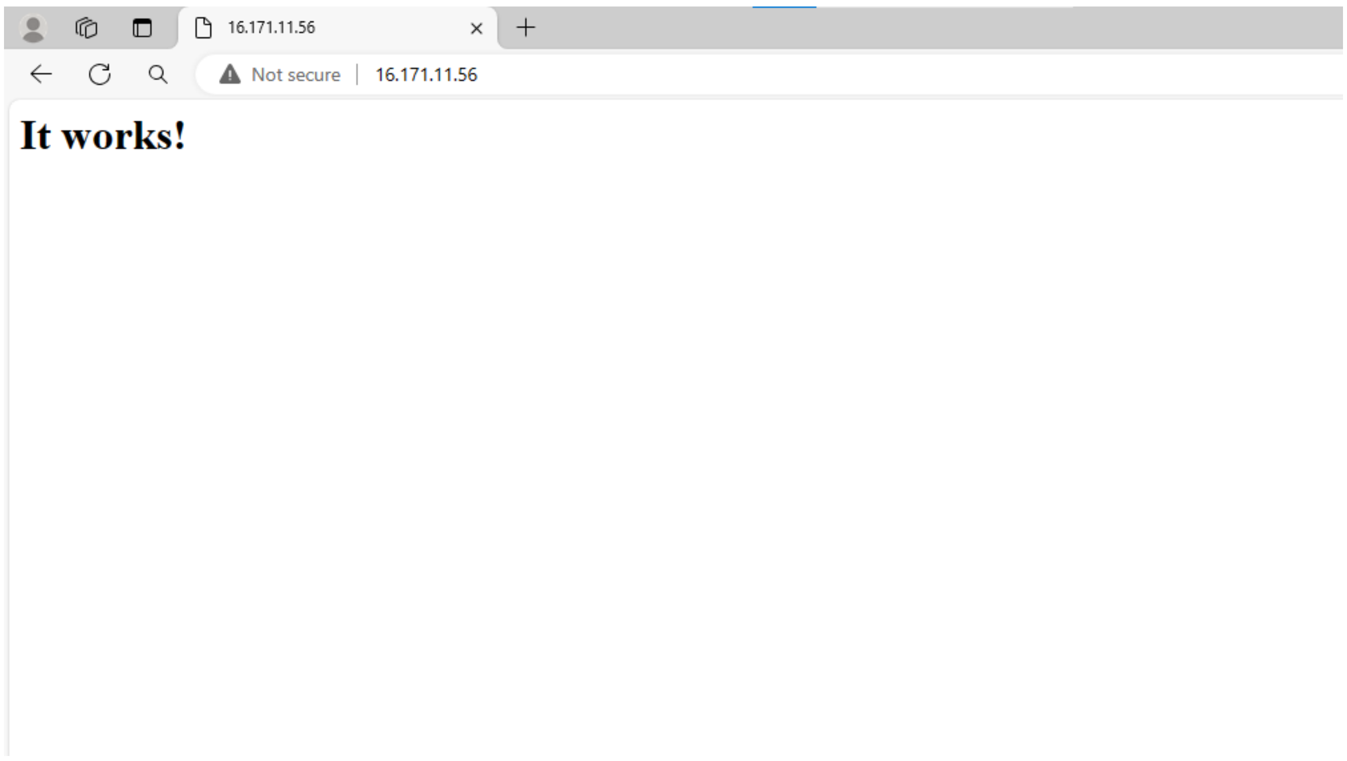
6. Testing and Validation

i. Execute the Ansible playbook to deploy the containers.

To run the ansible playbook, use the ‘ansible-playbook’ command followed by the ‘playbook filename’. Run the following command: “sudo ansible-playbook deploy_apache_container.yml”.

ii. Verify that the containers are successfully deployed and running.

After executing the ansible playbook, verify that the containers are successfully deployed and running using docker commands. Here are a few commands: “docker ps”. This command lists all running containers. Make sure that the containers deployed are present in the list.



It works!

7. Conclusion

To sum up, the combination of Ansible with Docker offers a powerful way to streamline deployment and infrastructure management processes. Ansible's automated playbooks simplify the deployment of Apache web servers within Docker containers, guaranteeing a consistent, scalable, and effective deployment in a variety of IT scenarios. The creation of a GitHub repository strengthens version control and teamwork in developing infrastructure settings, making it easier to track changes and follow best practices. Furthermore, the infrastructure's resilience is strengthened by optimizing accessibility and security through Docker container networking. Essentially, the combination of Docker's lightweight containerization capabilities with Ansible's automation skills allows enterprises to move more quickly, reliably, and easily through the digital landscape, which in turn promotes long-term corporate growth and innovation.

8. References

https://abhayagarwal4483.medium.com/configuring-apache-httpd-server-python-on-docker-container-5552e0c40797

https://medium.com/@dwivedipiyush9754/set-up-the-apache-web-server-within-the-docker-container-utilizing-an-aws-ec2-instance-b12a6910af66

https://www.linkedin.com/pulse/devops-configuration-httpd-docker-container-target-node-sharma?utm_source=share&utm_medium=member_android&utm_campaign=share_via

9. Bibliography

https://www.youtube.com/watch?v=TWcFAi6rvlc

https://www.youtube.com/watch?v=1id6ERvfozo

https://kodekloud.com/courses/ansible-for-the-absolute-beginners-course/