

**Automated Container Deployment- Apache Ansible.**

**Network Systems and Administration**

**B9IS121**

**Professor Kingsley Ibomo**

**By**

L. Chandu 20026627

**Content**

|  |  |
| --- | --- |
| Introduction | 03 |
| Lab Setup | 04 |
| Ansible Playbook Creation | 06 |
| Deployment Process | 08 |
| Network Configuration | 10 |
| Network Diagram | 10 |
| Conclusions | 11 |
| References | 11 |

**Introduction**

The Network Systems and Administration CA focuses on testing students' Ansible-based Docker container deployment skills. The goal of this hands-on, lab-based evaluation is to gauge how well students comprehend and use automation and containerization technologies. A GitHub repository must be created, an Ansible playbook must be written, an Apache Docker container must be deployed, networking must be configured, and a network diagram must be created to show the deployment. The proficient execution of these assignments showcases the learners' proficiency in automating container deployment and administration procedures, which are crucial abilities in contemporary IT infrastructure management.

**Methodology**

The evaluation process consists of a set of structured assignments meant to gauge students' ability to deploy Docker containers using Ansible. It is mandatory for students to establish a publicly accessible GitHub repository that houses all pertinent materials, such as the Ansible playbook. The playbook uses Ansible's powerful features to automate the deployment process through careful design. Students launch an Apache Docker container, set networking parameters, and verify that the deployed services are accessible by carrying out pre-established activities. Students are also required to create a detailed network diagram that illustrates the deployment architecture and connections.

**Results**

The accomplishment of the assessment assignments shows how proficient the students are with automation and containerization technology. Students successfully deploy Docker containers using Ansible by carefully following the given specifications and recommended practices. Establishing a GitHub repository makes version control and teamwork easier, which guarantees smooth project management. Moreover, the deployment procedure is streamlined by the Ansible playbook, allowing for effective and repeatable container deployments. The deployment architecture is clearly visualized by the network diagram, which also emphasizes the connections between internal and external components.

**Lab Setup**

A proper lab setup is essential to guarantee the assessment's effective completion. In order to deploy containers, this system requires that Ansible be installed on the control machine for automation and Docker installed on the target machines. Furthermore, version control necessitates access to a public GitHub repository in order to enable cooperation and the sharing of deployment artefacts. A well-prepared lab environment guarantees that the deployment process runs well and gives students an effective opportunity to demonstrate their abilities.

**Installation of homebrew & Ansible**

On a Mac, you must first use Terminal to install Homebrew. Terminal is in the Applications directory's Utilities folder. Alternatively, Spotlight Search (Cmd + Space) to search Terminal.  
  
To install Homebrew, you must paste a command into Terminal after it is open. Using its official repository, this command installs Homebrew and downloads its installation script. This is the directive:

**/bin/bashc"$(curlfsSLhttps://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)”**

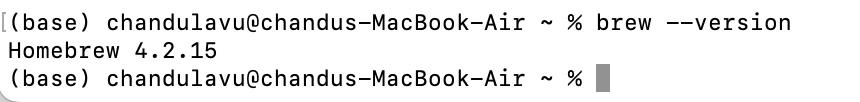
A screenshot of a computer program

Description automatically generated

**Fig 1 homebrew installation**

To run the command after pasting it, hit Enter. You may be asked to confirm the XCode license terms or provide your macOS user password during the installation process. As needed, adhere to the prompts. After the installation is finished, use the following command in Termina to confirm that Homebrew has been installed correctly. Once you hit Enter, the Terminal should show the Homebrew version that is installed on your machine. With Homebrew installed, you can use the command line to install a variety of programs and apps quickly and easily on your Mac. To install a package, just type its name followed by brew install, and Homebrew will do the rest.

“**brew –version”**

****

**Fig 1.1 homebrew version**

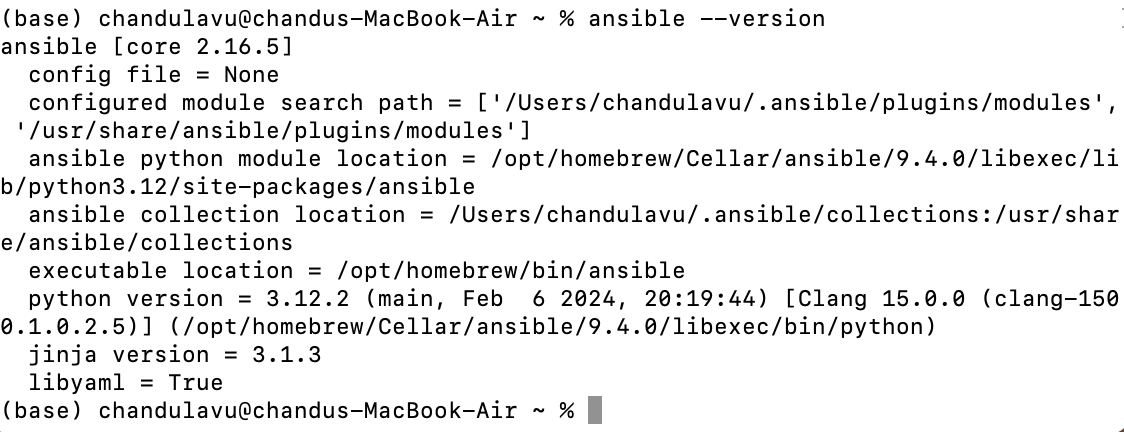
Now that Homebrew has been updated, you can install Ansible. Just type the following command into Terminal and hit Enter

**“brew install ansible”**

**A computer code with black text

Description automatically generated**

**Fig 1.2 ansible installation**

****

**Fig 2.1 ansible version**

**Installation of Docker Desktop**

Docker Desktop can be installed on a Mac by downloading the package from the Docker website, dragging the Docker icon into Applications, and starting the program. Turn on Hyper kit when requested. You can also sign into Docker Hub. Check the installation with the commands in Terminal. Your Mac can now create and manage containers using Docker Desktop.

**Ansible Playbook Creation**

---

- name: Deploy httpd container with Git installed

hosts: localhost

connection: local

gather\_facts: no

tasks:

- name: Create Docker network

docker\_network:

name: my\_network

driver: bridge

ipam\_config:

- subnet: 172.168.10.0/30

- name: Pull httpd Docker image

docker\_image:

name: httpd

source: pull

- name: Install Git inside the container

docker\_container:

name: my\_httpd\_container

image: httpd

state: started

networks:

- name: my\_network

ports:

- "8080:80"

command: /bin/bash -c "apt-get update && apt-get install -y git && exec httpd -DFOREGROUND"

The deployment of a httpd Docker container with Git installed is automated by this Ansible script. Let's dissect each task:

Establish a Docker Network: ‘Docker network' is the Ansible task. Action: Establishes a bridge driver and IP address management (IPAM) configuration for a Docker network called "my network," with the subnet "172.168.10.0/30" specified.  
Pull Docker Image for httpd: Task for Ansible: "docker image"  
Operation: Retrieves the httpd Docker image from the registry of Docker Hub. The image will be downloaded if it is not already on the host.

Essentially, this playbook ensures that a httpd container with Git installed and reachable on port 8080 of the host system is deployed by automating the establishment of a Docker environment, including network configuration, image retrieval, and container instantiation with particular parameters and commands.

The deployment of a httpd Docker container with Git installed is automated by this Ansible script. 'my network' is the name of the Docker network that is first created, along with a chosen subnet. The httpd Docker image is then retrieved from the Docker Hub registry. Next, using the 'httpd' image as a basis, it builds and launches a Docker container called 'my\_httpd\_container'. The httpd service can be accessed because this container is set up to use the 'my network' network and translates port 8080 from the host to port 80 on the container. Furthermore, by running commands to update package lists, install Git, and launch the httpd service in the foreground, the script installs Git inside the container. Because of this thorough process, the httpd container with Git installed is deployed smoothly and is ready to be used in the specified environment.

**Deployment Process**

To install an Apache Docker container, the steps specified in the Ansible playbook must be carried out. Students define tasks to start and control the containerization process using the docker container Ansible module. By following predetermined guidelines and goals, students successfully deploy the Apache service within the Docker container by carrying out their duties in a methodical manner.

Make sure Ansible is installed on your computer before attempting to launch an Ansible playbook file. Get into the directory where your playbook file is located by using a command prompt or terminal. To launch the playbook, type "ansible-playbook" and then the filename of the playbook (e.g., docker\_deploy.yml). If encrypted variables are used in your playbook, answer with the Ansible Vault password when prompted. Once the playbook execution is finished, check the summary of tasks completed and modifications made, and keep an eye on the playbook execution in real-time for task status updates.

A screenshot of a computer program

Description automatically generated

**Fig 2.2 Running of yml file**

Then if we check the local host the result would be like this for this docker\_deploy.yml

A screenshot of a chat

Description automatically generated

**Fig 3.1 Result for running of yml file**  
After this there are some tasks to be checked as to inspect the container which is running and have t check if it is running on the specified subnet or not here is the proof for thatA white screen with black text

Description automatically generated

**Fig 3.2 Inspection of the container**  
  
  
A screenshot of a computer

Description automatically generated

The Ansible playbook deployment method makes it possible to provision an Apache Docker container with ease. Students carefully plan and carry out activities to start and manage containerization using the docker container Ansible module, making sure that all needs and goals are accomplished. This methodical technique demonstrates expertise in automation and containerization technologies and guarantees the successful deployment of the Apache service within the Docker container.

**Network Configuration**

In order to facilitate communication between deployed containers and external entities, networking configuration is essential. Students define tasks in the Ansible playbook to set up networking for the Apache Docker container as part of this assessment. This involves making sure the Apache service is reachable from the host computer and configuring the container to use the designated subnet (172.168.100/30). Students create a reliable communication architecture by carefully setting networking parameters, which makes it possible for deployed containers to communicate with external systems.

**Network Diagram**

**A diagram of a computer

Description automatically generated**

The Ansible playbook and related artefacts are hosted on the GitHub Repository. The machine known as the "host machine," or "localhost," runs the playbook and coordinates the Docker container's deployment. The Apache service is housed in a container called "my\_httpd\_container," which is deployed using Ansible. By connecting to the Docker network "my\_network" using IP address 172.168.10.1, it makes networked environments more conducive to communication. By simplifying the deployment procedure, this configuration makes it possible to integrate and administer the Apache server inside the Docker container with ease.

**Conclusion**

Finally, the Network Systems and Administration CA exam gives us a real-world chance to show that they know how to use Ansible to deploy Docker containers. we demonstrate our ability to automate complex IT infrastructure operations by successfully completing activities relating to networking configuration, playbook writing, container deployment, and verification. we make contributions to the development of automation and containerization technologies—two crucial elements of contemporary IT environments—by adhering to best practices and paying close attention to detail.

**References**

Rehkopf, B. M. (n.d.). *Epics | Atlassian*. Atlassian. <https://www.atlassian.com/agile/project-management/epics>

Benselim, M. S., & Seridi-Bouchelaghem, H. (2013). Extending UML class diagram notation for the development of context-aware applications. *Journal of Emerging Technologies in Web Intelligence*, *5*(1), 35-44. <http://www.jetwi.us/uploadfile/2014/1219/20141219052249905.pdf>

Li, Q., & Chen, Y. L. (2009). Data flow diagram. In *Modeling and Analysis of Enterprise and Information Systems* (pp. 85-97). Springer, Berlin, Heidelberg. <https://link.springer.com/chapter/10.1007/978-3-540-89556-5_4>

Song, I. Y., Evans, M., & Park, E. K. (1995). A comparative analysis of entity-relationship diagrams. *Journal of Computer and Software Engineering*, *3*(4), 427-459. <https://www.cin.ufpe.br/~in1008/aulas/A%20Comparative%20Analysis%20of%20Entity-Relationship%20Diagrams.pdf>

McGill, M. J. (2001). UML Class Diagram Syntax: An Empirical Study of Comprehension. <https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=fae00885887732fcf1769170d4b8514e3730e358>

Al-Fedaghi, S. (2021). UML sequence diagram: an alternative model. *arXiv preprint arXiv:2105.15152*. <https://arxiv.org/abs/2105.15152>

Suriya, S., & Nivetha, S. (2023). Design of UML Diagrams for WEBMED-Healthcare Service System Services. *EAI Endorsed Transactions on e-Learning*, *8*(1). <https://eudl.eu/doi/10.4108/eetel.v8i1.3015>

Ma, L., Zhao, H., You, S. J., & Ge, W. (2018, May). Analysis and design of hospital management information system based on UML. In *AIP conference proceedings* (Vol. 1967, No. 1). AIP Publishing. <https://pubs.aip.org/aip/acp/article-abstract/1967/1/040012/821299>