



Automated Container Deployment and Administration

Instructor: Kingsley Ibomo

Student Name: Md. Ashraf Uddin

Student Number: 20024497

Student Name: P S Aakash Krishna

Student Number: 20019783

Summary:

The report is all about the automatic container deployment and administration using Ansible, Docker and Apache HTTP Service.

The main objective of doing this report is to deploy an IP address using ansible playbook and run it to the Apache server.

The key features of this Assignment:

Ansible Playbook: An ansible playbook is used to utilize the automatic deployment of the process, which also ensures the consistency and the efficiency. Ansible playbooks can be used to define the desired state of your infrastructure and execute the tasks which has been given to the command.

Docker Containers: Docker container is an executable package which contains everything needed to run a machine, including the codes, tools, libraries and all the other settings. It is based on the docker images. On the docker images everything is written step by step and docker containers just perform it based on the images.

- **Apache container:** Here we I installed and used apache container service with a static web page which allows for HTTP access only.

Network Configuration:

This is one of the most important things which ensure connectivity and the accessibility between the containers and the host machine.

- I used a specific subnet which has been given for the network isolation and this is because of the security too just to filter the network traffic and minimizing the unauthorized access.
- I set up the port mapping for seamless interaction between the host machine and container.
- The accessibility is ensured between the host machine and the container itself.

This deployment and the administration process offers a lot benefits which includes automation, consistency, scalability, and portability.

Table of Contents:

SL	Topics	Page No.
1.	Introduction	3
2.	Installation Process All Installation on the Control/Host Machine	3
	Establishing the connection between Control Node and Managed Node:	6
	Installing Docker and Apache into the Managed Node	6
3.	Ansible Playbook creation	7
	Network Diagram	10
	Results	10
4.	Conclusions	10
5.	References	11
6.	Results	12

1. Introduction

This report is prepared for the documentation of deployment of a web server using docker containers and ansible. This report provides a full detailed overview and walk through of the whole deployment process and steps taken on every step.

Objective:

1. This report has demonstrated the clear understanding of the practical application and usage for deploying docker containers using ansible.
2. It includes the step-by-step process for setting up the virtual environment, host, and target machines, creating customized ansible playbook, docker deployment, network configuration using custom subnet and verify the accessibility of the network.
3. Used a virtual environment like AWS for making two separate virtual machines and installed ansible and docker individually to the controller and target machines.
4. A network diagram has been provided to illustrate how the works has been done in the control and target node.

2. Installation Process:

Initially I have created an account to the AWS amazon for creating two virtual machines. After that I created two virtual machines in the Amazon EC2 instance named VM-1-Control Node and another one is VM-2-Managed Node and I installed Amazon Linux on it.

All Installation on the Control/Host Machine:

On the control machine first, I updated all the things. Then I have switched to the root user.

Command:

'sudo yum update' this command is used for updating the installed packages on the system to the latest versions which is available.

'sudo yum install ansible' this command is used for installing the ansible in my control machine. Then I press 'y' for confirming the installation without any restrictions.

'ansible --version' this command is used for verifying whether the ansible is installed properly or not.

Now I have created an inventory for the Ansible but Ansible does not know the location of the inventory. As a result, first I have created a configuration file for ansible which gives the location of the inventory. Whenever ansible runs any command, it first goes to the config file which has been set and then run the command. I named it as ansible.cfg

Command:

'vim /etc/ansible/ansible.cfg' after running this command there an edit option has come and I pressed 'i' for inserting all my commands inside it. I have set the following things in the configuration file

[defaults]

Inventory = /home/ec2-user/myhosts.txt

Ansible_host_key_checking=FALSE

Host_key_checking=FALSE

After all these commands the default configuration is set up for the ansible's communication.

In the /home/ec2-user directory I have created a file named myhosts.txt where I put a name for the Target node and it is Ip address.

Command:

'vim myhosts.txt' it directs to the edit option and after that I clicked 'i' to insert whatever I needed to. I inserted

[Managed_Node1] → named

172.31.30.123 → The target/managed node's Ip address

Then I have created an index.html file which will be showed in the website when it loads.

Command:

'vim index.html' It goes to the file and

```
<html>
```

```
<head></head>
```

```
<body style="background-color:powderblue;" data-new-gr-c-s-check-loaded="14.1162.0" data-gr-ext-installed="">
```

```
<h1 style="color:red; text-align: center;">!!! Automated Container deployment and Administration !!!</h1>
```

```
<h1 style="color:purple;text-align: center;">Deploy a Docker container using Ansible, running Apache service with a static web page</h1>
```

```
<ul>
```

```
<h3 style="color:green;text-align: center; font-size:30px">!! Docker !!</h3>
```

```
<h3 style="color:brown;text-align: center; font-size:30px">!! Ansible !!</h3>
```

```
</ul>
```

```
</body>
```

```
<grammarly-desktop-integration data-grammarly-shadow-root="true"></grammarly-desktop-integration>
```

```
</html>
```

This is the Html code I have used.

Establishing the connection between Control Node and Managed Node:

After configuring these two files I have established a connection between the two nodes Control Node and the Managed Node. For establishing the connection, I have used the following commands and generated a key which has been used both for connecting simultaneously.

Command in Control Node:

'ssh-keygen' it creates three keys called 'authorized_keys', 'id_rsa', 'id_rsa_pub' .

Then I have gone to the directory of rsa_pub for the rsa key. I copied the rsa key from the Control Node and go to the Managed Node.

'vim /.ssh/authorized_keys' I configured the Control Node's key and pasted it there for connecting the two devices with common key.

Installing Docker and Apache into the Managed Node:

In the managed node I have installed Docker and apache server for running the docker, start, and enable it.

Command:

'sudo yum update' for updating everything which is already installed in the managed node linux system.

'sudo yum install docker' I ran this command for installing the docker.

'docker --version' for checking whether it's installed correctly or not.

'sudo systemctl start docker' for starting the docker service in the managed node.

'sudo systemctl enable docker' for enabling docker service to start on boot.

This is the full docker installation and running commands in the managed node I have followed.

Apache:

'yum install httpd' This command is used for installing Apache HTTP Server on the managed node.

'systemctl start httpd' is used to start the Apache service.

'systemctl enable httpd' for enabling Apache service to start on boot.

'systemctl status httpd' For checking the status whether Apache service is running properly or not.

3. Ansible Playbook creation

Inside to Control Node I have named an Ansible playbook file `docker_deploy.yml` where the full deployment of the docker containers will be occurred. Here I have deployed the docker container for running the apache container with a web page and a dedicated IP subnet in it. I have run the `docker_deploy.yml` file playbook.

The following command is used for creating and structuring the `docker_deploy.yml` file.

Command:

'`vim docker_deploy.yml`' for inserting all the commands inside the yml file.

The ansible playbook contents are:

name: Deploy Apache Docker Container using Ansible

hosts: Managed_Node1

become: true

tasks:

- name: Configure Docker Repository

yum_repository:

name: Docker

description: "Docker Repo"

baseurl: "https://download.docker.com/linux/centos/docker-ce.repo"

gpgcheck: no

when: ansible_os_family == "RedHat"

- name: Install Docker (Debian)

apt:

name: docker.io

state: present

when: ansible_os_family == "Debian"

- name: Starting Docker Daemon (Debian)

service:

name: docker

```

    state: started
when: ansible_os_family == "Debian"

- name: Pull Apache Docker Image
  docker_image:
    name: httpd
    tag: latest
    source: pull
  register: z

- name: Creating a persistence Volume Directory
  file:
    path: "/home/ec2-user/httpd"
    state: directory

- name: Copy index.html to host directory
  copy:
    src: "/home/ec2-user/index.html"
    dest: "/home/ec2-user/httpd/index.html"

- name: Configure Networking for Apache Container
  docker_network:
    name: apache_network
    driver: bridge
  ipam_config:
    - subnet: "172.168.10.0/30"

- name: Run Apache Docker Container
  docker_container:
    name: apache-httpd
    image: httpd

```



```
state: started
restart_policy: always
exposed_ports:
  - "80"
ports:
  - "8888:80"
volumes:
  - "/home/ec2-user/httpd:/usr/local/apache2/htdocs"
networks:
  - name: apache_network
```

This playbook command is used to deploy Apache within a docker container.

For running the playbook, I used the following command 'ansible-playbook docker_deploy.yml' I have run this command to execute the ansible playbook which has a set of instructions and ansible will follow it to automate the process on deploying the remote servers. Ansible reads this playbook and defined the tasks automatically that on the target hosts it should be specified.

Here,

'name' is indicating the purpose of the playbook.

'hosts' it specifies the tasks will be executed on the node which is named 'Managed_Node1'

'become: true': It indicates that the task is using the required elevated permissions before executing the tasks.

'tasks: ': It contains the list of tasks which ansible will perform actually.

'name: ': Configure Docker Repository' Here it indicates that Ansible will configure the Docker repository and this is the descriptive name of the task.

'yum_repository: ': This is used for managing YUM repositories

The rest will do the same work for the Ansible playbook as given.

For running the apache web server, I have setup the firewall security from the AWS firewall setting and changed the inbound settings which will allow all the traffics from 8888 port.

Network Diagram:

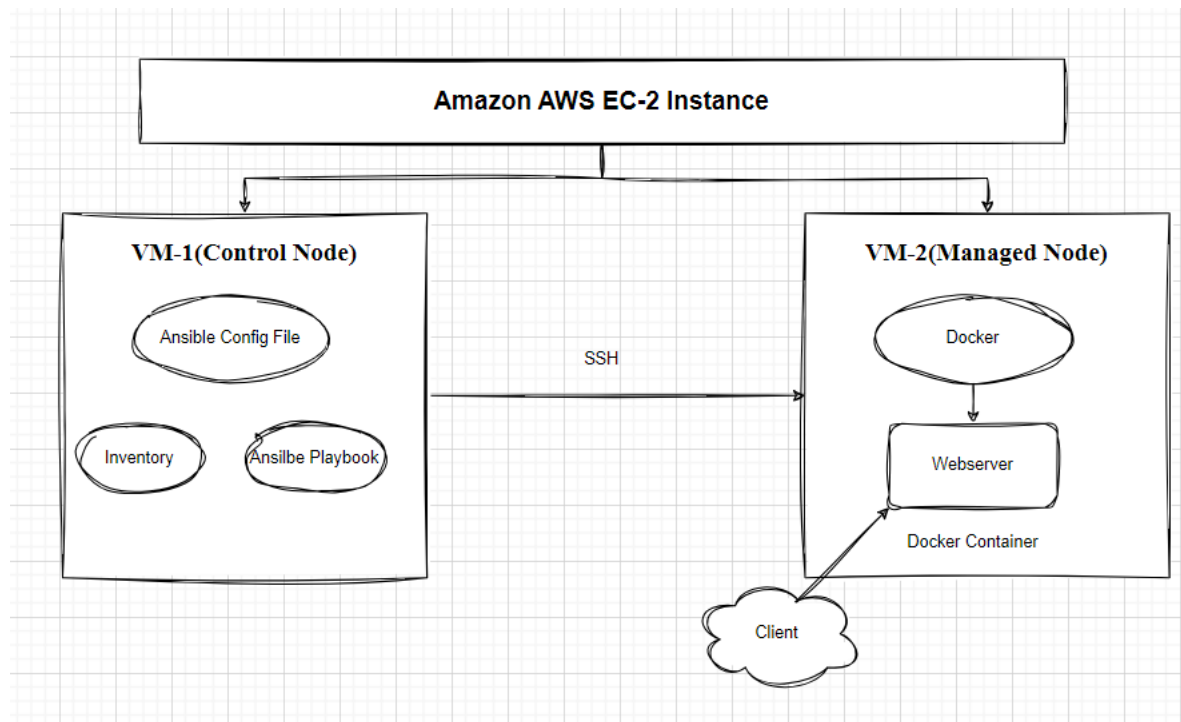


Fig: 1 (Network Diagram)

Results:

- Docker container is running and deployed successfully on the Apache Service using the Ansible.
- Configured a specified subnet (172.168.10.0/30) for the networking apache container and it is accessible from the host (managed node) machine.
- Public GitHub repository is created for containing the ansible playbook file, host file and other configuration and necessary instructions.
- Provided a network diagram for illustrating the process and for better understandings.

4. Conclusions:

This docker containers deployment and administration were achieved through the Ansible automation process. We have used AWS EC-2 Instances for establishing two Virtual Machines for running this full deployment process and Vm-1 is for Control Node and Vm-2 is for Managed Node and we have streamlined the process and ensured the consistency and reliability. The configuration was successful for the Apache container and it established seamless communication with the host machine.

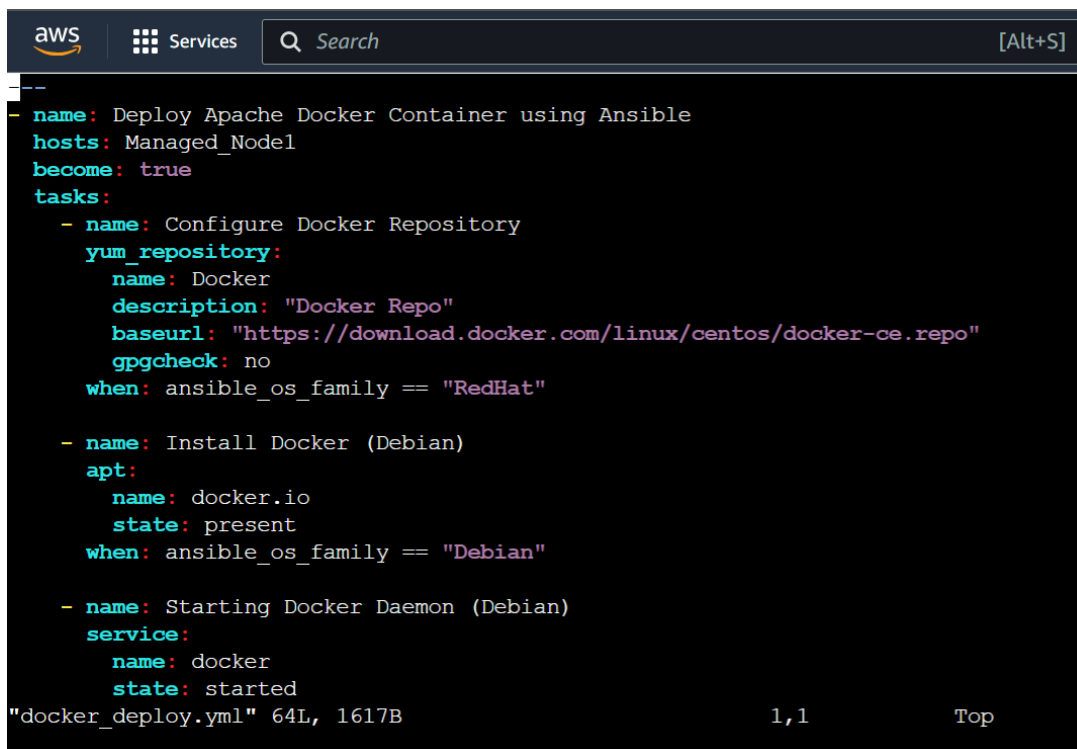
Also, a public repository in GitHub is being created for storing the Ansible playbook file, host.txt file, and the demonstrating the full process of the work in the GitHub Readme file.

Overall, this assessment is done for demonstrating the practical application of deploying a docker container based on the docker image using ansible from the control node and it also highlights the very importance of automation and proper networking configuration which can be controlled or centralized in one machine only.

5. References

- [1] "Home." (2024, March 7). Docker Documentation. <https://docs.docker.com/>
- [2] Community, A. (n.d.). Ansible documentation. <https://docs.ansible.com/>
- [3] Mudasir. (2023, May 18). Build and Deploy Code on Containers Using Ansible and automate the whole process through Jenkins. Medium. <https://medium.com/@mudasirhaji/build-and-deploy-code-on-containers-using-ansible-and-automate-the-whole-process-through-jenkins-ea0123f35a55>
- [4] Rajpurohit, P. S. (2021, December 15). Deploy Apache Web-Server in Docker Container using Ansible-Playbook. Medium. <https://rajpurohitprakash04.medium.com/deploy-apache-web-server-in-docker-container-using-ansible-playbook-320794bc2268>
- [5] Prakash Singh Rajpurohit. (2020, August 5). Deploy Web Application in Docker Container using Ansible. [Video]. YouTube. <https://www.youtube.com/watch?v=TWcFAi6rvlc>
- [6] Abhishek.Veeramalla. (2023, January 25). Day-15 | Ansible Zero to Hero | #ansible #devops [Video]. YouTube. <https://www.youtube.com/watch?v=Z6T2r3Xhk5k>
- [7] ChatGPT. (n.d.). <https://chat.openai.com/>
- [8] Intro to playbooks — ansible documentation. (n.d.). https://docs.ansible.com/ansible/2.9/user_guide/playbooks_intro.html
- [9] "Install Docker desktop on Linux." (2024, February 15). Docker Documentation. <https://docs.docker.com/desktop/install/linux-install/>
- [10] Tcarriga. (2023, January 12). Using ssh-keygen and sharing for key-based authentication in Linux. Enable Sysadmin. <https://www.redhat.com/sysadmin/configure-ssh-keygen#:~:text=You>

6. Results:

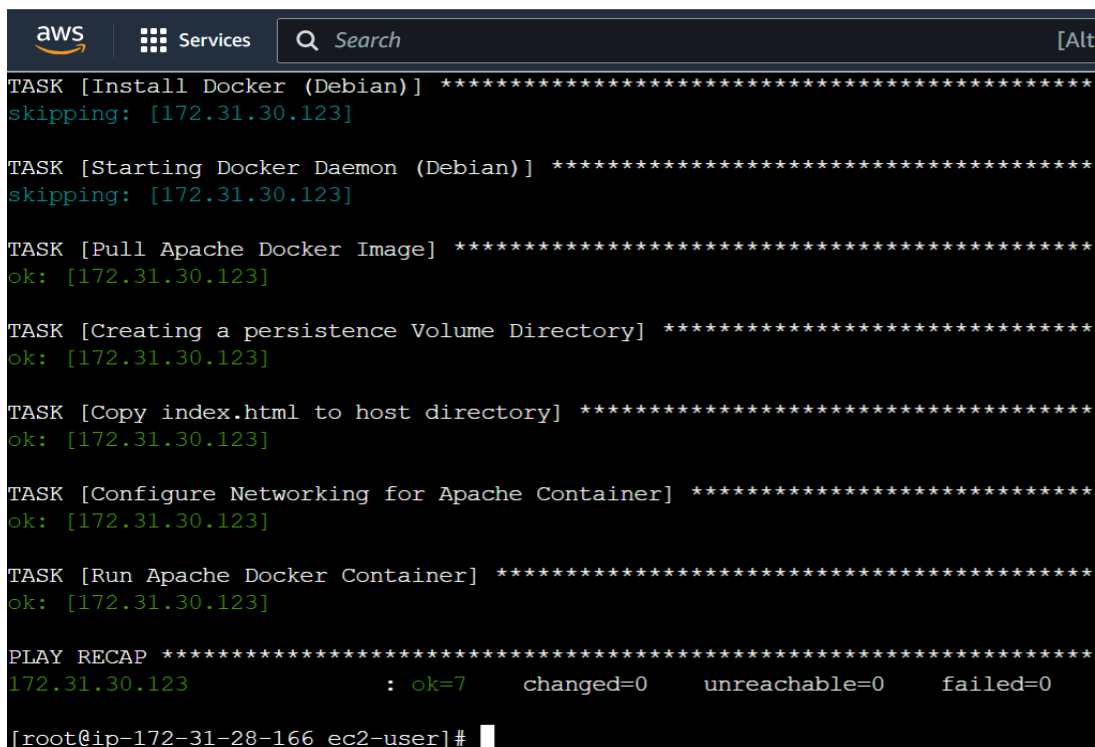


```
--
- name: Deploy Apache Docker Container using Ansible
  hosts: Managed_Node1
  become: true
  tasks:
    - name: Configure Docker Repository
      yum_repository:
        name: Docker
        description: "Docker Repo"
        baseurl: "https://download.docker.com/linux/centos/docker-ce.repo"
        gpgcheck: no
        when: ansible_os_family == "RedHat"

    - name: Install Docker (Debian)
      apt:
        name: docker.io
        state: present
        when: ansible_os_family == "Debian"

    - name: Starting Docker Daemon (Debian)
      service:
        name: docker
        state: started
"docker_deploy.yml" 64L, 1617B                               1,1           Top
```

Fig: 6.1 (docker_deploy.yml)



```
aws  Services  Search  [Alt]

TASK [Install Docker (Debian)] *****
skipping: [172.31.30.123]

TASK [Starting Docker Daemon (Debian)] *****
skipping: [172.31.30.123]

TASK [Pull Apache Docker Image] *****
ok: [172.31.30.123]

TASK [Creating a persistence Volume Directory] *****
ok: [172.31.30.123]

TASK [Copy index.html to host directory] *****
ok: [172.31.30.123]

TASK [Configure Networking for Apache Container] *****
ok: [172.31.30.123]

TASK [Run Apache Docker Container] *****
ok: [172.31.30.123]

PLAY RECAP *****
172.31.30.123      : ok=7    changed=0    unreachable=0    failed=0

[root@ip-172-31-28-166 ec2-user]#
```

Fig: 6.2 (Running the Ansible Playbook)



Fig: 6.3 (Deployed Web-Server)