Automated Container deployment and Administration

SUMMARY

This project focuses mainly on using Ansible to automate the deployment of Apache and FTP Docker containers, along with their network configuration. Key tasks include creating an Ansible playbook to deploy containers, configuring subnets to each of the containers, and ensuring service accessibility. We were able to draw out conclusions and were able to represent it in a presentable format.

The network diagram helps in visualization of the setup and to get more clarity on how things are displayed. The final Verification is done through a step-by-step confirmation manner.

The deployment was successful!!

This project highlights the efficiency and reliability of automation in container management and networking and helped in learning their modules, images, the running, and deployment. Tasks such as HTTP and FTP were also able to study in detail. The connection establishment was via SSH so yes that it was also covered during the same time.

CONTENTS

INTRODUCTION

Project Objective:

1) Explain the significance of deploying Docker containers and configuring their networking.
2) Provide an overview of the Ansible playbook's role in automating the deployment process.

Project Aim:

1) To build an apache container (static httpd web page hosting) and assign it a different ip address other than the target machines Ip.(subnet).This should be accessible by the remote user.
2) To build an FTP container which permits and allows ftp functions and assign it under a different ip address other than the target machines ip, this should be accessible by the remote user.

Library:

1) Container
   A container is a standalone executable package that includes everything needed to run a piece of software.
2) Ansible
   In real-life scenarios, it simplifies repetitive tasks and system administration by allowing users to automate processes and deploy them to "n" number of machines.
   Playbook –where the script is written (YAML format) for automation.
   Inventory-where the target machines ip is to be mentioned.
3) Docker

   It is an application which helps in building containers.

SOLUTION

The PLAYBOOK **docker_deploy.yml** can be broken down into separate tasks .

TASK_1: Installing Docker

We have pushed docker into all the target machines via this task.

TASK_2: Starting Docker

The docker is a service, so in order for a service to function it has to be loaded and enabled.

TASK_3: Creates a Docker network for Apache:

The `docker_network` module to create a bridge network named `Apache network` with a specified subnet (172.10.0.0/16).

TASK_4: Creates a Docker network for FTP:

Uses the `docker network` module to create a bridge network named `FTP_network` with a specified subnet (172.20.0.0/16).

TASK_5:Runs the Apache Container:

a) The `docker_container` module to create and start a Docker container named `apache_container` with the `httpd:latest` image.
b) We can Map port 80 inside the container to port 80 on the host.
c) Connects the container to the `apache_network` with the specified IPv4 address (172.10.0.100).
d) Mounts a volume from the host to the container at `/kali/local/apache2/htdocs`.

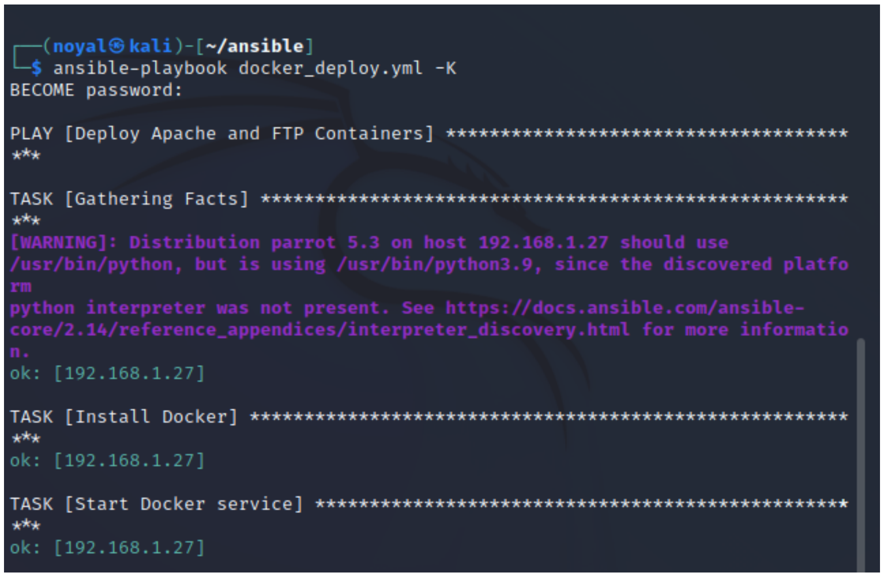
TASK_6:Runs the FTP Container:

a) The `docker_container` module to create and start a Docker container named `ftp_container` with the `fauria/vsftpd:latest` image.
b) Maps port 21 inside the container to port 21 on the host.
c) Connects the container to the `FTP_network` with the specified IPv4 address (172.20.0.100).
d) Sets the environment variables for FTP user and password are defined. (FTP_USER: kali, FTP_PASS: kali).

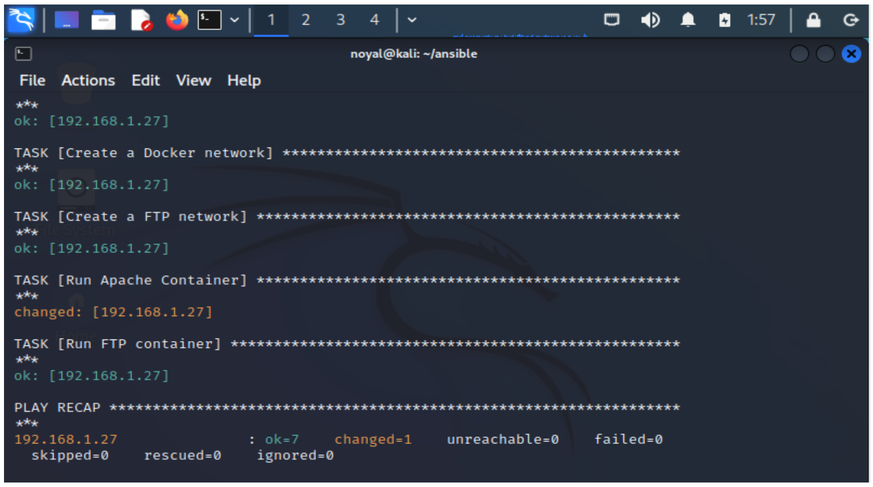
VERIFICATION:

To check the Playbook:

The –K is given so that we can enter SUDO password.



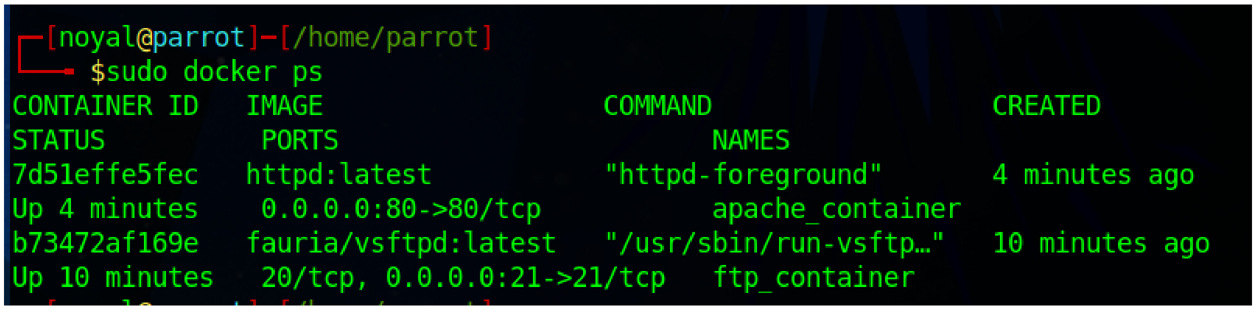
```
┌──(noyal㉿kali)-[~/ansible]
└─$ ansible-playbook docker_deploy.yml -K
BECOME password:

PLAY [Deploy Apache and FTP Containers] *************************************
***

TASK [Gathering Facts] ******************************************************
***
[WARNING]: Distribution parrot 5.3 on host 192.168.1.27 should use
/usr/bin/python, but is using /usr/bin/python3.9, since the discovered platfo
rm
python interpreter was not present. See https://docs.ansible.com/ansible-
core/2.14/reference_appendices/interpreter_discovery.html for more informatio
n.
ok: [192.168.1.27]

TASK [Install Docker] *******************************************************
***
ok: [192.168.1.27]

TASK [Start Docker service] *************************************************
***
ok: [192.168.1.27]
```



noyal@kali: ~/ansible

```
***
ok: [192.168.1.27]

TASK [Create a Docker network] **********************************************
***
ok: [192.168.1.27]

TASK [Create a FTP network] *************************************************
***
ok: [192.168.1.27]

TASK [Run Apache Container] *************************************************
***
changed: [192.168.1.27]

TASK [Run FTP container] ****************************************************
***
ok: [192.168.1.27]

PLAY RECAP ******************************************************************
***
192.168.1.27               : ok=7    changed=1    unreachable=0    failed=0
  skipped=0    rescued=0    ignored=0
```

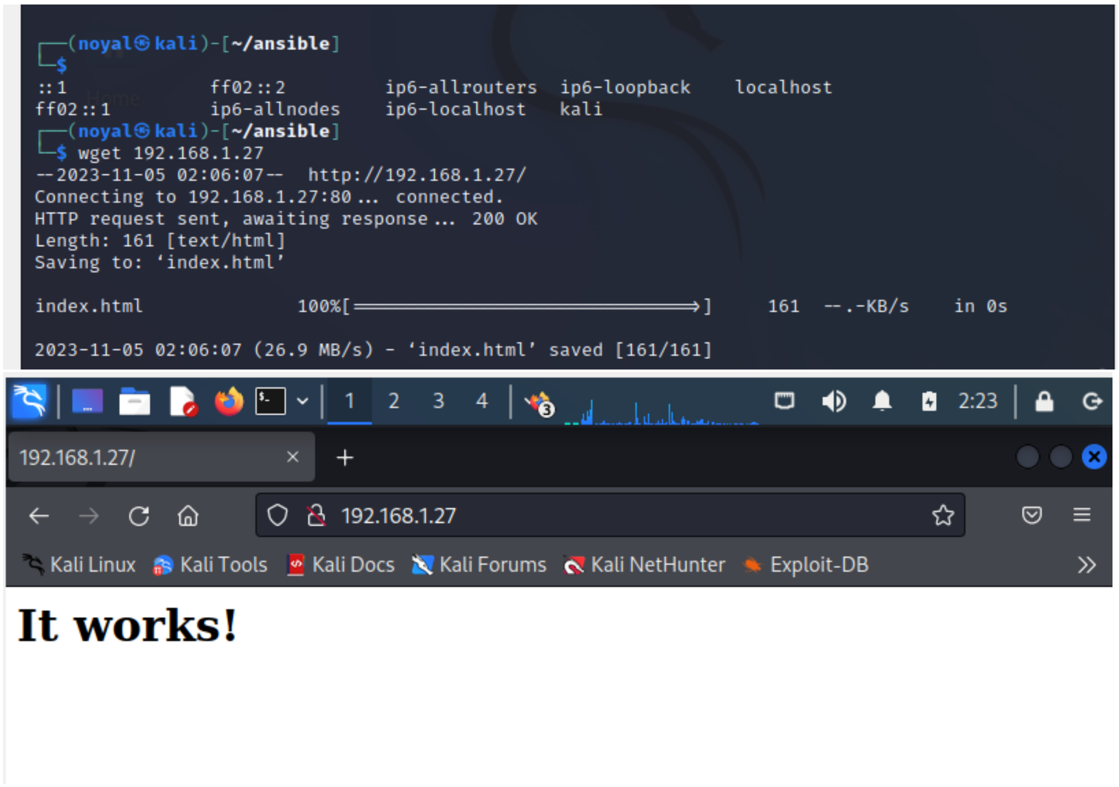
The Dockers listed in the target Machine:



```
┌─[noyal@parrot]─[/home/parrot]
└──╼ $sudo docker ps
CONTAINER ID   IMAGE                    COMMAND                   CREATED
STATUS          PORTS                              NAMES
7d51effe5fec   httpd:latest             "httpd-foreground"        4 minutes ago
Up 4 minutes    0.0.0.0:80->80/tcp                 apache_container
b73472af169e   fauria/vsftpd:latest     "/usr/sbin/run-vsftp…"    10 minutes ago
Up 10 minutes   20/tcp, 0.0.0.0:21->21/tcp   ftp_container
```

Apache Container Details:

Verifying the output:

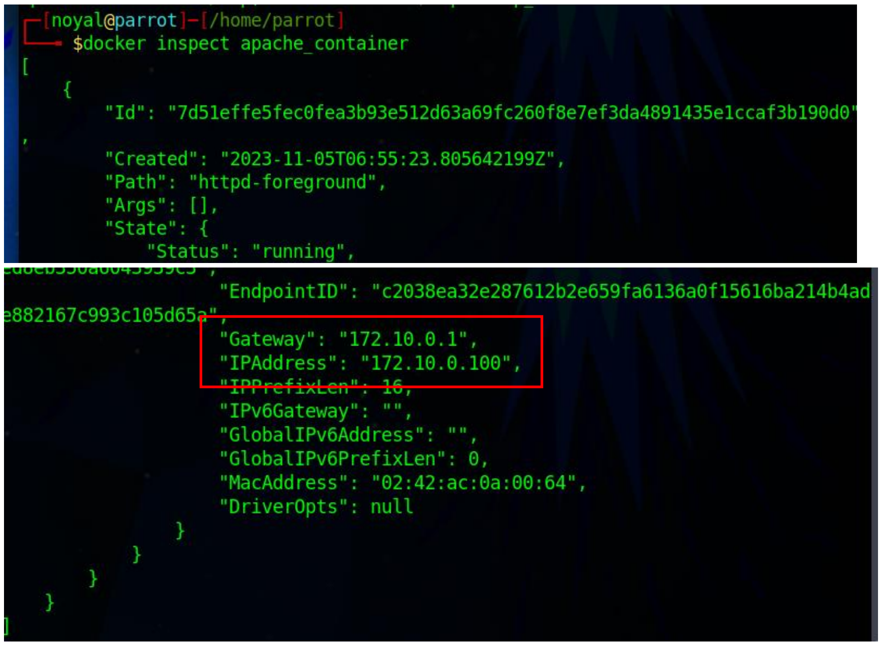


```
┌──(noyal㉿kali)-[~/ansible]
└─$ 
::1           ff02::2        ip6-allrouters  ip6-loopback    localhost
ff02::1       ip6-allnodes   ip6-localhost   kali
┌──(noyal㉿kali)-[~/ansible]
└─$ wget 192.168.1.27
--2023-11-05 02:06:07--  http://192.168.1.27/
Connecting to 192.168.1.27:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 161 [text/html]
Saving to: 'index.html'

index.html            100%[===========================>]     161  --.-KB/s    in 0s

2023-11-05 02:06:07 (26.9 MB/s) - 'index.html' saved [161/161]
```

It works!

The apache container details pulled from target machine .

NOTE :The subnetted IP address is given below.

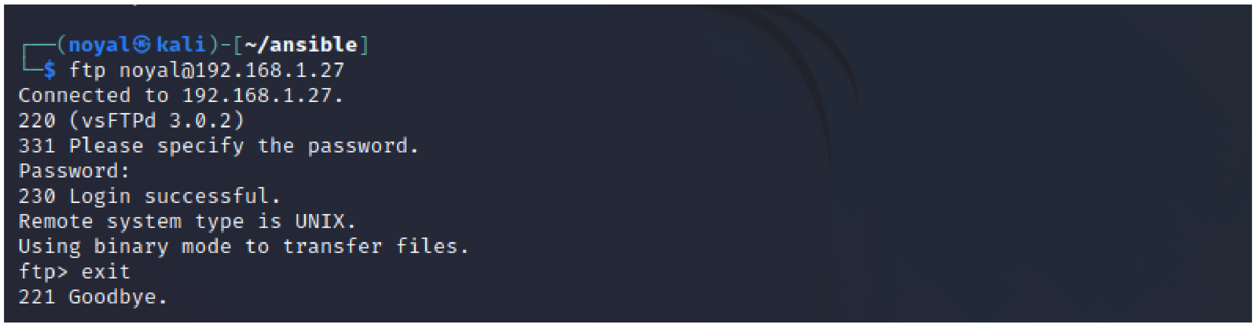


```
┌─[noyal@parrot]─[/home/parrot]
└──╼ $docker inspect apache_container
[
    {
        "Id": "7d51effe5fec0fea3b93e512d63a69fc260f8e7ef3da4891435e1ccaf3b190d0"
,
        "Created": "2023-11-05T06:55:23.805642199Z",
        "Path": "httpd-foreground",
        "Args": [],
        "State": {
            "Status": "running",
```

```
                    "EndpointID": "c2038ea32e287612b2e659fa6136a0f15616ba214b4ad
e882167c993c105d65a",
                    "Gateway": "172.10.0.1",
                    "IPAddress": "172.10.0.100",
                    "IPPrefixLen": 16,
                    "IPv6Gateway": "",
                    "GlobalIPv6Address": "",
                    "GlobalIPv6PrefixLen": 0,
                    "MacAddress": "02:42:ac:0a:00:64",
                    "DriverOpts": null
                }
            }
        }
    }
]
```

FTP Container Details:

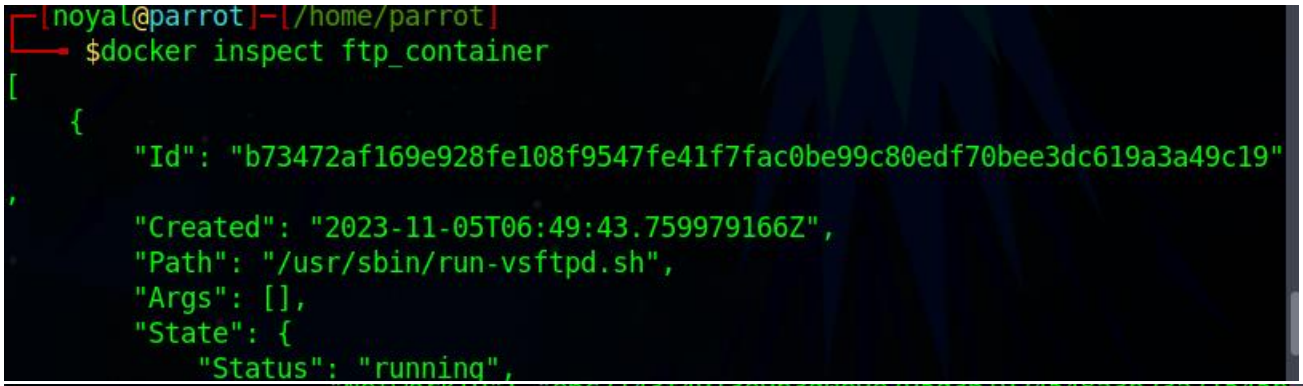
Verifying the output:



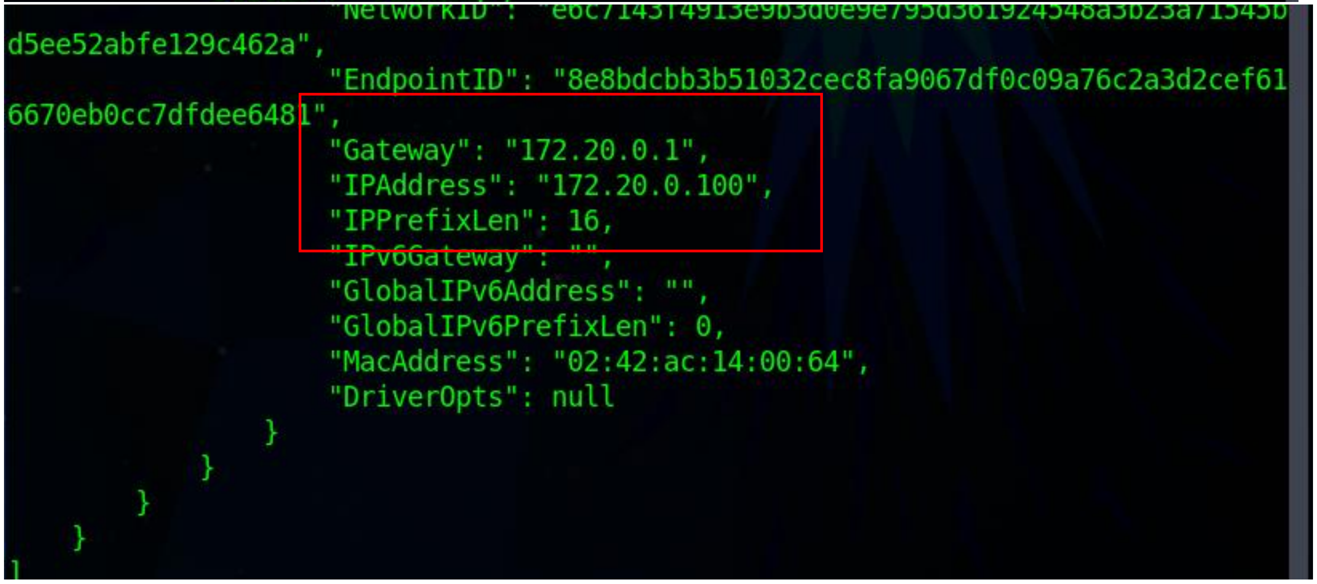
```
┌──(noyal㉿kali)-[~/ansible]
└─$ ftp noyal@192.168.1.27
Connected to 192.168.1.27.
220 (vsFTPd 3.0.2)
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> exit
221 Goodbye.
```

The FTP container details pulled from target machine .

NOTE :The subnetted IP address is given below.

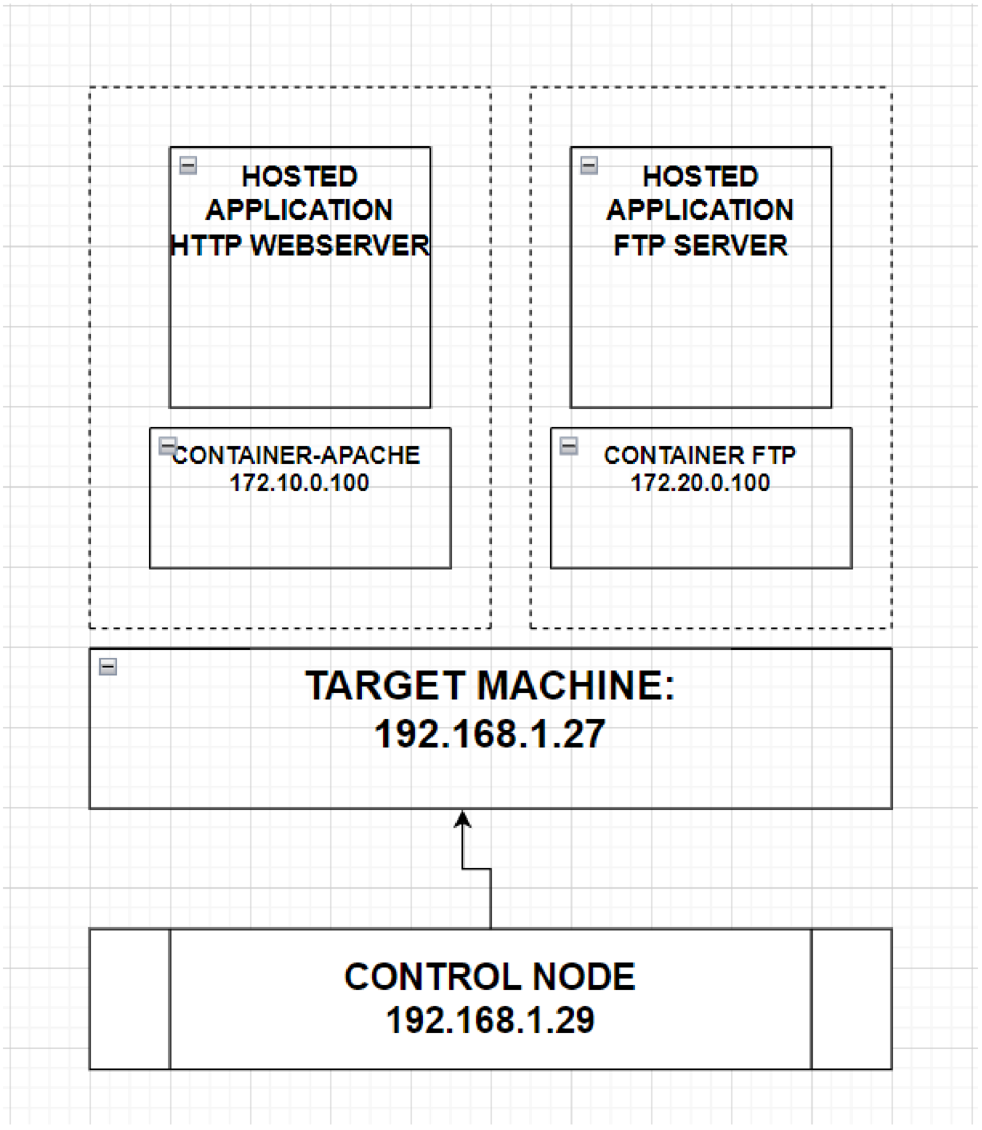


```
┌─[noyal@parrot]─[/home/parrot]
└──╼ $docker inspect ftp_container
[
    {
        "Id": "b73472af169e928fe108f9547fe41f7fac0be99c80edf70bee3dc619a3a49c19"
,
        "Created": "2023-11-05T06:49:43.759979166Z",
        "Path": "/usr/sbin/run-vsftpd.sh",
        "Args": [],
        "State": {
            "Status": "running",
```



```
d5ee52abfe129c462a",
                    "EndpointID": "8e8bdcbb3b51032cec8fa9067df0c09a76c2a3d2cef61
6670eb0cc7dfdee6481",
                    "Gateway": "172.20.0.1",
                    "IPAddress": "172.20.0.100",
                    "IPPrefixLen": 16,
                    "IPv6Gateway": "",
                    "GlobalIPv6Address": "",
                    "GlobalIPv6PrefixLen": 0,
                    "MacAddress": "02:42:ac:14:00:64",
                    "DriverOpts": null
                }
            }
        }
    }
]
```

NETWORK DIAGRAM


HOSTED APPLICATION HTTP WEBSERVER
HOSTED APPLICATION FTP SERVER
CONTAINER-APACHE 172.10.0.100
CONTAINER FTP 172.20.0.100
TARGET MACHINE: 192.168.1.27
CONTROL NODE 192.168.1.29

Referenced from :

RED-HAT SYSTEM ADMINISTARTION 3:Linux Automation(Edition 1)

Authors: Trey Feagle,Herve Quatremain

Chapter1 :Running -Containers

CONCLUSION

The containers were build and deployed successfully into different subnets.The KALI machine and PARROT machine communicated suucessfully and we were able to ping and access the containers.

REFERENCES

1) To Install Docker on KALI:
https://www.kali.org/docs/containers/installing-docker-on-kali/

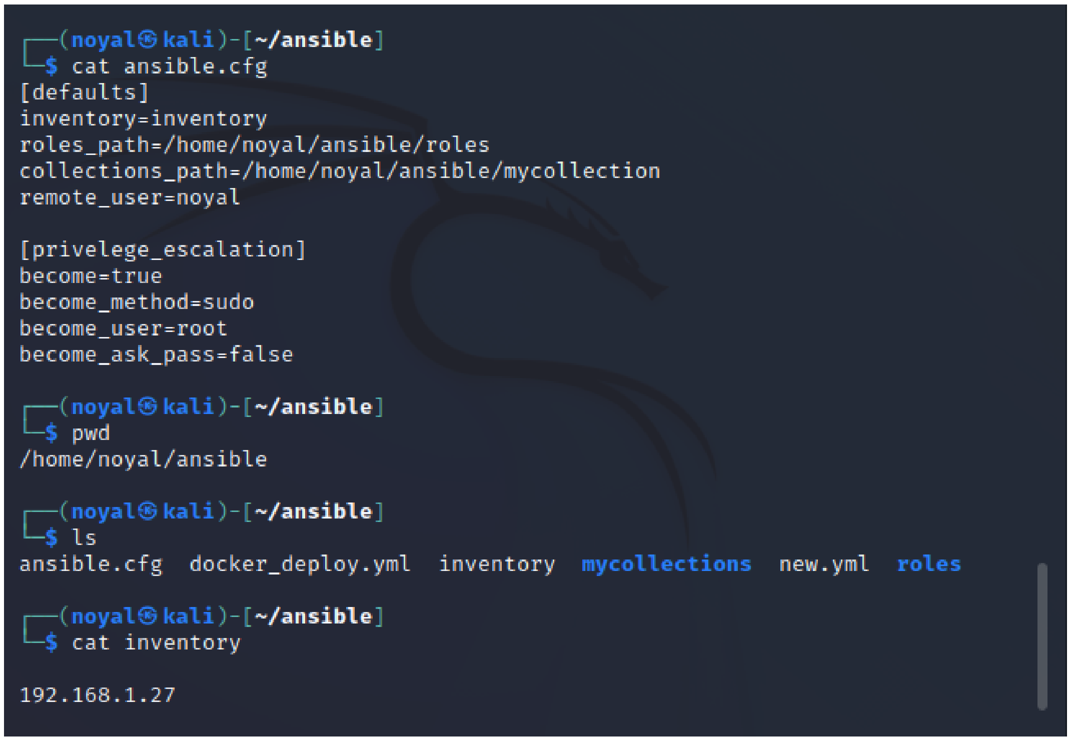
2) To install ANSIBLE:
https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html

3) To connect 2 machines using SSH by ansible :
https://docs.ansible.com/ansible/latest/inventory_guide/connection_details.html

4) docker container building apache:
https://medium.com/@sarthaksriw/apache-webserver-configuration-in-docker-using-ansible-1b19e0e6b3c5

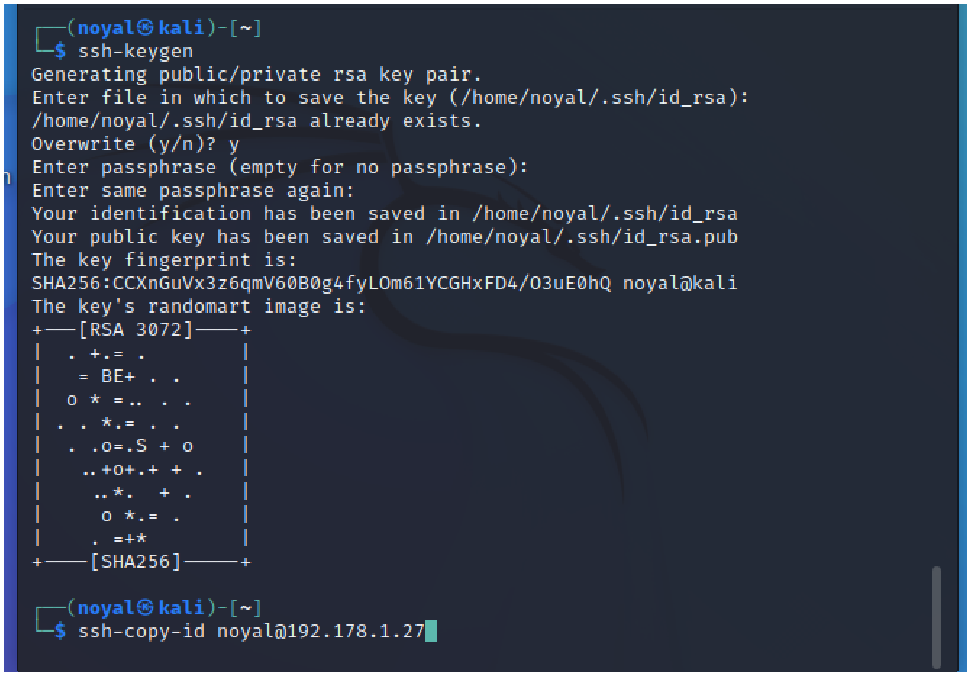
APPENDICES

1) Exploring the Ansible Machine



```
┌──(noyal㉿kali)-[~/ansible]
└─$ cat ansible.cfg
[defaults]
inventory=inventory
roles_path=/home/noyal/ansible/roles
collections_path=/home/noyal/ansible/mycollection
remote_user=noyal

[privelege_escalation]
become=true
become_method=sudo
become_user=root
become_ask_pass=false

┌──(noyal㉿kali)-[~/ansible]
└─$ pwd
/home/noyal/ansible

┌──(noyal㉿kali)-[~/ansible]
└─$ ls
ansible.cfg  docker_deploy.yml  inventory  mycollections  new.yml  roles

┌──(noyal㉿kali)-[~/ansible]
└─$ cat inventory

192.168.1.27
```

2) Keys-Generated for SSH AUTHENTICATION



```
┌──(noyal㉿kali)-[~]
└─$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/noyal/.ssh/id_rsa):
/home/noyal/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/noyal/.ssh/id_rsa
Your public key has been saved in /home/noyal/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:CCXnGuVx3z6qmV60B0g4fyLOm61YCGHxFD4/O3uE0hQ noyal@kali
The key's randomart image is:
+---[RSA 3072]----+
|  . +.= .        |
|   = BE+ . .     |
|  o * =.. . .    |
| . . *.= . .     |
|  . .o=.S + o    |
|   ..+o+.+ + .   |
|    ..*.  + .    |
|     o *.= .     |
|    . =+*        |
+----[SHA256]-----+

┌──(noyal㉿kali)-[~]
└─$ ssh-copy-id noyal@192.178.1.27
```