

---

# SETUP

---



JANUARY 2024

Abdulfatah Abdillahi

## Contents

Introduction .....	3
Tasks .....	3
Create SEED VM .....	3
Docker Review for SEED Labs .....	4
REFERENCES .....	13

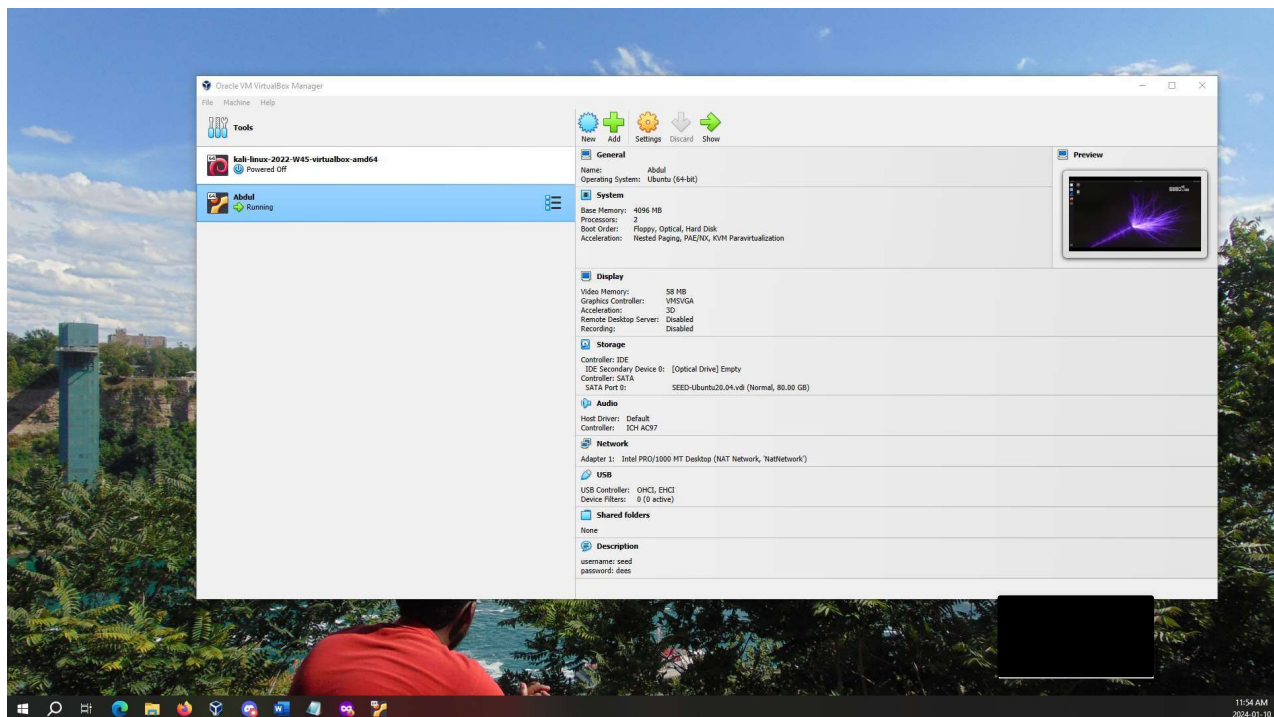
## Introduction

In this lab, have set up and understood the environment required to use the SEED Labs, and create some containers.

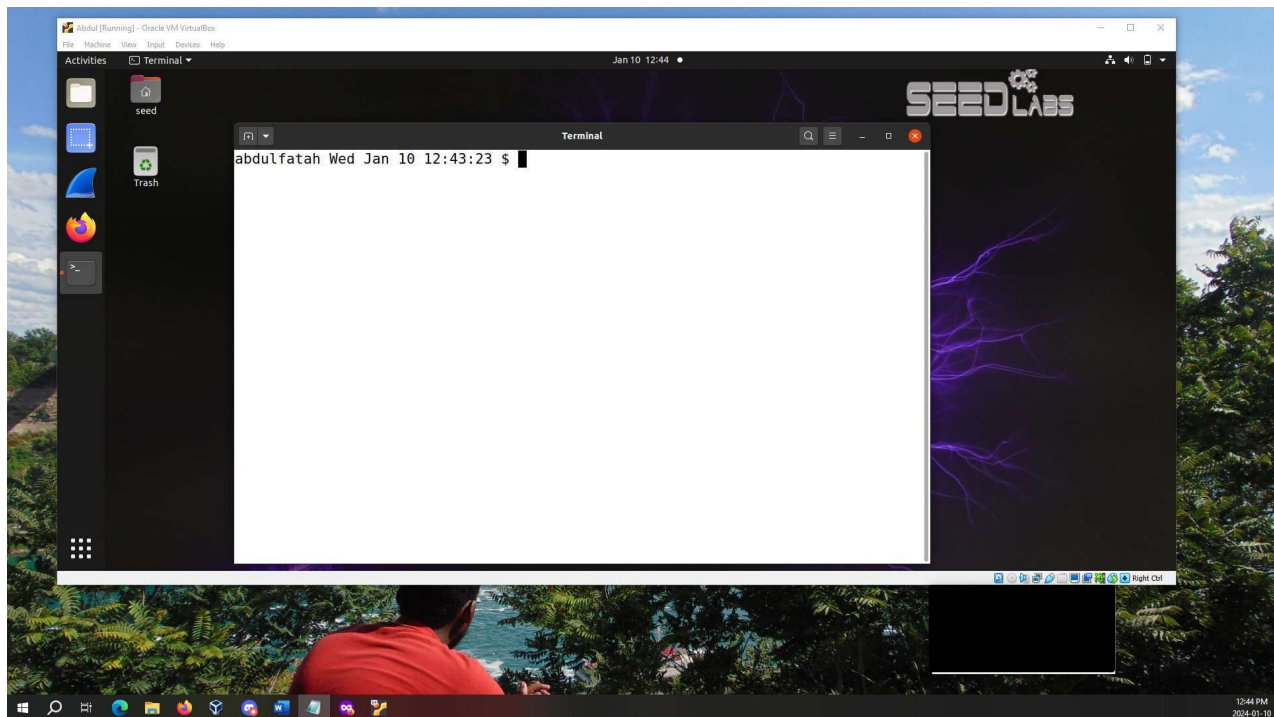
## Tasks

### Create SEED VM

Here we start off the lab by setting up our environment and creating a SEED VM using the pre-built SEED Ubuntu 20.04 image. This will be used for all future labs. The below screenshot shows the settings that have been configured for the VM as outlined in the seedvm-manual.



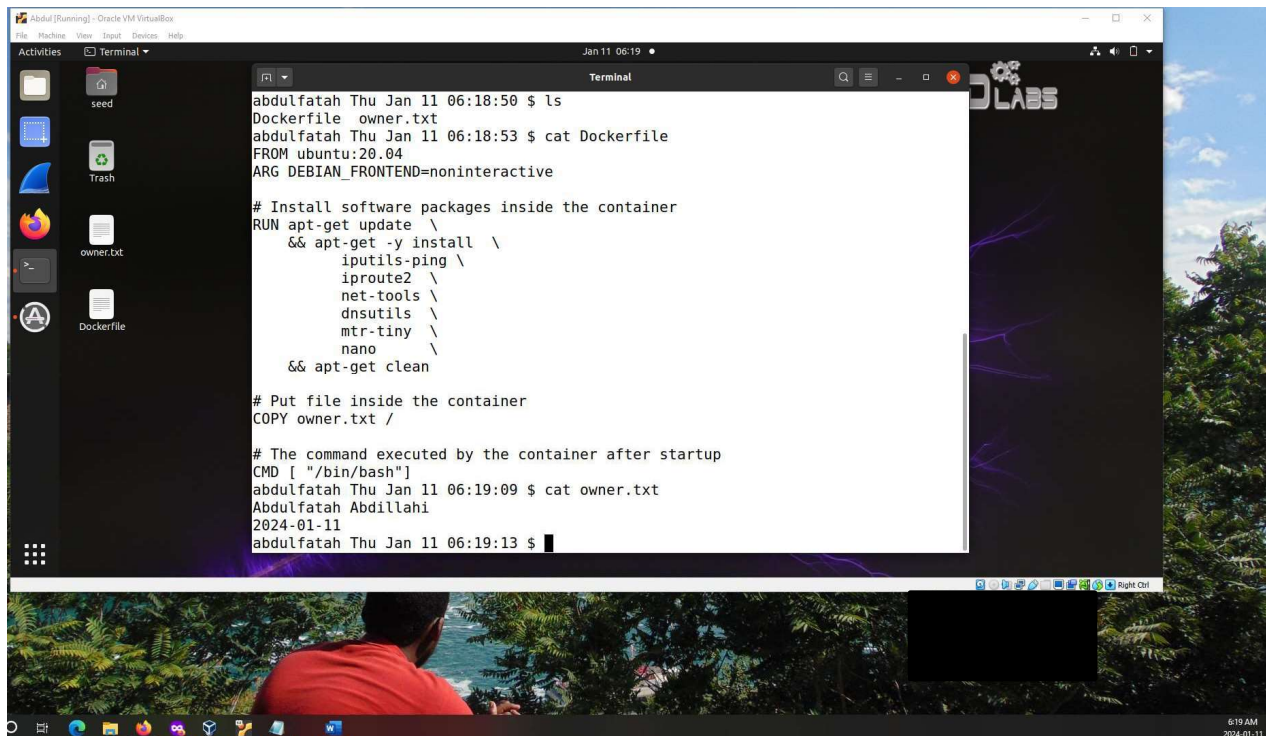
Upon running the VM for the first time, I made sure to change the command prompt as instructed to include my name (Abdulfatah), date, and time.



## Docker Review for SEED Labs

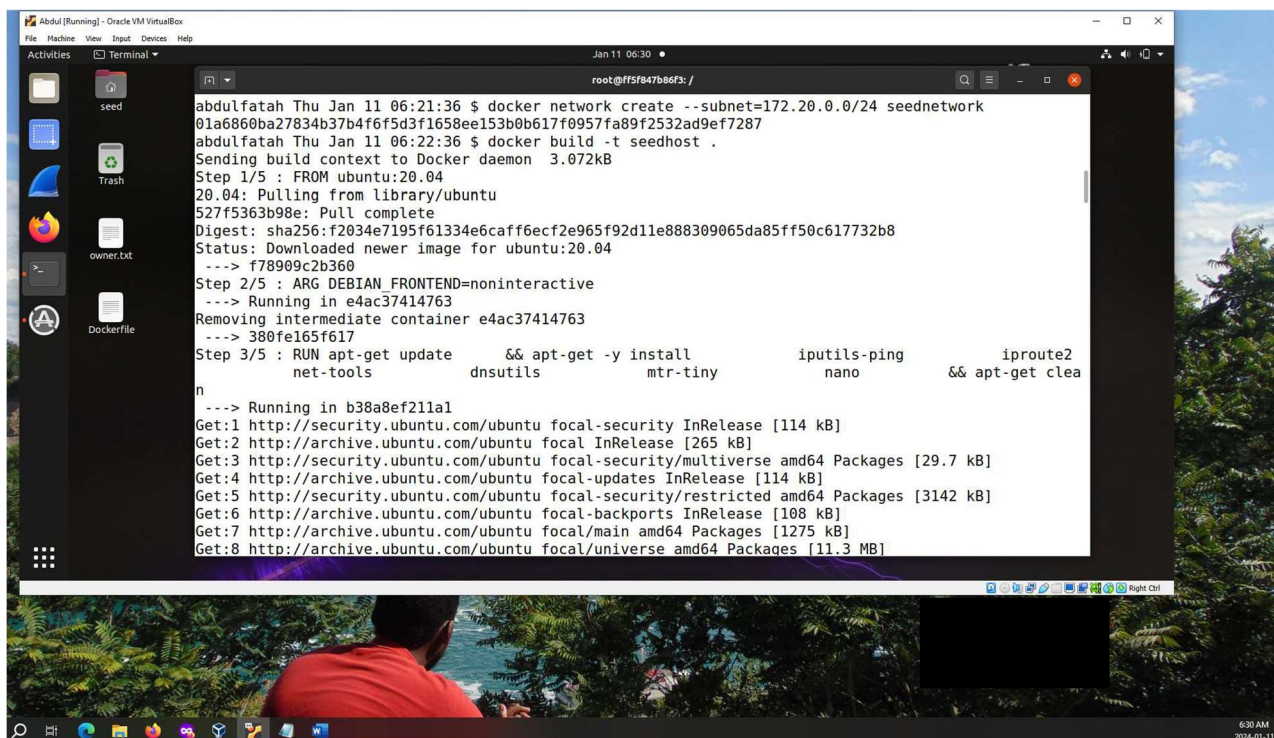
### Writing the Dockerfile

For this part of the lab, we have built a custom Dockerfile from which a docker image can be built. As you can see in the screenshot, in the Dockerfile we specified that the container will be built based on the official Ubuntu 20.04 Docker image. We also installed numerous software packages and ran the “apt-get clean” command to empty the local repository and thereby decrease the overall size of the image. Moreover, as per the instruction in Blackboard I have the file copy line to copy a file named owner.txt, which included my name and the date.



## Building and starting the container

Here, we are assigning a static IP address to our container but before doing that we must first attach our container to a network. Here we use the command “docker network create --subnet=172.20.0.0/24 seednetwork” to create a network with the prefix 172.20.0.0/24 called *seednetwork*. Then I run the command “docker build -t seedhost .” to create a container called *seedhost* using the *Dockerfile* we just created beforehand.





Lastly, I ran the command “`docker run --net seednetwork --ip 172.20.0.5 --rm -it seedhost`” to start an interactive session with the container and specify an Ip address from the *seedhost* network (we just created above). I then ran some commands (`whoami`, `ls`, `ip --br a`) to prove that my container was working like it was supposed to, including the assigned IP address and `owner.txt` file being copied.

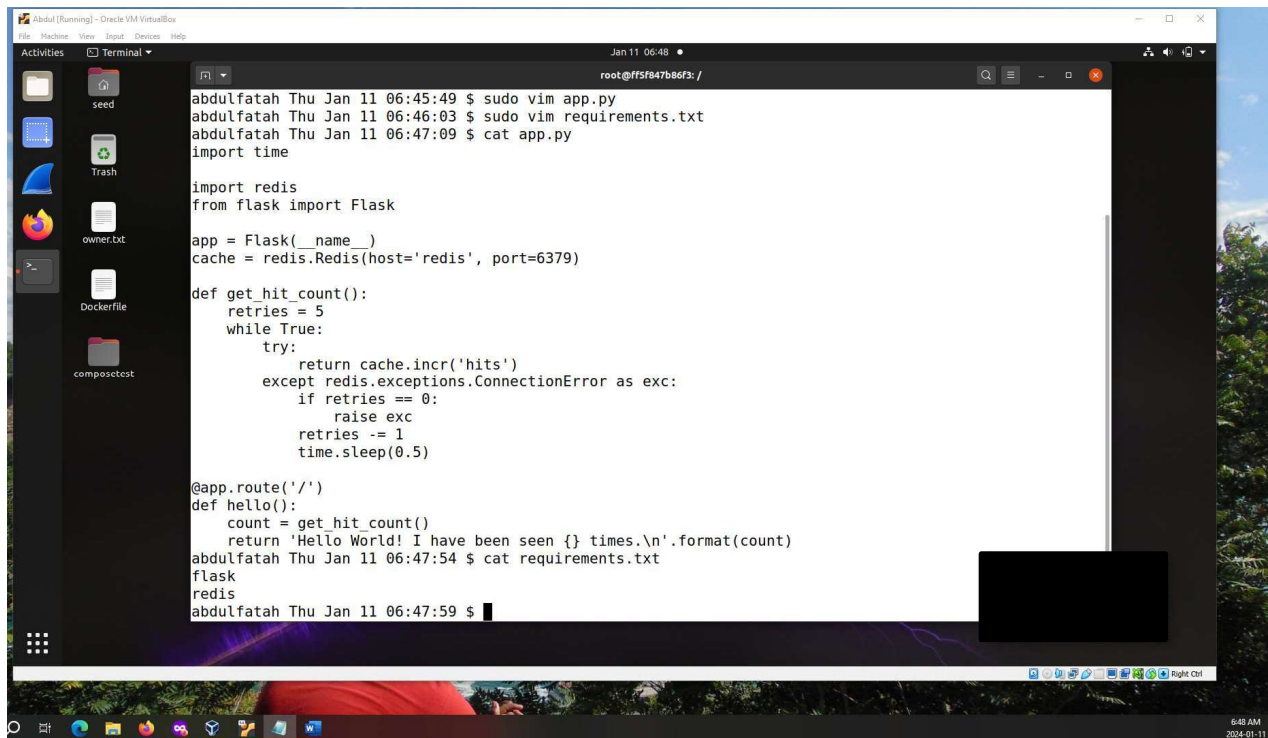
```

Setting up dnstools (1:9.16.1-0ubuntu2.16) ...
Processing triggers for libc-bin (2.31-0ubuntu9.14) ...
Removing intermediate container b38a8ef211a1
--> b47f79eb6801
Step 4/5 : COPY owner.txt /
--> 73167a7b2b57
Step 5/5 : CMD [ "/bin/bash" ]
--> Running in db0fa988c090
Removing intermediate container db0fa988c090
--> d0d10a75510f
Successfully built d0d10a75510f
Successfully tagged seedhost:latest
abdulfatah Thu Jan 11 06:27:05 $ docker run --net seednetwork --ip 172.20.0.5 --rm -it seedhost
root@ff5f847b86f3:/# whoami
root
root@ff5f847b86f3:/# pwd
/
root@ff5f847b86f3:/# ls
bin  dev  home  lib32  libx32  mnt  owner.txt  root  sbin  sys  usr
boot  etc  lib  lib64  media  opt  proc      run  srv   tmp  var
root@ff5f847b86f3:/# ip --br a
lo                UNKNOWN
eth0@if8          UP           127.0.0.1/8
eth0@if8          UP           172.20.0.5/24
root@ff5f847b86f3:/#

```

## Docker Compose

Here, we explore some concepts of docker compose by building a python web application. We first started creating a file named *app.py* which contained python code to create a simple Flask web application that uses Redis for caching. Also, I made another file called *requirements.txt* that contains the words “flask” and “redis”, which will come in handy when later creating the Dockerfile.



```

root@ff5f847b86f3: /
abduLfatah Thu Jan 11 06:45:49 $ sudo vim app.py
abduLfatah Thu Jan 11 06:46:03 $ sudo vim requirements.txt
abduLfatah Thu Jan 11 06:47:09 $ cat app.py
import time

import redis
from flask import Flask

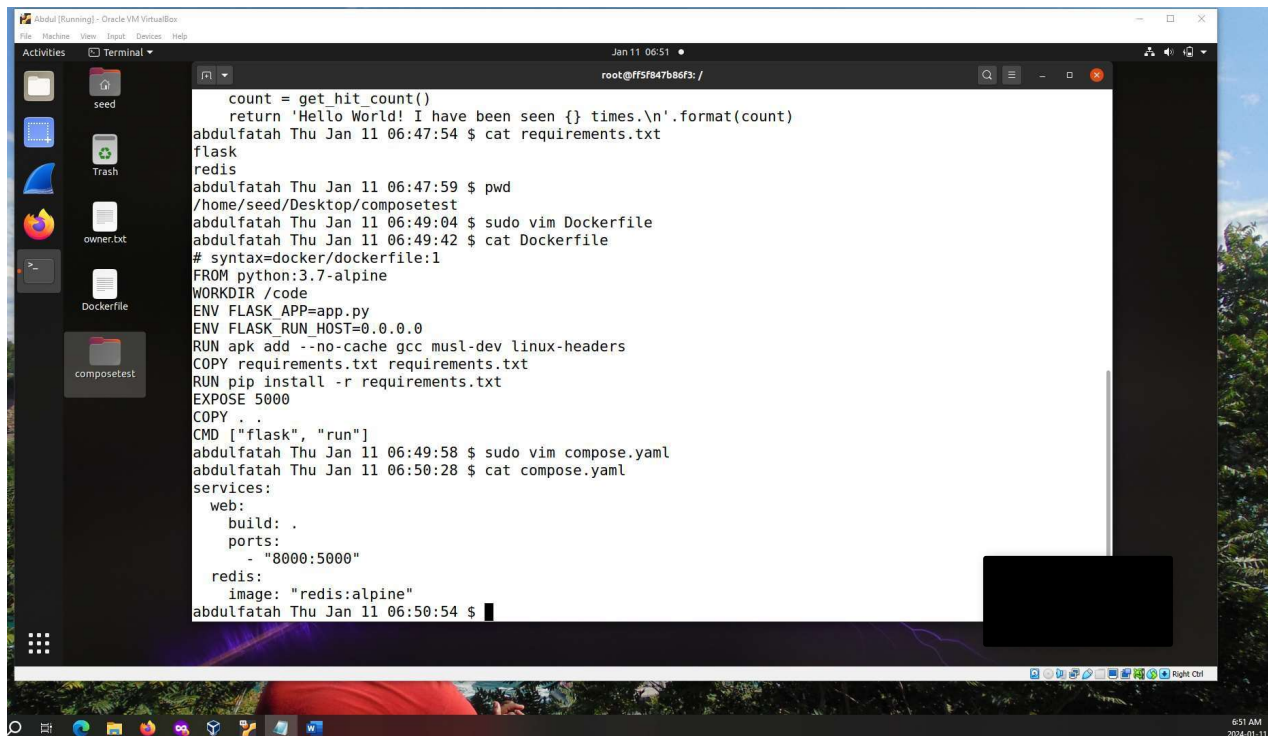
app = Flask(__name__)
cache = redis.Redis(host='redis', port=6379)

def get_hit_count():
    retries = 5
    while True:
        try:
            return cache.incr('hits')
        except redis.exceptions.ConnectionError as exc:
            if retries == 0:
                raise exc
            retries -= 1
            time.sleep(0.5)

@app.route('/')
def hello():
    count = get_hit_count()
    return 'Hello World! I have been seen {} times.\n'.format(count)
abduLfatah Thu Jan 11 06:47:54 $ cat requirements.txt
flask
redis
abduLfatah Thu Jan 11 06:47:59 $

```

Here, I have created a Docker file which will be used to build a docker image. This file contains all the required dependencies needed for the proper functionality of a Python application, as well as the Python application itself. The services (web and redis) were later defined in a compose file named compose.yaml where the internal and exposed ports were configured and the Dockerfile image (alpine) was used.



```

root@ff5f847b86f3: /
count = get_hit_count()
return 'Hello World! I have been seen {} times.\n'.format(count)
abduLfatah Thu Jan 11 06:47:54 $ cat requirements.txt
flask
redis
abduLfatah Thu Jan 11 06:47:59 $ pwd
/home/seed/Desktop/composetest
abduLfatah Thu Jan 11 06:49:04 $ sudo vim Dockerfile
abduLfatah Thu Jan 11 06:49:42 $ cat Dockerfile
# syntax=docker/dockerfile:1
FROM python:3.7-alpine
WORKDIR /code
ENV FLASK_APP=app.py
ENV FLASK_RUN_HOST=0.0.0
RUN apk add --no-cache gcc musl-dev linux-headers
COPY requirements.txt requirements.txt
RUN pip install -r requirements.txt
EXPOSE 5000
COPY . .
CMD ["flask", "run"]
abduLfatah Thu Jan 11 06:49:58 $ sudo vim compose.yaml
abduLfatah Thu Jan 11 06:50:28 $ cat compose.yaml
services:
  web:
    build: .
    ports:
      - "8000:5000"
  redis:
    image: "redis:alpine"
abduLfatah Thu Jan 11 06:50:54 $

```

Here, I am starting the application by running the command “docker-compose up”.

```

Abdul [Running] - Oracle VM VirtualBox
File Edit View Host Devices Help
Activities Terminal
root@ff5f847b6f3: /
abduLfatah Thu Jan 11 06:56:09 $
abduLfatah Thu Jan 11 06:56:14 $ docker-compose up
Creating network "composetest_default" with the default driver
Building web
Step 1/10 : FROM python:3.7-alpine
3.7-alpine: Pulling from library/python
96526aa774ef: Pull complete
9875af95546d: Pull complete
4819c95424fc: Pull complete
148762f75a1f: Pull complete
eal518237b37: Pull complete
Digest: sha256:f3d31c8677d03f0b3c724446077f229a6ce9d3ac430f5c08cd7dff00292048c3
Status: Downloaded newer image for python:3.7-alpine
--> 1bac8ae77e4a
Step 2/10 : WORKDIR /code
--> Running in 8264efbcdec9
Removing intermediate container 8264efbcdec9
--> aa669fa8d21d
Step 3/10 : ENV FLASK_APP=app.py
--> Running in b94ebc5699e4
Removing intermediate container b94ebc5699e4
--> 9e4939a2805f
Step 4/10 : ENV FLASK_RUN_HOST=0.0.0
--> Running in c2a52a30e2ab
Removing intermediate container c2a52a30e2ab
--> b13045e762c4
Step 5/10 : RUN apk add --no-cache gcc musl-dev linux-headers
--> Running in bc4cc6451251
fetch https://dl-cdn.alpinelinux.org/alpine/v3.18/main/x86_64/APKINDEX.tar.gz
fetch https://dl-cdn.alpinelinux.org/alpine/v3.18/community/x86_64/APKINDEX.tar.gz

```

I follow this by browsing to port 8000 on the localhost and this shows that the application works as expected.

```

Abdul [Running] - Oracle VM VirtualBox
File Edit View Host Devices Help
Activities Firefox Web Browser
root@ff5f847b6f3: /
redis_1 | 1:C 11 Jan 2024 11:56:58.136 # WARNING Memory overcommit must be enabled! Without it, a back
redis_1 | 1:C 11 Jan 2024 11:56:58.136 * o000o000o000o Redis is starting o000o000o000o
redis_1 | 1:C 11 Jan 2024 11:56:58.136 * Redis version=7.2.4, bits=64, commit=00000000, modified=0, pi
redis_1 | 1:C 11 Jan 2024 11:56:58.136 # Warning: no config file specified, using the default config.
redis_1 | 1:M 11 Jan 2024 11:56:58.136 * monotonic clock: POSIX clock_gettime
redis_1 | 1:M 11 Jan 2024 11:56:58.137 * Running m
redis_1 | 1:M 11 Jan 2024 11:56:58.138 * Server in
redis_1 | 1:M 11 Jan 2024 11:56:58.138 * Ready to
web_1 | * Serving Flask app 'app.py'
web_1 | * Debug mode: off
web_1 | WARNING: This is a development server. Do not use it in production.
web_1 | * Running on all addresses (0.0.0.0)
web_1 | * Running on http://127.0.0.1:5000
web_1 | * Running on http://172.18.0.3:5000
web_1 | Press CTRL+C to quit
web_1 | 172.18.0.1 - - [11/Jan/2024 13:11:55] "G
redis_1 | 1:M 11 Jan 2024 13:11:55.259 * 1 changes
redis_1 | 1:M 11 Jan 2024 13:11:55.259 * Background
redis_1 | 18:C 11 Jan 2024 13:11:55.278 * DB saved on disk
redis_1 | 18:C 11 Jan 2024 13:11:55.278 * Fork CoW for RDB: current 0 MB, peak 0 MB, ave
web_1 | 172.18.0.1 - - [11/Jan/2024 13:11:55] "GET /favicon.ico HTTP/1.1" 404 -
redis_1 | 1:M 11 Jan 2024 13:11:55.360 * Background saving terminated with success

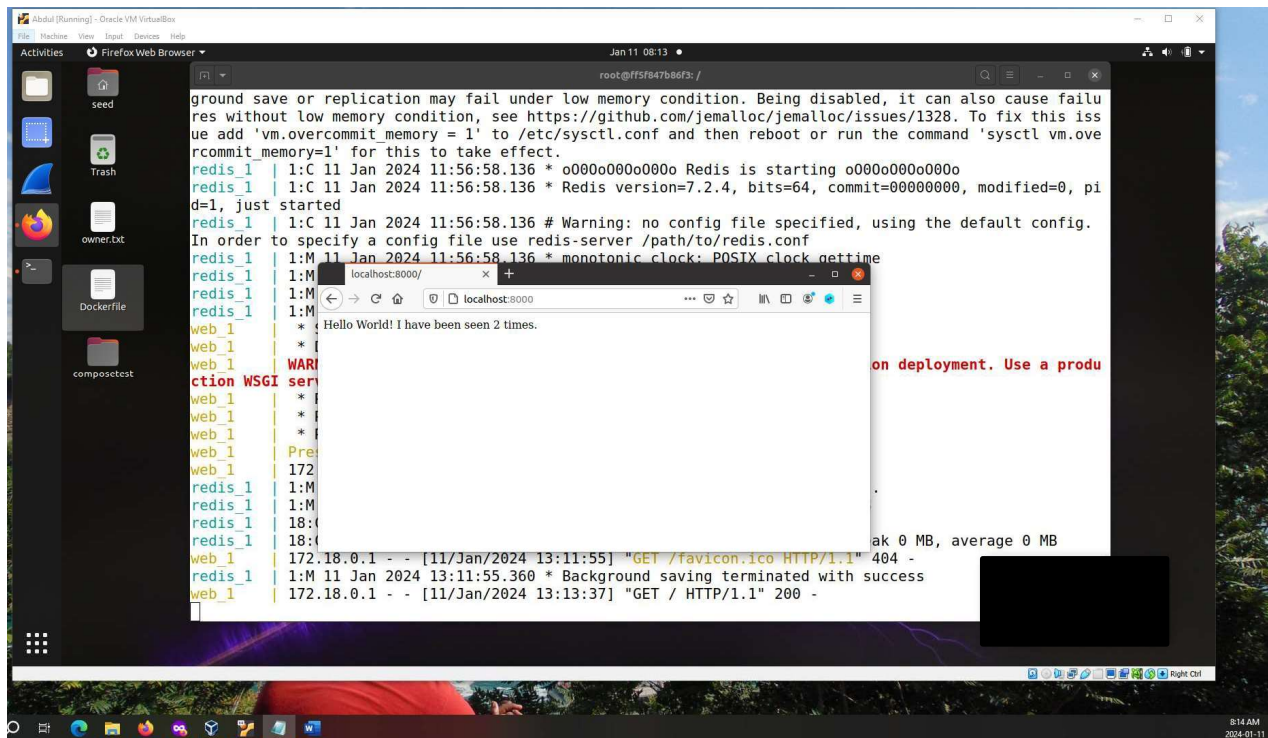
```

localhost:8000

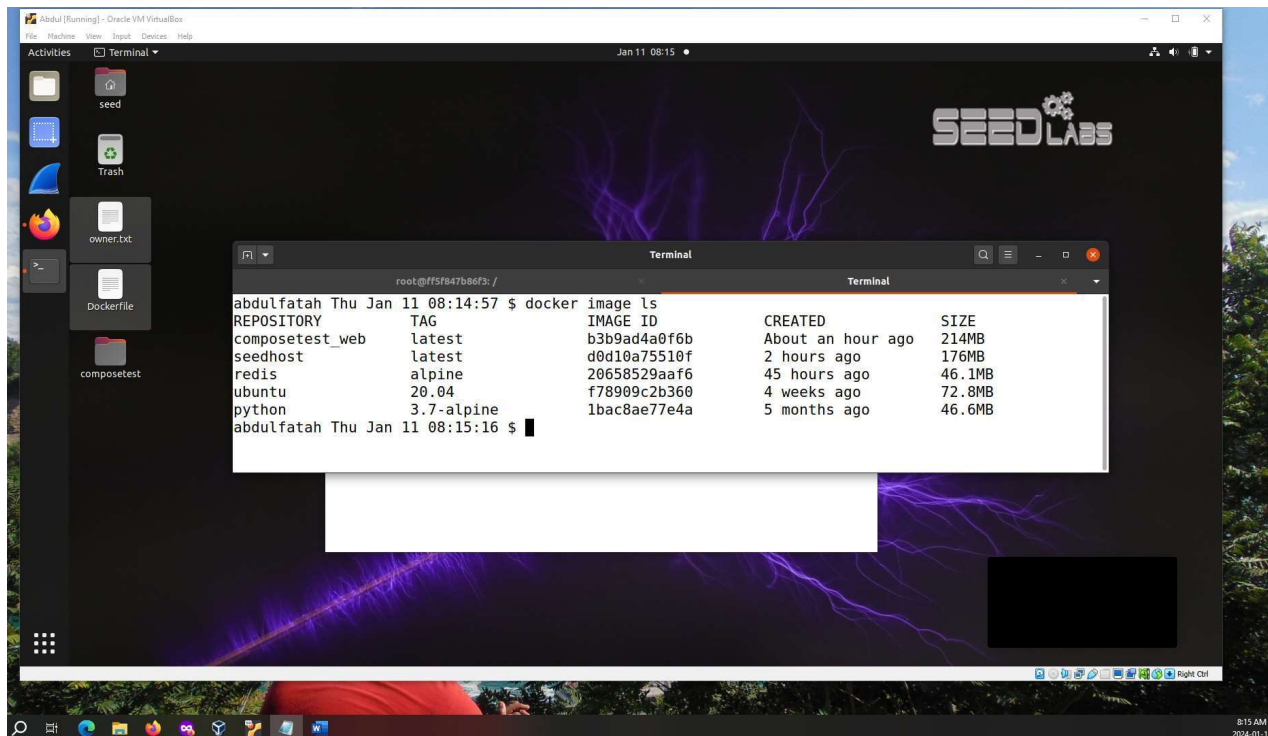
Hello World! I have been seen 1 times.

Upon refreshing the page, we see the increment in the number displayed in the message, proving that our Python code is working as we hoped.

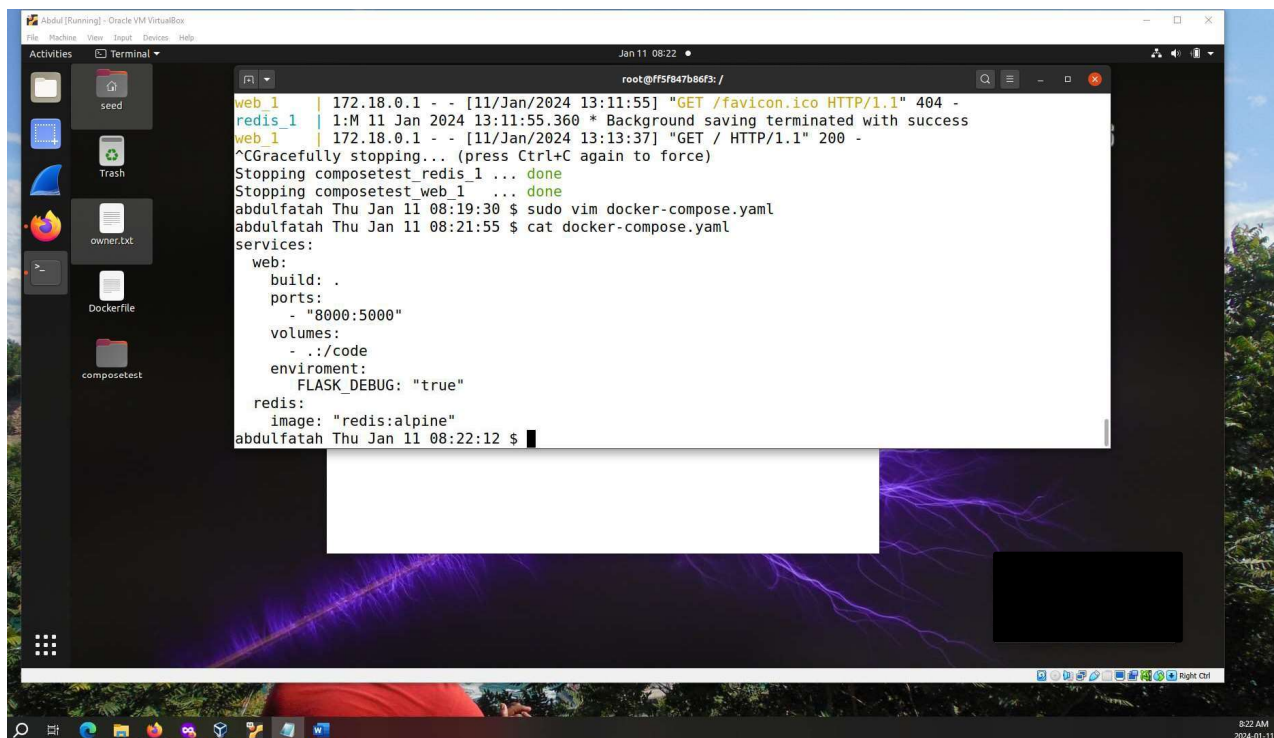




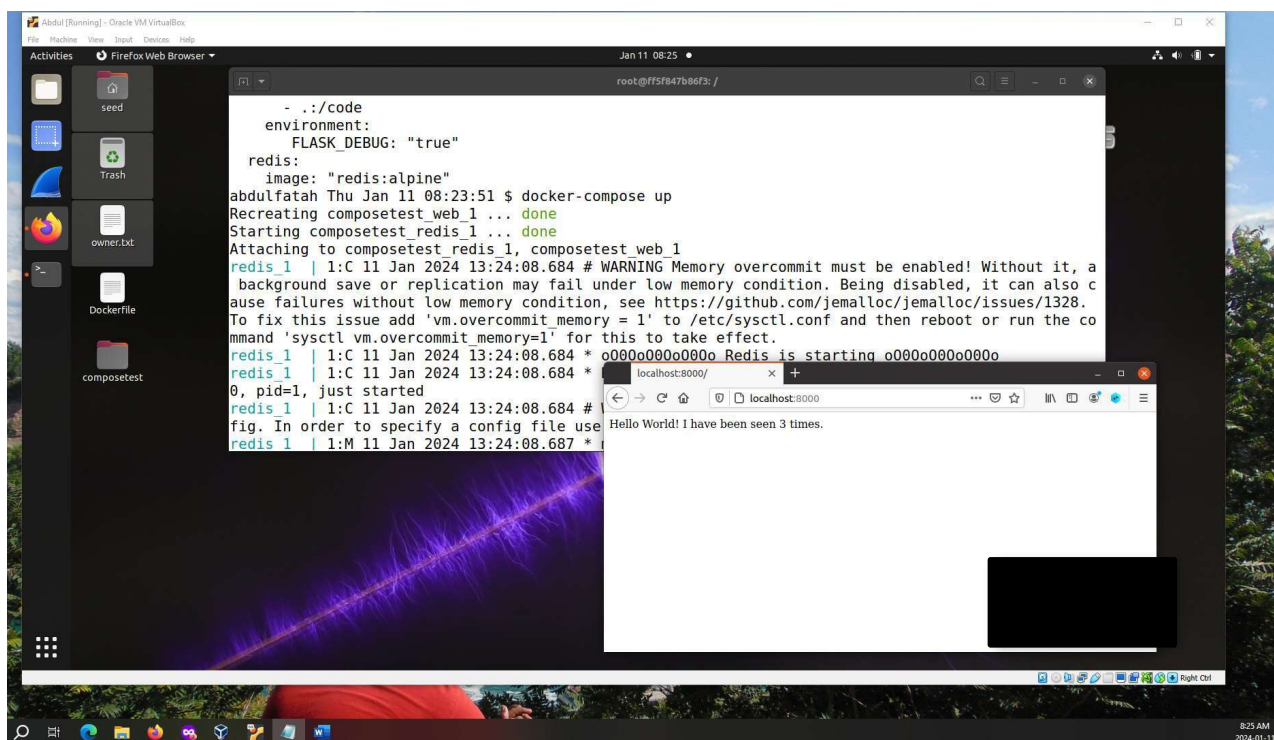
This is showing all the present docker images in the local repository up until this moment.



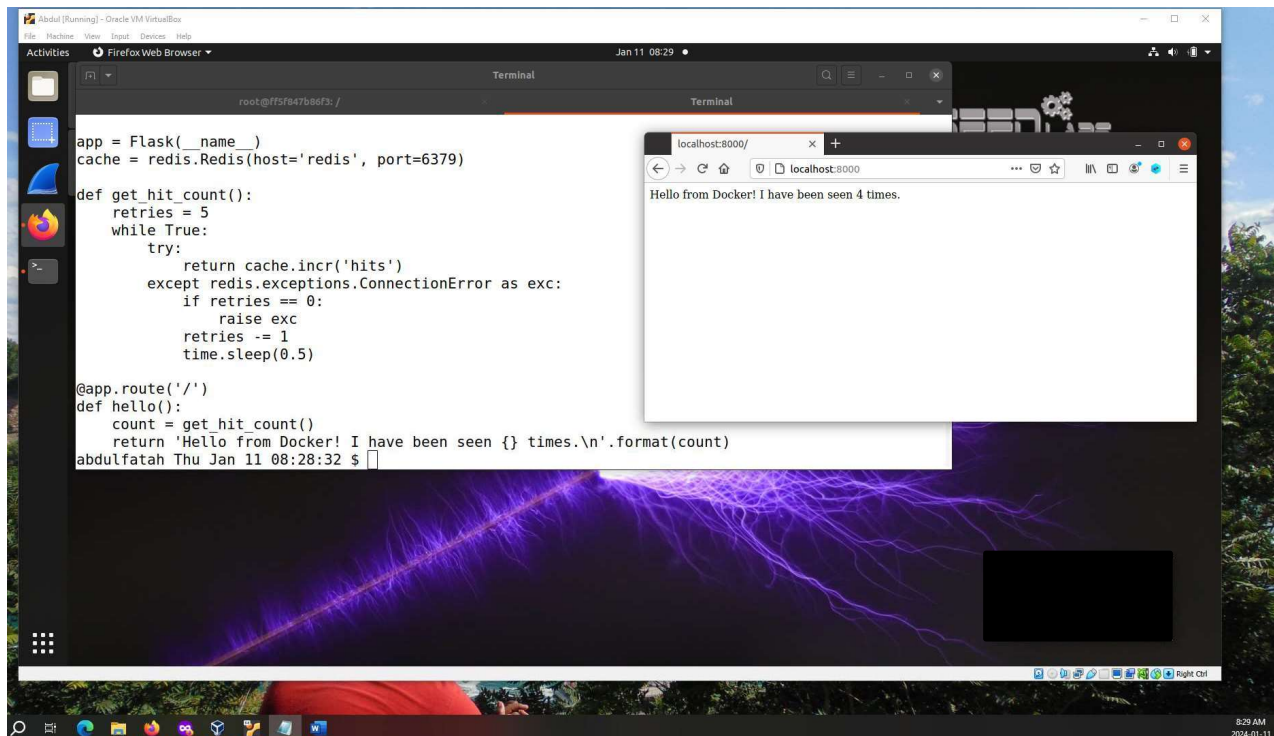
Here, I went back to editing the compose file to include a bind mount. This is used to mount the file on the host machine to a container.



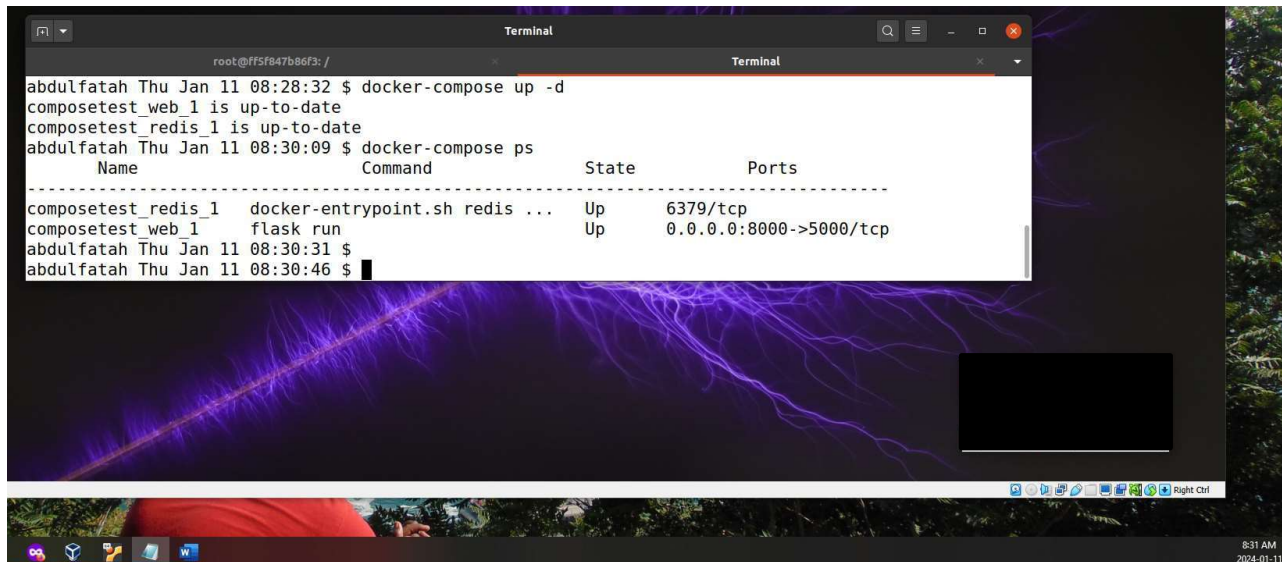
I am now running “docker-compose up” to build the app with the updated compose file. I then refresh the page and see the count increment.



Here, the displayed message in the Python code was updated from “Hello World!” to “Hello from Docker!”. And since we have previously configured the bind mount, we are no longer required to update the image after making changes to the code as the changes will take place instantly.

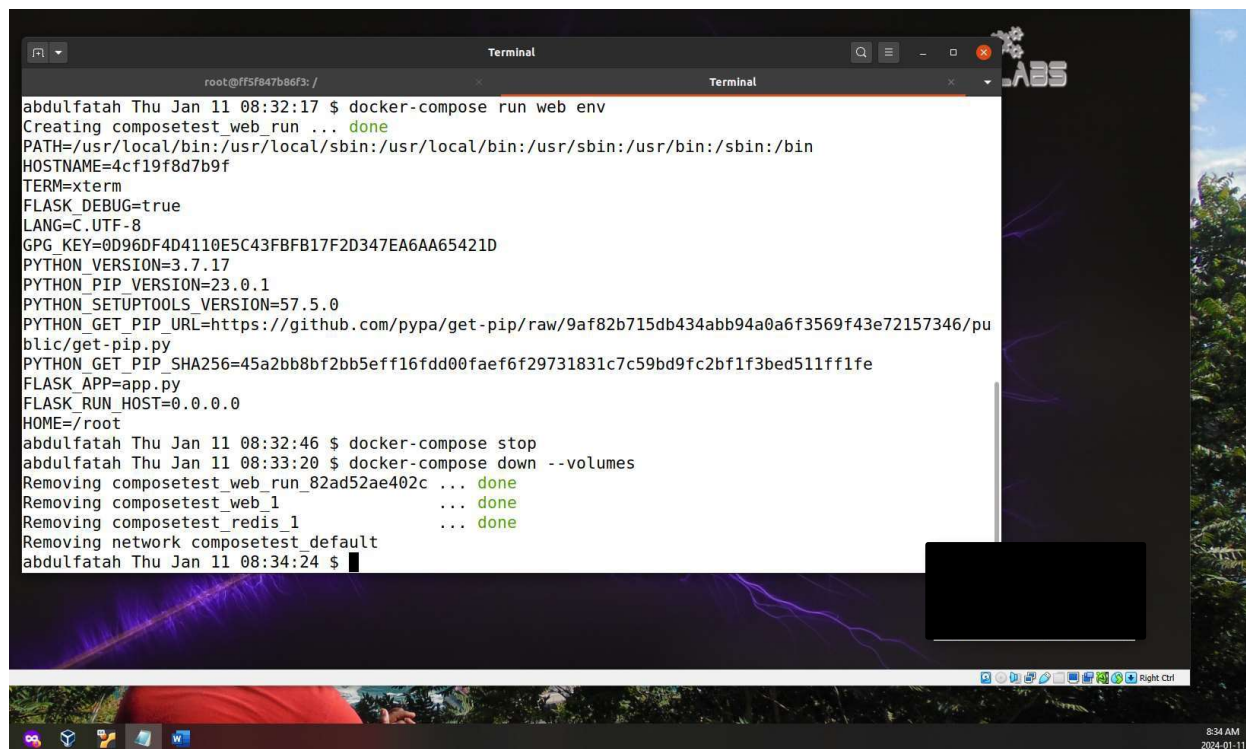


Here, we use the `-d` flag with `compose-up` to run the services in detached mode and leave them in the background. I also ran the command `"docker-compose ps"` to see all any active services defined in the compose file.



Finally, I ran the command `"docker compose run web env"` to see what environmental variables are available on the web service. I then ran the command `"docker compose stop"` to eliminate all the services running in the background (in detached mode). Also, with the `"docker compose down --volumes"` command all containers have been completely removed and the data volume used by any service (Redis) is also removed.





```
root@ff5f847b86f3: /  
Terminal  
abdufatah Thu Jan 11 08:32:17 $ docker-compose run web env  
Creating composetest_web_run ... done  
PATH=/usr/local/bin:/usr/local/sbin:/usr/bin:/sbin:/bin  
HOSTNAME=4cf19f8d7b9f  
TERM=xterm  
FLASK_DEBUG=true  
LANG=C.UTF-8  
GPG_KEY=0D96DF4D4110E5C43FBFB17F2D347EA6AA65421D  
PYTHON_VERSION=3.7.17  
PYTHON_PIP_VERSION=23.0.1  
PYTHON_SETUPTOOLS_VERSION=57.5.0  
PYTHON_GET_PIP_URL=https://github.com/pypa/get-pip/raw/9af82b715db434abb94a0a6f3569f43e72157346/public/get-pip.py  
PYTHON_GET_PIP_SHA256=45a2bb8bfb2bb5eff16fdd0faef6f29731831c7c59bd9fc2bf1f3bed511ff1fe  
FLASK_APP=app.py  
FLASK_RUN_HOST=0.0.0.0  
HOME=/root  
abdufatah Thu Jan 11 08:32:46 $ docker-compose stop  
abdufatah Thu Jan 11 08:33:20 $ docker-compose down --volumes  
Removing composetest_web_run_82ad52ae402c ... done  
Removing composetest_web_1 ... done  
Removing composetest_redis_1 ... done  
Removing network composetest_default  
abdufatah Thu Jan 11 08:34:24 $
```

8:34 AM  
2024-01-11



## References

[1] “seed-labs/seed-labs,” *GitHub*, Jan. 11, 2023. <https://github.com/seed-labs/seed-labs/blob/master/manuals/docker/docker.md>

[2] “Get started with Docker Compose,” *Docker Documentation*, Jul. 22, 2020. <https://docs.docker.com/compose/gettingstarted/> (accessed Jan. 12, 2024).

[3] “seed-labs/seed-labs,” *GitHub*, Jan. 12, 2020. <https://github.com/seed-labs/seed-labs/blob/master/manuals/vm/seedvm-manual.md> (accessed Jan. 12, 2024).