Flask Web Application in Docker

The application is a lightweight web service built using the Flask framework, a popular micro web framework written in Python. It is designed to serve a static HTML page as its main interface, which greets users with a simple Hello message.

The directory structure will be:

- `/dockerized_flask_app`
  - `Dockerfile`
  - `app.py`
  - `requirements.txt`
  - `/templates`
    - `Index.html`

The code in the ***dockerfile***:

Dockerfile

```
# Use an official Python runtime as the base image
FROM python:3.8-slim

# Set the working directory inside the container
WORKDIR /app

# Copy the requirements file into the container
COPY requirements.txt .

# Install Flask and any other dependencies
RUN pip install --no-cache-dir -r requirements.txt

# Copy the rest of the application into the container
COPY . .

# Make port  available to the host
EXPOSE 8080

# Define the command to run the Flask application
CMD ["python", "app.py"]
```

The ***app.py*** looks like this :

```python
app.py

from flask import Flask, render_template

app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html')

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0')
```

The ***requirements*** that were installed to make the file execute

```
requirements.txt

Flask==1.1.2
Jinja2==2.11.3
MarkupSafe==1.1.1
itsdangerous==1.1.0
Werkzeug==0.16.1
```

The ***Index.html page*** showing the output that will be displayed on output

```html
index.html

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Welcome to Flask with Docker!</title>
</head>
<body>
    <h1>Hello from Flask!</h1>
    <p>Welcome to your new Flask project running inside a Docker.</p>
</body>
</html>
```

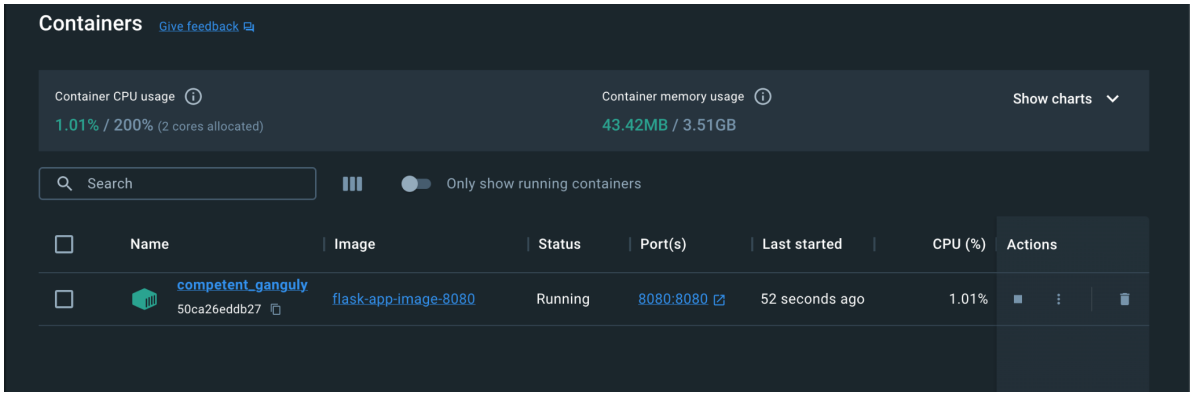
Building the Docker image from Docker file

```
dockerized_flask_app — -zsh — 88×28
aviajmera@MacBook-Air-17 dockerized_flask_app % docker build -t flask-app-image-8080 .

[+] Building 3.0s (11/11) FINISHED                      docker:desktop-linux
 => [internal] load build definition from Dockerfile                    0.1s
 => => transferring dockerfile: 913B                                    0.0s
 => [internal] load .dockerignore                                       0.0s
 => => transferring context: 2B                                         0.0s
 => [internal] load metadata for docker.io/library/python:3.8-slim      2.4s
 => [auth] library/python:pull token for registry-1.docker.io           0.0s
 => [1/5] FROM docker.io/library/python:3.8-slim@sha256:64951435db9e09  0.0s
 => [internal] load build context                                       0.0s
 => => transferring context: 1.52kB                                     0.0s
 => CACHED [2/5] WORKDIR /app                                           0.0s
 => CACHED [3/5] COPY requirements.txt .                                0.0s
 => CACHED [4/5] RUN pip install --no-cache-dir -r requirements.txt     0.0s
 => [5/5] COPY . .                                                      0.1s
 => exporting to image                                                  0.0s
 => => exporting layers                                                 0.0s
 => => writing image sha256:ac5b68a5c79aceca6aaeb5125f7551174f3a2f6c70  0.0s
 => => naming to docker.io/library/flask-app-image-8080                 0.0s

What's Next?
  View summary of image vulnerabilities and recommendations → docker scout quickview
aviajmera@MacBook-Air-17 dockerized_flask_app %
```

Running the DockerImage in the container

```
aviajmera@MacBook-Air-17 dockerized_flask_app % docker run -p 8080:8080 flask-app-image-8080

 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 160-568-745
```

The STATS of the container:


Containers
Give feedback
Container CPU usage
1.01% / 200% (2 cores allocated)
Container memory usage
43.42MB / 3.51GB
Show charts
Search
Only show running containers
Name
Image
Status
Port(s)
Last started
CPU (%)
Actions
competent_ganguly
50ca26eddb27
flask-app-image-8080
Running
8080:8080
52 seconds ago
1.01%

The Output :


localhost:8080
Hello from Flask!
Welcome to your new Flask project running inside a Docker.

Flask Web Application in Vagrant

This project encapsulates a lightweight Flask-based web service within a Vagrant-managed virtual environment. It leverages the power of Vagrant to create a reproducible, consistent environment that can be easily shared and deployed across various development setups.

Project Directory Structure:

- `/project_folder`
  - `Vagrantfile`
  - `/app`
    - `app.py`
    - `requirements.txt`
    - `/templates`
      - `index.html`

The ***VAGRANT FILE***

Vagrantfile

```
Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/bionic64"
  config.vm.network "forwarded_port", guest: 5000, host: 8080

  config.vm.provision "shell", inline: <<-SHELL
    sudo apt-get update
    sudo apt-get install -y python3-pip
    sudo pip3 install -r /vagrant/app/requirements.txt
  SHELL
end
```

The flask application file - ***App.py***

```python
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html')

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0')
```

The requirements to be installed in the ***requirements.txt*** file

```
Flask==1.1.2
```

The ***Index page***

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Welcome to Flask with Vagrant!</title>
</head>
<body>
    <h1>Hello from Flask!</h1>
    <p>Welcome to your new Flask project running inside a Vagrant box.</p>
</body>
</html>
```

Running the Vagrant file to set up the environment

```
aviajmera@MacBook-Air-17 project_folder % vagrant up

Bringing machine 'default' up with 'virtualbox' provider...
==> default: Importing base box 'ubuntu/bionic64'...
==> default: Matching MAC address for NAT networking...
==> default: Checking if box 'ubuntu/bionic64' version '20230607.0.0' is up to date...
==> default: Setting the name of the VM: project_folder_default_1694747353249_63427
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
    default: Adapter 1: nat
==> default: Forwarding ports...
    default: 5000 (guest) => 8080 (host) (adapter 1)
    default: 22 (guest) => 2222 (host) (adapter 1)
==> default: Running 'pre-boot' VM customizations...
==> default: Booting VM...
==> default: Waiting for machine to boot. This may take a few minutes...
    default: SSH address: 127.0.0.1:2222
    default: SSH username: vagrant
    default: SSH auth method: private key
    default: Warning: Connection reset. Retrying...
    default:
    default: Vagrant insecure key detected. Vagrant will automatically replace
    default: this with a newly generated keypair for better security.
    default:
    default: Inserting generated public key within guest...
    default: Removing insecure key from the guest if it's present...
    default: Key inserted! Disconnecting and reconnecting using new SSH key...
==> default: Machine booted and ready!
==> default: Checking for guest additions in VM...
    default: The guest additions on this VM do not match the installed version of
    default: VirtualBox! In most cases this is fine, but in rare cases it can
    default: prevent things such as shared folders from working properly. If you see
    default: shared folder errors, please make sure the guest additions within the
    default: virtual machine match the version of VirtualBox you have installed on
    default: your host and reload your VM.
    default:
    default: Guest Additions Version: 5.2.42
    default: VirtualBox Version: 7.0
==> default: Mounting shared folders...
    default: /vagrant => /Users/aviajmera/Downloads/project_folder
==> default: Running provisioner: shell...
    default: Running: inline script
    default: Hit:1 http://archive.ubuntu.com/ubuntu bionic InRelease
    default: Get:2 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
    default: Get:3 http://archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
    default: Get:4 http://archive.ubuntu.com/ubuntu bionic-backports InRelease [83.3 kB]
    default: Get:5 http://security.ubuntu.com/ubuntu bionic-security/main amd64 Packages [2717 kB]
    default: Get:6 http://archive.ubuntu.com/ubuntu bionic/universe amd64 Packages [8570 kB]
    default: Get:7 http://security.ubuntu.com/ubuntu bionic-security/main Translation-en [467 kB]
    default: Get:8 http://security.ubuntu.com/ubuntu bionic-security/restricted amd64 Packages [1317 kB]
    default: Get:9 http://security.ubuntu.com/ubuntu bionic-security/restricted Translation-en [182 kB]
    default: Get:10 http://security.ubuntu.com/ubuntu bionic-security/universe amd64 Packages [1303 kB]
    default: Get:11 http://security.ubuntu.com/ubuntu bionic-security/universe Translation-en [308 kB]
    default: Get:12 http://security.ubuntu.com/ubuntu bionic-security/multiverse amd64 Packages [19.8 kB]
    default: Get:13 http://security.ubuntu.com/ubuntu bionic-security/multiverse Translation-en [3928 B]
    default: Get:14 http://archive.ubuntu.com/ubuntu bionic/universe Translation-en [4941 kB]
    default: Get:15 http://archive.ubuntu.com/ubuntu bionic/multiverse amd64 Packages [151 kB]
    default: Get:16 http://archive.ubuntu.com/ubuntu bionic/multiverse Translation-en [108 kB]
    default: Get:17 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [3045 kB]
    default: Get:18 http://archive.ubuntu.com/ubuntu bionic-updates/main Translation-en [553 kB]
    default: Get:19 http://archive.ubuntu.com/ubuntu bionic-updates/restricted amd64 Packages [1347 kB]
    default: Get:20 http://archive.ubuntu.com/ubuntu bionic-updates/restricted Translation-en [187 kB]
```

Entering the Terminal of the environment created

```
zsh: no such user or named directory: vagrant
[aviajmera@MacBook-Air-17 project_folder % vagrant ssh
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 4.15.0-212-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Fri Sep 15 03:12:45 UTC 2023

  System load:  0.29               Processes:             104
  Usage of /:   4.2% of 38.70GB    Users logged in:       0
  Memory usage: 15%                IP address for enp0s3: 10.0.2.15
  Swap usage:   0%


Expanded Security Maintenance for Infrastructure is not enabled.

11 updates can be applied immediately.
10 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

59 additional security updates can be applied with ESM Infra.
Learn more about enabling ESM Infra service for Ubuntu 18.04 at
https://ubuntu.com/18-04

New release '20.04.6 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


vagrant@ubuntu-bionic:~$
```

Running the python file in the environment

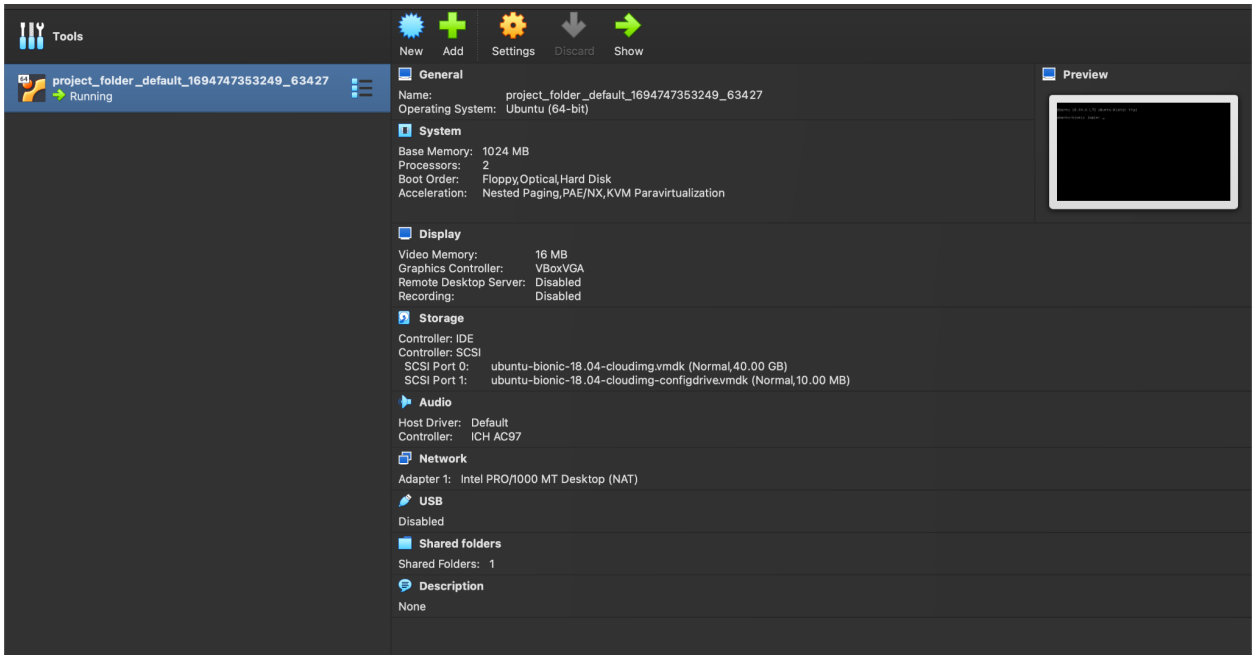
```
vagrant@ubuntu-bionic:~$ cd /vagrant/app
vagrant@ubuntu-bionic:/vagrant/app$ python3 app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://10.0.2.15:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 955-762-305
```

The output

Hello from Flask!

Welcome to your new Flask project running inside a Vagrant box.

The environment running in the virtual box



Following are the comparisons between the docker container and the VM.
VM using vagrant

- Start-Up time: 2-3 minutes
- Memory consumed: 4.2% of 38.7 GB

Docker Container

- Start Up time: 3 seconds to build the app.
- Memory Consumed: 43.42MB
- CPU Utilization: 1.01%