Docker Image

A. Prerequisites

- A Linux system
- Access to the command-line/terminal window
- Access to a user account with root or sudo privileges
- Docker installed and configured

B. How to Create a Dockerfile

The first thing you need to do is to create a directory in which you can store all the Docker images you build.

1. As an example, we will create a directory named MyDockerImages with the command:

```
$ mkdir MyDockerImages
```

2. Move into that directory and create a new empty file (Dockerfile) in it by typing:

```
$ cd MyDockerImages
```

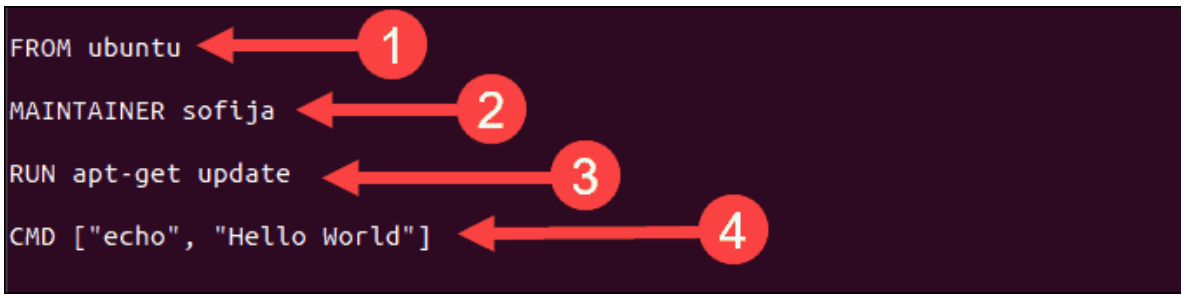
```
$ touch Dockerfile
```

3. Open the file with a text editor of your choice. In this example, we opened the file using Vim:

```
$ vim Dockerfile
```

4. Then, add the following content:

```
FROM ubuntu
MAINTAINER sofija
RUN apt-get update
CMD ["echo", "Hello World"]
```



- **FROM** – Defines the base of the image you are creating. You can start from

a parent image (as in the example above) or a base image. When using a parent image, you are using an existing image on which you base a new one. Using a base image means you are starting from scratch (which is exactly how you would define it: FROM scratch).

- **MAINTAINER** – Specifies the author of the image. Here you can type in your first and/or last name (or even add an email address). You could also use the LABEL instruction to add metadata to an image.
- **RUN** – Instructions to execute a command while building an image in a layer on top of it. In this example, the system searches for repository updates once it starts building the Docker image. You can have more than one RUN instruction in a Dockerfile.
- **CMD** – There can be only one CMD instruction inside a Dockerfile. Its purpose is to provide defaults for an executing container. With it, you set a default command. The system will execute it if you run a container without specifying a command.

5. Save and exit the file.

```
:wq
```

6. You can check the content of the file by using the cat command:

```
$ cat Dockerfile
```

C. Build a Docker Image with Dockerfile

The basic syntax used to build an image using a Dockerfile isA system running Ubuntu 22.04.

```
$ docker build [OPTIONS] PATH | URL | -
```

To build a docker image, you would therefore use:

```
$ docker build [location of your dockerfile]
```

If you are already in the directory where the Dockerfile is located, put a . instead of the location:

```
$ docker build .
```

By adding the -t flag, you can tag the new image with a name which will help you

when dealing with multiple images:

```
$ docker build -t my_first_image .
```

Once the image is successfully built, you can verify whether it is on the list of local images with the command:

```
$ docker images
```

The output should show my_first_image available in the repository.

Try create a new container based on the docker image you created in, at the previous step.

```
$ docker run --name test my_first_image
```

The Hello World message should appear in the command line, as seen in the image above.

D. Dockerize a Flask Application

Flask is a popular Python micro web framework that helps you develop lightweight web applications and APIs quickly and easily.

As a web framework, it provides greater flexibility, customization, and scalability for simple web applications while remaining highly compatible with cutting-edge technologies.

- **Set Up the Project**

  Basic directory structure, After completing the following steps, our application directory structure will look like this:

  flask-docker
  ├── app.py
  ├── Dockerfile
  ├── requirements.txt
  └── venv

- **Create virtual environment**

  ```
  $ python3 -m venv venv
  ```

- **Active the environment**

  ```
  $ . venv/bin/activate
  Or
  $ source venv/bin/activate
  ```

- **Edit requirements.txt**

```
click==8.0.3
Flask==2.0.2
itsdangerous==2.0.1
Jinja2==3.0.2
MarkupSafe==2.0.1
Werkzeug==2.0.2
```

- Install the requirements

```
$ pip install -r requirements.txt
```

- Modify app.py

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_geek():
    return '<h1>Hello from Flask & Docker</h2>'


if __name__ == "__main__":
    app.run(debug=True)
```

- Modify Dockerfile

```
# syntax=docker/dockerfile:1

FROM python:3.8-slim-buster

WORKDIR /python-docker

COPY requirements.txt requirements.txt
RUN pip3 install -r requirements.txt

COPY . .

CMD [ "python3", "-m" , "flask", "run", "--host=0.0.0.0"]
```

- Run the application

```
$ python app.py
```

- Build a Docker Image

After that, all that remains is to build our image. Using docker build, we can now

enlist Docker's help in building the image. You can combine the build command with other tags, such as the "--tag" flag, to specify the image name.

```
$ docker build --tag python-docker .
```

- Check images

```
$ docker images
```

- Run an image as a container

```
$ docker run -d -p 5000:5000 python-docker
```

Reference

- https://phoenixnap.com/kb/create-docker-images-with-dockerfile
- https://www.freecodecamp.org/news/how-to-dockerize-a-flask-app/