Here’s a step-by-step guide for someone who has the same Dockerfile and wants to run the Flask app from the beginning until accessing it on `localhost:8000`. This guide assumes they have Docker installed and their project structure is similar to yours.

---

**Step 1: Build the Docker Image**

Before running the container, you need to build the Docker image. Navigate to the directory containing your `Dockerfile` and execute the following command:

```bash

sudo docker build -t my-flask-mongo-app .

```

This command builds the Docker image and tags it as `my-flask-mongo-app`. The `.` at the end specifies the current directory as the build context .

---

**Step 2: Run the Container in Interactive Mode**

To ensure you can debug or manually start the Flask app, run the container interactively with a bind mount for real-time updates:

```bash

sudo docker run -it -p 8000:8000 -v "$(pwd)/app:/app" my-flask-mongo-app /bin/bash

```

- `-it`: Runs the container in interactive mode with a terminal session.

- `-p 8000:8000`: Maps port `8000` on your local machine to port `8000` in the container.

- `-v "$(pwd)/app:/app"`: Binds your local `/app` directory to the `/app` directory in the container, ensuring changes are reflected immediately .

- `/bin/bash`: Opens a shell session inside the container.

---

**Step 3: Start MongoDB Inside the Container**

Once inside the container, start MongoDB in the background:

```bash

mongod --fork --logpath /var/log/mongodb.log

```

Verify that MongoDB is running by checking its log file:

```bash

cat /var/log/mongodb.log

```

If MongoDB starts successfully, you should see no errors in the log file.

---

**Step 4: Navigate to the `/app` Directory**

Change to the `/app` directory where your Flask application is located:

```bash

cd /app

```

---

**Step 5: Run the Flask App**

Start the Flask app (`server.py`) using Python:

```bash

python3 server.py

```

Ensure that your `server.py` file is configured to listen on `0.0.0.0` (not `127.0.0.1`) so it accepts connections from outside the container. For example:

```python

if \_\_name\_\_ == '\_\_main\_\_':

app.run(host='0.0.0.0', port=8000)

```

---

**Step 6: Access the Web App in Your Browser**

Open your browser and navigate to:

```

http://localhost:8000

```

You should now see your Flask app running. Any changes you make to the `/app` directory on your local machine will be immediately reflected in the container due to the bind mount .

---

**Optional: Automate the Process**

If you want to avoid manually starting MongoDB and the Flask app every time, update the `CMD` instruction in your Dockerfile to automate this process:

```Dockerfile

CMD ["sh", "-c", "mongod --fork --logpath /var/log/mongodb.log && python3 /app/server.py"]

```

**Rebuild the image:**

```bash

sudo docker build -t my-flask-mongo-app .

```

**Run the container in detached mode:**

```bash

sudo docker run -d -p 8000:8000 -v "$(pwd)/app:/app" my-flask-mongo-app

```

Now, both MongoDB and the Flask app will start automatically when the container runs .

---

Summary

1. Build the Docker image: `sudo docker build -t my-flask-mongo-app .`

2. Run the container interactively: `sudo docker run -it -p 8000:8000 -v "$(pwd)/app:/app" my-flask-mongo-app /bin/bash`

3. Start MongoDB: `mongod --fork --logpath /var/log/mongodb.log`

4. Navigate to `/app`: `cd /app`

5. Run the Flask app: `python3 server.py`

6. Access the app in your browser: `http://localhost:8000`

By following these steps, anyone can successfully run your Flask app inside a Docker container and access it locally .