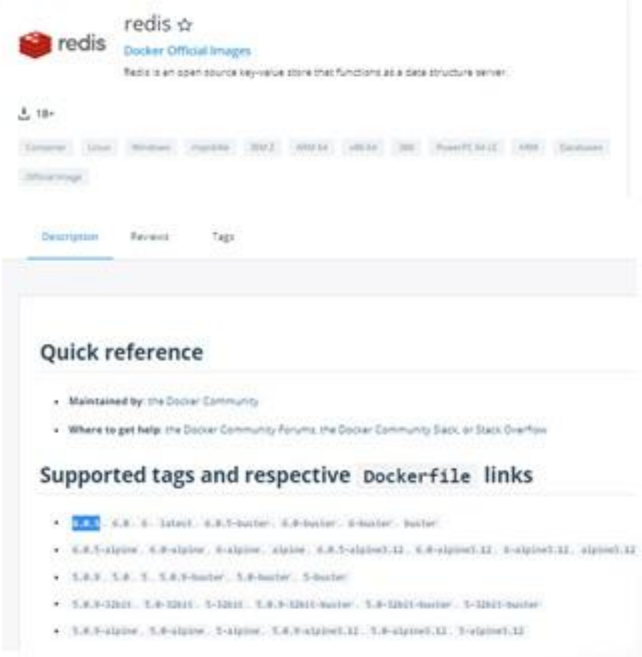
Docker Run

Docker Commands - Run

- Run a Redis Container
  - **docker run redis :latest**
- Run a Redis Container with specific version
  - **docker run redis:4.0**

TAG

- Find information about versions
  - Open https://hub.docker.com
  - Type redis
  - Open the official redis repository
  - Here you will get information about all supported tags



Redis is an open source key-value store that functions as a data structure server.

Quick reference

- **Maintained by:** the Docker Community
- **Where to get help:** the Docker Community Forums, the Docker Community Slack, or Stack Overflow

Supported tags and respective `Dockerfile` links

- `6.0.5`, `6.0`, `6`, `latest`, `6.0.5-buster`, `6.0-buster`, `6-buster`, `buster`
- `6.0.5-alpine`, `6.0-alpine`, `6-alpine`, `alpine`, `6.0.5-alpine3.12`, `6.0-alpine3.12`, `6-alpine3.12`, `alpine3.12`
- `5.0.9`, `5.0`, `5`, `5.0.9-buster`, `5.0-buster`, `5-buster`
- `5.0.9-32bit`, `5.0-32bit`, `5-32bit`, `5.0.9-32bit-buster`, `5.0-32bit-buster`, `5-32bit-buster`
- `5.0.9-alpine`, `5.0-alpine`, `5-alpine`, `5.0.9-alpine3.12`, `5.0-alpine3.12`, `5-alpine3.12`

Exercise

- Run Ubuntu 14.04 and print the release information

SkillAssure

Docker Commands – Run (STDIN)

- Run a simple prompt app
  - **./app.sh**
  - **It will ask for prompt**

- Run the dockerized simple prompt app
  - **docker run prabhavagrawal/simple-prompt**
  - **Whereas if you run the same with docker it does not ask.**

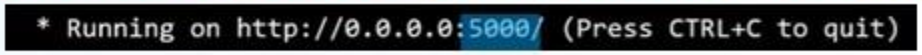
- Run it in interactive mode
  - **docker run -i prabhavagrawal/simple-prompt**
  - **When I input my name it prints the expected output. Something missing?**
  - **Welcome prompt is missing**

- Run it in interactive mode
  - **docker run -it prabhavagrawal/simple-prompt**
  - **When I input my name it prints the expected output**

SkillAssure

Docker Commands – Run (Port Mapping)

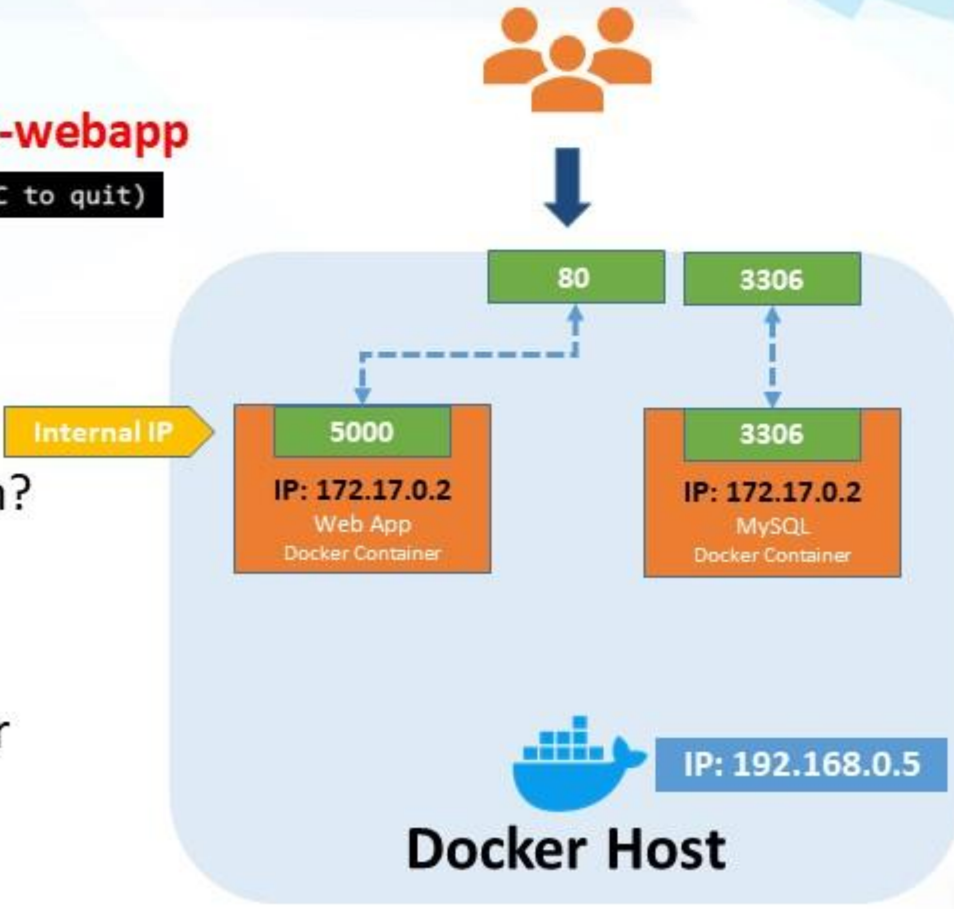
- Run a web application
  - **docker run prabhavagrawal/simple-webapp**



```
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```

- How do I access my application?
  - Using IP Address and port
  - But this is only internal IP
- How do my users access the application?
- Use Docker Host IP Address

- Map port from docker host to container
  - **docker run -p 80:5000 prabhavagrawal/simple-webapp**
- Run other container on different port
  - **docker run -p 3306:3306 mysql**



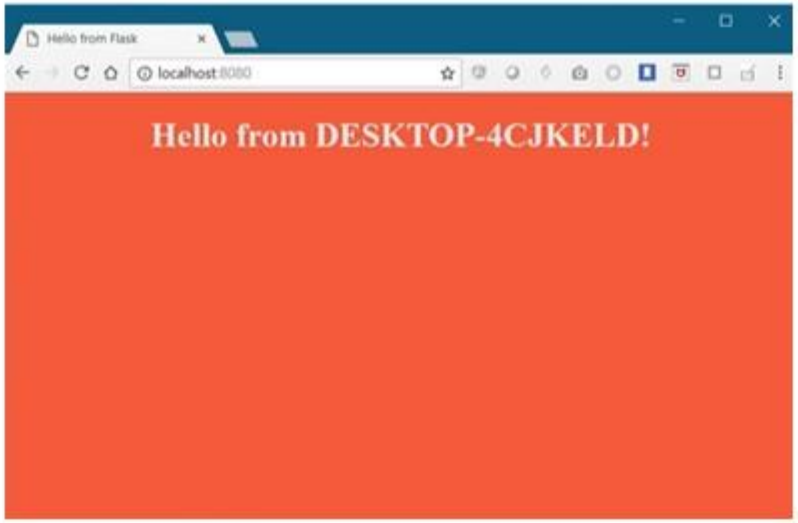
Exercise

- Run Nginx container and map it to port 8088
- Run a MySQL container and map it to port 3306
- Log in to MySQL container

Docker Commands – Run (Environment Variables)

app.py

```
import os
from flask import Flask

app = Flask(__name__)


...
...

color = "red"

@app.route("/")
def main():
    print(color)
    return render_template('hello.html', color=color)


if __name__ == "__main__":
    app.run(host="0.0.0.0", port="8080")
```



Hello from DESKTOP-4CJKELD!

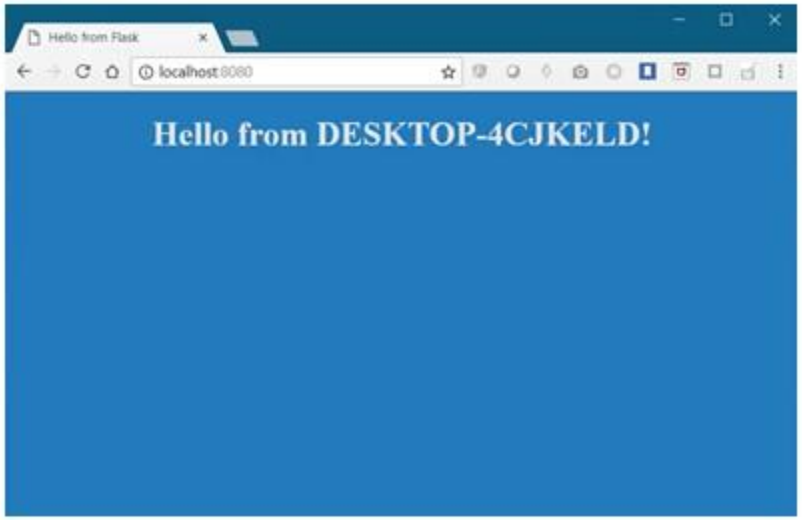
```
python app.py
```

SkillAssure

Docker Commands – Run (Environment Variables)

app.py

```
import os
from flask import Flask

app = Flask(__name__)

…
…

color = os.environ.get('APP_COLOR')

@app.route("/")
def main():
    print(color)
    return render_template('hello.html', color=color)


if __name__ == "__main__":
    app.run(host="0.0.0.0", port="8080")
```


Hello from Flask

localhost:8080

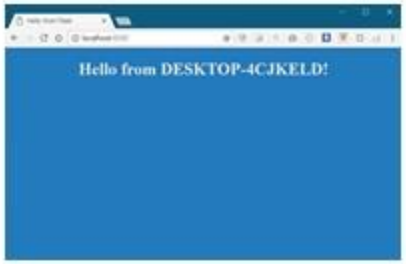
Hello from DESKTOP-4CJKELD!

```
export APP_COLOR=blue; python app.py
```

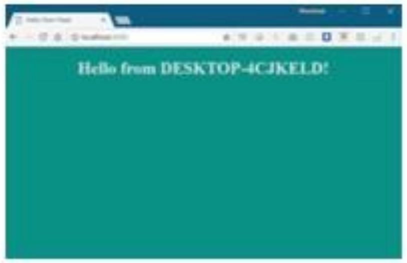
Docker Commands – Run (Environment Variables)

- Pass the environment variable while running the container

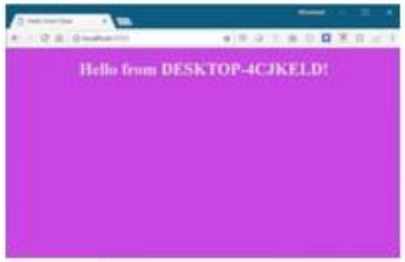
```
docker run -e APP_COLOR=blue prabhavagrawal/simple-webapp-color
```


Hello from DESKTOP-4CJKELD!

```
docker run -e APP_COLOR=green prabhavagrawal/simple-webapp-color
```


Hello from DESKTOP-4CJKELD!

```
docker run -e APP_COLOR=pink prabhavagrawal/simple-webapp-color
```


Hello from DESKTOP-4CJKELD!

Exercise

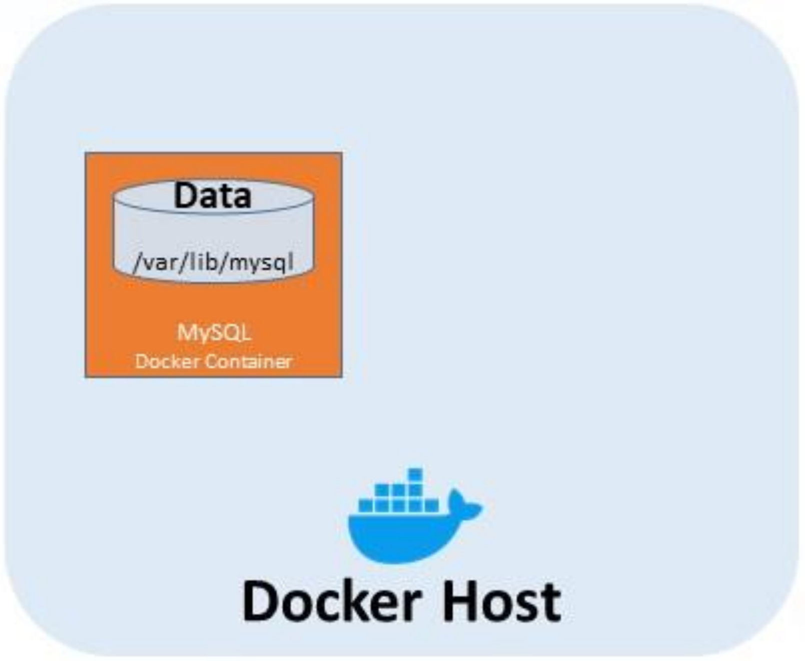
- Create a MySQL Container using environment variable
- Login to MySQL container and create a database

- Stop the MySQL container
- Remove the MySQL container
- Launch another MySQL container and see if the database exists

SkillAssure

Docker Commands – Run (Volume)

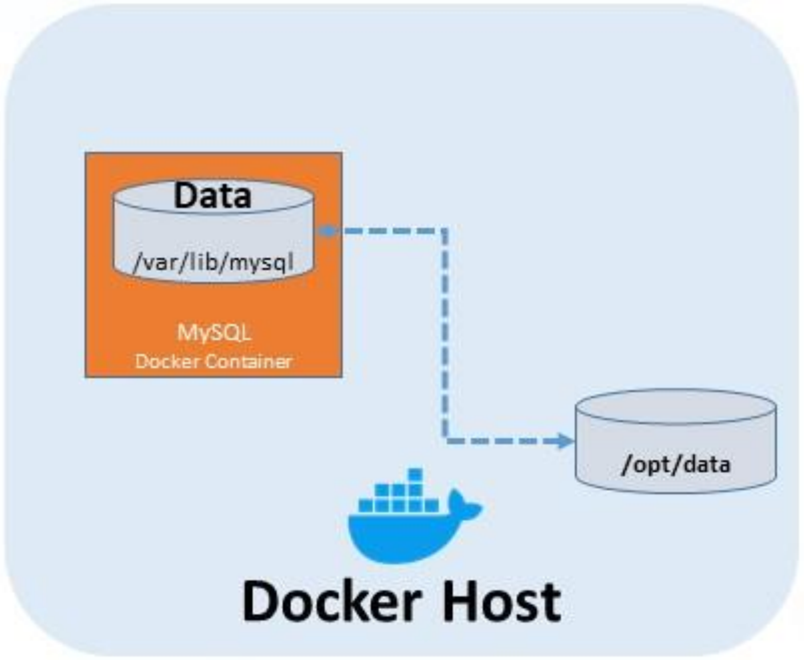
- Run a MySQL container
  - **docker run mysql**
- Now you dump a lot of data on this container
- Stop the container
  - **docker stop mysql**
- Remove the container
  - **docker rm mysql**
- Your container along with the data is gone



SkillAssure

Docker Commands – Run (Volume)

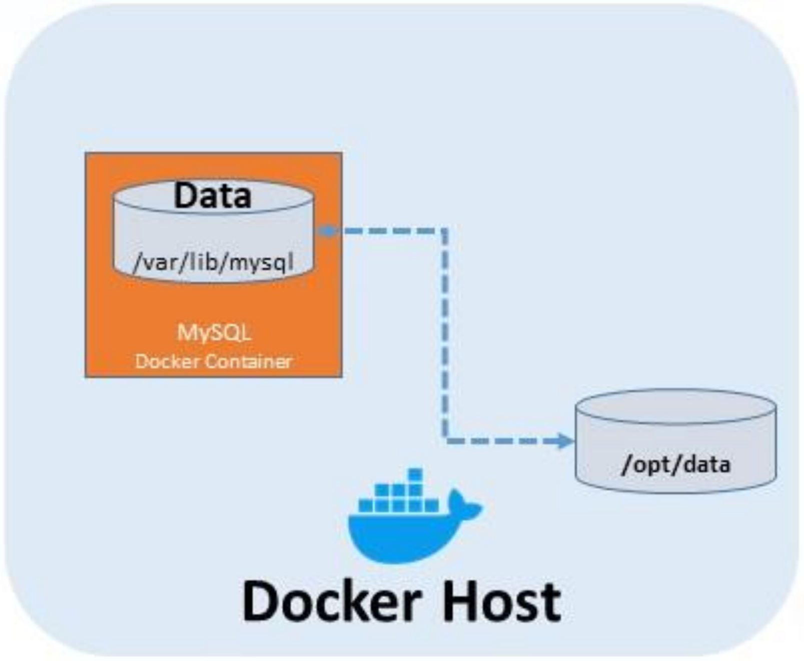
- Run a MySQL container with volume mapping
  - **docker run –v /opt/data:/var/lib/mysql mysql**



Docker Commands – Run (Volume)

- All your data will be stored in /opt/data

- **Data will remain even when you delete the container**



Exercise

- Run a Jenkins container
  - Host Port - 8089
  - Host Volume - /root/Jenkins-data

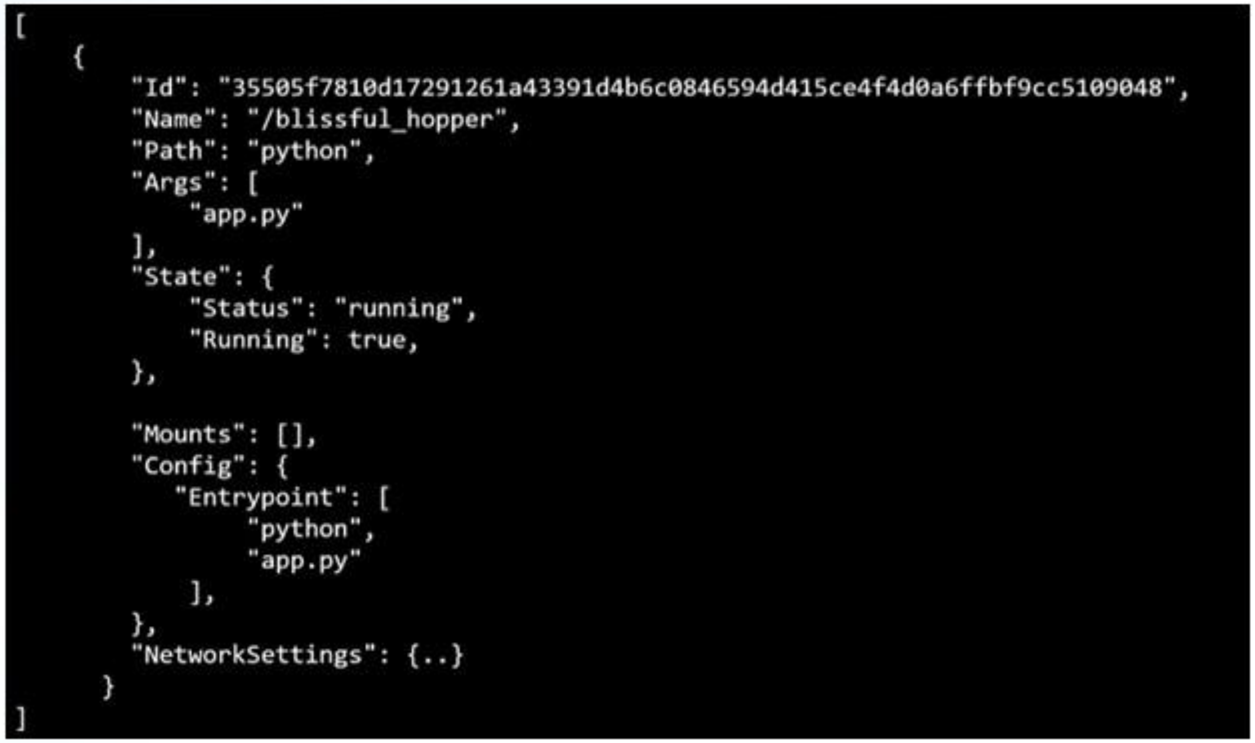
- Perform the Jenkins configuration
- Create a sample Jenkins job
- Delete the Jenkins container

- Create a new Jenkins container
  - Host Port - 8089
  - Host Volume - /root/Jenkins-data

SkillAssure

Docker Commands – Inspect

- Get more details about a container
  - **docker inspect <Container_ID>**



```
[
    {
        "Id": "35505f7810d17291261a43391d4b6c0846594d415ce4f4d0a6ffbf9cc5109048",
        "Name": "/blissful_hopper",
        "Path": "python",
        "Args": [
            "app.py"
        ],
        "State": {
            "Status": "running",
            "Running": true,
        },

        "Mounts": [],
        "Config": {
           "Entrypoint": [
                "python",
                "app.py"
            ],
        },
        "NetworkSettings": {..}
      }
]
```

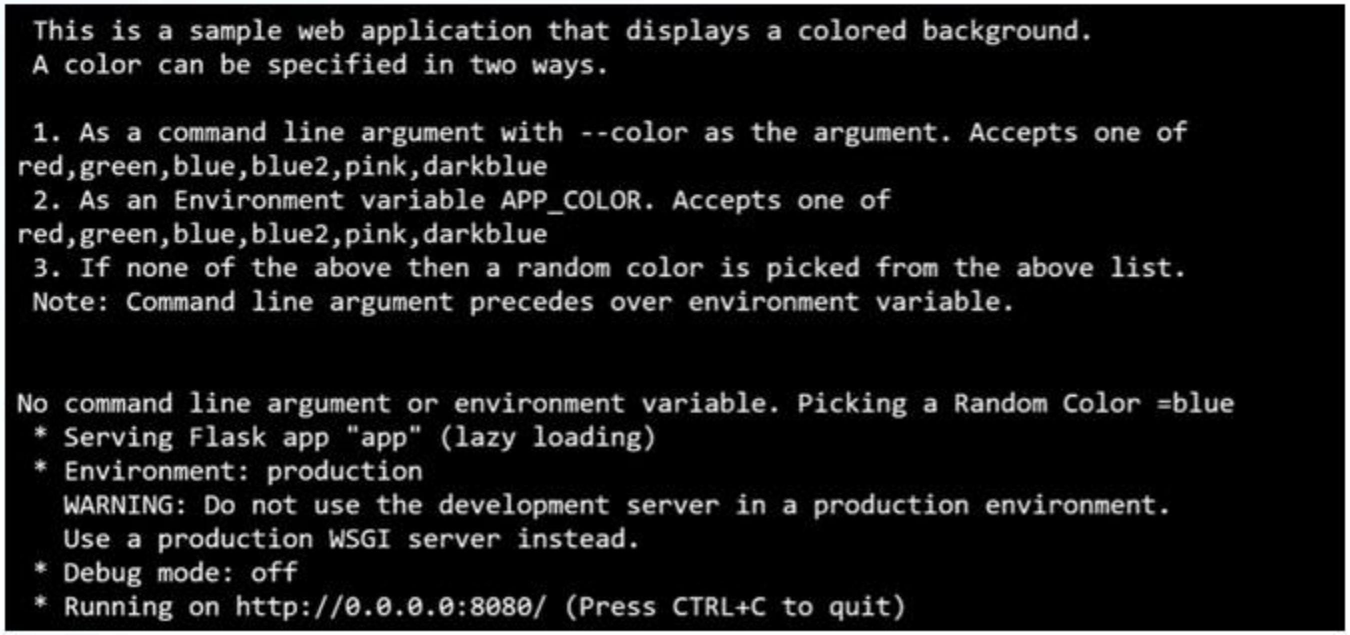
Exercise

- Find the container IP of a running mysql container.

SkillAssure

Docker Commands – Logs

- Get logs of a running container
  - **docker logs <Container_ID>**



```
 This is a sample web application that displays a colored background.
 A color can be specified in two ways.

 1. As a command line argument with --color as the argument. Accepts one of
red,green,blue,blue2,pink,darkblue
 2. As an Environment variable APP_COLOR. Accepts one of
red,green,blue,blue2,pink,darkblue
 3. If none of the above then a random color is picked from the above list.
 Note: Command line argument precedes over environment variable.


No command line argument or environment variable. Picking a Random Color =blue
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://0.0.0.0:8080/ (Press CTRL+C to quit)
```

Docker Commands

- To check Live Performance of a container
  - **docker stats**
- To see container process
  - **docker top <containerID>**
- To shell inside a running container
  - **docker exec -it <containerID> bash**

Exercise