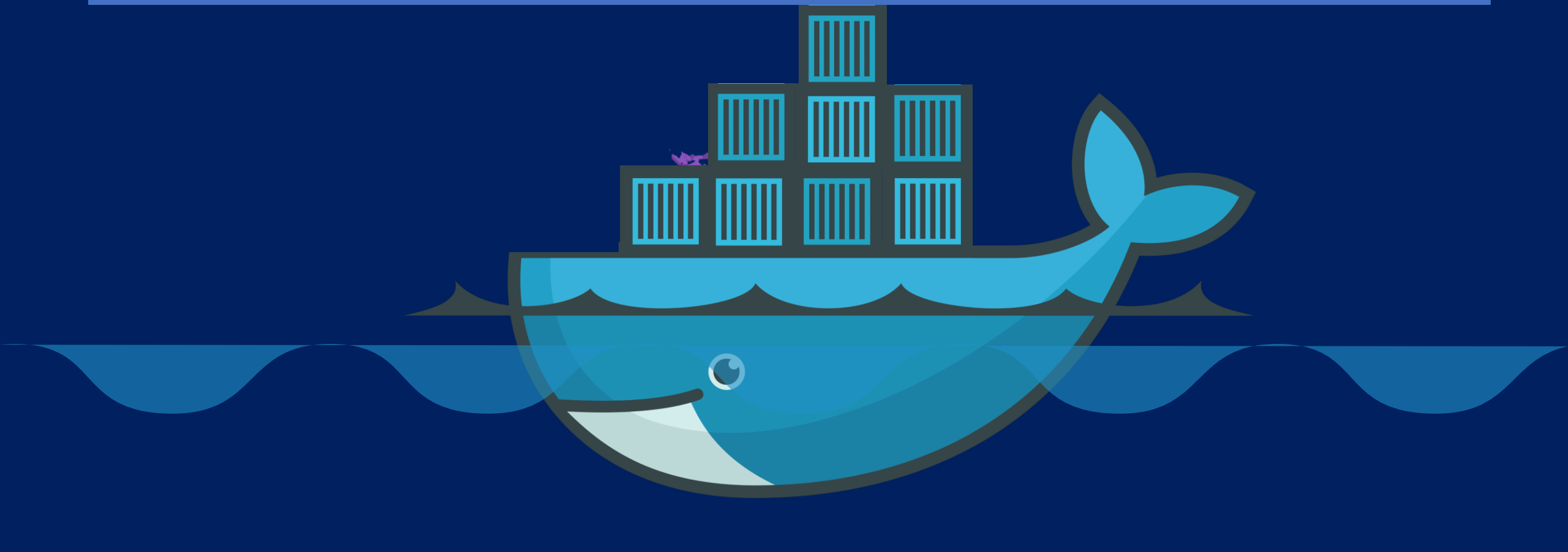
# MACHINE LEARNING OPERATIONS

WEEK 10


UNLIMITED

Presented by **Asst. Prof. Dr. Tuchsanai Ploysuwan**

# Docker Command



# ps – list containers

```
docker ps
```

```
CONTAINER ID   IMAGE   COMMAND                  CREATED          STATUS          PORTS     NAMES
796856ac413d   nginx   "nginx -g 'daemon of…"   7 seconds ago    Up 6 seconds    80/tcp    silly_sammet
```

```
docker ps -a
```

```
CONTAINER ID   IMAGE   COMMAND                  CREATED          STATUS                      NAMES
796856ac413d   nginx   "nginx -g 'daemon of…"   7 seconds ago    Up 6 seconds                silly_sammet
cff8ac918a2f   redis   "docker-entrypoint.s…"   6 seconds ago    Exited (0) 3 seconds ago    relaxed_aryabhata
```

# STOP – stop a container

```
docker ps
```

```
CONTAINER ID   IMAGE   COMMAND                  CREATED          STATUS          PORTS     NAMES
796856ac413d   nginx   "nginx -g 'daemon of…"   7 seconds ago    Up 6 seconds    80/tcp    silly_sammet
```

```
docker stop silly_sammet
```

```
silly_sammet
```

```
docker ps -a
```

```
CONTAINER ID   IMAGE   COMMAND                  CREATED          STATUS                      NAMES
796856ac413d   nginx   "nginx -g 'daemon of…"   7 seconds ago    Exited (0) 3 seconds ago    silly_sammet
cff8ac918a2f   redis   "docker-entrypoint.s…"   6 seconds ago    Exited (0) 3 seconds ago    relaxed_aryabhata
```

To stop all running containers

```
docker stop  $(docker ps -aq)
```

docker ps -aq lists all container IDs on the Docker host, whether they are running or stopped.

# Rm – Remove a container

```
docker rm silly_sammet
```

```
silly_sammet
```

```
docker ps -a
```

```
CONTAINER ID   IMAGE     COMMAND                  CREATED              STATUS                          NAMES
cff8ac918a2f   redis     "docker-entrypoint.s…"   6 seconds ago        Exited (0) 3 seconds ago        relaxed_aryabhata
```

To delete all containers

```
docker rm  $(docker ps -aq)
```

This command uses the docker ps command with the -aq flags to list all running and stopped containers in your system, and then passes the list of container IDs to the docker rm command to remove them all.

```
docker container prune -f
```

This command will remove all stopped containers, including their networks and volumes.
The --force flag is used to skip the confirmation prompt and delete the containers directly.

# images – List images

```
docker images
```

```
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
nginx               latest              f68d6e55e065        4 days ago          109MB
redis               latest              4760dc956b2d        15 months ago       107MB
ubuntu              latest              f975c5035748        16 months ago       112MB
alpine              latest              3fd9065eaf02        18 months ago       4.14MB
```

# rmi – Remove images

```
docker rmi nginx
```

```
Untagged: nginx:latest
Untagged: nginx@sha256:96fb261b66270b900ea5a2c17a26abbfabe95506e73c3a3c65869a6dbe83223a
Deleted: sha256:f68d6e55e06520f152403e6d96d0de5c9790a89b4cfc99f4626f68146fa1dbdc
Deleted: sha256:1b0c768769e2bb66e74a205317ba531473781a78b77feef8ea6fd7be7f4044e1
Deleted: sha256:34138fb60020a180e512485fb96fd42e286fb0d86cf1fa2506b11ff6b945b03f
Deleted: sha256:cf5b3c6798f77b1f78bf4e297b27cfa5b6caa982f04caeb5de7d13c255fd7a1e
```

! Delete all dependent containers to remove image

To delete all images

```
docker rmi -f $(docker images -aq)
```

To clean up unused images.

```
docker image prune -f
```

# LAB 1 :

**Create two containers with the names "mycontainer1" and "mycontainer2" and "mycontainer3":**

```
docker run -d --name mycontainer1 nginx:latest
docker run -d --name mycontainer2 mysql:latest
docker run -d --name mycontainer3 redis:latest
```

**List all running containers:**

```
docker ps -a
```

**Stop the container with the name "mycontainer1":**

```
docker stop mycontainer1
```

**Remove the container with the name "mycontainer1":**

```
docker rm mycontainer1
```

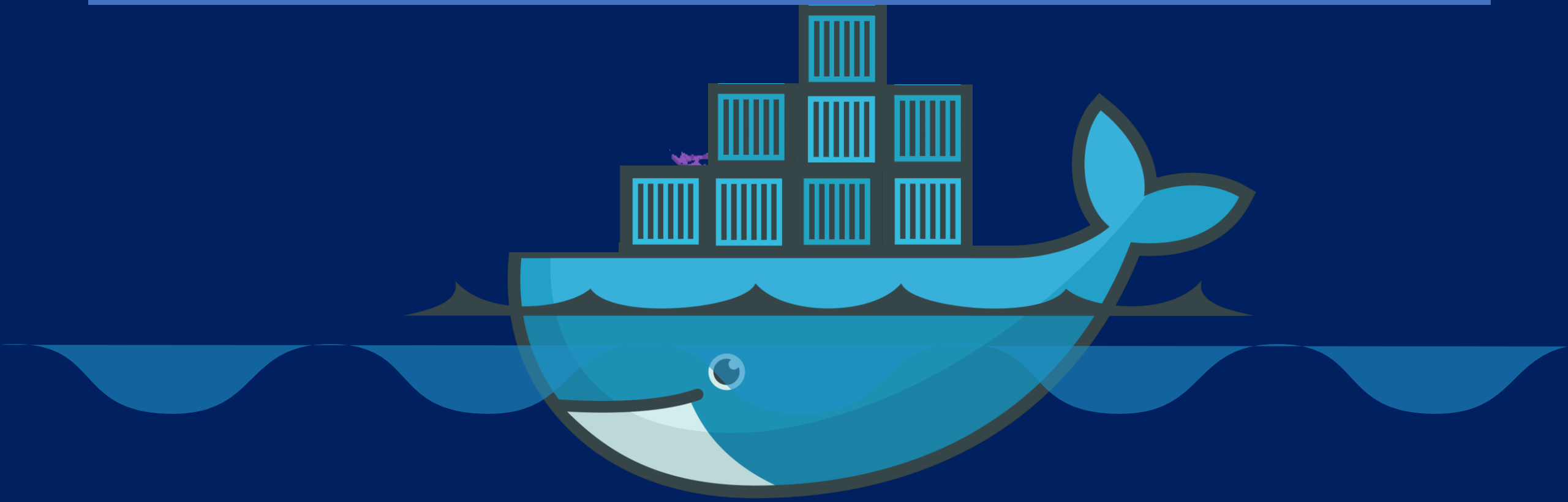
**Stop all running containers:**

```
docker stop $(docker ps -a -q)
```

**Remove all containers**

```
docker rm $(docker ps -a -q)
```

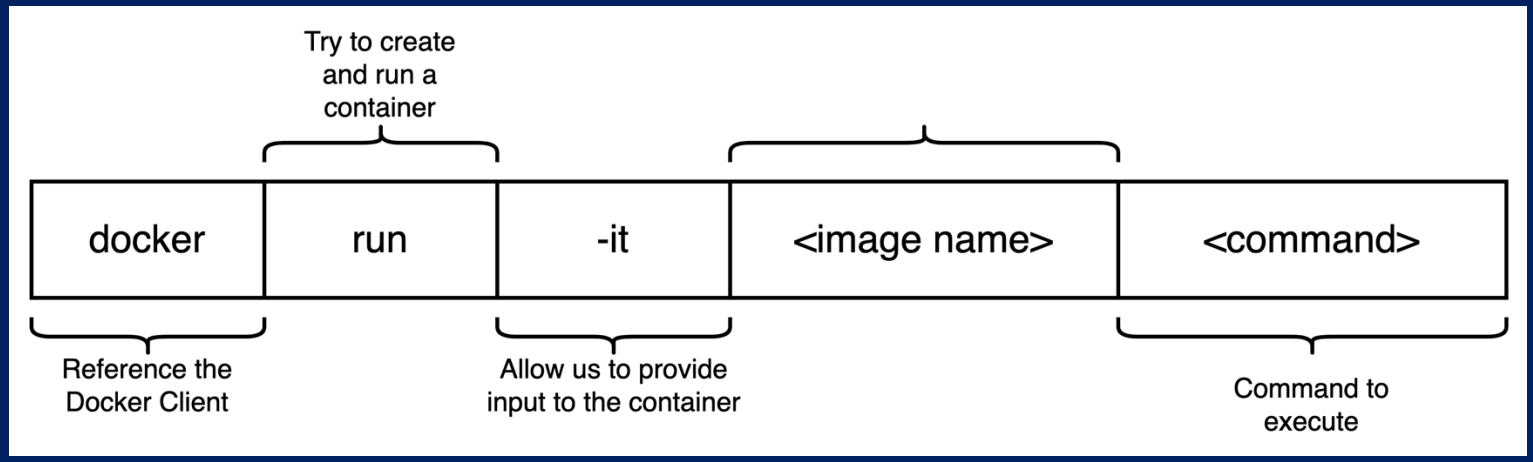
# More Docker run and Docker exec



# Run – stdin

```
docker pull ubuntu
docker run –it  ubuntu  sh
```

```
# ls
bin   dev  home  media  opt   root  sbin  sys  usr
boot  etc  lib   mnt    proc  run   srv   tmp  var
# echo hello world
hello world
#
```



# Run – attach and detach

```
docker run -d --name lab nginx
```

```
latest: Pulling from library/nginx
66dbba0fb1b5: Pull complete
6a4b1f0b5a90: Pull complete
16ea4daad357: Pull complete
646b2422838c: Pull complete
c6036fb71e57: Pull complete
dc0e78f15ad0: Pull complete
Digest: sha256:aa0afebbb3cfa473099a62c4b32e9b3fb73ed23f2a75a65ce1d4b4f55a5c2ef2
Status: Downloaded newer image for nginx:latest
252b74d97da0dd09d01cc50358ea86766894ba016dc8c03d22a01399d4298eb6
```

```
docker run  --name lab nginx
```

```
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
66dbba0fb1b5: Pull complete
6a4b1f0b5a90: Pull complete
16ea4daad357: Pull complete
646b2422838c: Pull complete
c6036fb71e57: Pull complete
dc0e78f15ad0: Pull complete
2023/03/10 03:40:11 [notice] 1#1: start worker processes
2023/03/10 03:40:11 [notice] 1#1: start worker process 29
2023/03/10 03:40:11 [notice] 1#1: start worker process 30
2023/03/10 03:40:11 [notice] 1#1: start worker process 31
2023/03/10 03:40:11 [notice] 1#1: start worker process 32
2023/03/10 03:40:11 [notice] 1#1: start worker process 33
```

# Run – tag

```
docker run redis
```

```
Using default tag: latest
latest: Pulling from library/redis f5d23c7fed46: Pull
complete Status: Downloaded newer image for
                                                    redis:latest

1:C 31 Jul 2019 09:02:32.624 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
1:C 31 Jul 2019 09:02:32.624 #            Redis version=5.0.5,          bits=64, commit=00000000, modified=0,          pid=1, just started
1:M 31 Jul 2019 09:02:32.626 # Server initialized
```

```
docker run redis:4.0
```

TAG

```
Unable to find image 'redis:4.0' locally 4.0: Pulling from
library/redis e44f086c03a2: Pull complete
Status: Downloaded newer image for redis:4.0


1:C 31 Jul 09:02:56.527 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
1:C 31 Jul 09:02:56.527 # Redis          version=4.0.14,          bits=64, commit=00000000, modified=0, pid=1, just started
1:M 31 Jul 09:02:56.530 # Server initialized
```

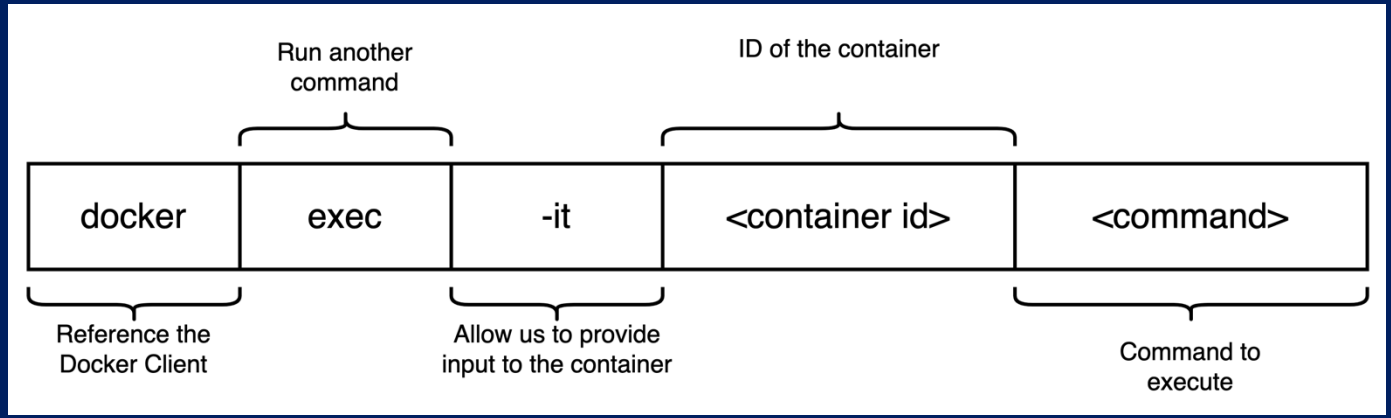
# Exec– execute a command

```
docker ps -a
```

```
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              NAMES
538d037f94a7        ubuntu              "sleep 100“         6 seconds ago       Up 4 seconds        distracted_mcclintock
```

```
docker exec distracted_mcclintock cat /etc/hosts
```

```
127.0.0.1       localhost
::1     localhost ip6-localhost ip6-loopback fe00::0 ip6-
localnet
ff00::0 ip6-mcastprefix ff02::1 ip6-
allnodes ff02::2 ip6-allrouters
172.18.0.2      538d037f94a7
```



# Inspect Container

```
docker inspect blissful_hopper
```


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

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

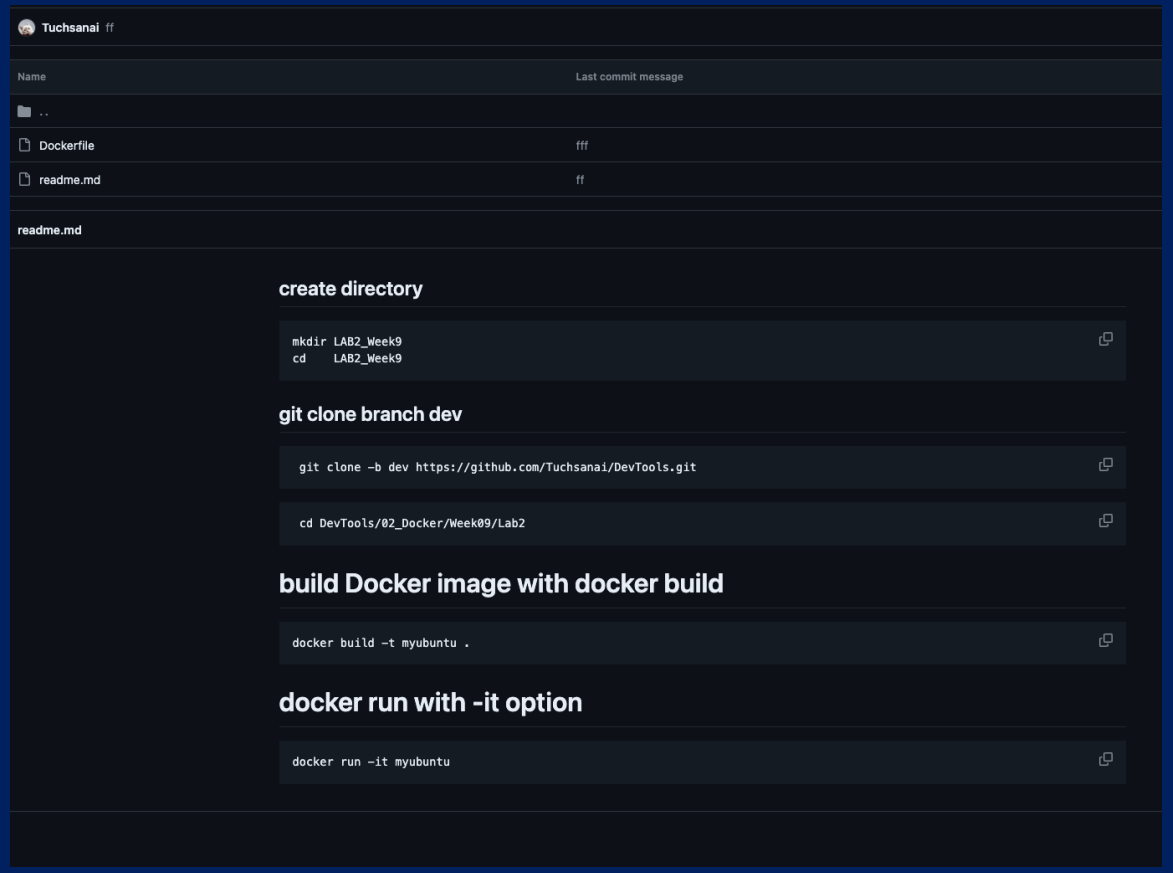
# Container Logs

```
docker logs blissful_hopper
```

```
 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)
```

# LAB 2 :

## Run – stdin

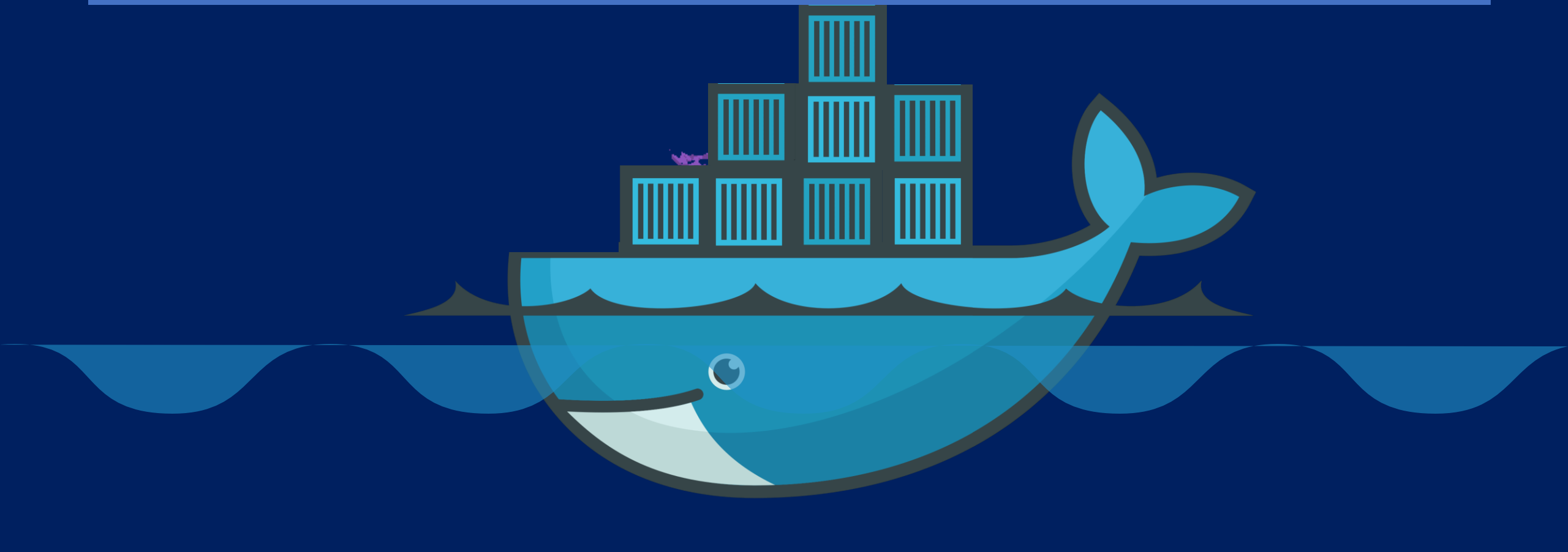



python Copy code

```
docker run -it ubuntu /bin/bash
```

คำสั่งดังกล่าวจะสร้าง Docker container จาก Ubuntu image และเปิด shell ของ Ubuntu ให้คุณสามารถเข้าถึงและใช้งานได้ทันที หากไม่มี Ubuntu image อยู่ในเครื่องของคุณ ระบบจะดาวน์โหลด Ubuntu image มาให้อัตโนมัติก่อนเริ่มทำงานใน Docker container นี้

# Docker Environment Varibles

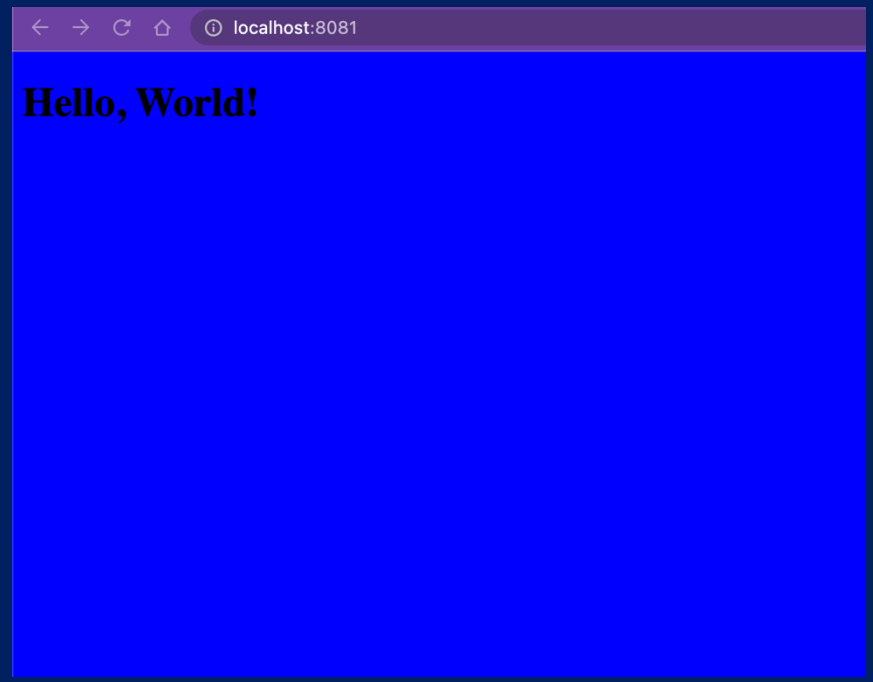


# Flask

```python
import os
from flask import Flask, render_template

app = Flask(__name__,template_folder="")


@app.route('/')
def home():

    env_var_colour = os.environ['APP_COLOR']

    return render_template("index.html", color= env_var_colour)


@app.route('/<string:name>')
def template(name):
    return render_template("index.html", color=name)


if __name__ == '__main__':
    app.run(host="0.0.0.0",port="8081")
```

index.html > html

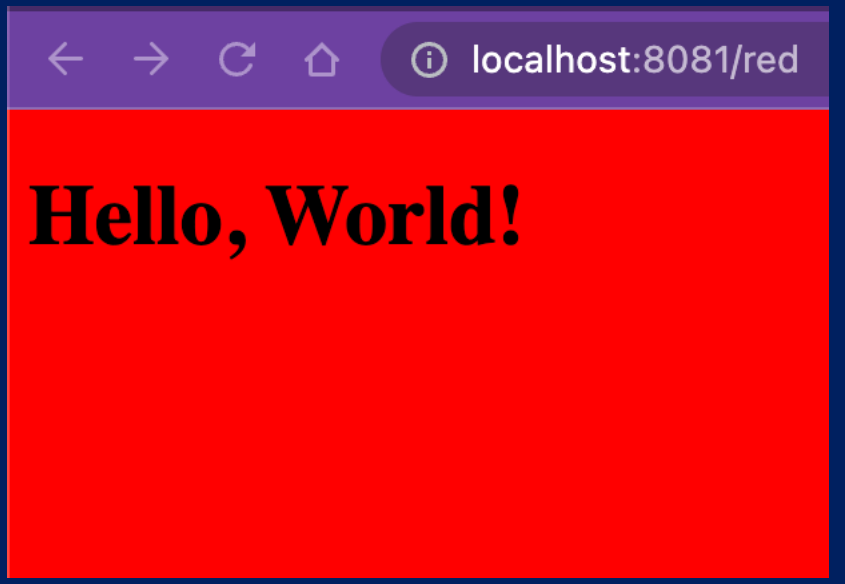
```html
<!DOCTYPE html>
<html>
<head>
    <title>Flask HTML Template</title>
    <style>
        body {
            background-color: {{ color }};
        }
    </style>
</head>
<body>
    <h1>Hello, World!</h1>
</body>
</html>
```

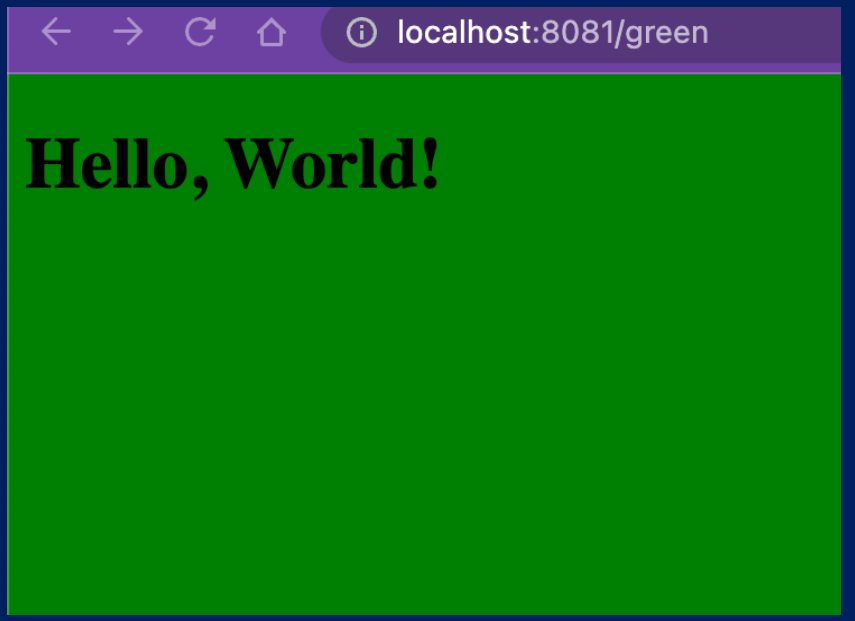



export APP_COLOR=blue; python app.py

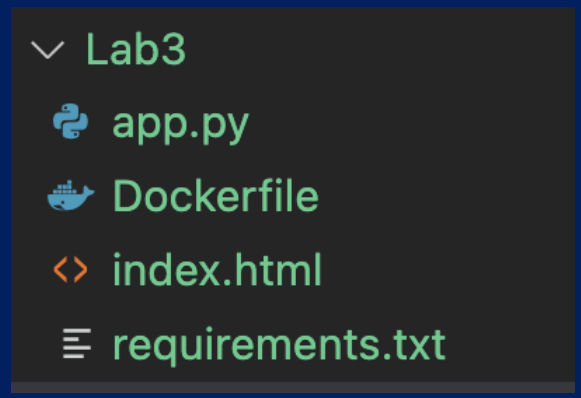
```python
import os
from flask import Flask, render_template

app = Flask(__name__,template_folder="")


@app.route('/')
def home():

    env_var_colour = os.environ['APP_COLOR']

    return render_template("index.html", color= env_var_colour)


@app.route('/<string:name>')
def template(name):
    return render_template("index.html", color=name)


if __name__ == '__main__':
    app.run(host="0.0.0.0",port="8081")
```

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



```
docker build -t flask-docker-app .
```




Lab3 > requirements.txt

```
1  flask
2
```

Lab3 > Dockerfile > ...

```
1  FROM python:3.8-alpine
2
3  WORKDIR /app
4  COPY requirements.txt requirements.txt
5  COPY . .
6
7  RUN pip install -r requirements.txt
8  EXPOSE 8081
9  CMD ["python", "app.py"]
```

## ENV Variables in Docker

```
docker run -p 8081:8081 -e APP_COLOR=red flask-docker-app
```

```
docker run -p 8081:8081 -e APP_COLOR=green flask-docker-app
```

# docker inspect flask-docker-app


```
[
    {
        "Id": "sha256:2b702818c29556 1f3ab6cd973da54cdafcc9d55df29ccb9a0206fa0457155 8d8",
        "RepoTags": [
            "flask-docker-app:latest"
        ],
        "RepoDigests": [],
        "Parent": "",
        "Comment": "buildkit.dockerfile.v0",
        "Created": "2023-03-10T08:38:08.766249342Z",
        "Container": "",
        "ContainerConfig": {
            "Hostname": "",
            "Domainname": "",
            "User": "",
    },
        "DockerVersion": "",
        "Author": "",
        "Config": {
            "Hostname": "",
            "Domainname": "",
            "User": "",
    "ExposedPorts": {
                "8081/tcp": {}
            },
            "Tty": false,
            "OpenStdin": false,
            "StdinOnce": false,
            "Env": [
                "PATH=/usr/local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin",
                "LANG=C.UTF-8",
                "GPG_KEY=E3FF2839C048B25C084DEBE9B26995E310250568",
                "PYTHON_VERSION=3.8.16",
                "PYTHON_PIP_VERSION=22.0.4",
                "PYTHON_SETUPTOOLS_VERSION=57.5.0",
                "PYTHON_GET_PIP_URL=https://github.com/pypa/get-pip/raw/d5cb0afaf23b8520f1bbcfed521017b4a95f5c01/public/get-pip.py",
                "PYTHON_GET_PIP_SHA256=394be00f13fa1b9aaa47e911bdb59a09c3b2986472130f30aa0bfaf7f3980637"
            ],
            "Cmd": [
                "python",
                "app.py"
            ],
            "ArgsEscaped": true,
            "Image": "",
            "Volumes": null,
            "WorkingDir": "/app",
            "Entrypoint": null,
            "OnBuild": null,
            "Labels": null
        },
        "Architecture": "arm64",
        "Variant": "v8",
        "Os": "linux",
        "Size": 637963 17,
        "VirtualSize": 637963 17,
        "GraphDriver": {
            "Data": {
    },
            "Name": "overlay2"
        },
        "RootFS": {
            "Type": "layers",
            "Layers": [
                "sha256:edf70074bd40c0b121636 7c29c18d1453b43cc69e5123268ba66dd45d86a9e8a8",
    ",
                "sha256:3f41e4baa3422c67da82b3e986cf3838faec79b4e802924e22ff8c3076892ace",
                "sha256:8beee4777 11ab384d9ba02beb52a2eaab3f25997dabdb92b64f0543c8a6edf07"
            ]
        },
        "Metadata": {
            "LastTagTime": "2023-03-10T08:38:08.836492133Z"
        }
    }
]
```

# LAB 3 :

### create directory

```
mkdir LAB3_Week9
cd    LAB3_Week9
```

### git clone branch dev

```
git clone -b dev https://github.com/Tuchsanai/DevTools.git
```

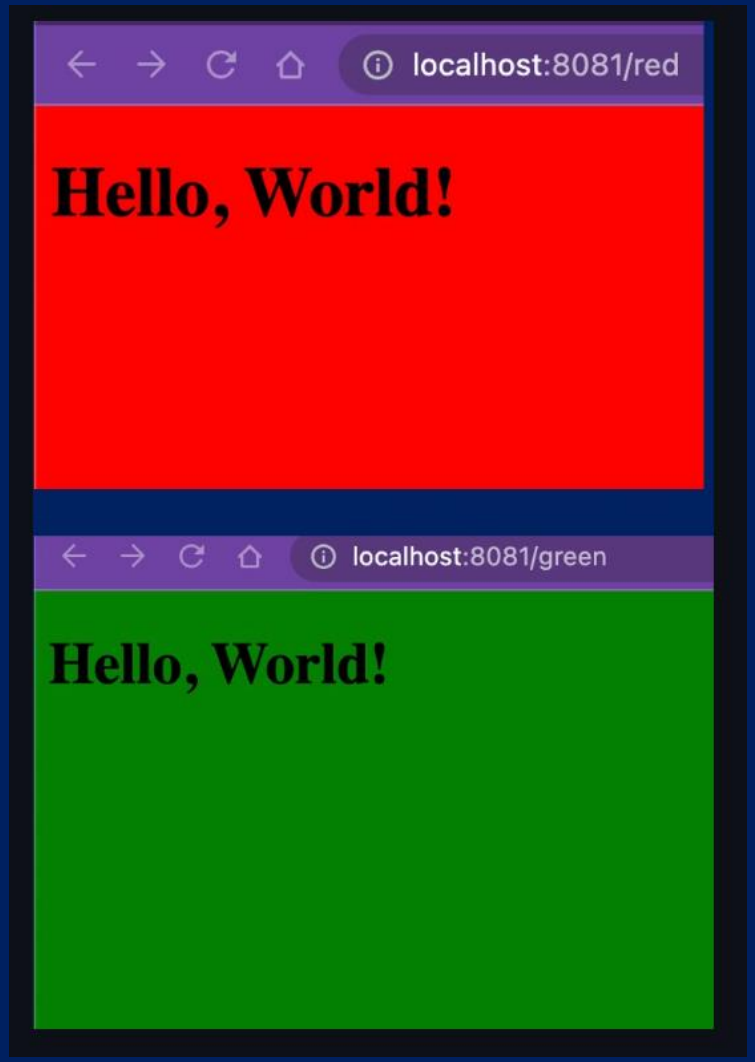
```
cd DevTools/02_Docker/Week09/Lab3
```

### build Docker image with docker build

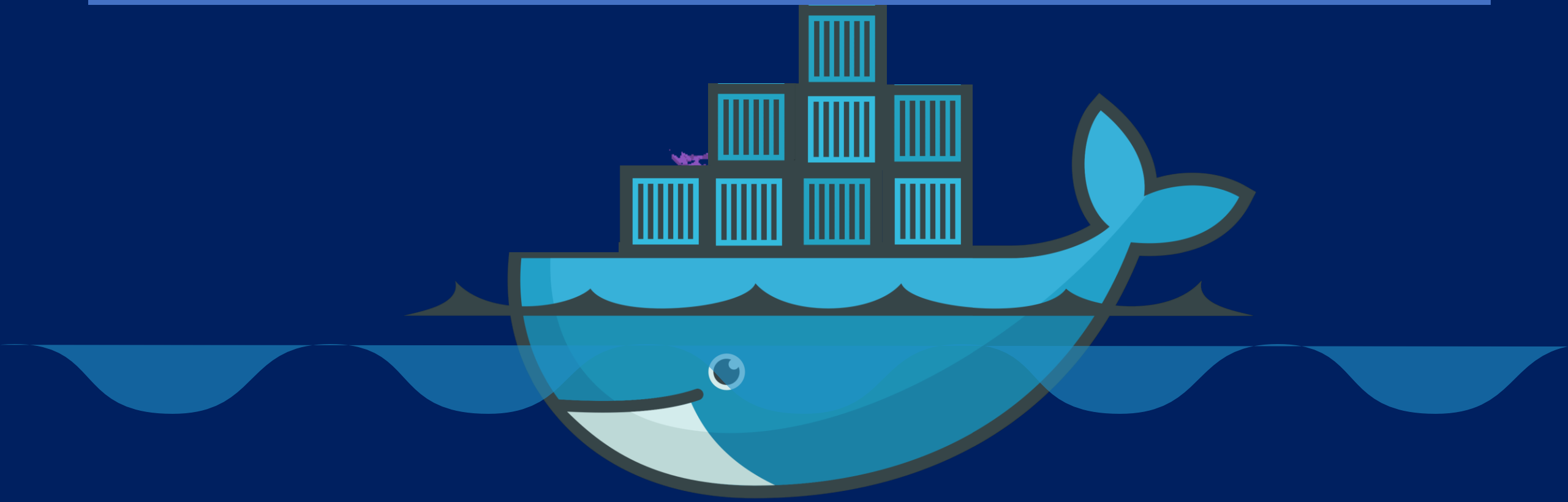
```
docker build -t flask-docker-app .
```

### docker run with -it option

```
docker run -p 8081:8081 -d --name container_red  -e APP_COLOR=red flask-docker-app
docker run -p 8085:8081 -d --name container_green  -e APP_COLOR=green flask-docker-app
```



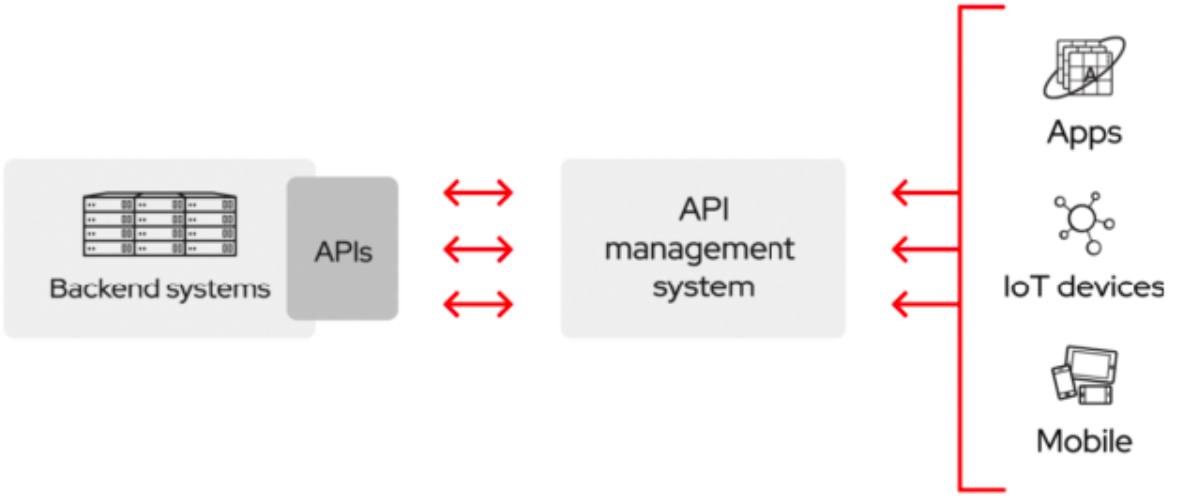
# Docker Play with FAST-API



# What is an API?

**API** (Application Programming Interface) creates an entry point for an application, through HTTP requests.

API: Application Abstraction + Simplification of Third Party Integration



## Status codes and HTTP methods

### HTTP methods

- **GET**: retrieve an existing resource (read only)
- **POST**: Create a new resource/send information
- **PUT**: Update an existing resource
- **PATCH**: partially update an existing resource
- **DELETE**: Delete a resource

### HTTP status code

- **2xx**: Successful operation
- **3xx**: Redirect
- **4xx**: client error
- **5xx**: Server Error

# Fast API Basics

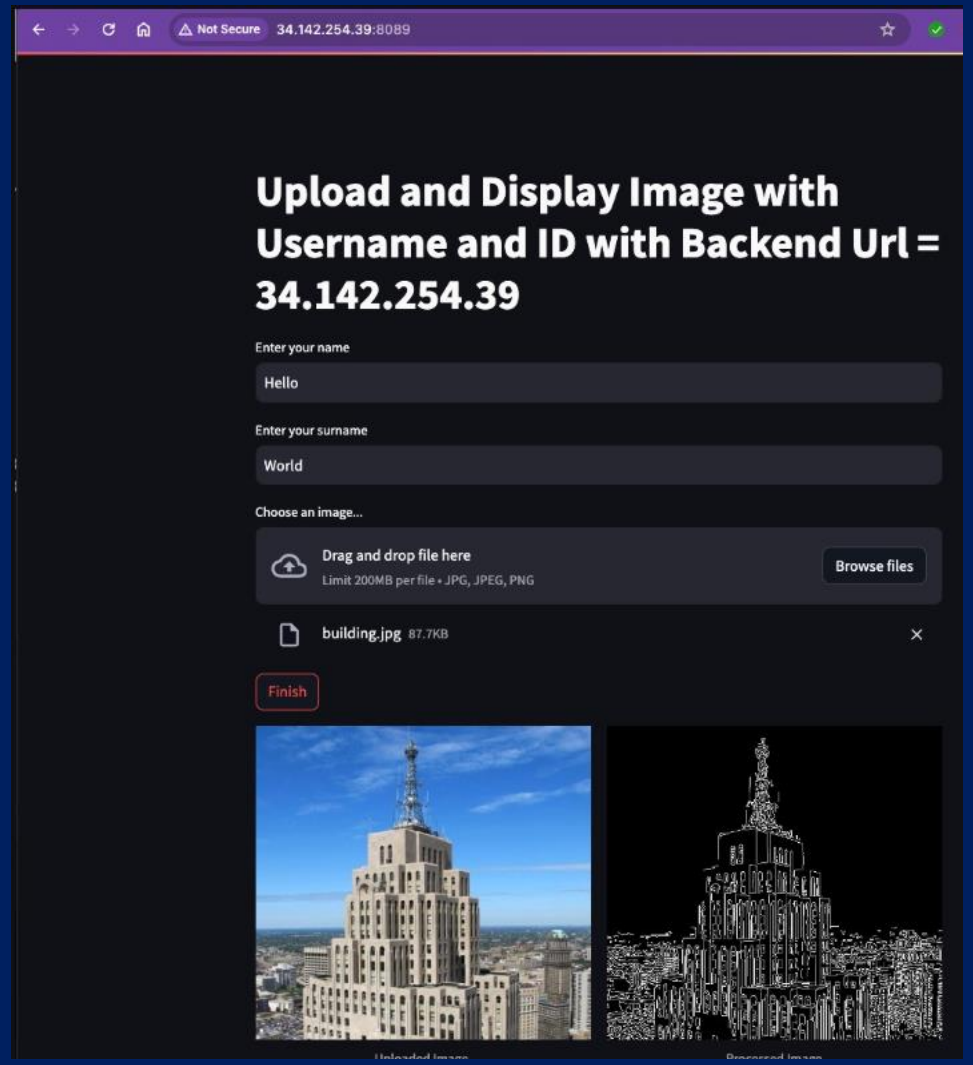
**Basic function**

```
import uvicorn
from fastapi import FastAPI

app = FastAPI()

@app.get("/")
def home():
    return {"Hello": "World"}

if __name__ == "__main__":
    uvicorn.run("hello_world_fastapi:app")
```

**Methods**

```
@app.get("/")
def home():
return {"Hello": "GET"}

@app.post("/")
def home_post():
return {"Hello": "POST"}
```

# Frontend




## Build and run the container as before:

### create directory

```
mkdir LAB4_Week9_Frontend
cd    LAB4_Week9_Frontend
```

### git clone branch dev

```
git clone -b dev https://github.com/Tuchsanai/DevTools.git
```

```
cd DevTools/02_Docker/Week09/Lab4/frontend
```

1. **Build the Docker Image:**

```
docker build -t streamlit-app_lab5  .
```

2. **Run the Docker Container:**

```
docker run -d --rm -p 8089:8501 -v  ./:/app -e BACKEND_URL=34.142.254.39  streamlit-app_lab5
```

# Backend

**create directory**

```
mkdir LAB4_Week9_Backend
cd    LAB4_Week9_Backend
```

**git clone branch dev**

```
git clone -b dev https://github.com/Tuchsanai/DevTools.git
```

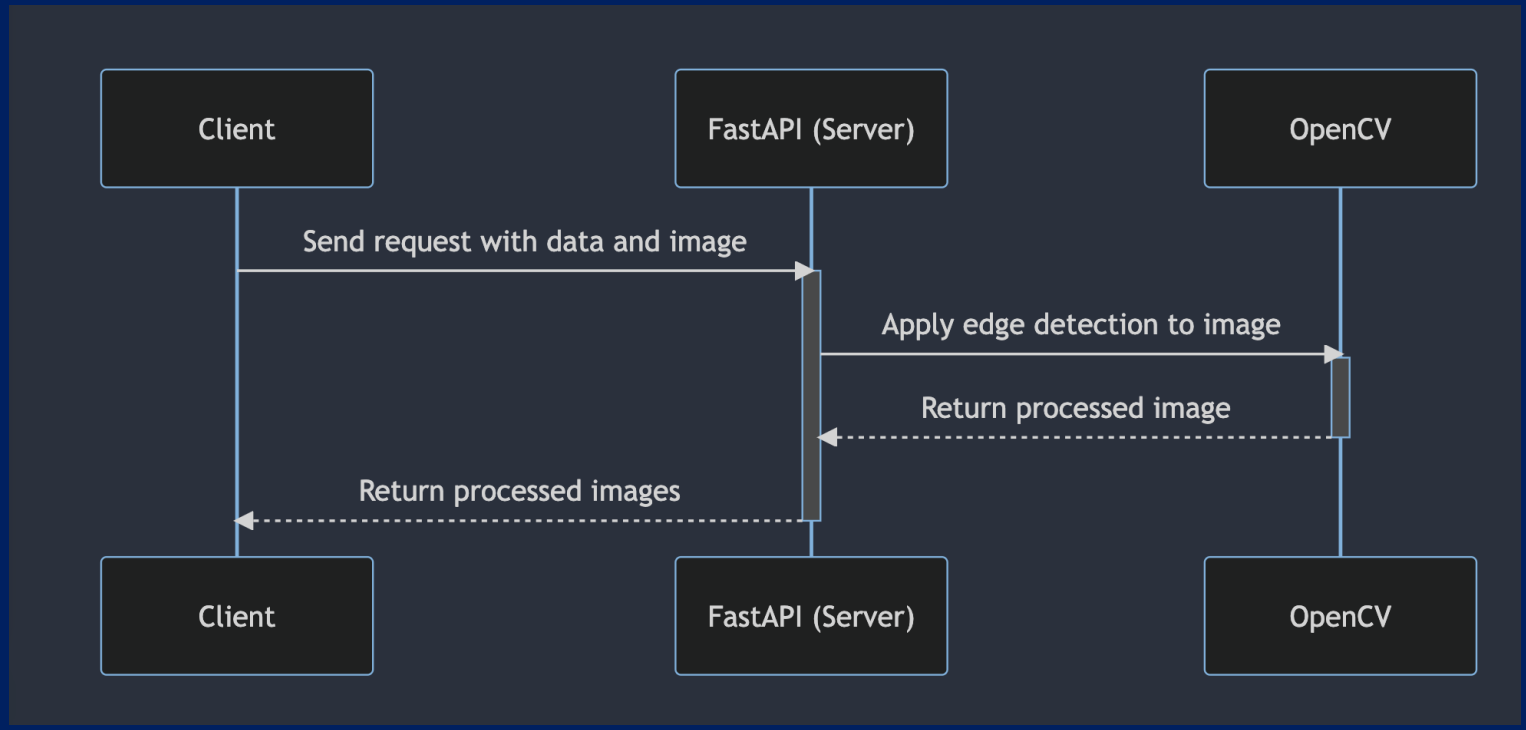
```
cd DevTools/02_Docker/Week09/Lab4/backend
```

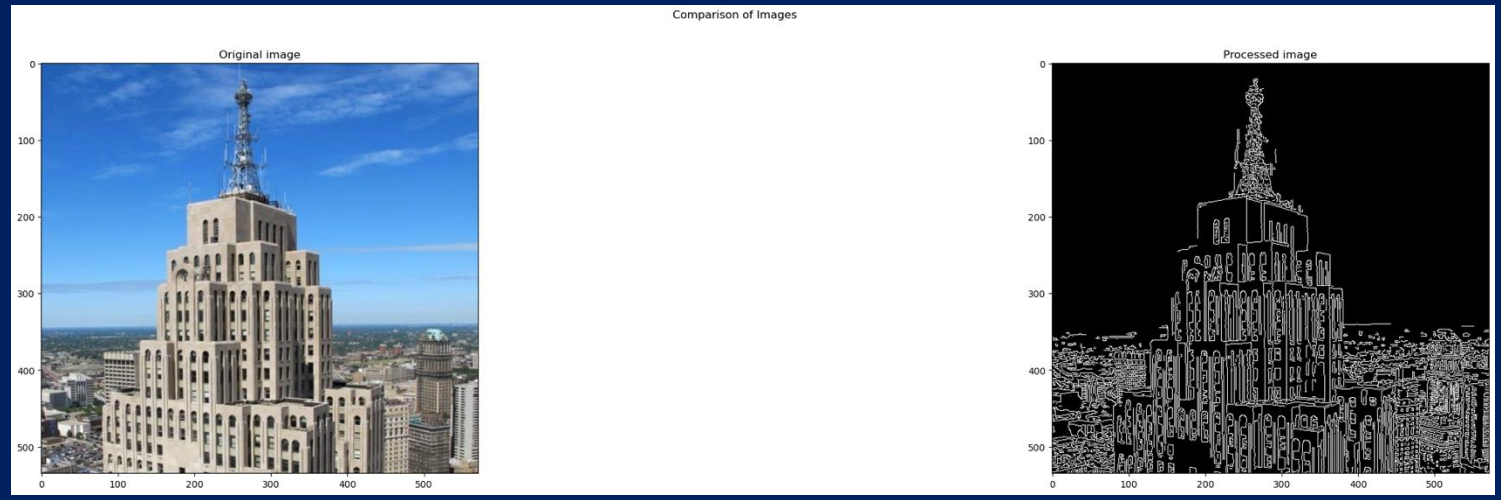
**build Docker image with docker build**

```
docker build -t fastapi-docker_lab4 .
```

**Run the Docker container by executing the following command:**

```
docker run -d -p 8088:80 fastapi-docker_lab4
```



# FAST-API

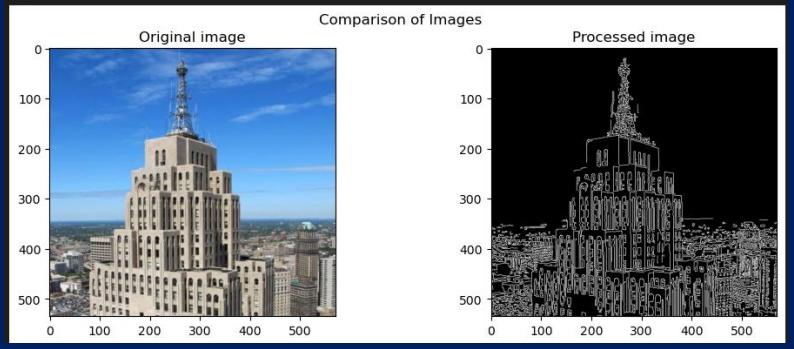
```python
import uvicorn
from fastapi import FastAPI
from pydantic import BaseModel
import numpy as np
import cv2
import base64

app = FastAPI()

class ImageRequest(BaseModel):
    image: str
    name: str
    surname: str
    numbers: List[int]


# encode image as base64 string
def encode_image(image):
    _, encoded_image = cv2.imencode(".jpg", image)
    return "" + base64.b64encode(encoded_image).decode()

# decode base64 string to image
def decode_image(image_string):
    encoded_data = image_string.split(',')[1]
    nparr = np.frombuffer(base64.b64decode(encoded_data), np.uint8)
    return cv2.imdecode(nparr, cv2.IMREAD_COLOR)

def apply_canny(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    edges = cv2.Canny(gray, 100, 200)
    return edges


@app.post("/process-image")
async def process_image(image_request: ImageRequest):
    image = decode_image(image_request.image)
    edges = apply_canny(image)
    processed_image = encode_image(edges)

    return {"name": image_request.name,
            "surname": image_request.surname,
            "numbers": image_request.numbers,
            "processed_image": processed_image}
```

# Client

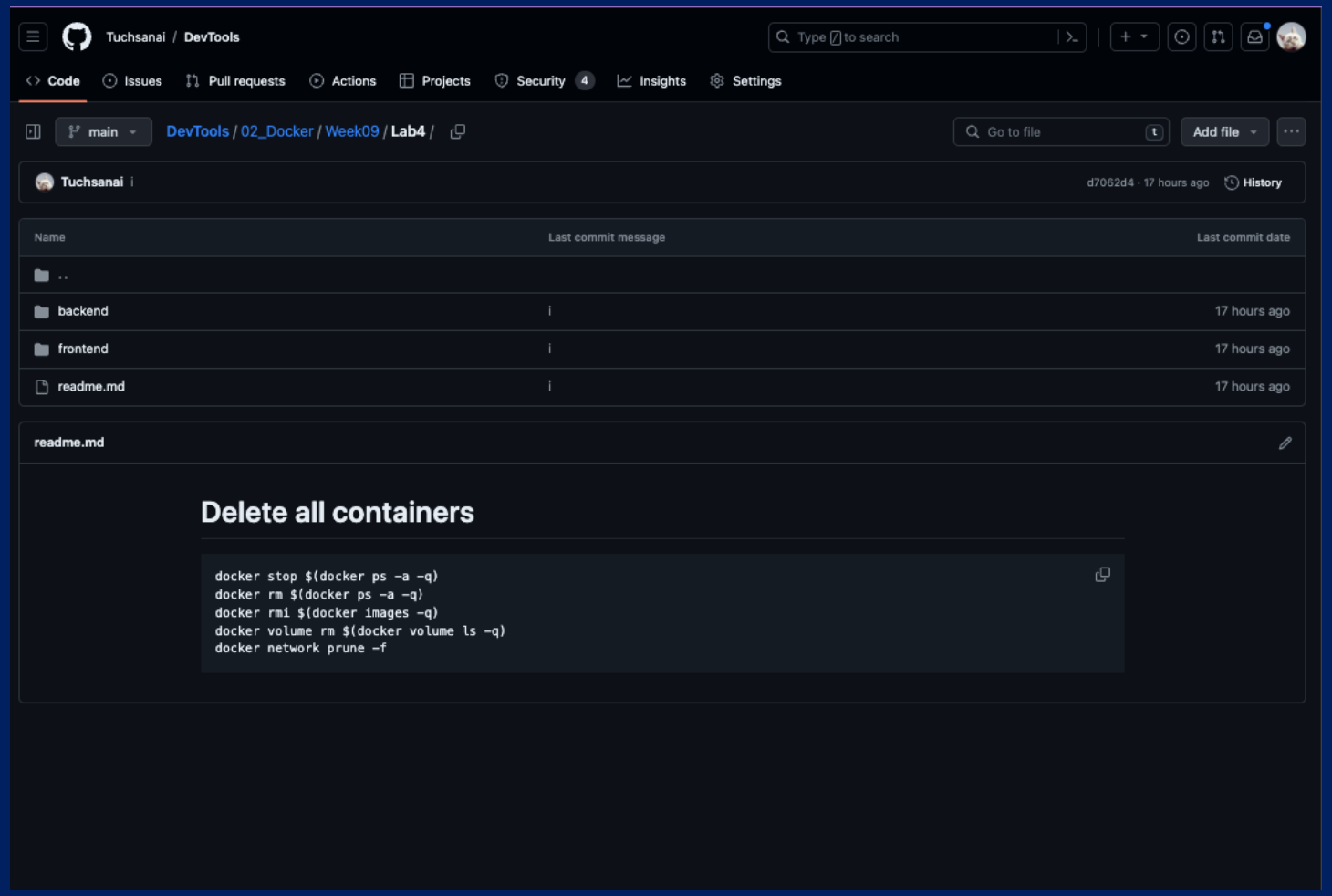
```python
import requests
import base64
import matplotlib.pyplot as plt
import numpy as np
import cv2


# encode image as base64 string
def encode_image(image):
    _, encoded_image = cv2.imencode(".jpg", image)
    return "" + base64.b64encode(encoded_image).decode()

# decode base64 string to image
def decode_image(image_string):
    encoded_data = image_string.split(',')[1]
    nparr = np.frombuffer(base64.b64decode(encoded_data), np.uint8)
    return cv2.imdecode(nparr, cv2.IMREAD_COLOR)



image_file = 'building.jpg'
url        = "http://localhost:8088"


# Load the image
image        = cv2.imread(image_file)
image        = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
image_string = encode_image(image)

payload = {
    "image": image_string,
    "name": "John",
    "surname": "Doe",
    "numbers": [1, 2, 3, 4, 5]
}

response = requests.post(f"{url}/process-image", json=payload)
data = json.loads(response.content)

processed_image_string = data["processed_image"]
processed_image        = decode_image(processed_image_string)
```

```python
# Create a figure and set the title
fig = plt.figure(figsize=(12, 4))
fig.suptitle('Comparison of Images')

# Add the first image to the left subplot
ax1 = fig.add_subplot(1, 2, 1)
ax1.imshow(image)
ax1.set_title('Original image')

# Add the second image to the right subplot
ax2 = fig.add_subplot(1, 2, 2)
ax2.imshow(processed_image)
ax2.set_title('Processed image')

# Show the plot
plt.show()
```

✓ 0.2s



# LAB 4 :


Tuchsanai / DevTools
Type / to search
Code
Issues
Pull requests
Actions
Projects
Security 4
Insights
Settings
main
DevTools / 02_Docker / Week09 / Lab4 /
Go to file
t
Add file
Tuchsanai
d7062d4 · 17 hours ago
History
Name
Last commit message
Last commit date
..
backend
i
17 hours ago
frontend
i
17 hours ago
readme.md
i
17 hours ago
readme.md
Delete all containers
docker stop $(docker ps -a -q)
docker rm $(docker ps -a -q)
docker rmi $(docker images -q)
docker volume rm $(docker volume ls -q)
docker network prune -f