LAB 5

CI/CD PIPELINE USING JENKINS, GITHUB AND DOCKER

Fullname: Nguyen Chi Hoang Minh

Student ID: B1809707

- Note: screenshots need to be clear and good-looking; submissions must be in PDF format.

1. Manually dockerize a Flask project

1.1. Deploy a Flask application

- Create a sample Flask application:

```
$mkdir cicd_tutorial ; cd cicd_tutorial
$nano flask_docker.py
```

flask_docker.py

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

@app.route('/')
def hello_world():
    return 'Hello FOSS'

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

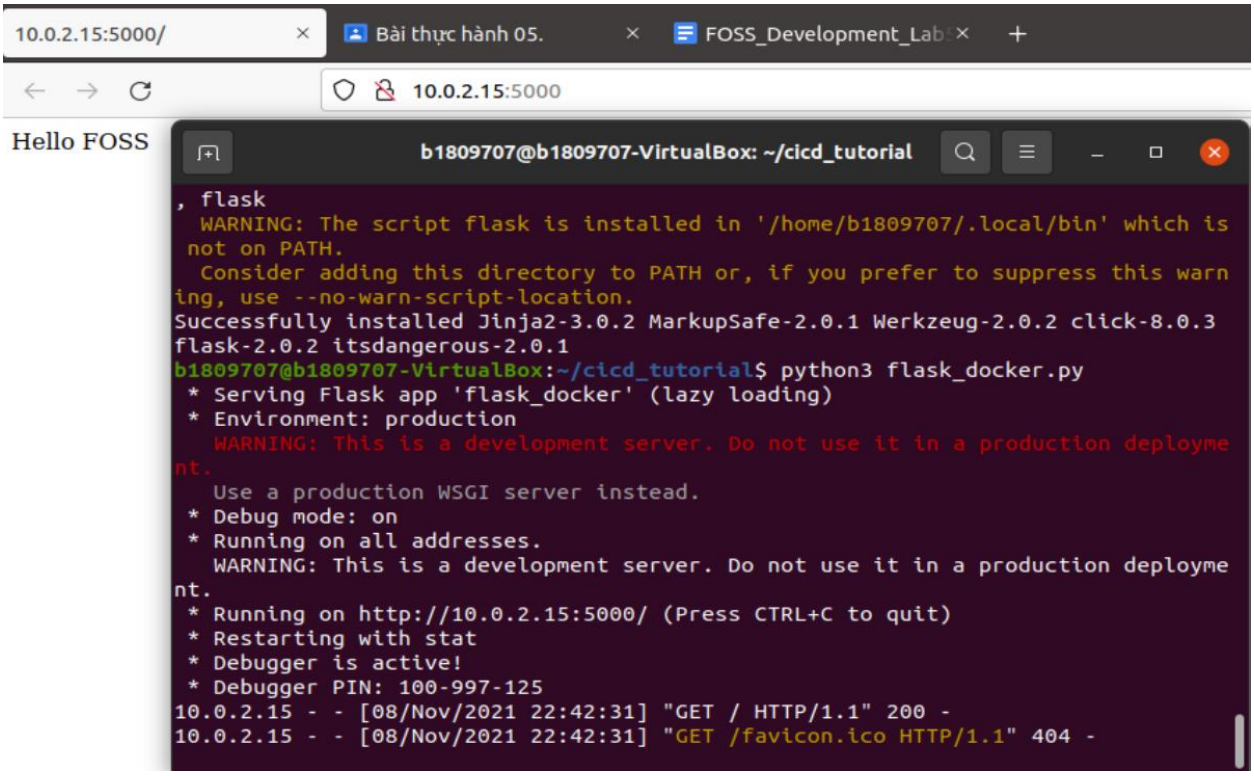
- Install pip (package installer for Python), and then the Flask framework

```
$sudo apt install python3-pip -y
$pip3 install flask
```

- We can test it out by running:

```
$python3 flask_docker.py
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 135-043-124
```

- Access the application from a browser (http://localhost:5000)



1.2. Dockerize a Flask application using Dockerfile

- Update the apt package index and install Docker

```
$sudo apt update
$sudo apt install docker.io -y
```

- Add current user to the docker group:

```
$sudo usermod -aG docker ${USER}
$su - ${USER}
```

- Check whether you can access and download images from Docker Hub

```
$docker run hello-world
```

The output will indicate that Docker is working correctly:

```
Hello from Docker!
      This message shows that your installation appears
to be working correctly.
```

- Create a requirements.txt file

```
$nano requirements.txt
```

requirements.txt

```
Flask==0.12.2
```

- Create a Dockerfile file

```
$nano Dockerfile
```

Dockerfile

```
FROM ubuntu:latest
MAINTAINER Tuan Thai "tuanthai@example.com"
RUN apt update -y
RUN apt install -y python3-pip python3-dev build-essential
ADD . /flask_app
WORKDIR /flask_app
RUN pip3 install -r requirements.txt
ENTRYPOINT ["python3"]
CMD ["flask_docker.py"]
```

- Create a Docker image whose name is "my-flask-image:latest", using the Dockerfile

```
$docker build -t my-flask-image:latest .
```

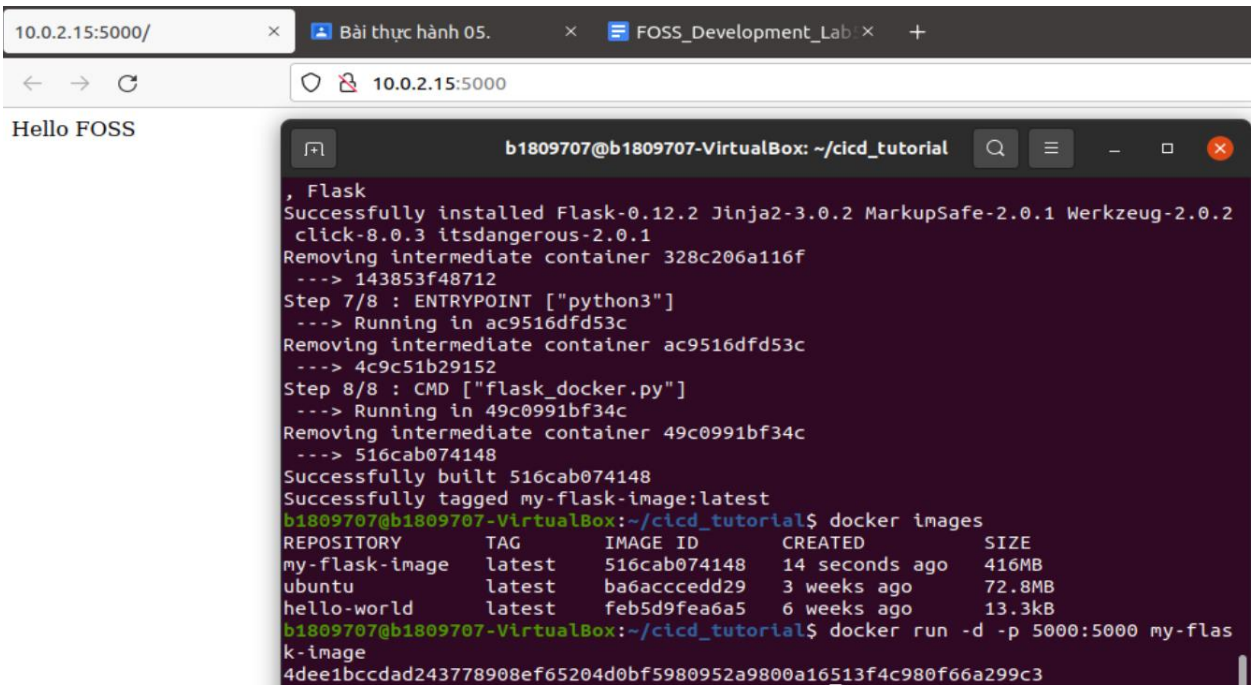
- Then see if your image is in Docker

```
$docker images
```

- Run your image

```
$docker run -d -p 5000:5000 my-flask-image
$docker ps
```

- Access the application from a browser (http://localhost:5000)



2. Automatically dockerize a Flask project using Jenkins

2.1. Push your code to a Github repository

- Create an account (or login) to GitHub at https://github.com
- Create a new repository, name it as "cicd_tutorial". Get the repository URL (for example: https://github.com/TuanThai/cicd_tutorial.git)
- Install and setup git on your computer (remember to set your name/email)

```
$sudo apt update ; sudo apt install git -y
$git config --global user.name "Firstname Lastname"
$git config --global user.email "example@.ctu.edu.vn"
```

- Initialize git, commit and push your flask project files to Github

```
$mv ~/cicd_tutorial
$git init
$git add .
$git commit -m "first commit"
$git remote add origin <your repository URL>
$git push -u origin master
```

```
b1809707@b1809707-VirtualBox:~/cicd_tutorial$ git push -u origin master
Username for 'https://github.com': 70g37h3r
Password for 'https://70g37h3r@github.com':
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 611 bytes | 305.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0)
To https://github.com/70G37H3R/cicd_tutorial
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

2.2. Install and configure Jenkins

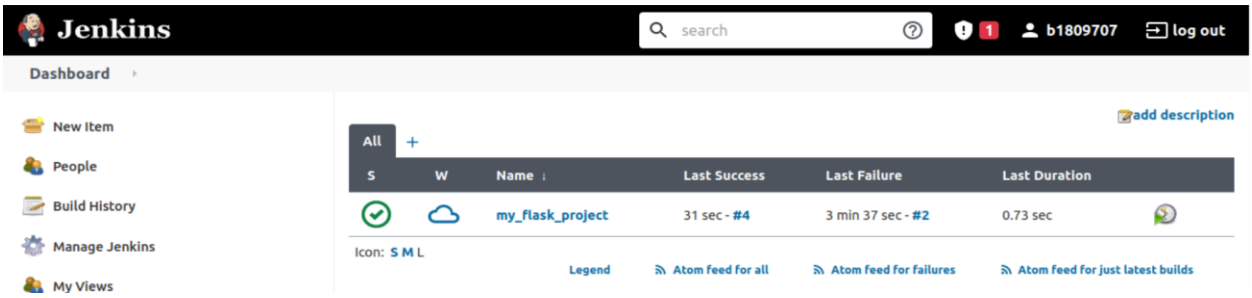
- Install Java and Jenkins

```
$sudo apt install openjdk-11-jdk -y
$wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -
$sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'
$sudo apt update ; sudo apt install jenkins -y
```

- Launch Jenkins

```
$sudo usermod -aG docker jenkins
$sudo systemctl restart jenkins.service
```

- Access Jenkins using a web browser (http://localhost:8080). Unlock Jenkins, install suggested plugins, create the first admin user.



2.3. Using Jenkins to automatically dockerize your application

- On Jenkins dashboard, cick "Create a new job", then choose "Freestyle project". Name your project as "my_flask_project"
- Under "Source Code Management" choose "Git", fill in your GitHub repository URL
- Under "Build Triggers" select "Build periodically", fill in "* * * * *" (build your project every minute)

```
* * * * *
```

- Under "Build" we will "Add build step", and select "Execute shell". Then fill in "docker build -t my-flask-image:latest ."

```
docker build -t my-flask-image:latest .
```

- Save your project. Then look at "Build history" to see that your project is built every minute.
- Then see if your image is in Docker

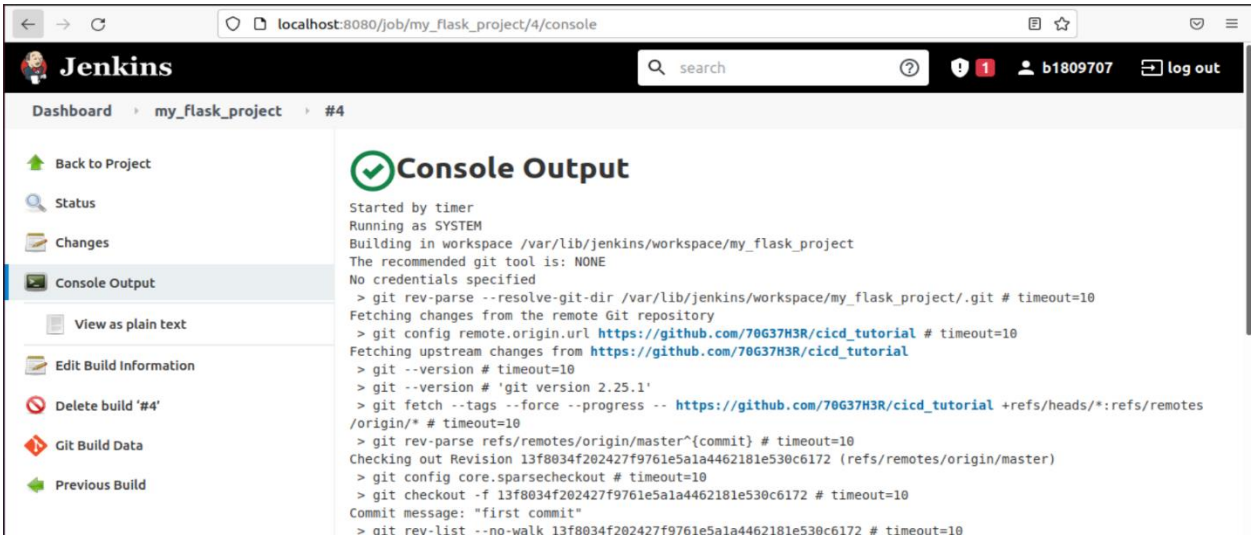
```
$docker images
```

- Modify your Flask application:

```
$nano flask_docker.py
```

```
from flask import Flask
app = Flask(__name__)
@app.route('/')

def hello_world():
    return 'Hello FOSS, Hello CI/CD using Jenkins'
if __name__ == '__main__':
    app.run(debug=True,host='0.0.0.0')
```




```
b1809707@b1809707-VirtualBox:~$ docker images
REPOSITORY        TAG       IMAGE ID       CREATED              SIZE
my-flask-image    latest    fd54a9f40f40   About a minute ago   416MB
<none>            <none>    516cab074148   About an hour ago    416MB
ubuntu            latest    ba6acccedd29   3 weeks ago          72.8MB
hello-world       latest    feb5d9fea6a5   6 weeks ago          13.3kB
```

- Commit and push your project files to GitHub

```
$git add .
$git commit -m "second commit"
$git push origin master
```

- Wait 1 minute, then run your image

```
$docker run -d -p 5000:5000 my-flask-image
$docker ps
```

- Access the application from a browser (http://localhost:5000)
- On your Jenkins project configure, under "Build Triggers", do not forget to deselect "Build periodically"

---END---