LAB 5

CI/CD PIPELINE USING JENKINS, GITHUB AND DOCKER

Fullname: Lam The Vinh

Student ID: B2206022

- 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

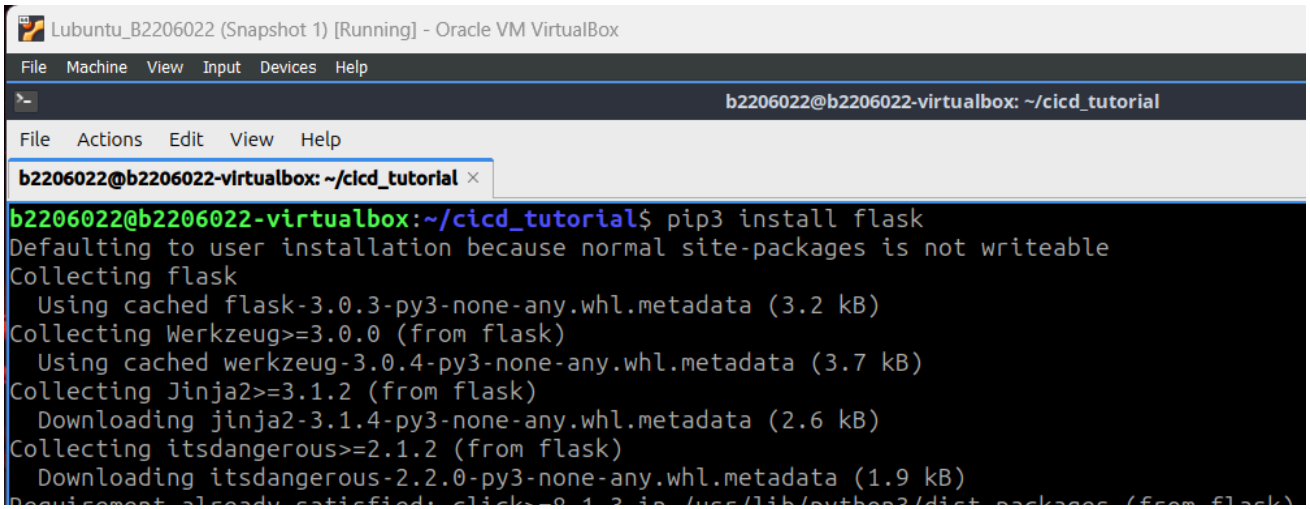
```
from flask import Flask
app = Flask(__name__)

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

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

```
Lubuntu_B2206022 [Running] - Oracle VM VirtualBox
File  Machine  View  Input  Devices  Help
b2206022@b2206022-virtualbox:
File  Actions  Edit  View  Help
b2206022@b2206022-virtualbox: ~/cicd_tutorial
b2206022@b2206022-virtualbox:~$ mkdir cicd_tutorial ; cd cicd_tutorial
b2206022@b2206022-virtualbox:~/cicd_tutorial$ nano flask_docker.py
b2206022@b2206022-virtualbox:~/cicd_tutorial$ tree
.
└── flask_docker.py

1 directory, 1 file
b2206022@b2206022-virtualbox:~/cicd_tutorial$ cat 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')
b2206022@b2206022-virtualbox:~/cicd_tutorial$
```

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

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

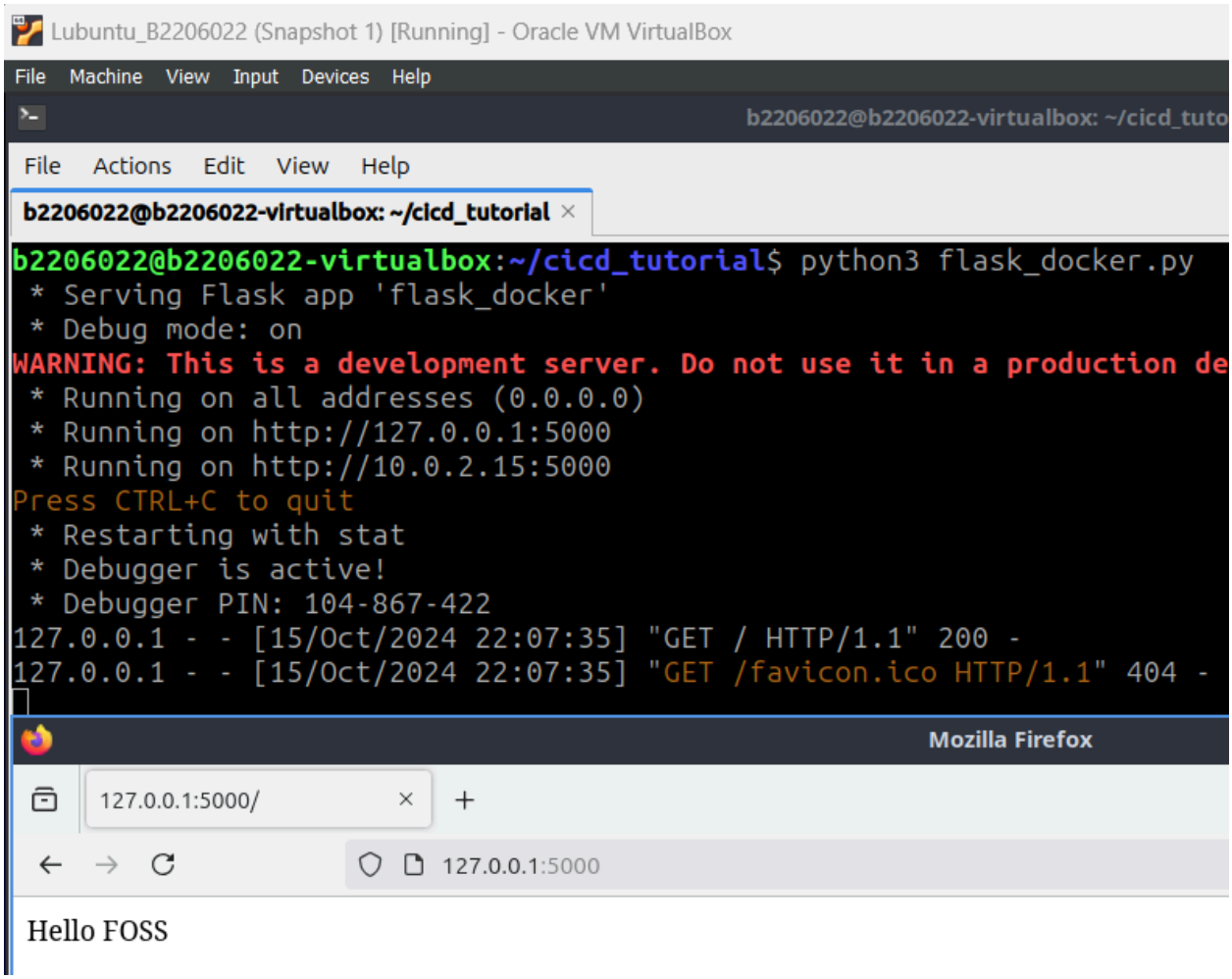
```
Lubuntu_B2206022 [Running] - Oracle VM VirtualBox
File  Machine  View  Input  Devices  Help
b2206022@b2206022-virtualbox: ~/cicd_tutorial
File  Actions  Edit  View  Help
b2206022@b2206022-virtualbox: ~/cicd_tutorial
b2206022@b2206022-virtualbox:~/cicd_tutorial$ sudo apt install python3-pip -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  javascript-common libexpat1-dev libjs-jquery libjs-sphinxdoc libjs-underscore l
  python3-setuptools python3-wheel python3.12-dev
Suggested packages:
  apache2 | lighttpd | httpd python-setuptools-doc
The following NEW packages will be installed:
  javascript-common libexpat1-dev libjs-jquery libjs-sphinxdoc libjs-underscore l
  python3-pip python3-setuptools python3-wheel python3.12-dev
0 upgraded, 12 newly installed, 0 to remove and 3 not upgraded.
```



- 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
```

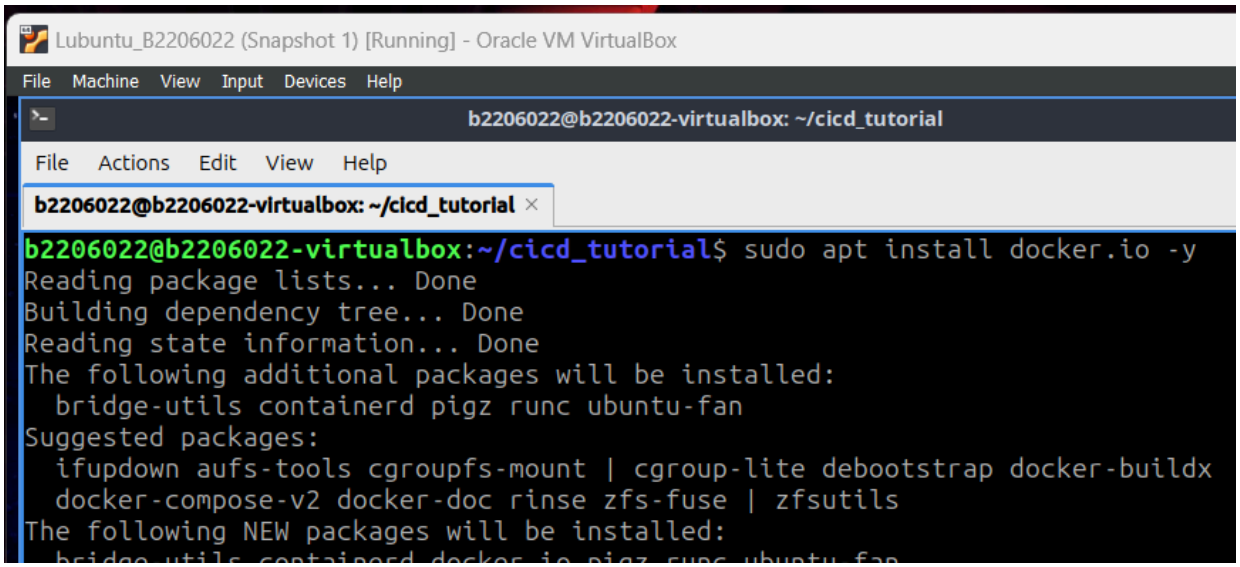
- Access the application from a browser (http://localhost:5000), (take a screenshot)



1.2. Dockerize a Flask application using Dockerfile

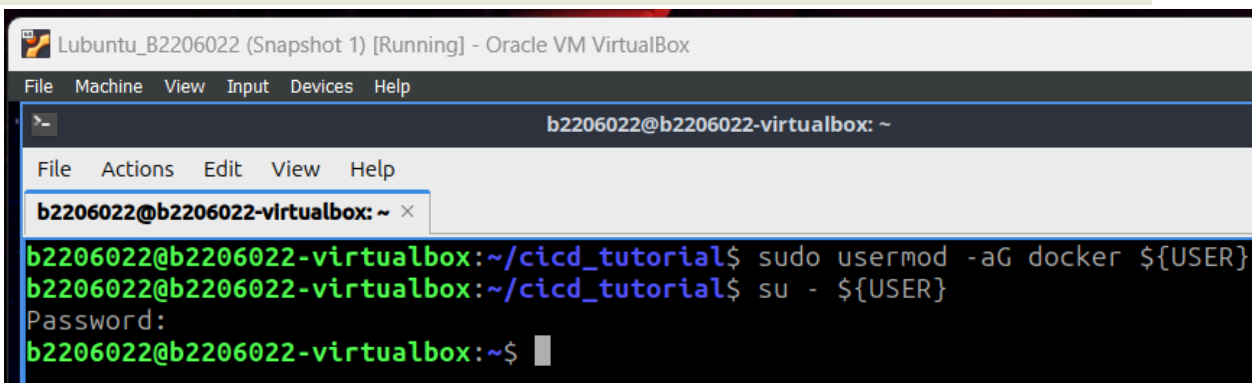
- Update the apt package index and install Docker

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



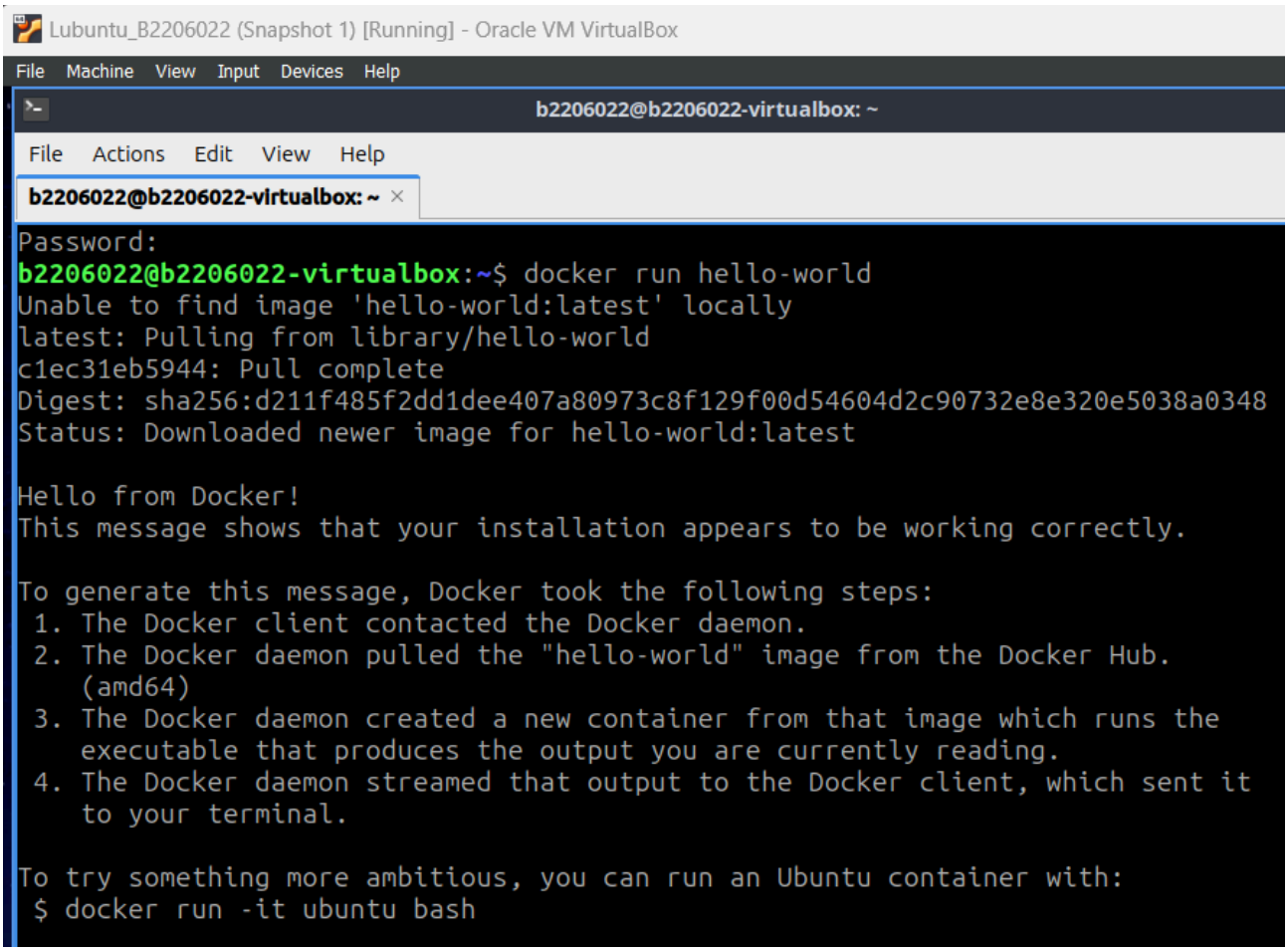
- Add current user to the docker group:

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



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

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



- Create a requirements.txt file

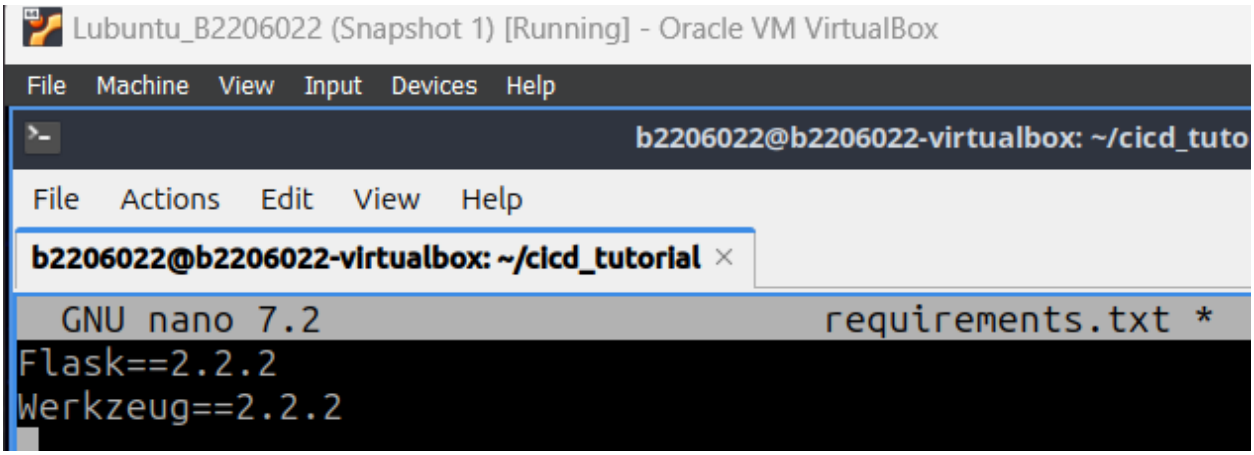
```
$nano requirements.txt
```

Requirements.txt

Creating an environment

- **venv** is a module in Python used to create virtual environments. A virtual environment is a self-contained directory that includes its own installation of Python and packages.
- Below configuration ensures Flask app will run inside the Docker container using Werkzeug's development server.

```
Flask==2.2.2
Werkzeug==2.2.2
```



- Create a Dockerfile file

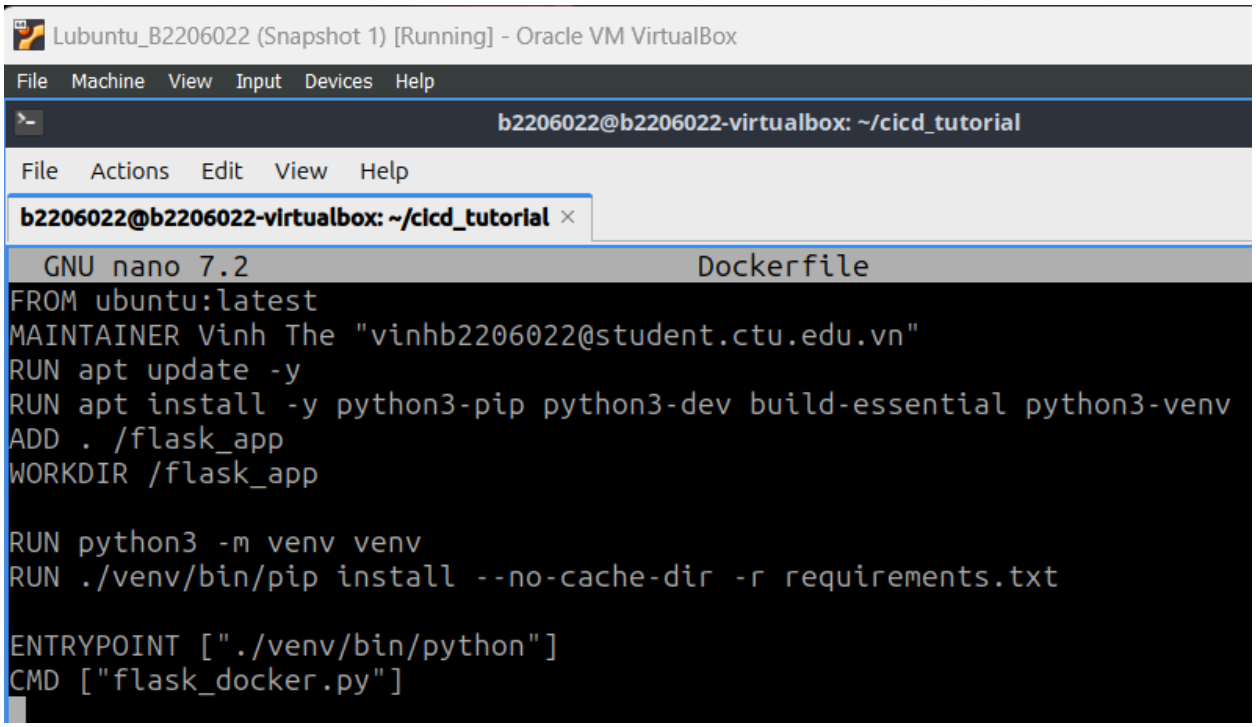
```
$nano Dockerfile
```

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"]
```

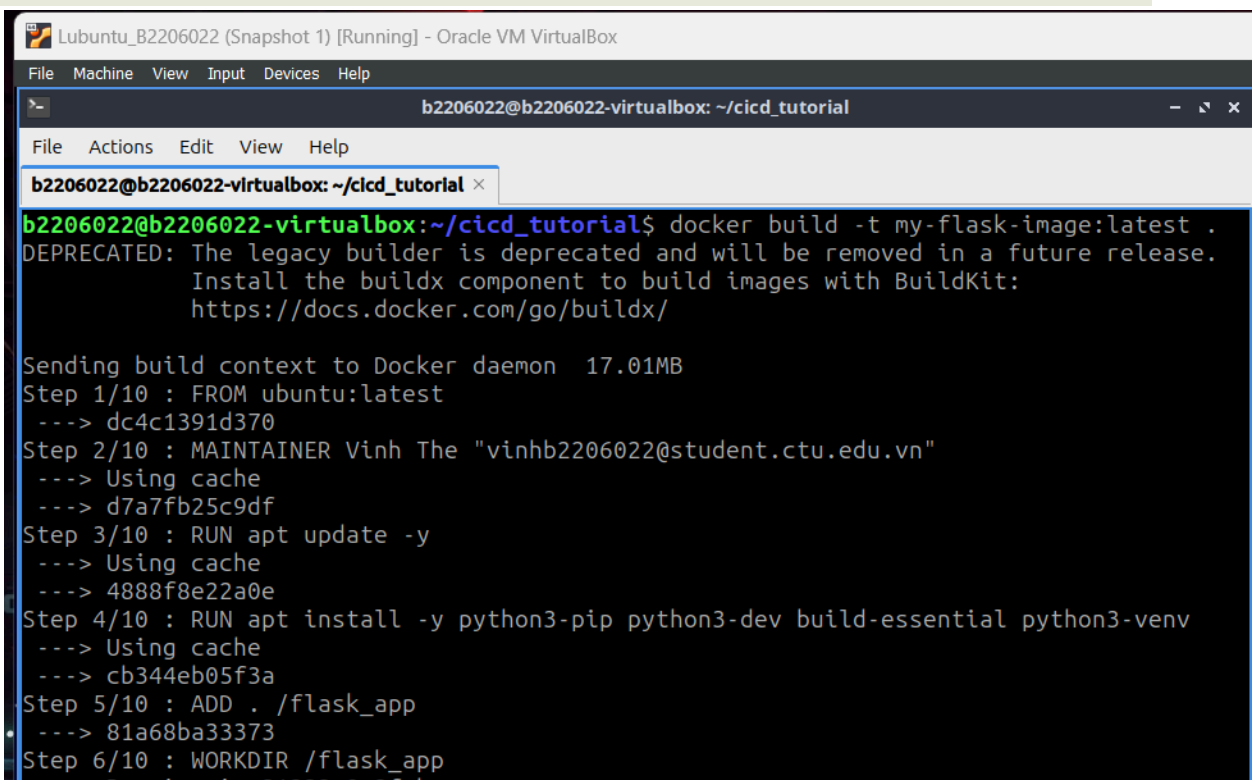
Modifying to suit with my computer

- Modify MAINTAINER NAME
  MAINTAINER Vinh The "vinhb2206022@student.ctu.edu.vn"
- Install python3-venv
- Create a virtual environment and install dependencies
  RUN python3 -m venv venv
  RUN ./venv/bin/pip install --no-cache-dir -r requirements.txt
- Change Entrypoint
  ENTRYPOINT ["./venv/bin/python"]



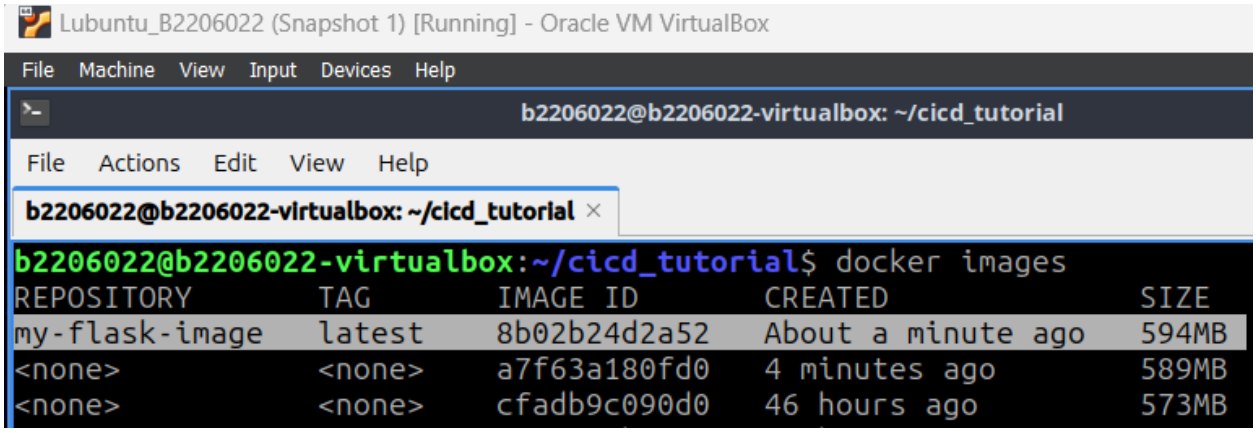
- Create a Docker image whose name is "my-flask-image:latest", using the Dockerfile

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

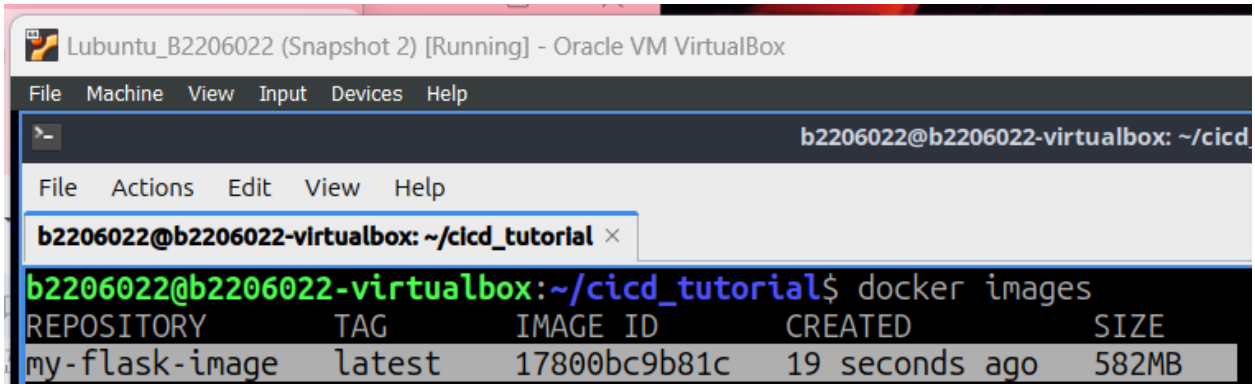


- Then see if your image is in Docker (take a screenshot)

```
$docker images
```

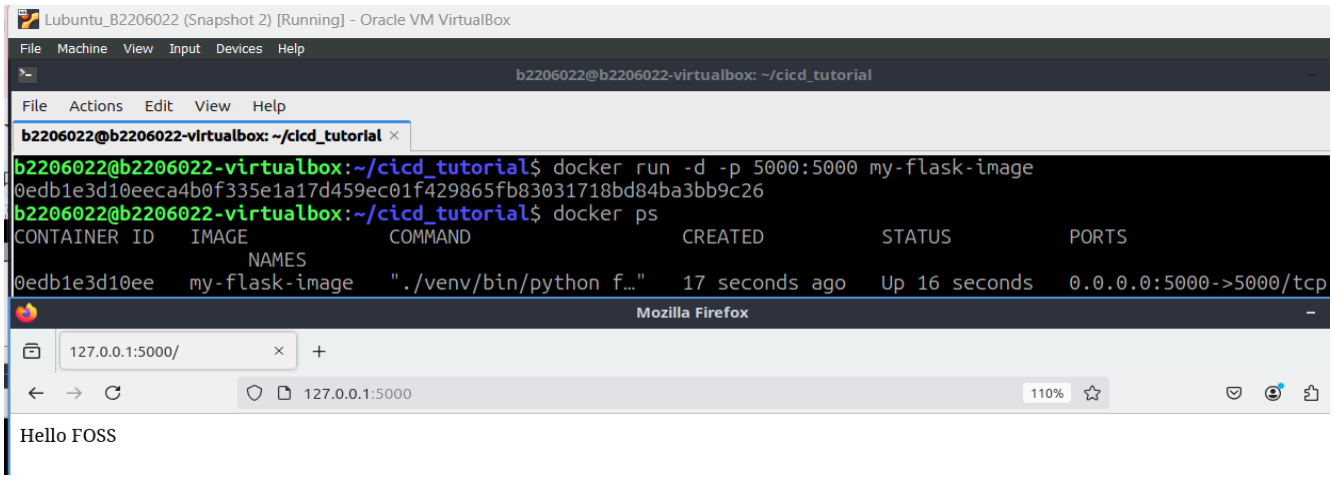


- Run your image (take a screenshot)

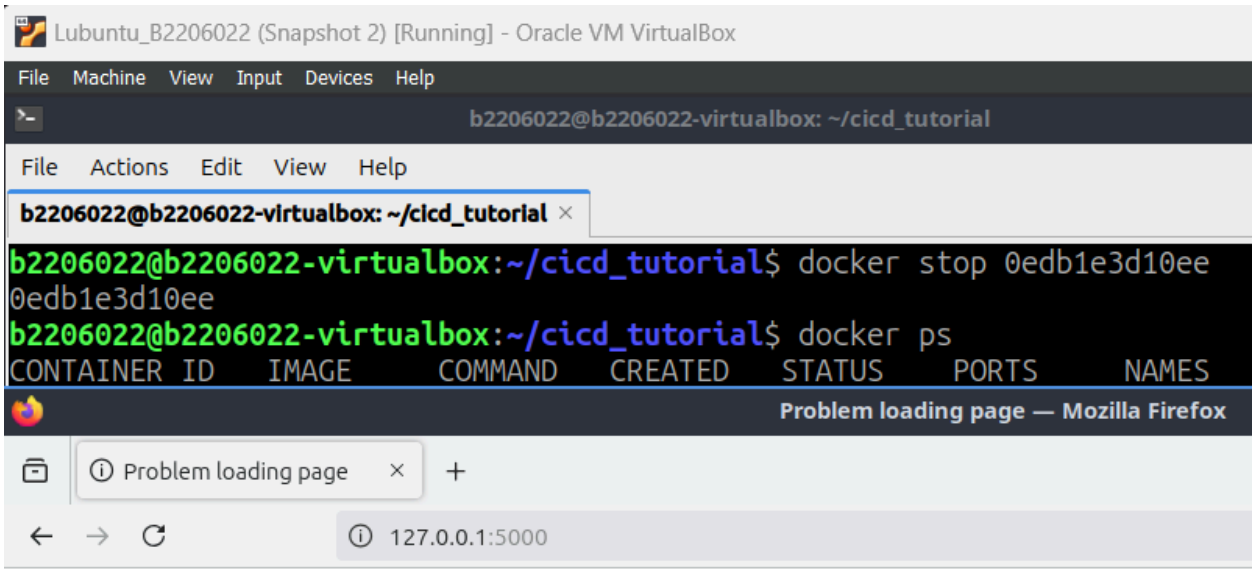



```
$docker run -d -p 5000:5000 my-flask-image
$docker ps
```

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



Stop docker by the way run: **docker stop <CONTAINER IID>**



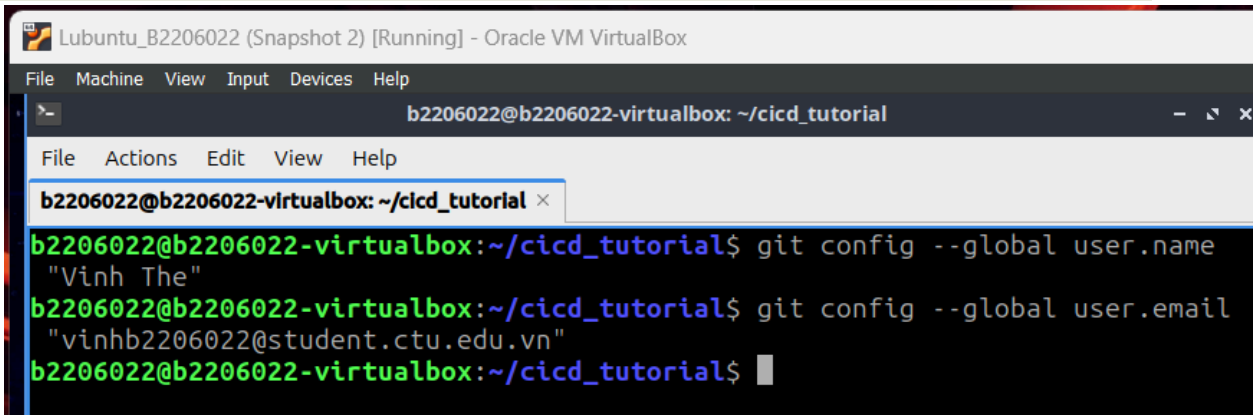
2. Automatically dockerize a Flask project using Jenkins

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)

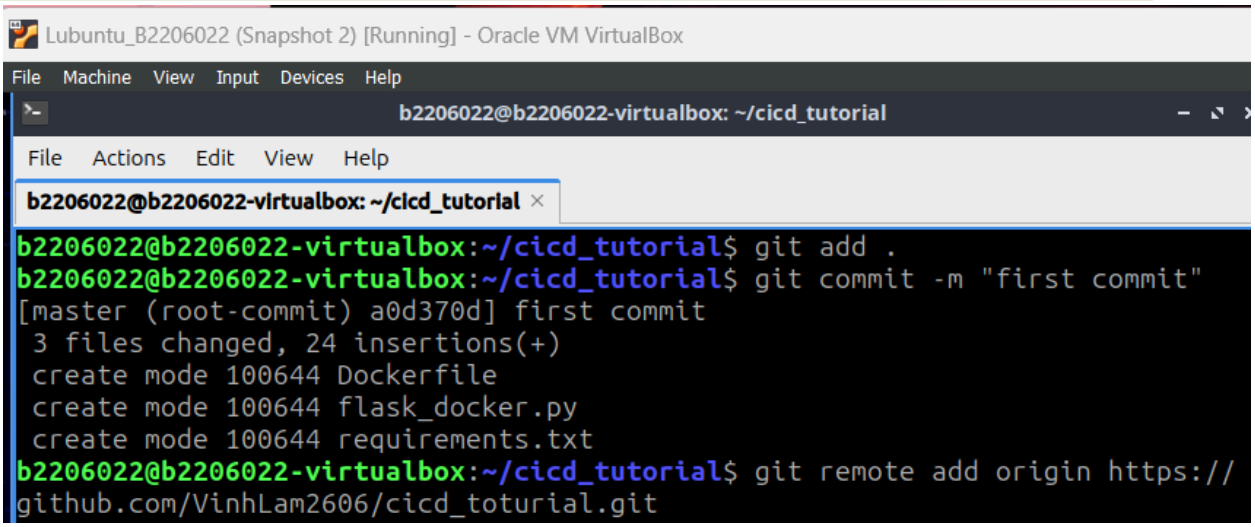
```
$sudo apt update ; sudo apt install git -y
$git config --global user.name "Firstname Lastname"
```

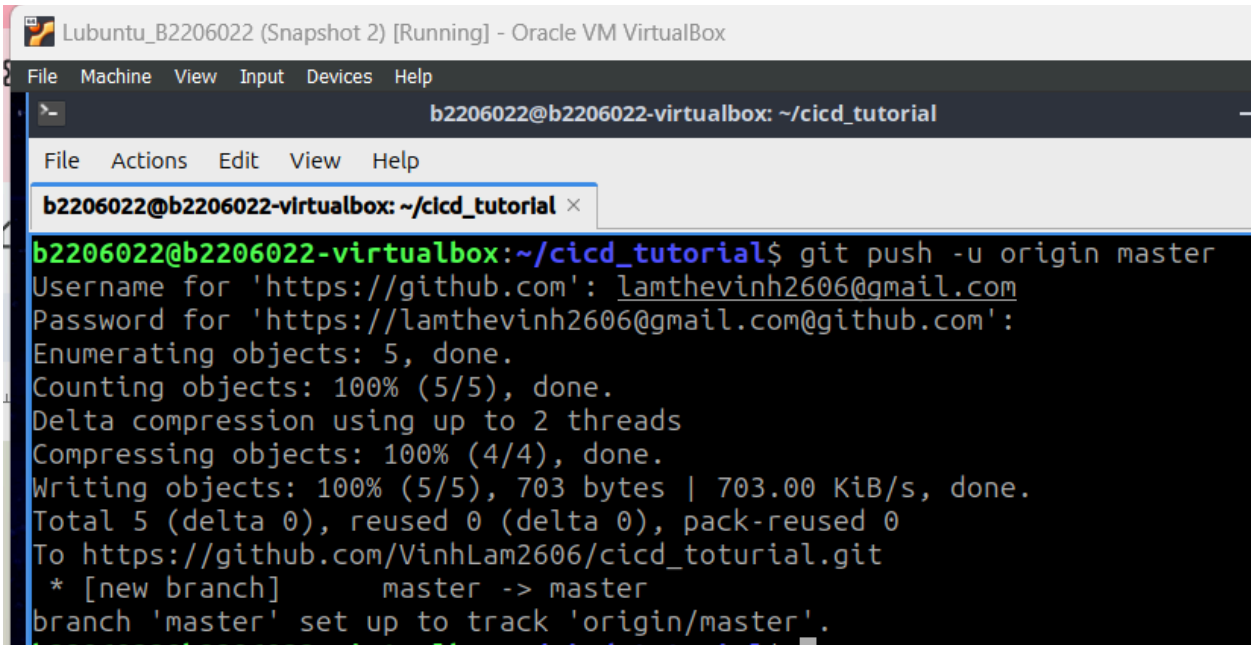
```
$git config --global user.email "example@.ctu.edu.vn"
```

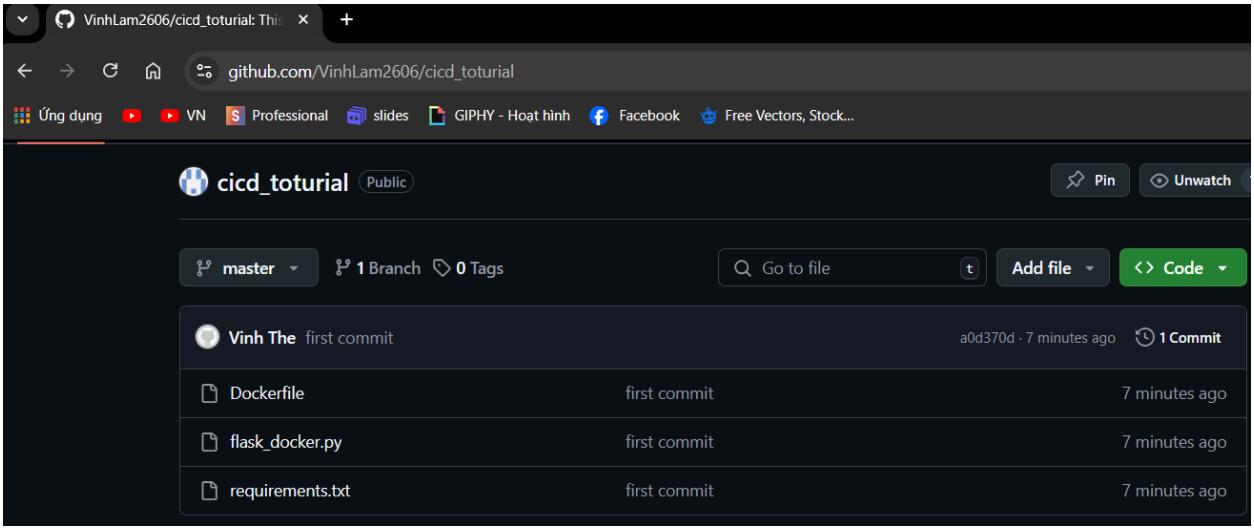


- Initialize git, commit and push your flask project files to Github

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



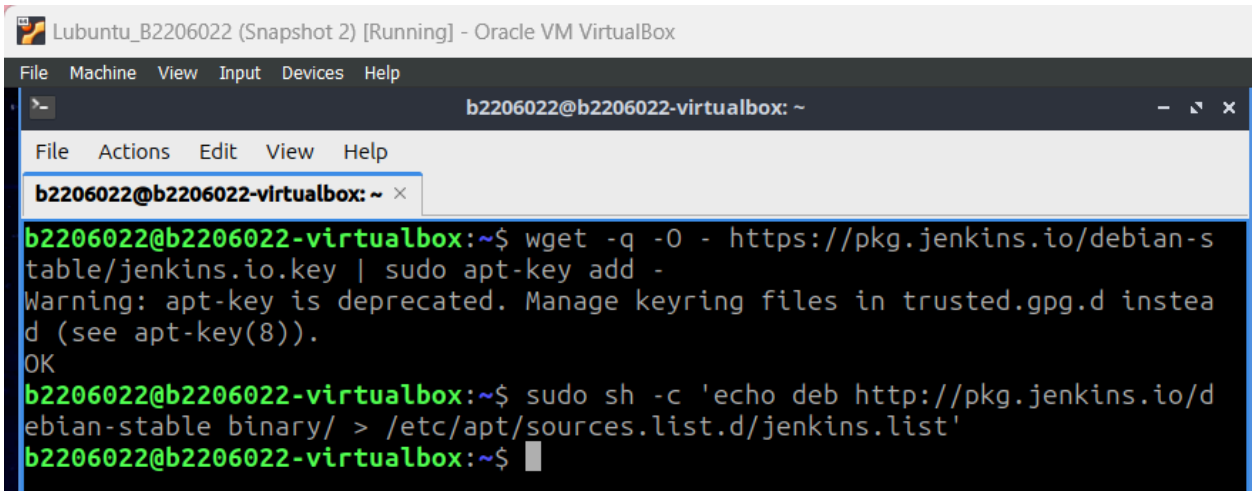


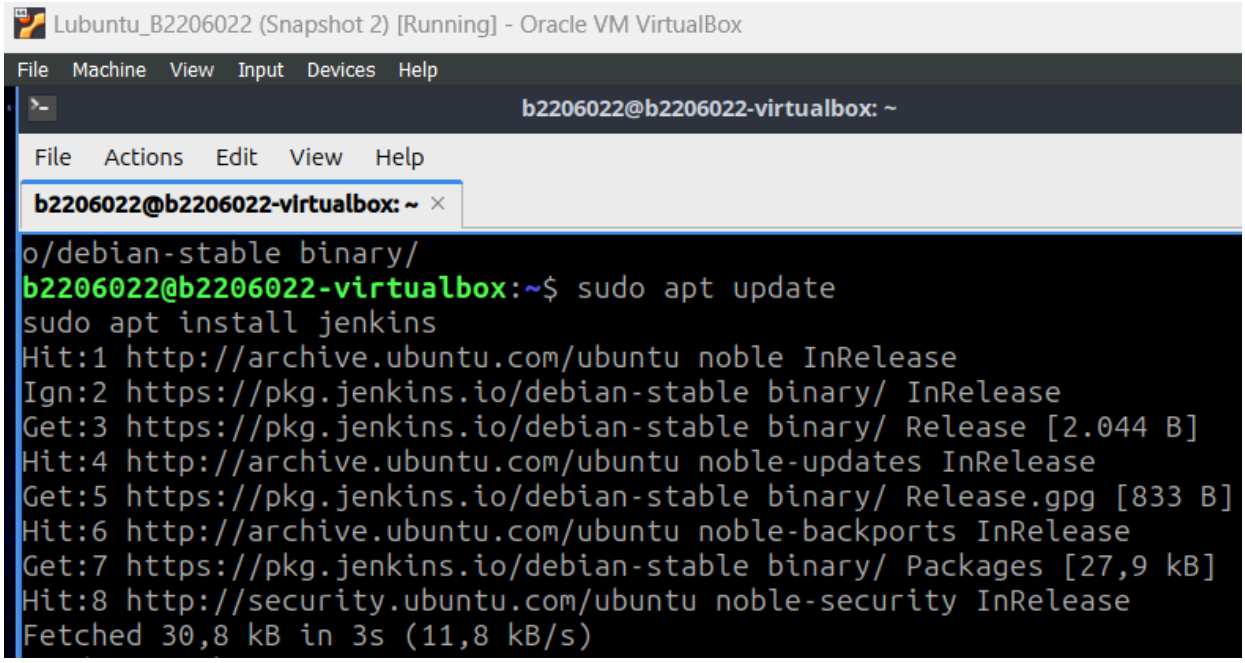


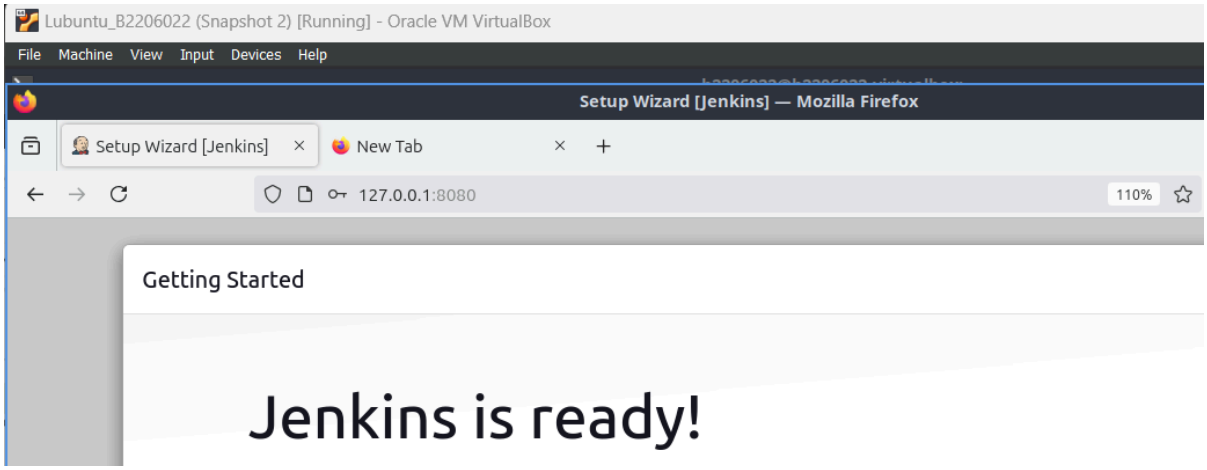
2.2. Install and configure Jenkins

- 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
```


Lubuntu_B2206022 (Snapshot 2) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
b2206022@b2206022-virtualbox: ~
File Actions Edit View Help
b2206022@b2206022-virtualbox: ~
b2206022@b2206022-virtualbox:~$ wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
OK
b2206022@b2206022-virtualbox:~$ sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'
b2206022@b2206022-virtualbox:~$


Lubuntu_B2206022 (Snapshot 2) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
b2206022@b2206022-virtualbox: ~
File Actions Edit View Help
b2206022@b2206022-virtualbox: ~
o/debian-stable binary/
b2206022@b2206022-virtualbox:~$ sudo apt update
sudo apt install jenkins
Hit:1 http://archive.ubuntu.com/ubuntu noble InRelease
Ign:2 https://pkg.jenkins.io/debian-stable binary/ InRelease
Get:3 https://pkg.jenkins.io/debian-stable binary/ Release [2.044 B]
Hit:4 http://archive.ubuntu.com/ubuntu noble-updates InRelease
Get:5 https://pkg.jenkins.io/debian-stable binary/ Release.gpg [833 B]
Hit:6 http://archive.ubuntu.com/ubuntu noble-backports InRelease
Get:7 https://pkg.jenkins.io/debian-stable binary/ Packages [27,9 kB]
Hit:8 http://security.ubuntu.com/ubuntu noble-security InRelease
Fetched 30,8 kB in 3s (11,8 kB/s)


Lubuntu_B2206022 (Snapshot 2) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Setup Wizard [Jenkins] — Mozilla Firefox
Setup Wizard [Jenkins]
New Tab
127.0.0.1:8080
110%
Getting Started
Jenkins is ready!

- Launch Jenkins

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

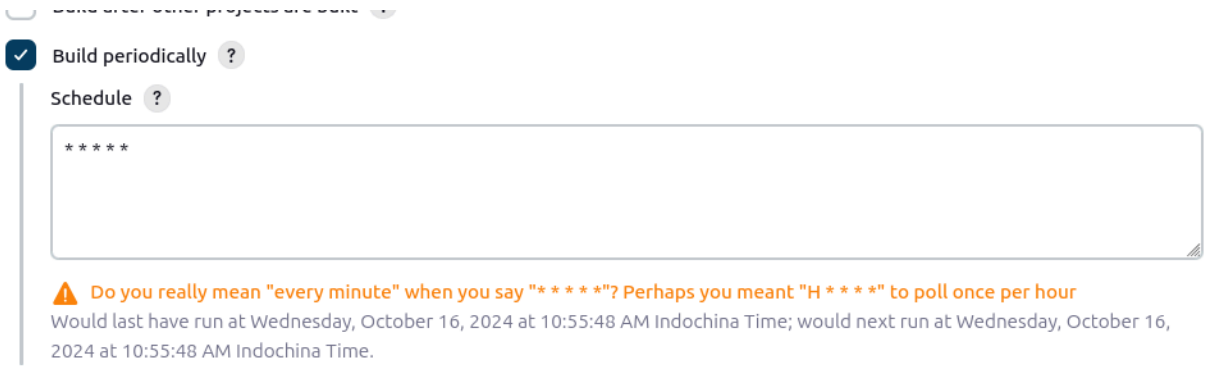
- Access Jenkins using a web browser (http://localhost:8080). Unlock Jenkins, install suggested plugins, create the first admin user.

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)

```
* * * * *
```

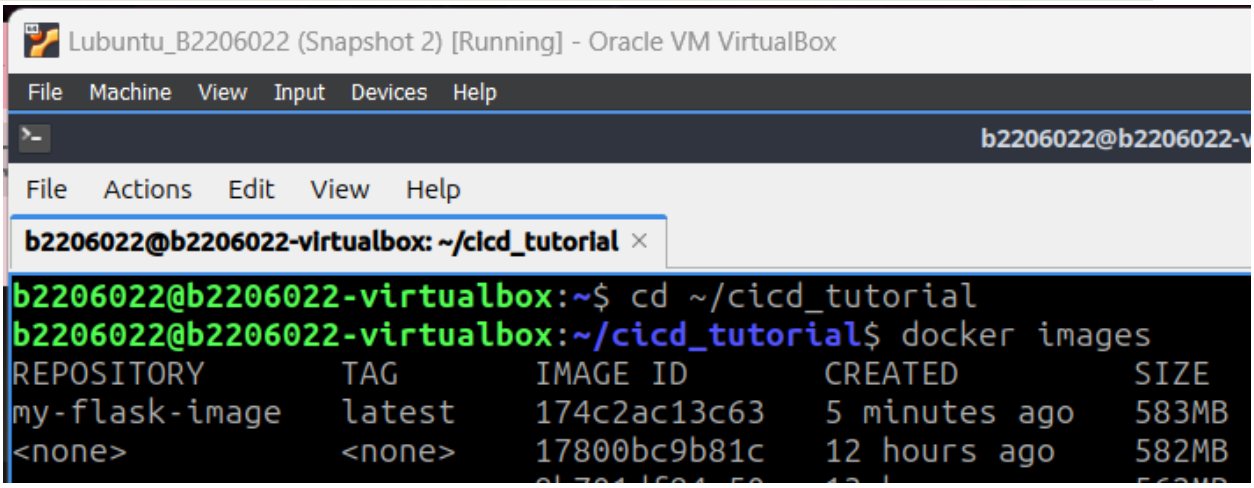


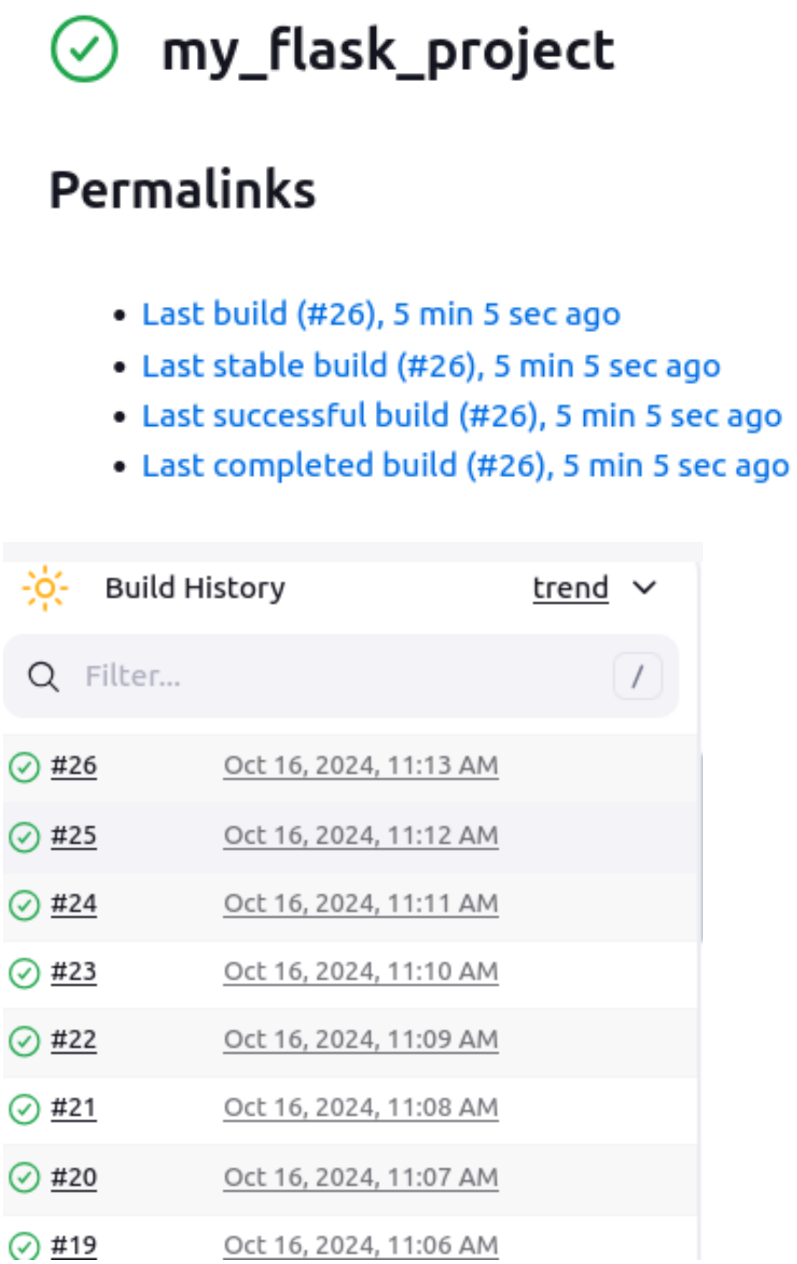
- Under "Build" we will "Add build step", and select "Execute shell". Then fill in "docker build -t my-flask-image:latest ."

```
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 (take a screenshot)

```
$docker images
```



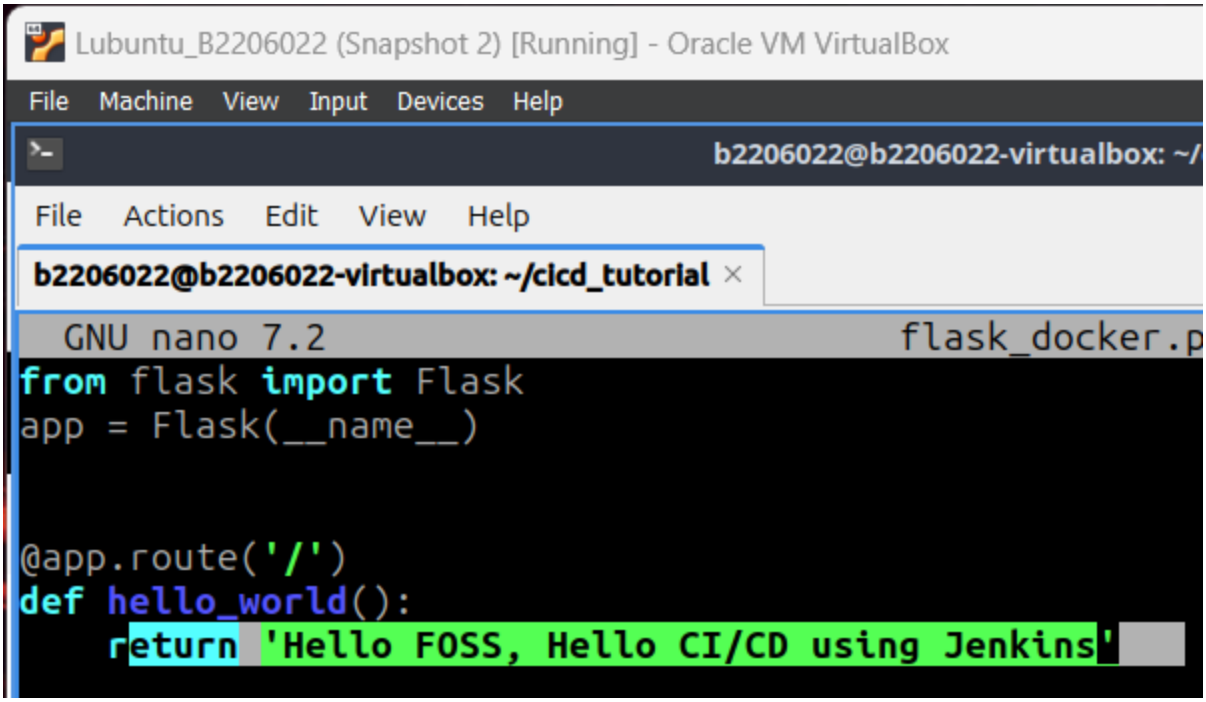


- Modify your Flask application:

```
$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')
```



- Commit and push your project files to GitHub

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

Lubuntu_B2206022 (Snapshot 2) [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

b2206022@b2206022-virtualbox: ~/cicd_tutorial

File Actions Edit View Help

b2206022@b2206022-virtualbox: ~/cicd_tutorial

```
b2206022@b2206022-virtualbox:~/cicd_tutorial$ git add .
b2206022@b2206022-virtualbox:~/cicd_tutorial$ git commit -m "second commit"
[master b7af34e] second commit
 1 file changed, 1 insertion(+), 1 deletion(-)
b2206022@b2206022-virtualbox:~/cicd_tutorial$ git push origin master
Username for 'https://github.com': lamthevinh2606@gmail.com
Password for 'https://lamthevinh2606@gmail.com@github.com':
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 2 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 317 bytes | 317.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/VinhLam2606/cicd_toturial.git
   a0d370d..b7af34e  master -> master
b2206022@b2206022-virtualbox:~/cicd_tutorial$
```

Vinh The second commit

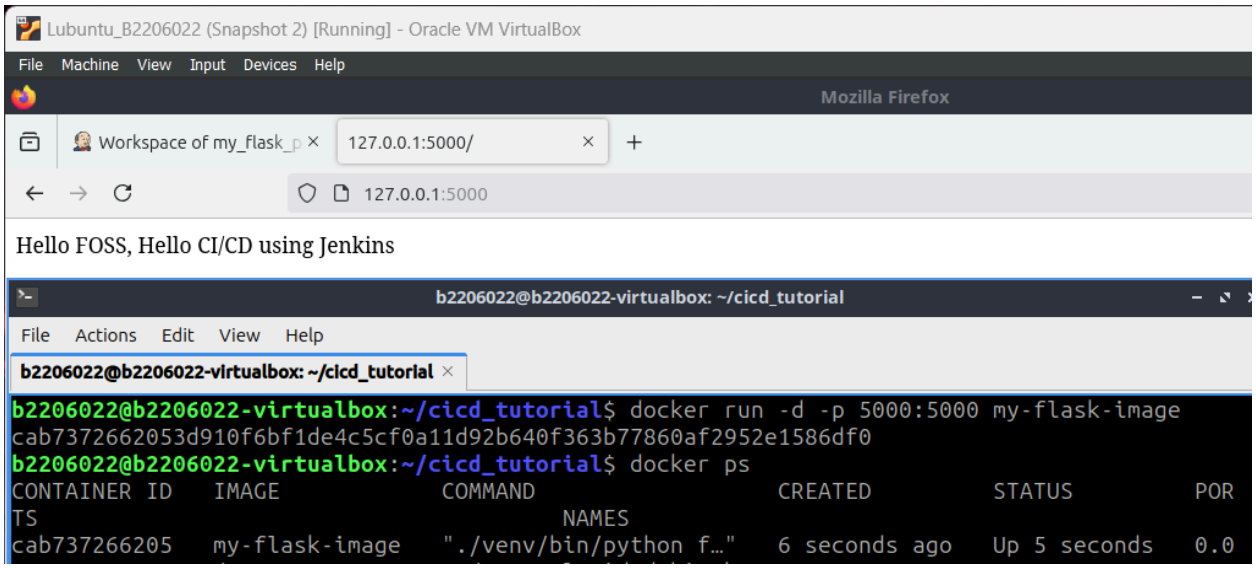
Code Blame 10 lines (7 loc) · 201 Bytes Code 55%

```python
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')
```

- 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) (take a screenshot)



- On your Jenkins project configure, under "Build Triggers", do not forget to deselect "Build periodically"

---END---