

## 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

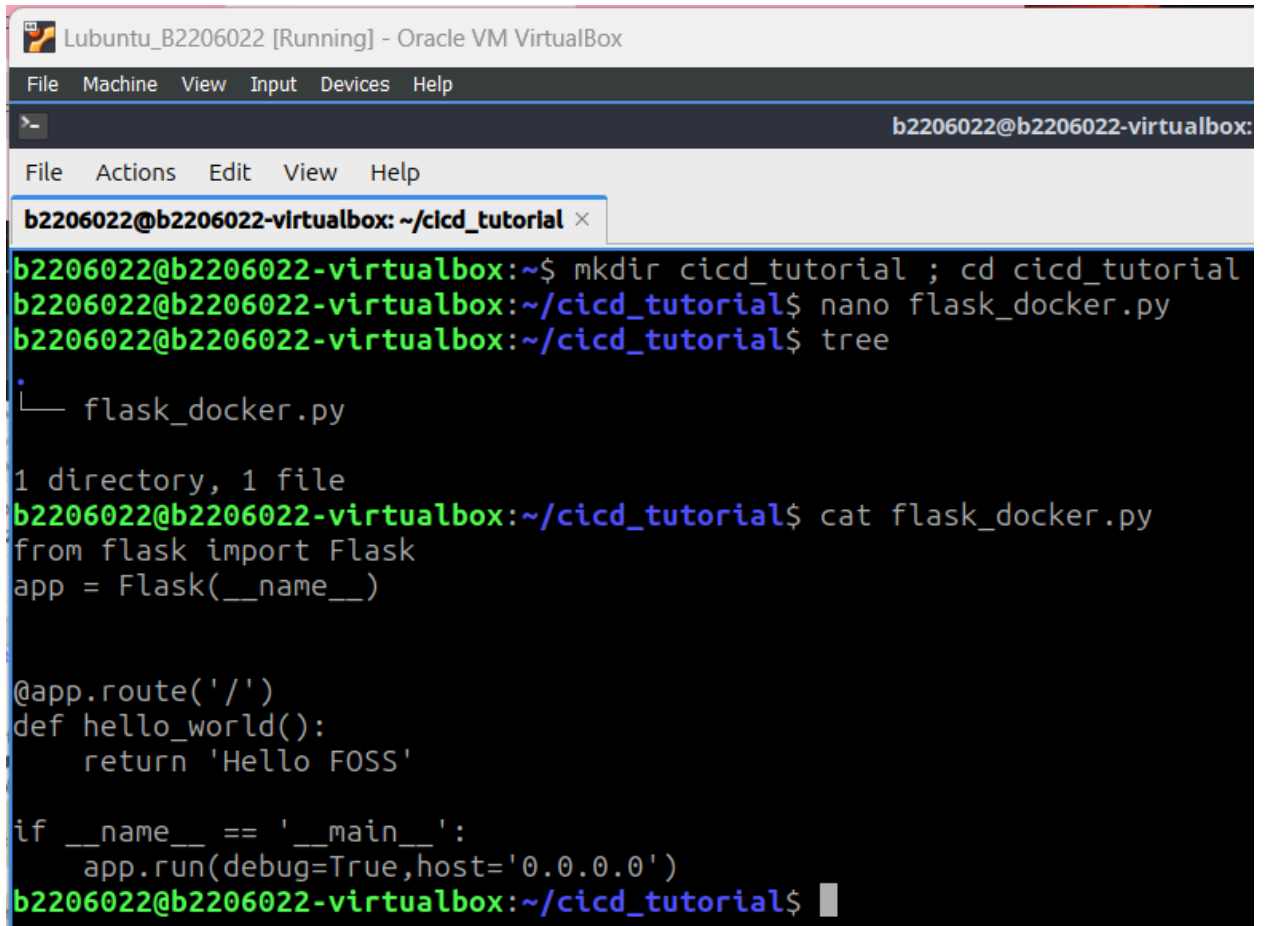
##### 1.1. Deploy a Flask application

- Create a sample Flask application:

```
$mkdir cicc_tutorial ; cd cicc_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')
```



The screenshot shows a terminal window titled "Lubuntu\_B2206022 [Running] - Oracle VM VirtualBox". The terminal prompt is "b2206022@b2206022-virtualbox: ~". The user has navigated to a directory named "cicd\_tutorial" and created a file named "flask\_docker.py". The file content is as follows:

```
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

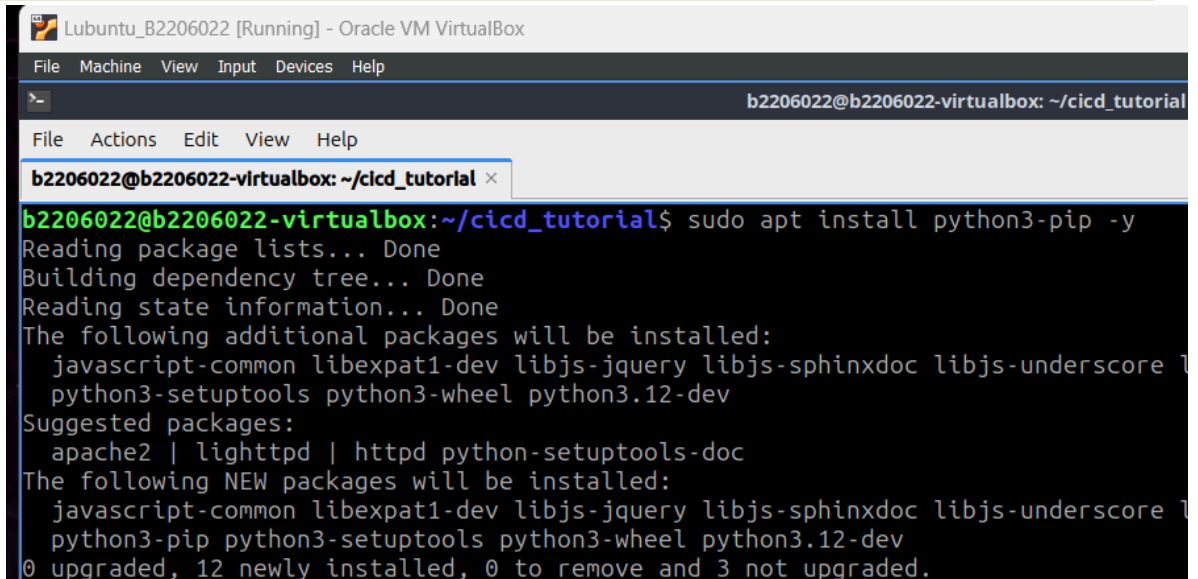
1 directory, 1 file
b2206022@b2206022-virtualbox:~/cicd_tutorial$ cat flask_docker.py
from flask import Flask
app = Flask(__name__)

@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$
```

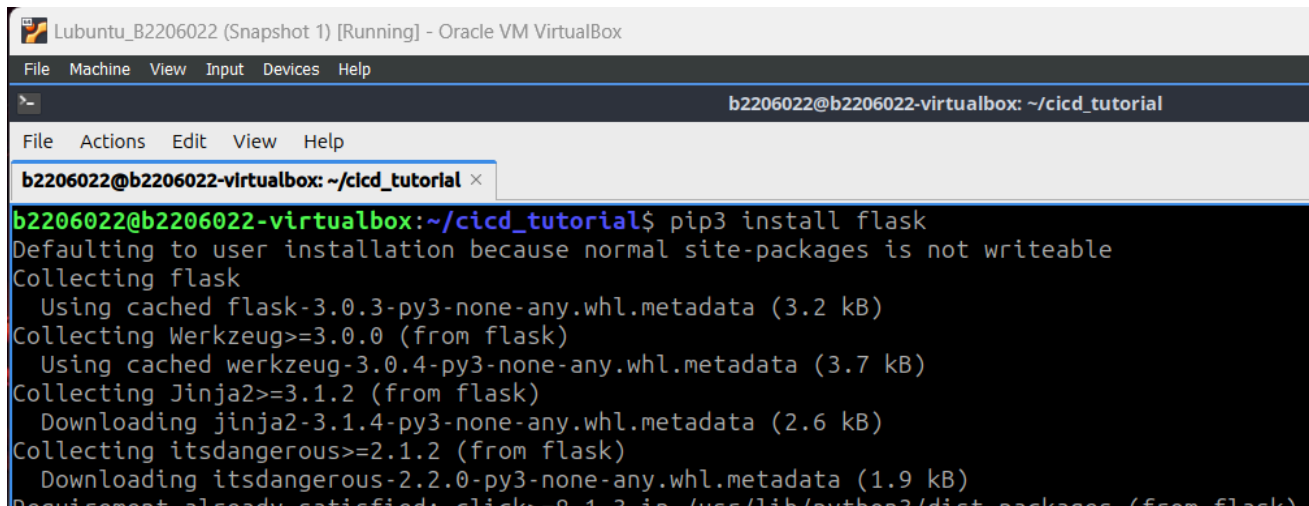
- Install pip (package installer for Python), and then the Flask framework

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



The screenshot shows a terminal window titled "Lubuntu\_B2206022 [Running] - Oracle VM VirtualBox". The terminal prompt is "b2206022@b2206022-virtualbox: ~/cicd\_tutorial". The user has run the command "sudo apt install python3-pip -y". The output is as follows:

```
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
  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
  python3-pip python3-setuptools python3-wheel python3.12-dev
0 upgraded, 12 newly installed, 0 to remove and 3 not upgraded.
```



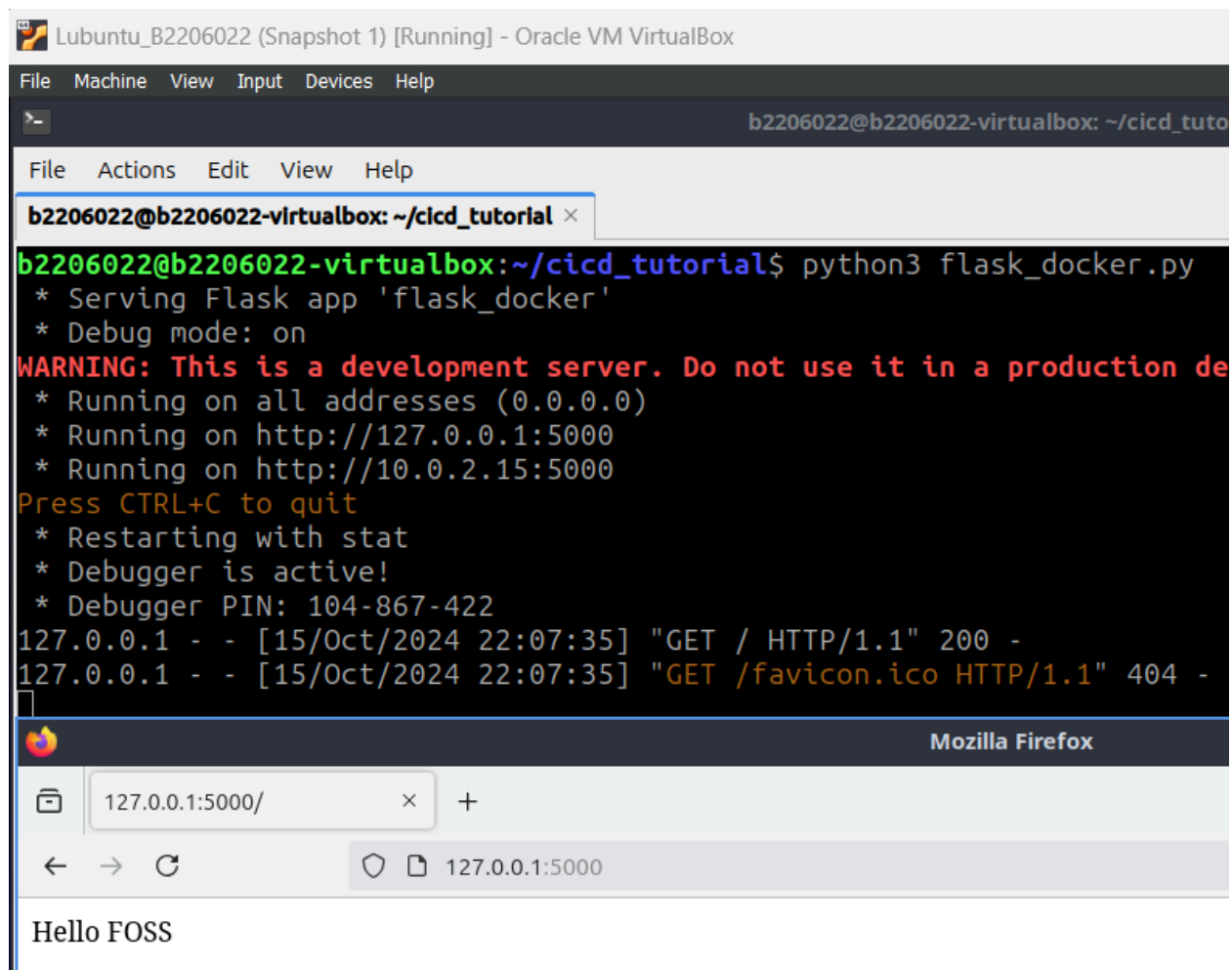
The screenshot shows a terminal window titled "Lubuntu\_B2206022 (Snapshot 1) [Running] - Oracle VM VirtualBox". The terminal prompt is "b2206022@b2206022-virtualbox: ~/cicd\_tutorial". The command "pip3 install flask" has been executed. The output shows the installation process, including collecting packages and downloading metadata for Flask, Werkzeug, Jinja2, and itsdangerous. The installation is successful, and the requirement is satisfied.

```
b2206022@b2206022-virtualbox: ~/cicd_tutorial$ pip3 install flask
Defaulting to user installation because normal site-packages is not writeable
Collecting flask
  Using cached flask-3.0.3-py3-none-any.whl.metadata (3.2 kB)
Collecting Werkzeug>=3.0.0 (from flask)
  Using cached werkzeug-3.0.4-py3-none-any.whl.metadata (3.7 kB)
Collecting Jinja2>=3.1.2 (from flask)
  Downloading jinja2-3.1.4-py3-none-any.whl.metadata (2.6 kB)
Collecting itsdangerous>=2.1.2 (from flask)
  Downloading itsdangerous-2.2.0-py3-none-any.whl.metadata (1.9 kB)
Requirement already satisfied: click>=8.1.3 in /usr/lib/python3/dist-packages (from 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>), (take a screenshot)



The screenshot shows a VirtualBox window titled "Lubuntu\_B2206022 (Snapshot 1) [Running] - Oracle VM VirtualBox". Inside the window, there is a terminal window and a Mozilla Firefox browser window. The terminal window shows the command `python3 flask_docker.py` being executed, which starts a Flask application in debug mode on port 5000. It displays a warning about using a development server in production and shows log messages for two GET requests: one to the root path (200 OK) and one to `/favicon.ico` (404 Not Found). The browser window shows the address `127.0.0.1:5000/` and displays the text "Hello FOSS".

```
b2206022@b2206022-virtualbox: ~/cicd_tutorial$ python3 flask_docker.py
* Serving Flask app 'flask_docker'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production de
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://10.0.2.15:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 104-867-422
127.0.0.1 - - [15/Oct/2024 22:07:35] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Oct/2024 22:07:35] "GET /favicon.ico HTTP/1.1" 404 -
```

127.0.0.1:5000/

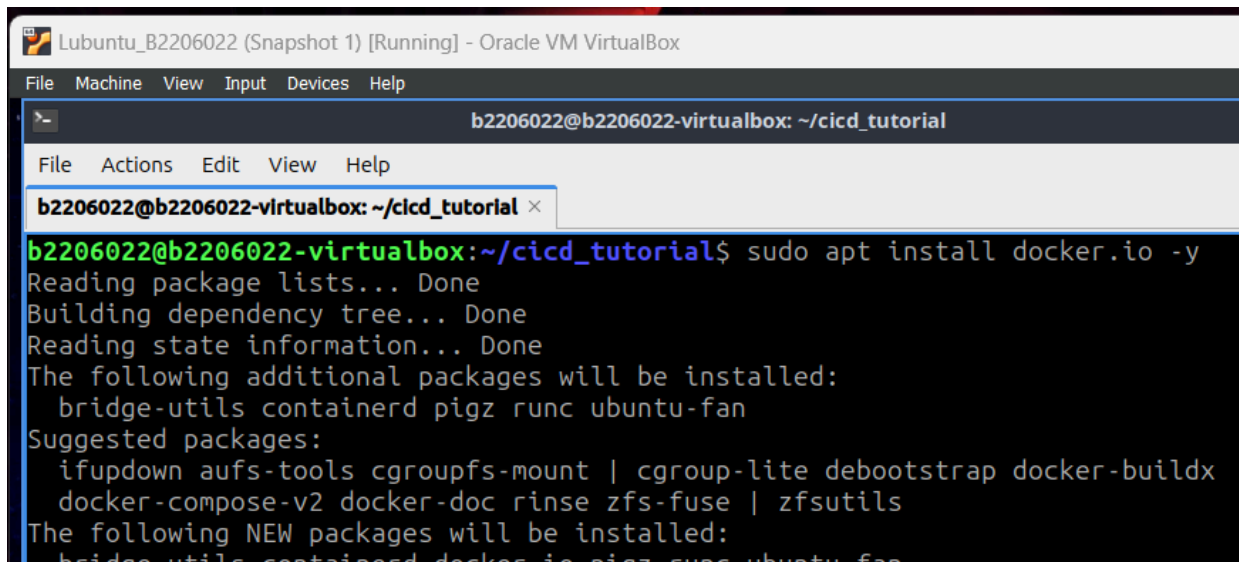
127.0.0.1:5000

Hello FOSS

## 1.2. Dockerize a Flask application using Dockerfile

- Update the apt package index and install Docker

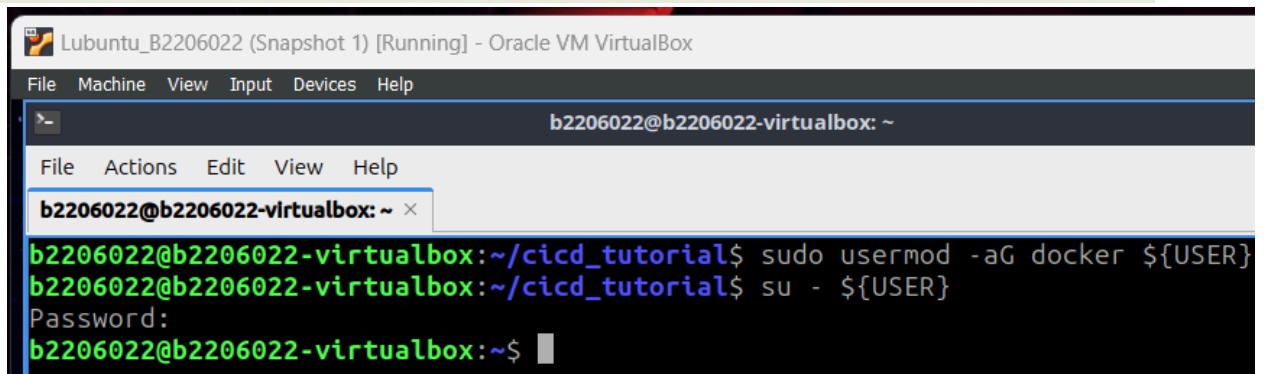
```
$sudo apt update
$sudo apt install docker.io -y
```



The screenshot shows a terminal window titled "b2206022@b2206022-virtualbox: ~/cicd\_tutorial". The user has executed the command `sudo apt install docker.io -y`. The terminal output shows the package lists being read, the dependency tree being built, and the state information being read. It lists additional packages to be installed: `bridge-utils containerd pigz runc ubuntu-fan`. It also lists suggested packages: `ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-buildx docker-compose-v2 docker-doc rinse zfs-fuse | zfsutils`. Finally, it lists the new packages to be installed: `bridge-utils containerd docker.io pigz runc ubuntu-fan`.

- Add current user to the docker group:

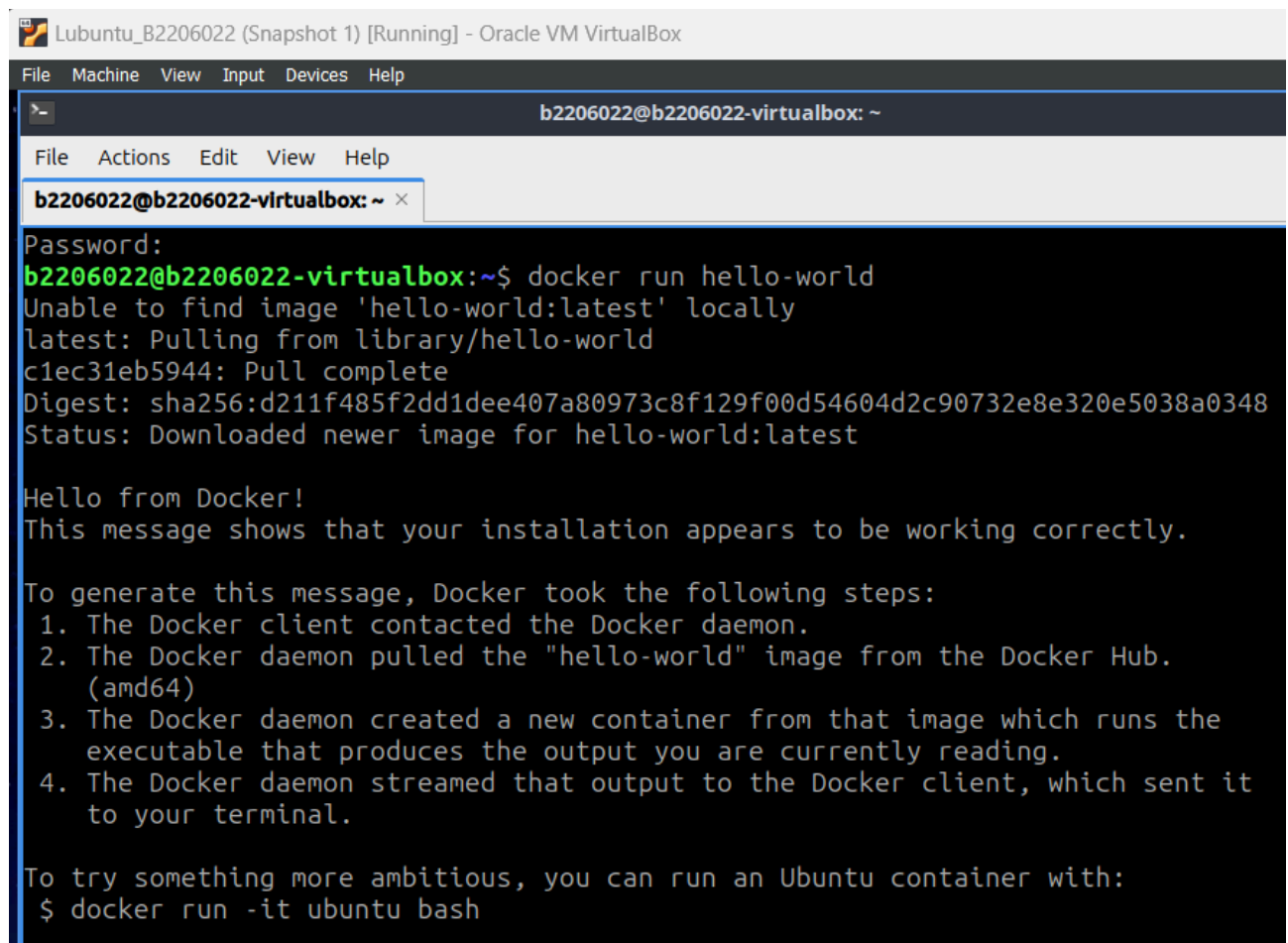
```
$sudo usermod -aG docker ${USER}
$su - ${USER}
```



The screenshot shows a terminal window titled "b2206022@b2206022-virtualbox: ~". The user has executed the command `sudo usermod -aG docker ${USER}`. The terminal output shows the command being executed. The user then executes `su - ${USER}` and enters the password. The terminal output shows the password being entered and the user being switched to the root user.

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

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



The screenshot shows a terminal window titled 'b2206022@b2206022-virtualbox: ~'. The user has entered the command 'docker run hello-world'. The output shows that Docker pulled the 'hello-world:latest' image from the Docker Hub and successfully ran the container, displaying 'Hello from Docker!' and a message about the installation. A list of four steps explains the process: 1. Docker client contacted the daemon, 2. daemon pulled the image, 3. daemon created a container, and 4. daemon streamed the output to the client. The terminal also suggests running 'docker run -it ubuntu bash' for a more ambitious test.

```
Lubuntu_B2206022 (Snapshot 1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
b2206022@b2206022-virtualbox: ~
File Actions Edit View Help
b2206022@b2206022-virtualbox: ~ x
Password:
b2206022@b2206022-virtualbox:~$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:d211f485f2dd1dee407a80973c8f129f00d54604d2c90732e8e320e5038a0348
Status: Downloaded newer image for hello-world:latest

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

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash
```

- Create a requirements.txt file

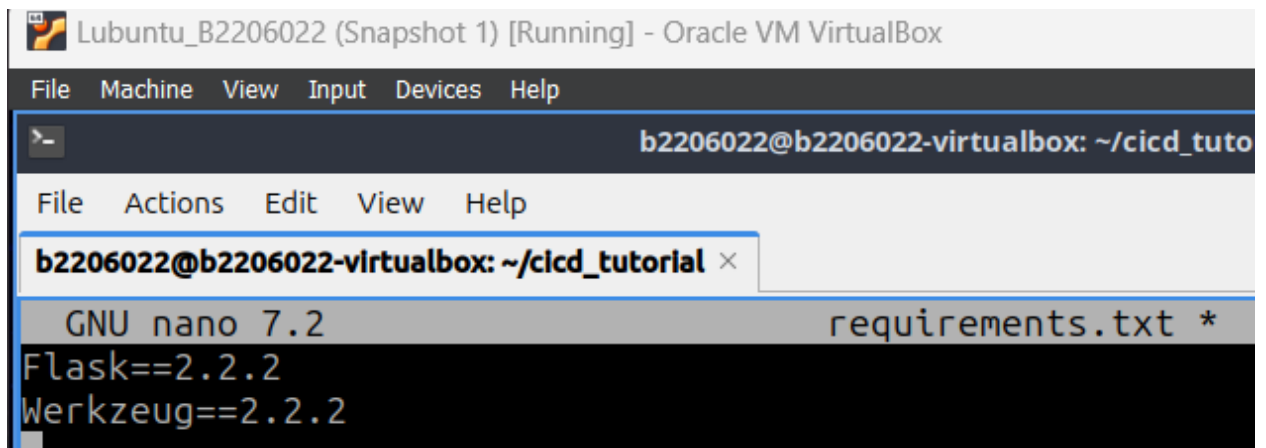
```
$nano requirements.txt
```

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
```



The screenshot shows a VirtualBox window titled 'Lubuntu\_B2206022 (Snapshot 1) [Running] - Oracle VM VirtualBox'. Inside the window, there is a terminal window with the prompt 'b2206022@b2206022-virtualbox: ~/cicd\_tuto'. Below the terminal, a nano editor window is open, showing the file 'requirements.txt'. The content of the file is:

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

- Create a Dockerfile file

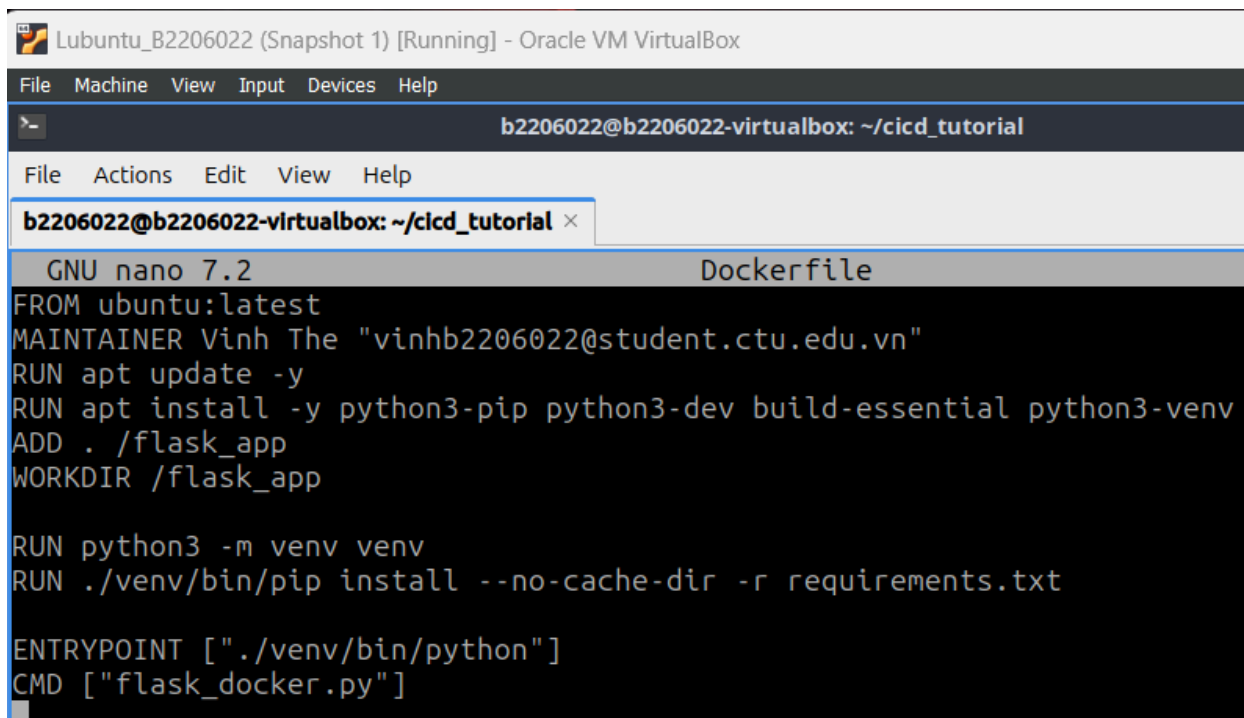
```
$nano Dockerfile
```

#### 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"]



The screenshot shows a terminal window titled "Lubuntu\_B2206022 (Snapshot 1) [Running] - Oracle VM VirtualBox". The terminal prompt is "b2206022@b2206022-virtualbox: ~/cicd\_tutorial". A file named "Dockerfile" is open in the nano text editor. The content of the Dockerfile is as follows:

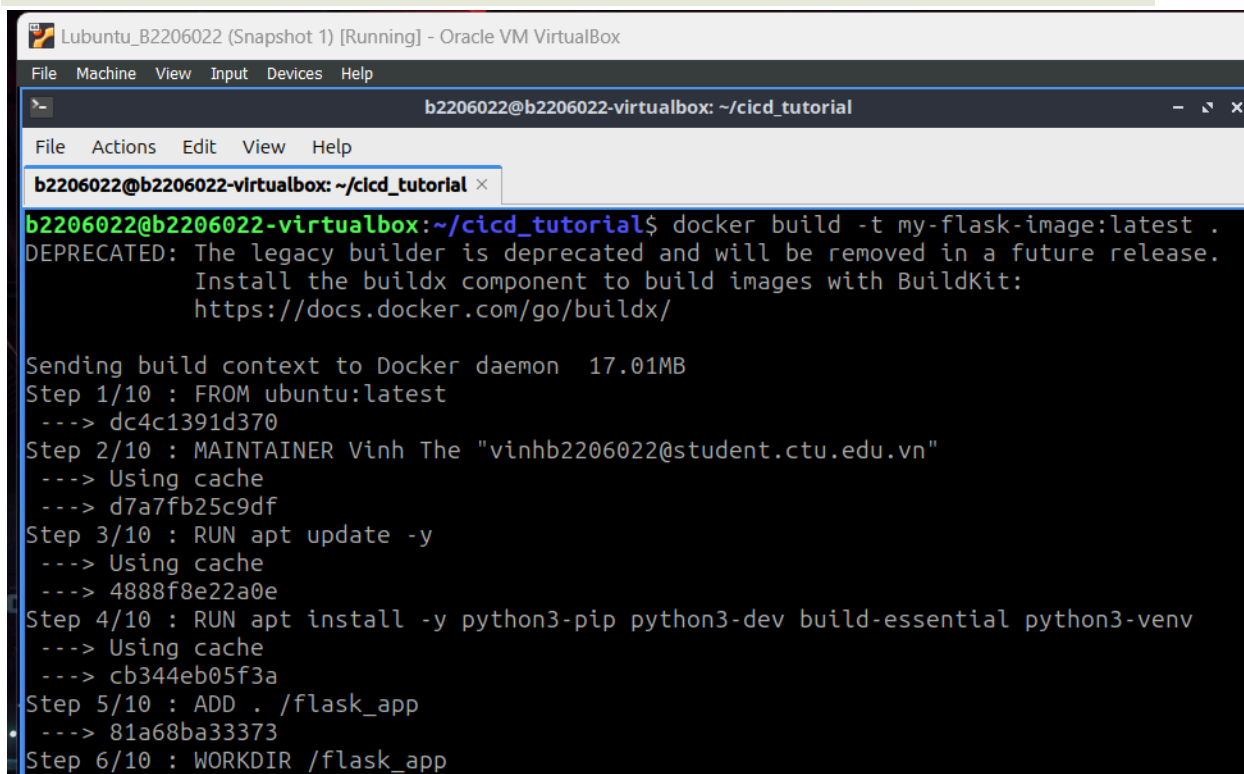
```
GNU nano 7.2 Dockerfile
FROM ubuntu:latest
MAINTAINER Vinh The "vinhb2206022@student.ctu.edu.vn"
RUN apt update -y
RUN apt install -y python3-pip python3-dev build-essential python3-venv
ADD . /flask_app
WORKDIR /flask_app

RUN python3 -m venv venv
RUN ./venv/bin/pip install --no-cache-dir -r requirements.txt

ENTRYPOINT ["/venv/bin/python"]
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 .
```



The screenshot shows the same terminal window after running the command "docker build -t my-flask-image:latest .". The output is as follows:

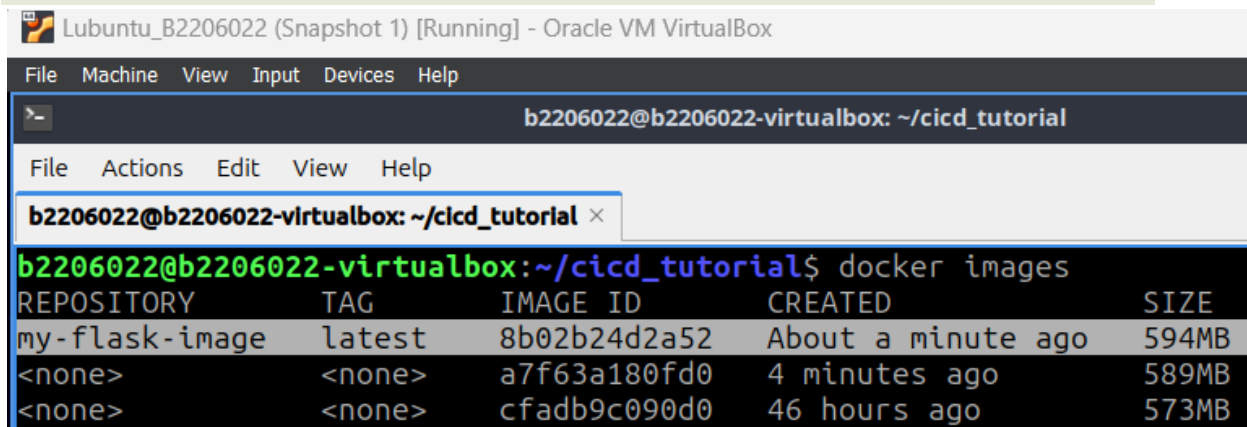
```
b2206022@b2206022-virtualbox:~/cicd_tutorial$ docker build -t my-flask-image:latest .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
             Install the buildx component to build images with BuildKit:
             https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  17.01MB
Step 1/10 : FROM ubuntu:latest
--> dc4c1391d370
Step 2/10 : MAINTAINER Vinh The "vinhb2206022@student.ctu.edu.vn"
--> Using cache
--> d7a7fb25c9df
Step 3/10 : RUN apt update -y
--> Using cache
--> 4888f8e22a0e
Step 4/10 : RUN apt install -y python3-pip python3-dev build-essential python3-venv
--> Using cache
--> cb344eb05f3a
Step 5/10 : ADD . /flask_app
--> 81a68ba33373
Step 6/10 : WORKDIR /flask_app
```



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

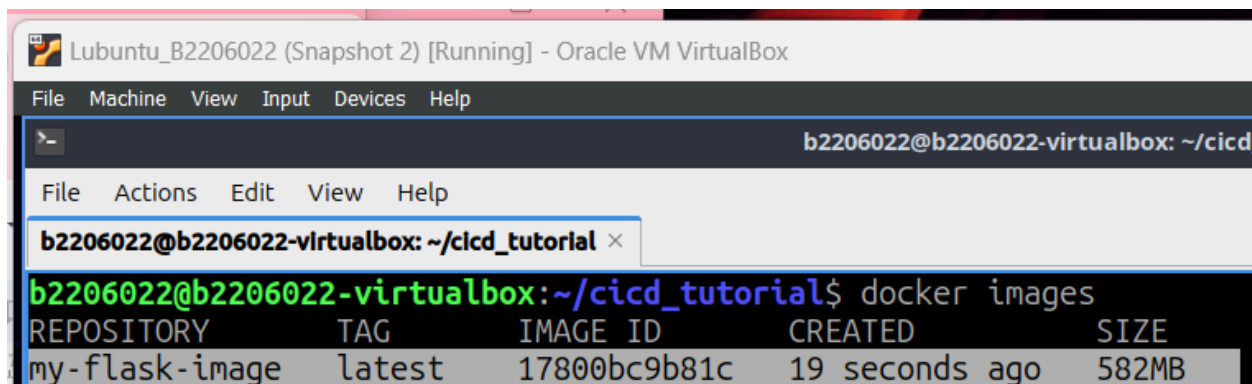
```
$docker images
```



The screenshot shows a terminal window titled "Lubuntu\_B2206022 (Snapshot 1) [Running] - Oracle VM VirtualBox". The terminal prompt is "b2206022@b2206022-virtualbox: ~/cicd\_tutorial". The command "docker images" has been executed, resulting in the following table:

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
my-flask-image	latest	8b02b24d2a52	About a minute ago	594MB
<none>	<none>	a7f63a180fd0	4 minutes ago	589MB
<none>	<none>	cfadb9c090d0	46 hours ago	573MB

- Run your image (take a screenshot)

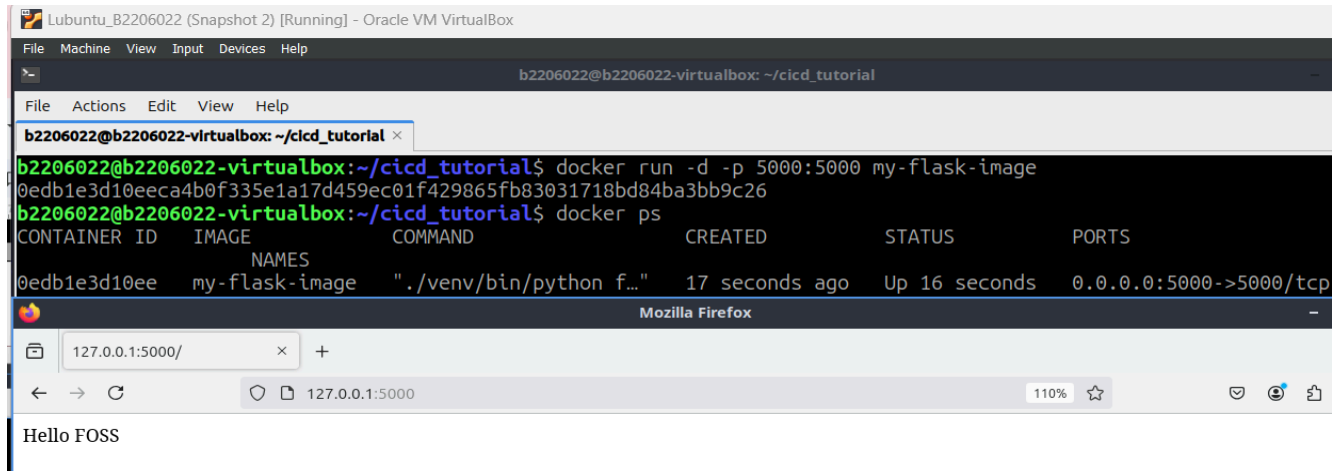


The screenshot shows a terminal window titled "Lubuntu\_B2206022 (Snapshot 2) [Running] - Oracle VM VirtualBox". The terminal prompt is "b2206022@b2206022-virtualbox: ~/cicd\_tutorial". The command "docker images" has been executed, resulting in the following table:

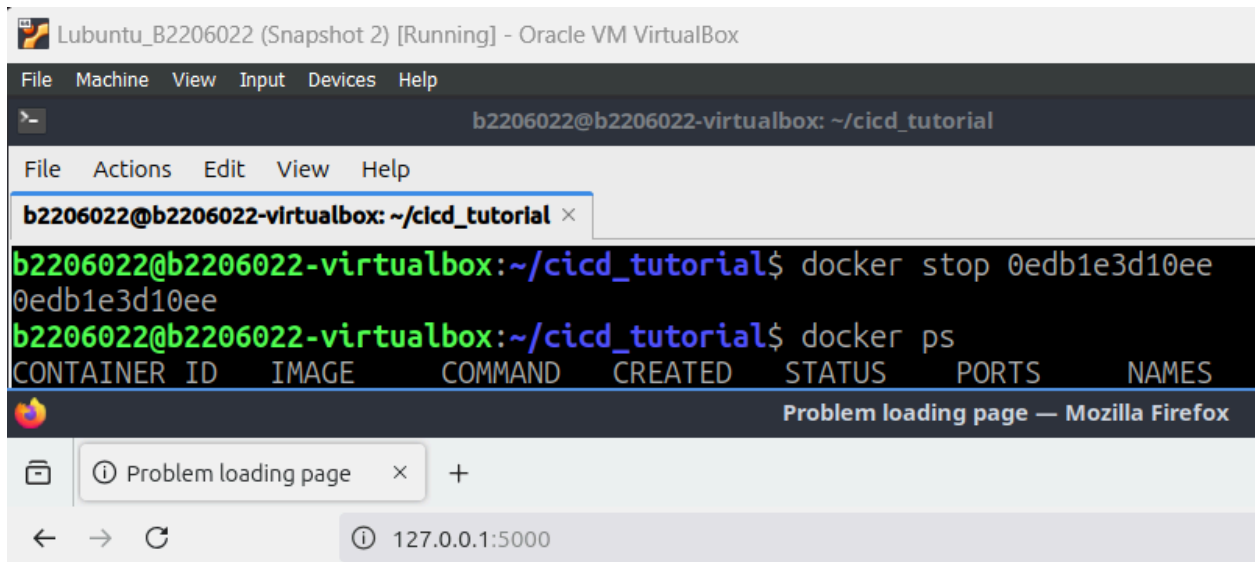
REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
my-flask-image	latest	17800bc9b81c	19 seconds ago	582MB

```
$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>**



## Unable to connect

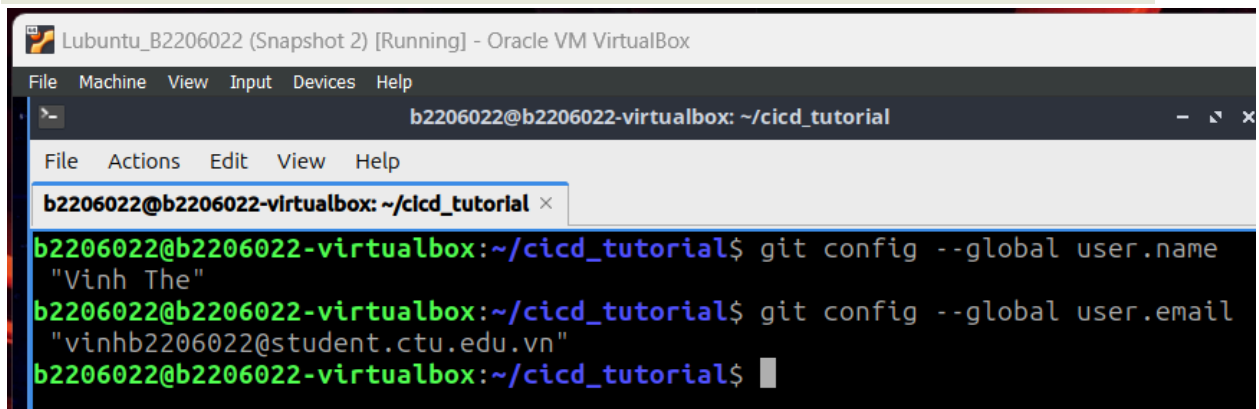
### 2. Automatically dockerize a Flask project using Jenkins

#### 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](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"
```

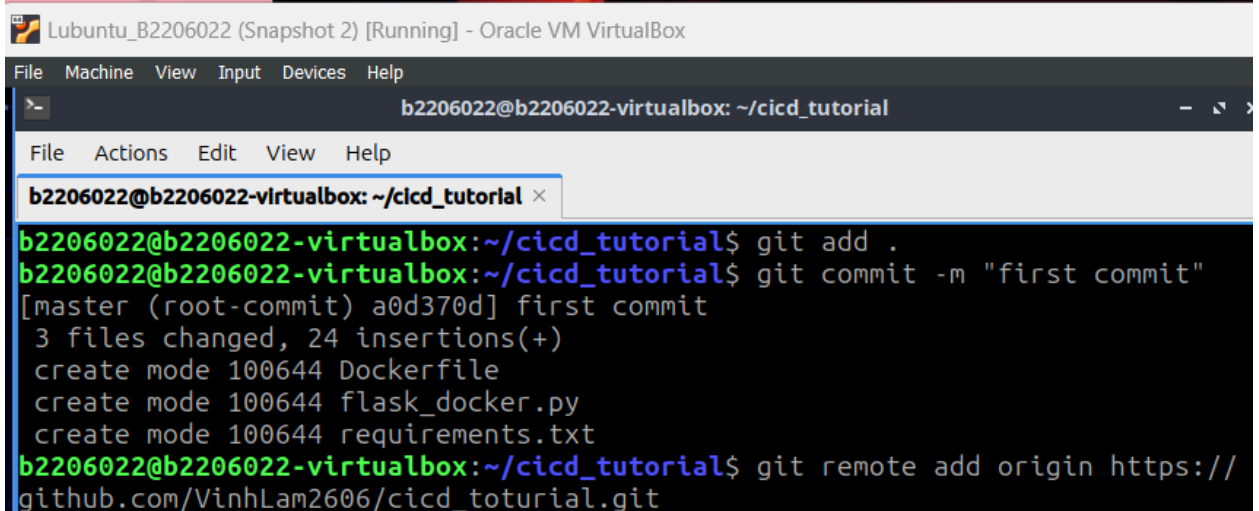


The screenshot shows a terminal window titled "b2206022@b2206022-virtualbox: ~/cicd\_tutorial". The terminal output shows the following commands and their results:

```
b2206022@b2206022-virtualbox:~/cicd_tutorial$ git config --global user.name "Vinh The"
b2206022@b2206022-virtualbox:~/cicd_tutorial$ git config --global user.email "vinhb2206022@student.ctu.edu.vn"
b2206022@b2206022-virtualbox:~/cicd_tutorial$
```

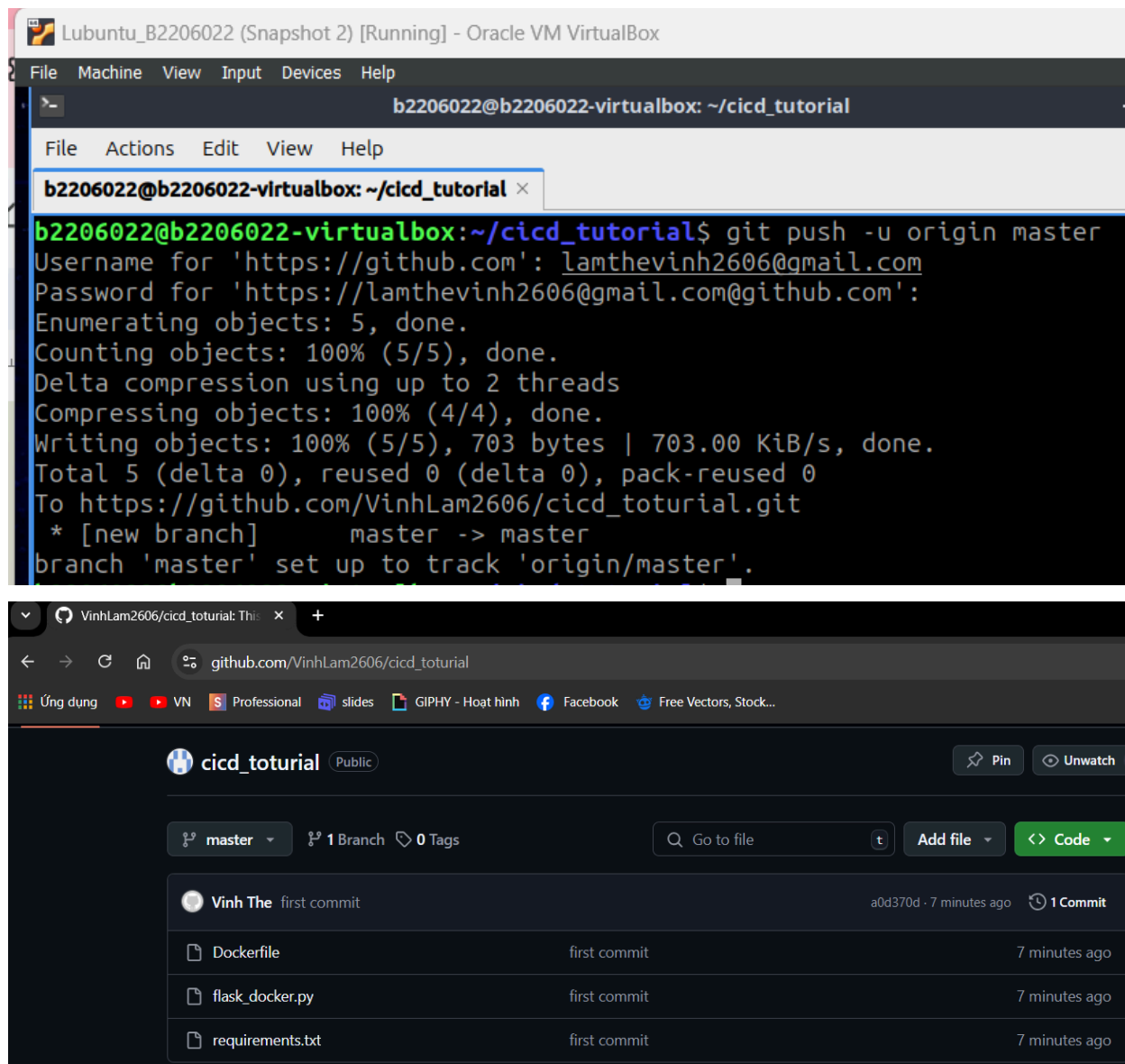
- Initialize git, commit and push your flask project files to Github

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



The screenshot shows a terminal window titled "b2206022@b2206022-virtualbox: ~/cicd\_tutorial". The terminal output shows the following commands and their results:

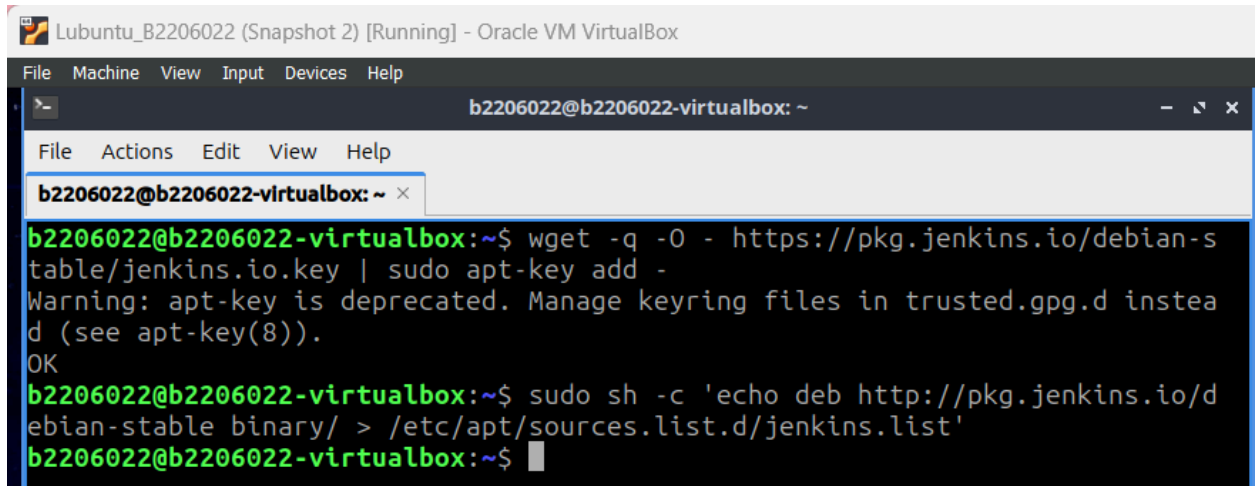
```
b2206022@b2206022-virtualbox:~/cicd_tutorial$ git add .
b2206022@b2206022-virtualbox:~/cicd_tutorial$ git commit -m "first commit"
[master (root-commit) a0d370d] first commit
3 files changed, 24 insertions(+)
create mode 100644 Dockerfile
create mode 100644 flask_docker.py
create mode 100644 requirements.txt
b2206022@b2206022-virtualbox:~/cicd_tutorial$ git remote add origin https://github.com/VinhLam2606/cicd_totutorial.git
```



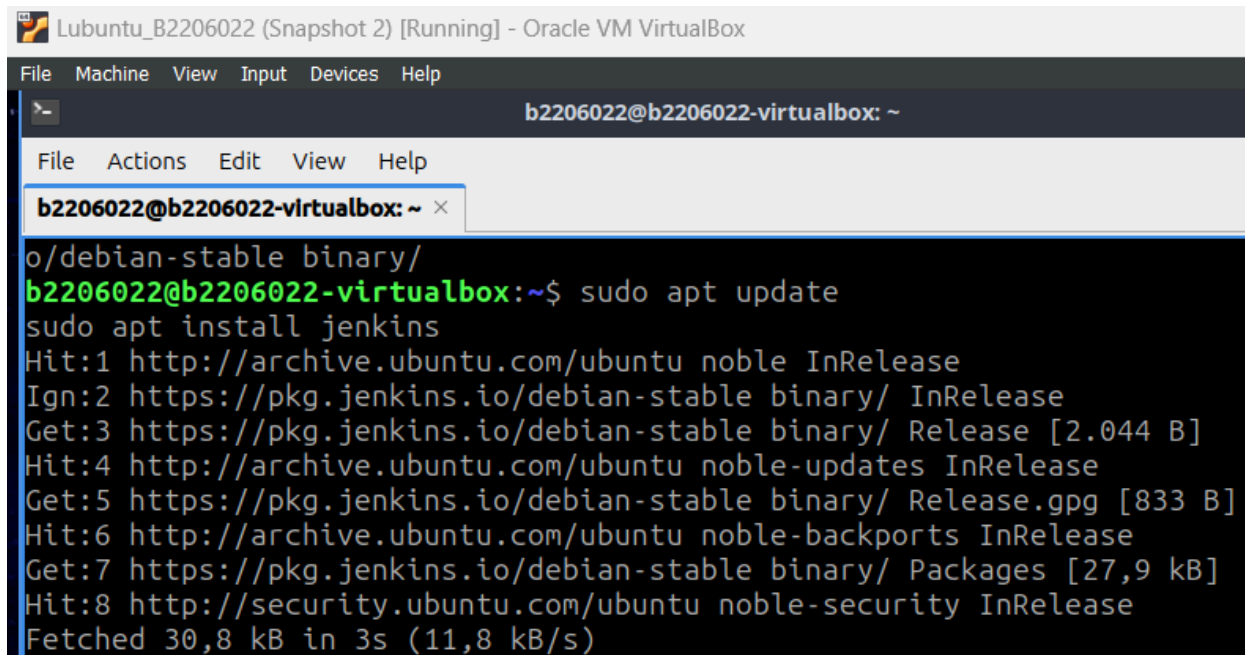
## 2.2. Install and configure Jenkins

### - Install Java and Jenkins

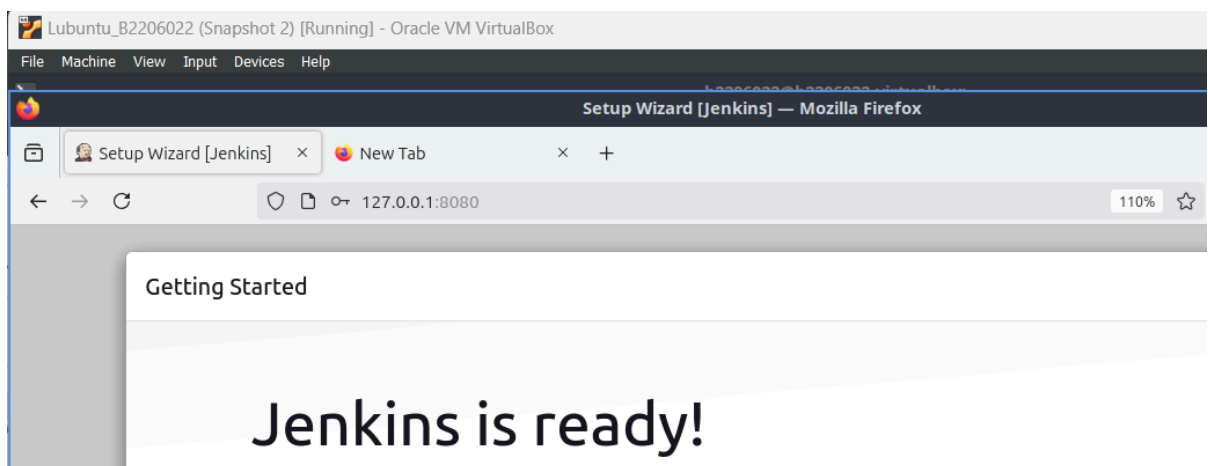
```
$sudo apt install openjdk-11-jdk -y
$wget -q -O -
https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo
apt-key add -
$sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable
binary/ > /etc/apt/sources.list.d/jenkins.list'
$sudo apt update ; sudo apt install jenkins -y
```



```
Lubuntu_B2206022 (Snapshot 2) [Running] - Oracle VM VirtualBox
b2206022@b2206022-virtualbox: ~
File Actions Edit View Help
b2206022@b2206022-virtualbox: ~ x
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
b2206022@b2206022-virtualbox: ~
File Actions Edit View Help
b2206022@b2206022-virtualbox: ~ x
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)
```



### - 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, click "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

Repositories ?

Repository URL ?

Credentials ?

- none -


+ Add

- Under "Build Triggers" select "Build periodically", fill in "\* \* \* \* \*" (build your project every minute)

\* \* \* \* \*

☒ Build periodically ?

Schedule ?

 Do you really mean "every minute" when you say "\* \* \* \* \*"? Perhaps you meant "H \* \* \* \*" to poll once per hour  
Would last have run at Wednesday, October 16, 2024 at 10:55:48 AM Indochina Time; would next run at Wednesday, October 16, 2024 at 10:55:48 AM Indochina Time.

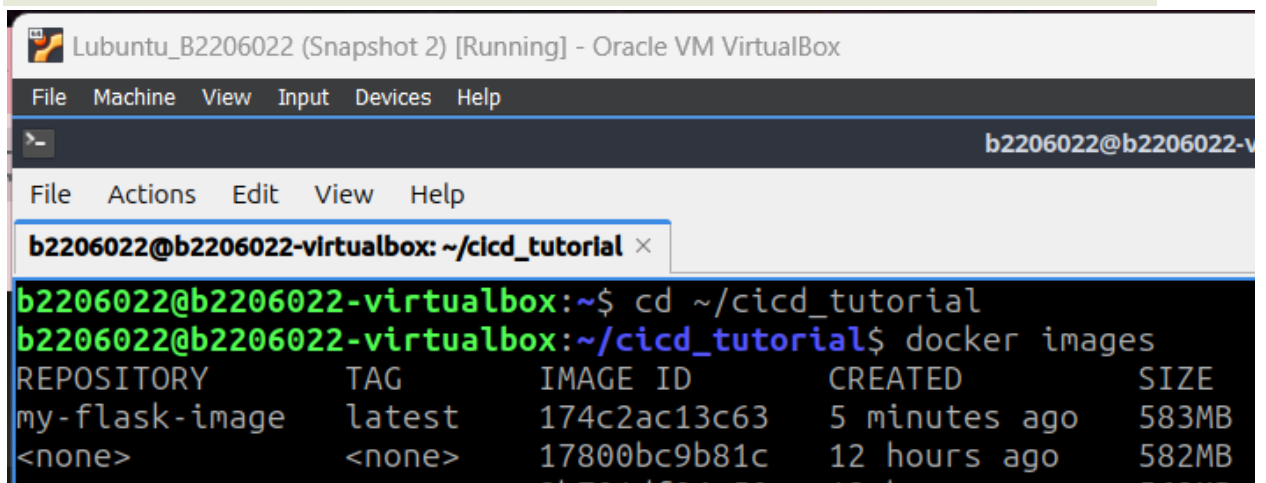
- Under "Build" we will "Add build step", and select "Execute shell". Then fill in "docker build -t my-flask-image:latest ."

```
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
```



## my\_flask\_project

### Permalinks

- [Last build \(#26\), 5 min 5 sec ago](#)
- [Last stable build \(#26\), 5 min 5 sec ago](#)
- [Last successful build \(#26\), 5 min 5 sec ago](#)
- [Last completed build \(#26\), 5 min 5 sec ago](#)

	Build History	trend ▾
	Filter...	/
	<a href="#">#26</a>	<a href="#">Oct 16, 2024, 11:13 AM</a>
	<a href="#">#25</a>	<a href="#">Oct 16, 2024, 11:12 AM</a>
	<a href="#">#24</a>	<a href="#">Oct 16, 2024, 11:11 AM</a>
	<a href="#">#23</a>	<a href="#">Oct 16, 2024, 11:10 AM</a>
	<a href="#">#22</a>	<a href="#">Oct 16, 2024, 11:09 AM</a>
	<a href="#">#21</a>	<a href="#">Oct 16, 2024, 11:08 AM</a>
	<a href="#">#20</a>	<a href="#">Oct 16, 2024, 11:07 AM</a>
	<a href="#">#19</a>	<a href="#">Oct 16, 2024, 11:06 AM</a>

- Modify your Flask application:

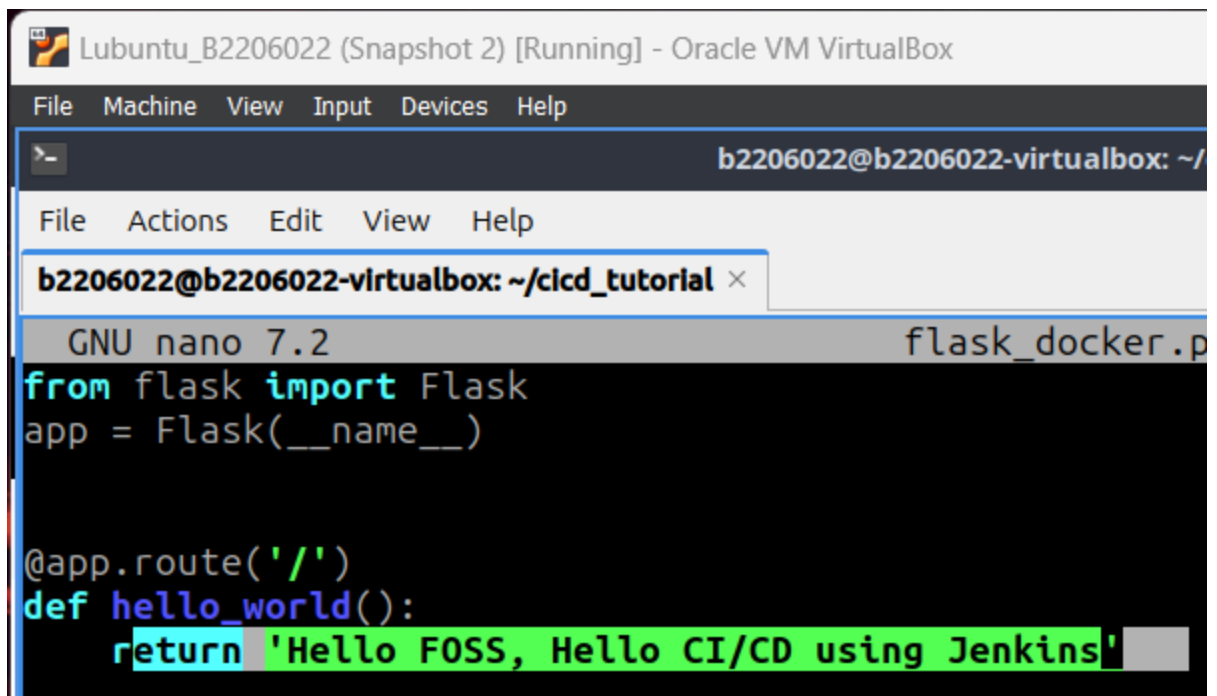
```
$nano flask_docker.py
```

```
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')
```

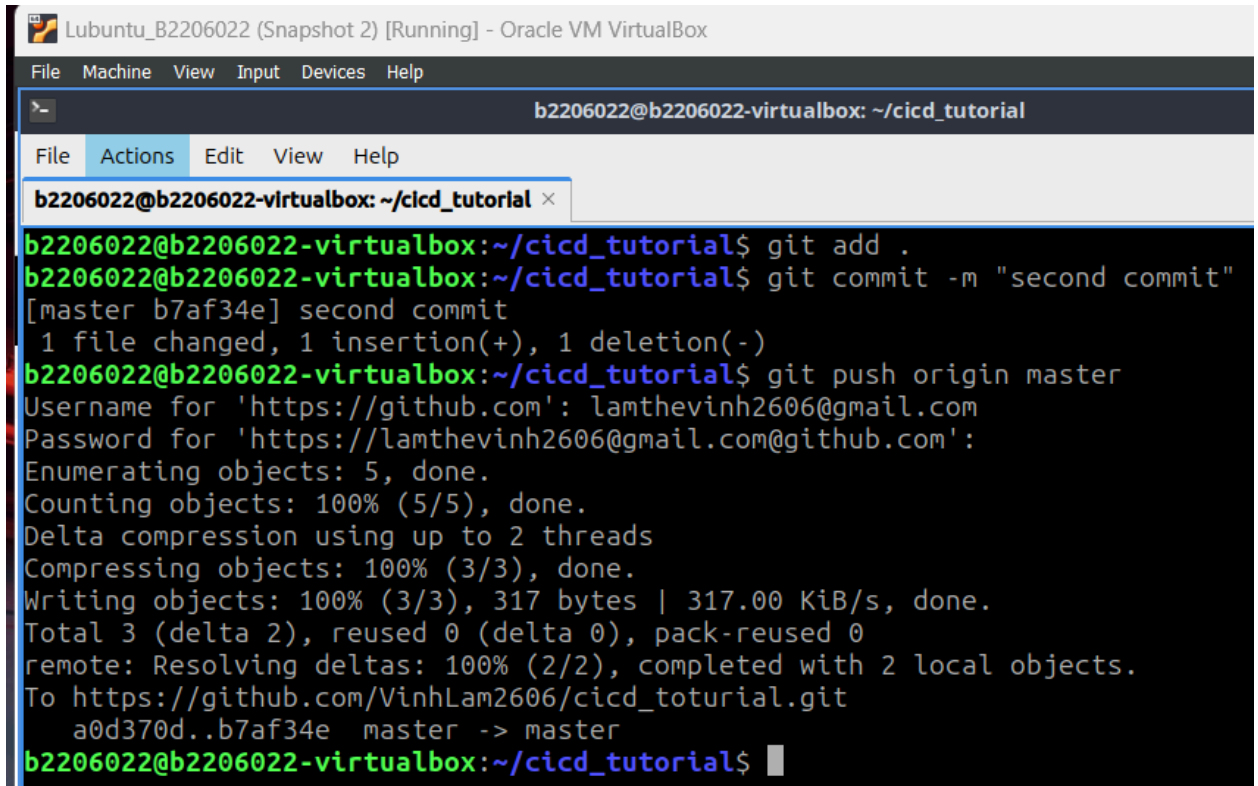


The screenshot shows a VirtualBox window titled "Lubuntu\_B2206022 (Snapshot 2) [Running] - Oracle VM VirtualBox". Inside the window is a terminal window with a menu bar (File, Machine, View, Input, Devices, Help) and a title bar "b2206022@b2206022-virtualbox: ~/". The terminal shows a nano editor session editing a file named "flask\_docker.p". The editor's menu bar includes File, Actions, Edit, View, and Help. The code being edited is a Flask application with a route for "/" and a "hello\_world" function that returns a message. The line "return 'Hello FOSS, Hello CI/CD using Jenkins'" is highlighted in green.

```
b2206022@b2206022-virtualbox: ~/flask_docker.p  
GNU nano 7.2  
from flask import Flask  
app = Flask(__name__)  
  
@app.route('/')  
def hello_world():  
    return 'Hello FOSS, Hello CI/CD using Jenkins'
```

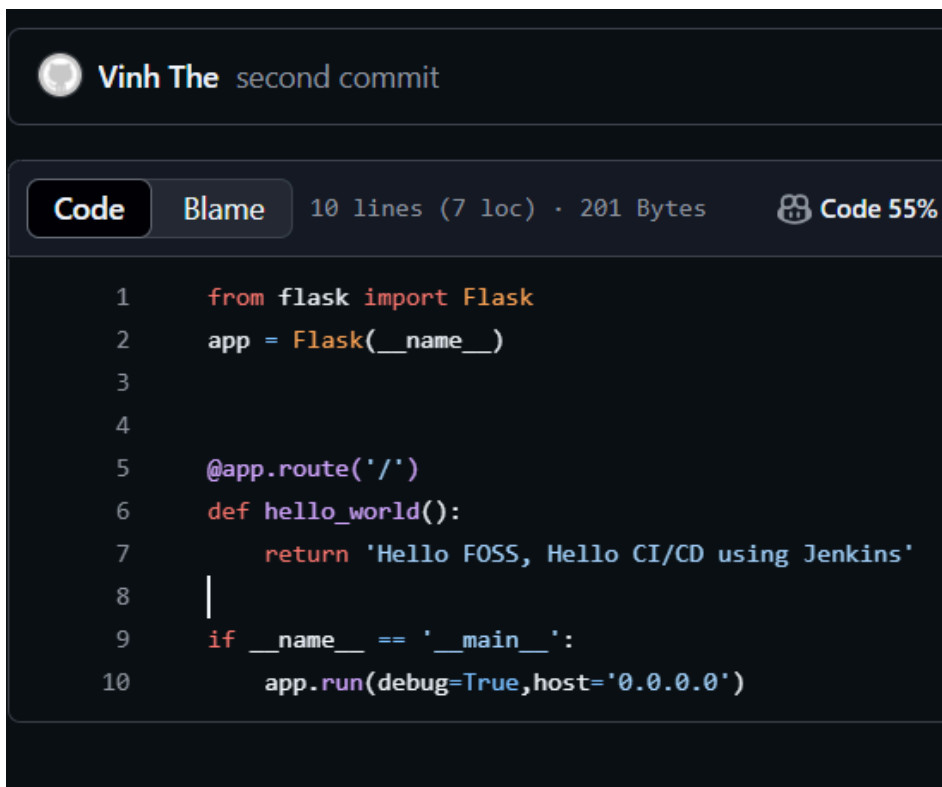
- Commit and push your project files to GitHub

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



The screenshot shows a terminal window titled "b2206022@b2206022-virtualbox: ~/cicd\_tutorial". The terminal displays the following commands and output:

```
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_totutorial.git
a0d370d..b7af34e master -> master
b2206022@b2206022-virtualbox:~/cicd_tutorial$
```



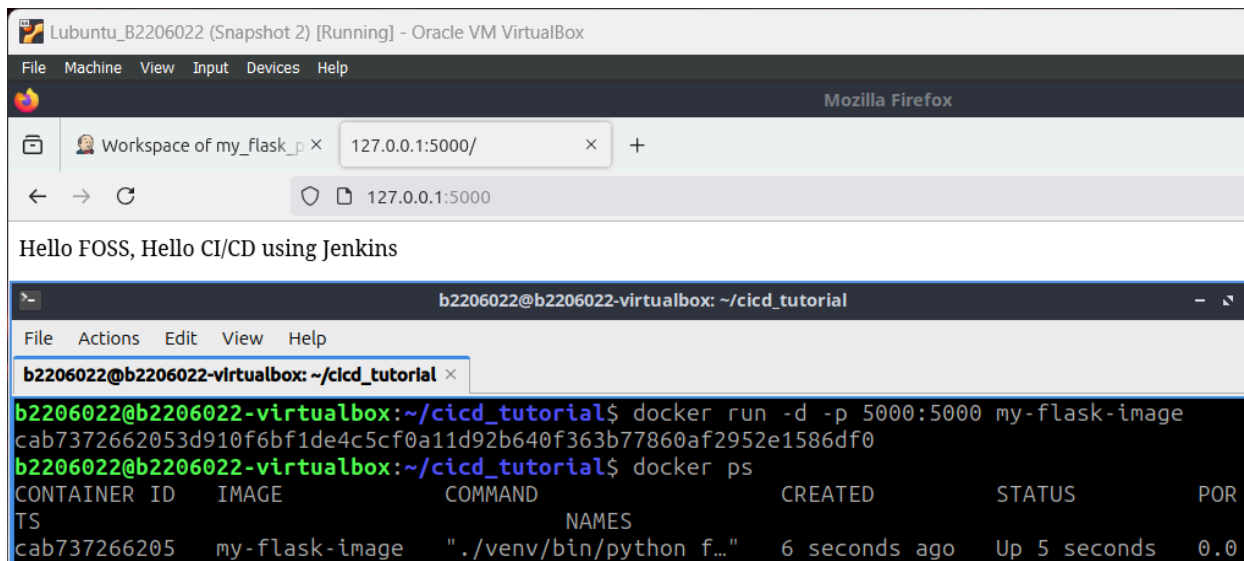
The screenshot shows a GitHub repository view for a commit titled "second commit" by user "Vinh The". The view includes a "Code" tab, a "Blame" tab, and a summary of 10 lines (7 loc) and 201 Bytes. The code is a Python Flask application with a 55% code coverage.

```
1  from flask import Flask
2  app = Flask(__name__)
3
4
5  @app.route('/')
6  def hello_world():
7      return 'Hello FOSS, Hello CI/CD using Jenkins'
8
9  if __name__ == '__main__':
10     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---