LAB 5

CI/CD PIPELINE USING JENKINS, GITHUB AND DOCKER

Fullname: Trương Đặng Trúc Lâm

Student ID: B2111933

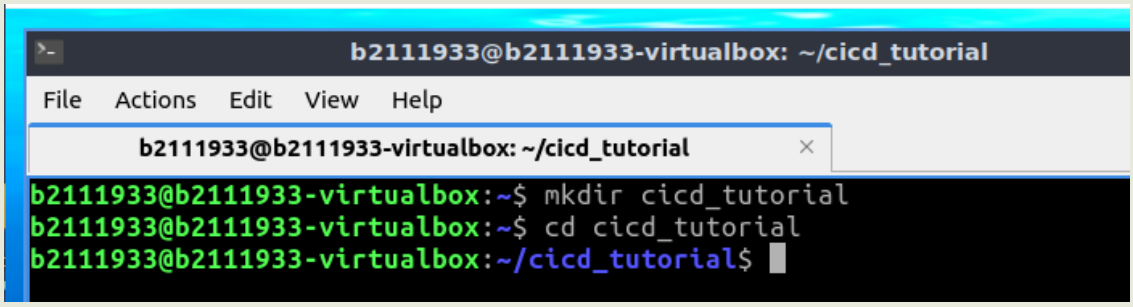
- Note: screenshots need to be clear and good-looking; submissions must be in PDF format.

1. Manually dockerize a Flask project

1.1. Deploy a Flask application

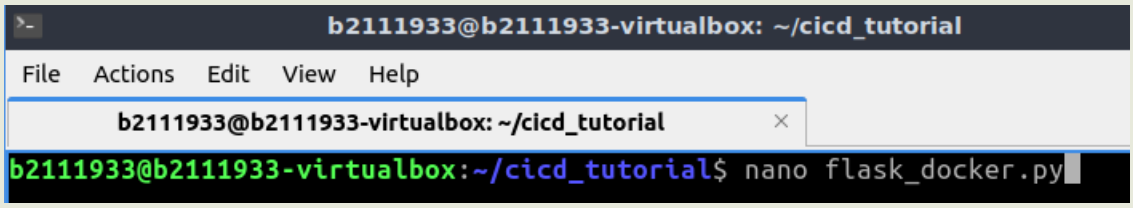
- Create a sample Flask application:

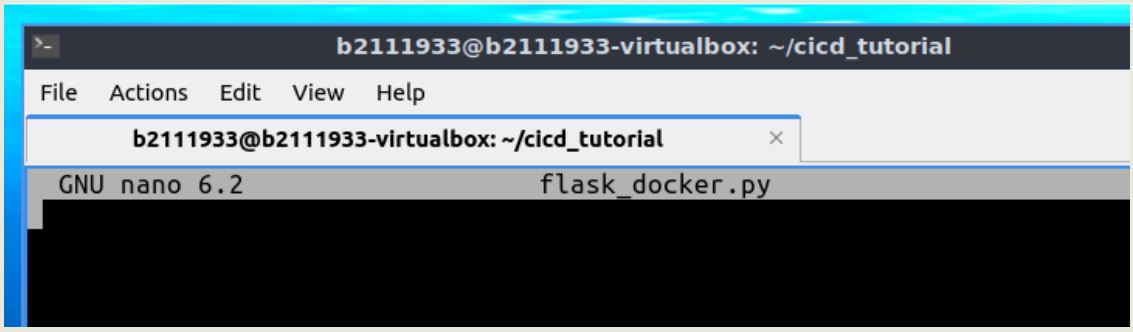
```
$mkdir cicd_tutorial ; cd cicd_tutorial
```



Create directory **cicd_tutorial** & change the working directory to **cicd_tutorial** directory

```
$nano flask_docker.py
```

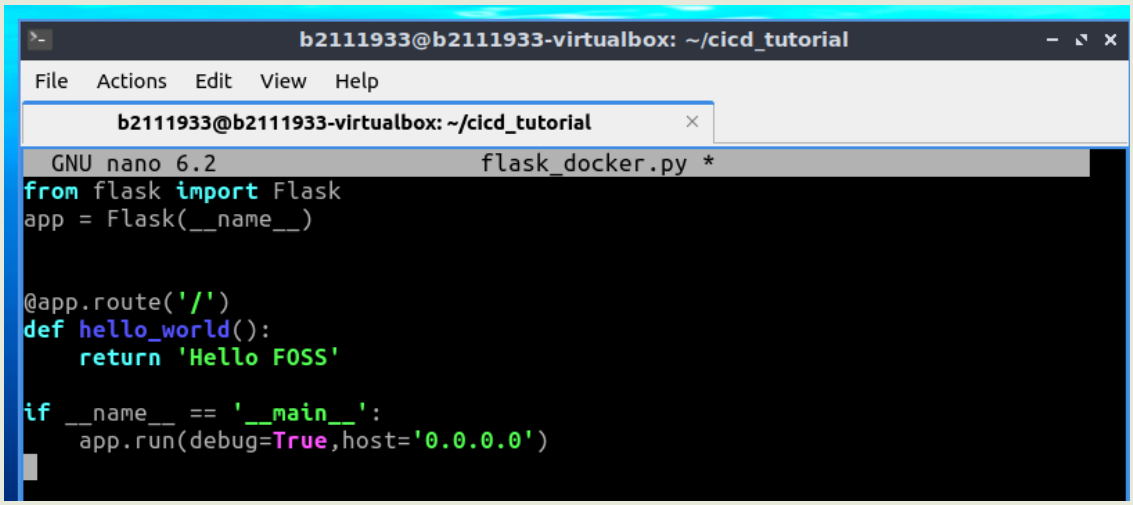




Create file **flask_docker.py** with **nano**

flask_docker.py

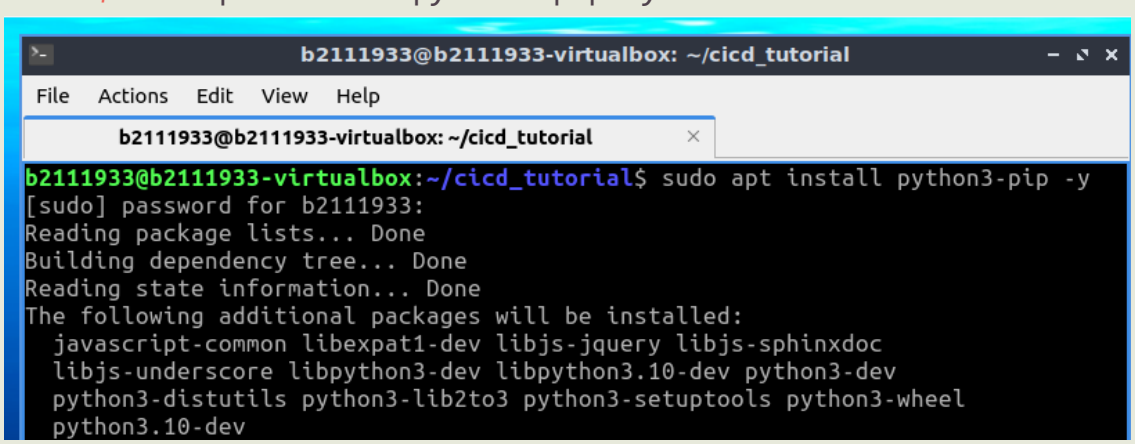
```
from flask import Flask
app = Flask(__name__)

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

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



The contents of file **flask_docker.py** is a simple website that runs on the IP address **0.0.0.0** . When we get access to this website, it will display "**Hello FOSS**"

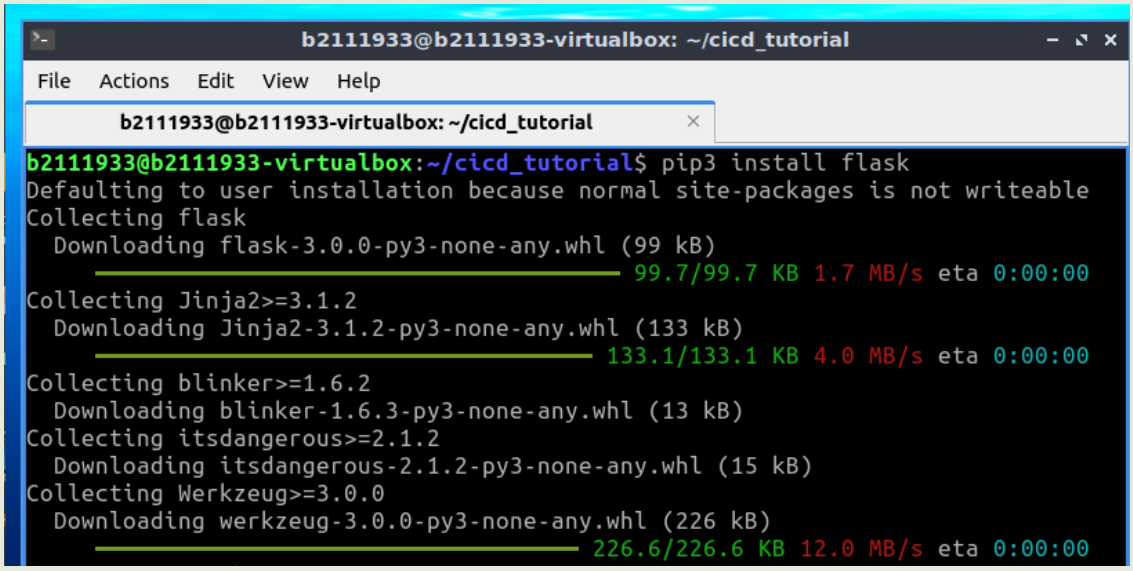
- Install pip (package installer for Python), and then the Flask framework

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



Install **pip** (package installer for Python)

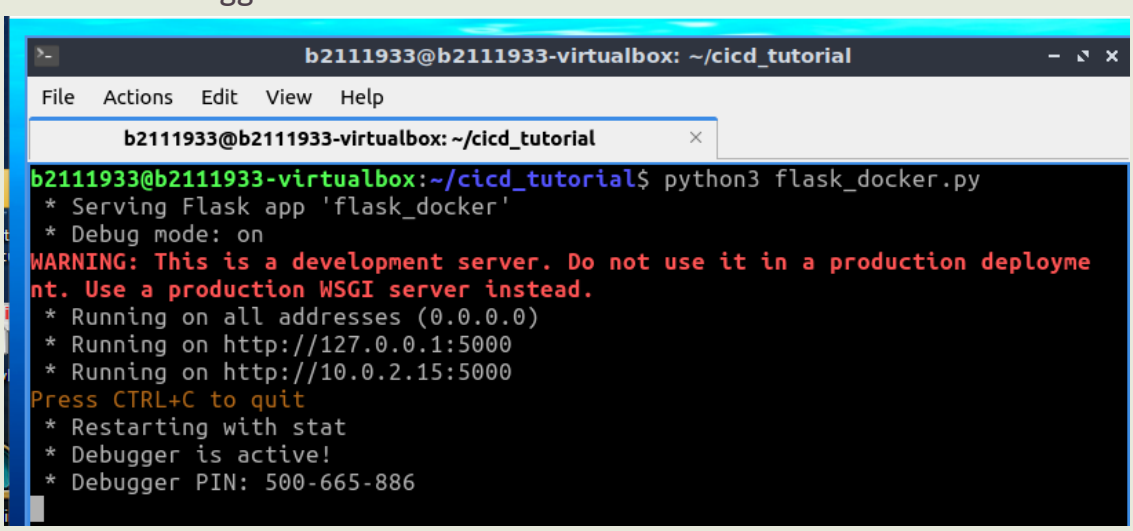
```
$pip3 install flask
```



Use **pip** to install the **Flask** framework (a necessary framework for our website)

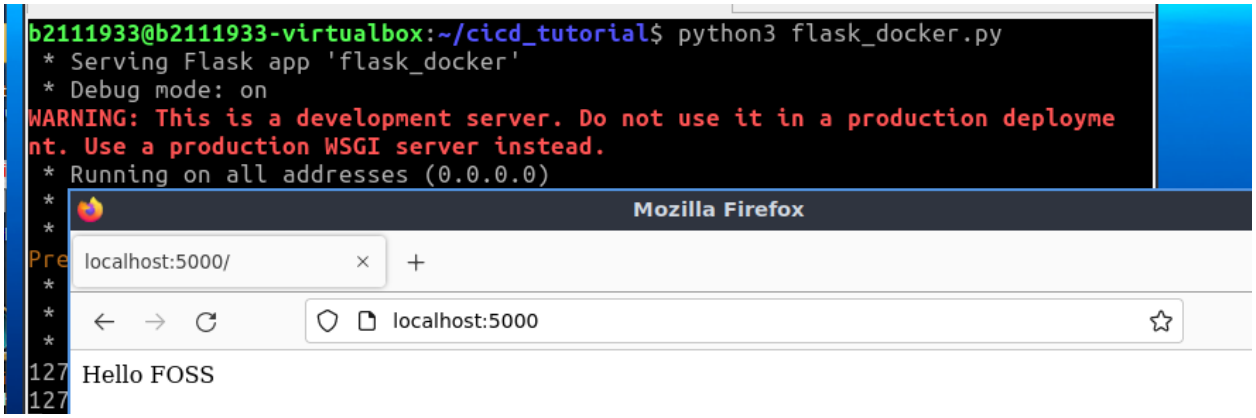
- We can test it out by running:

```
$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
```



The website is running on port **5000** for every IP address of my VM

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

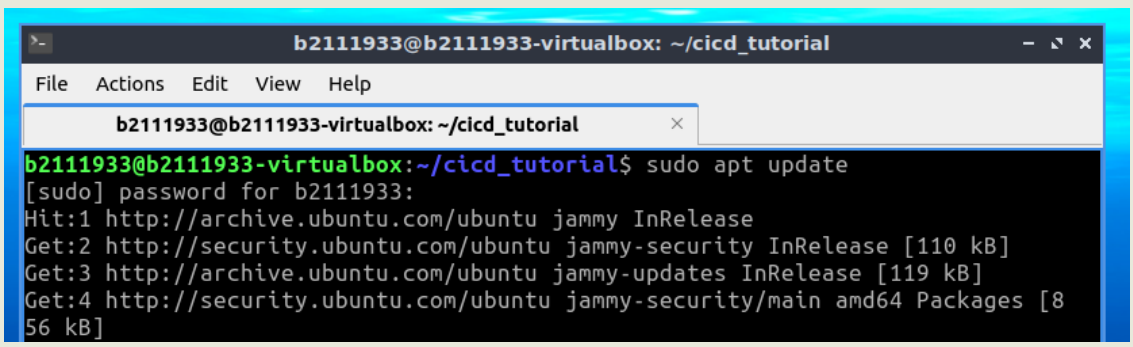


Access the website from **Mozilla Firefox** (**http://localhost:5000**)

1.2. Dockerize a Flask application using Dockerfile

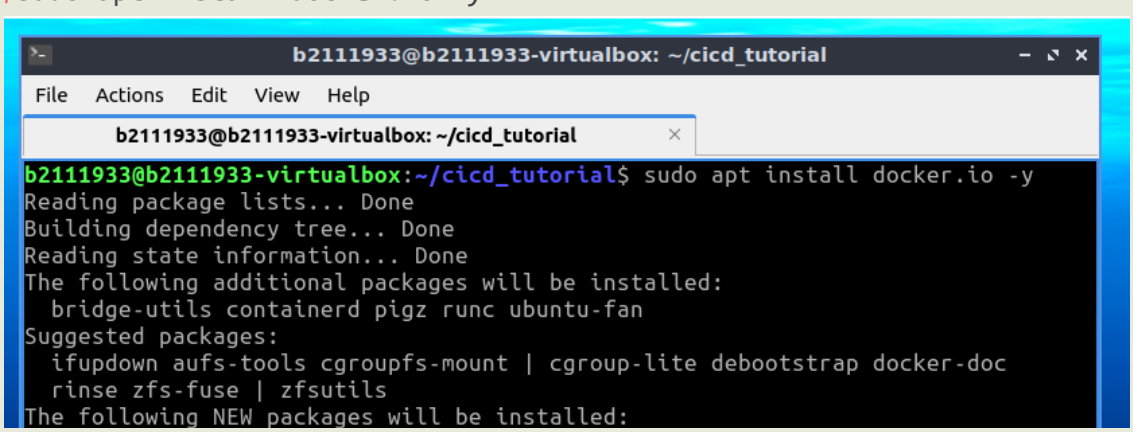
- Update the apt package index and install Docker

```
$sudo apt update
```



Update the **apt package index**

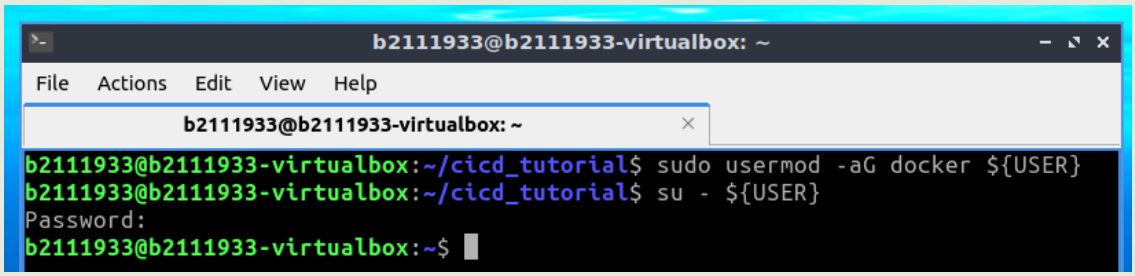
```
$sudo apt install docker.io -y
```



Install **docker.io**

- Add current user to the docker group:

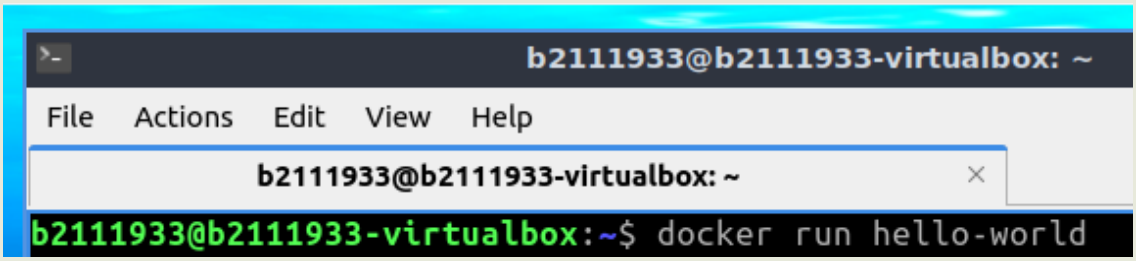
```
$sudo usermod -aG docker ${USER}
$su - ${USER}
```



Add **current user** to the **docker group**, then login again the current user
(no need to restart the VM or log out)

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

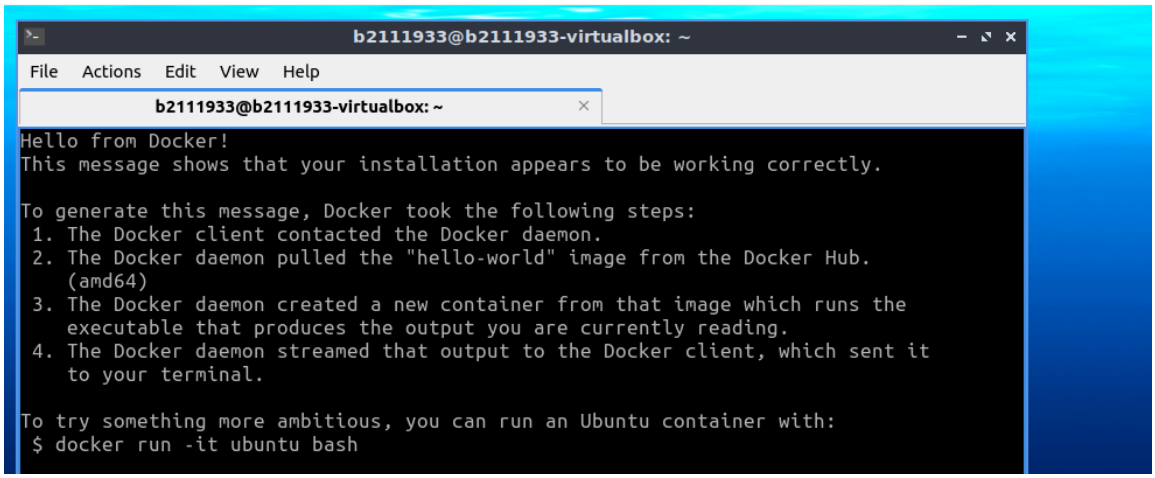
```
$docker run hello-world
```



Pull the image from **Docker** then execute it

The output will indicate that Docker is working correctly:

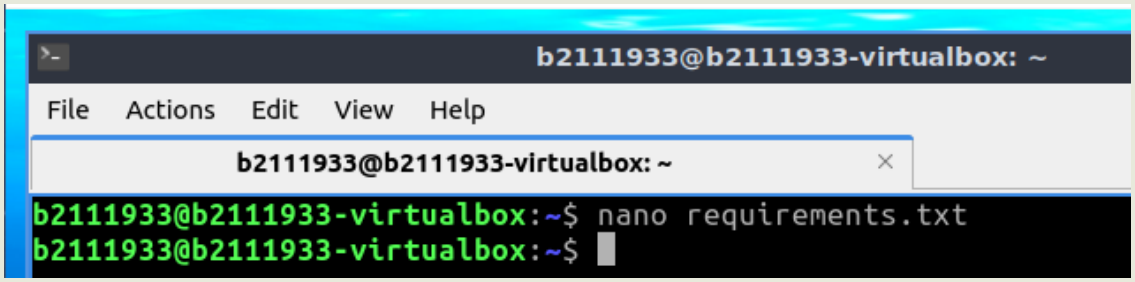
```
Hello from Docker!
This message shows that your installation appears to be working correctly.
```



We can see that **Docker** is working correctly

- Create a requirements.txt file

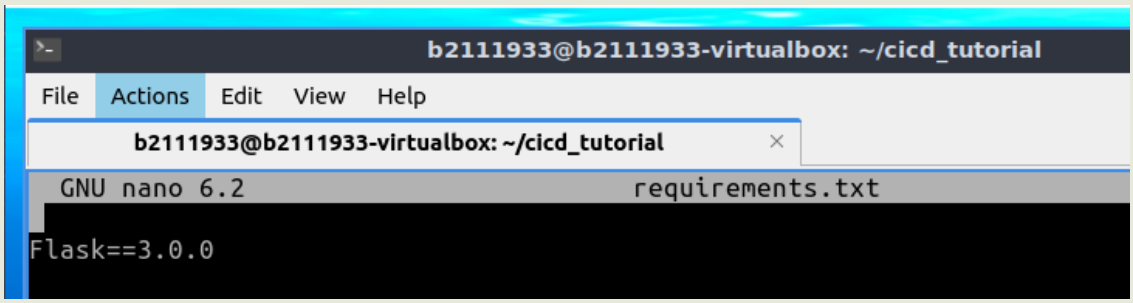
```
$nano requirements.txt
```



Create a **requirements.txt** file with **nano**

requirements.txt

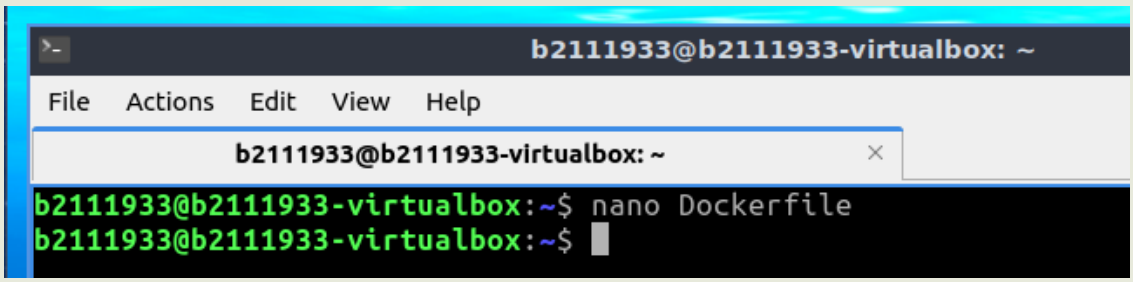
```
Flask==2.2.2
```



The contents of file including the **Flask framework version 3.0.0**
(Flask framework version 2.2.2 is not supported anymore)

- Create a Dockerfile file

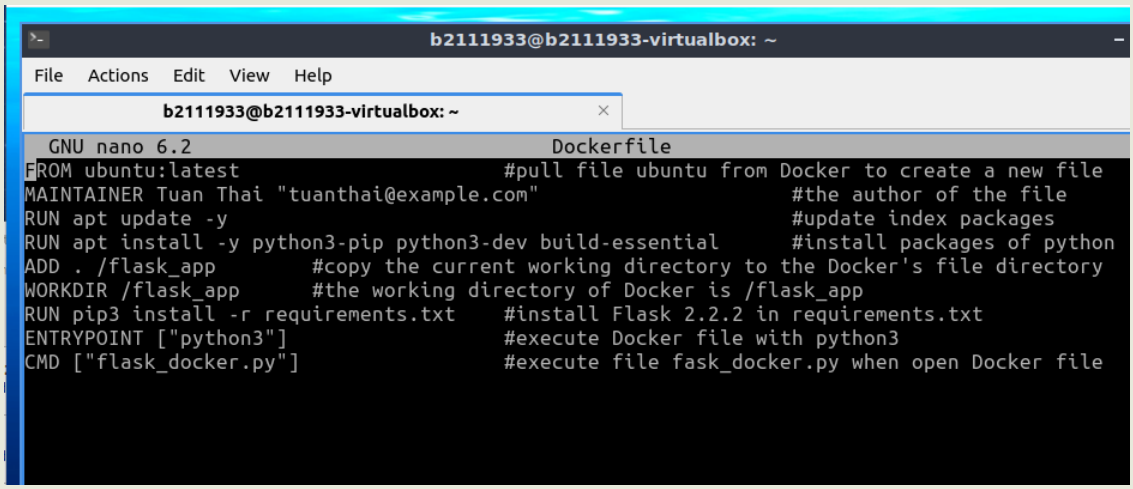
```
$nano Dockerfile
```



Create a **Dockerfile** file with **nano**

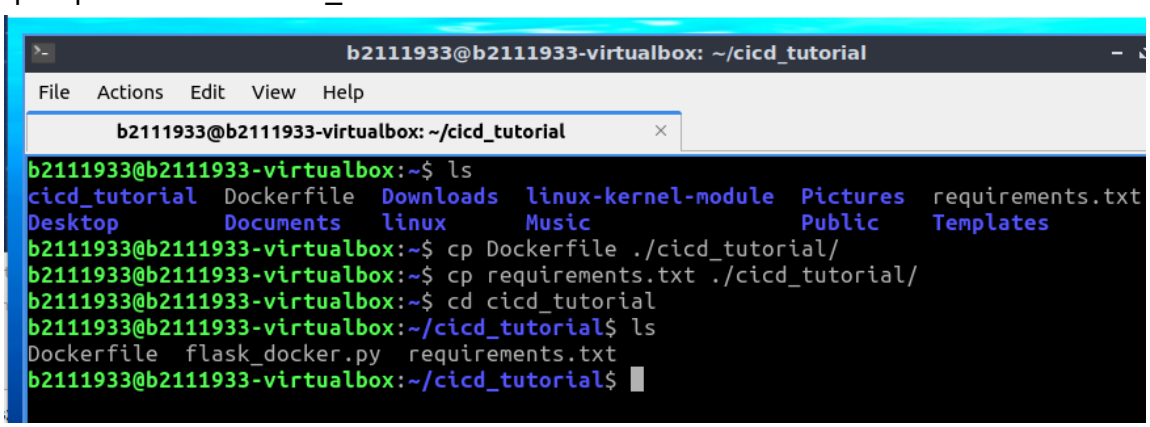
Dockerfile

```
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"]
```



The file will automatically do works on **Docker**

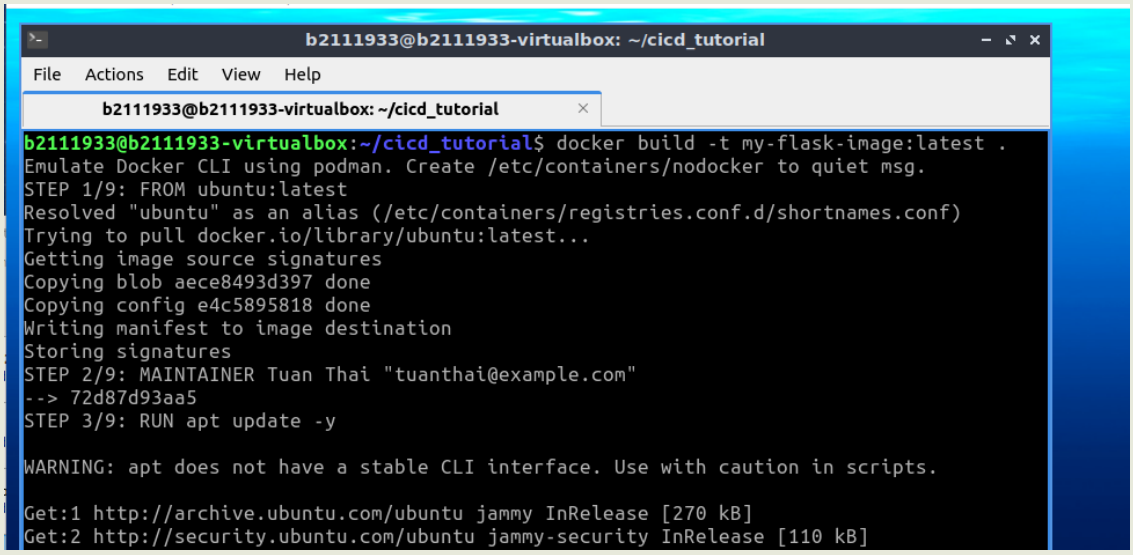
cp Dockerfile ./cicd_tutorial/
cp requirements.txt ./cicd_tutorial/

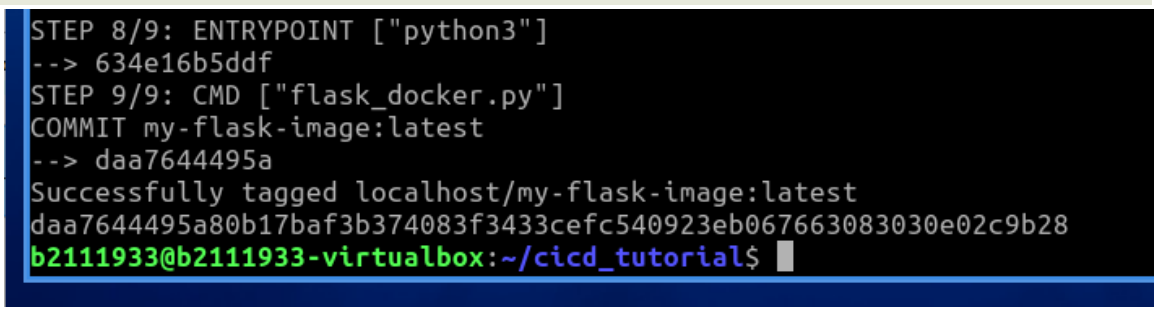


Copy those files to **Docker** file directory

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

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

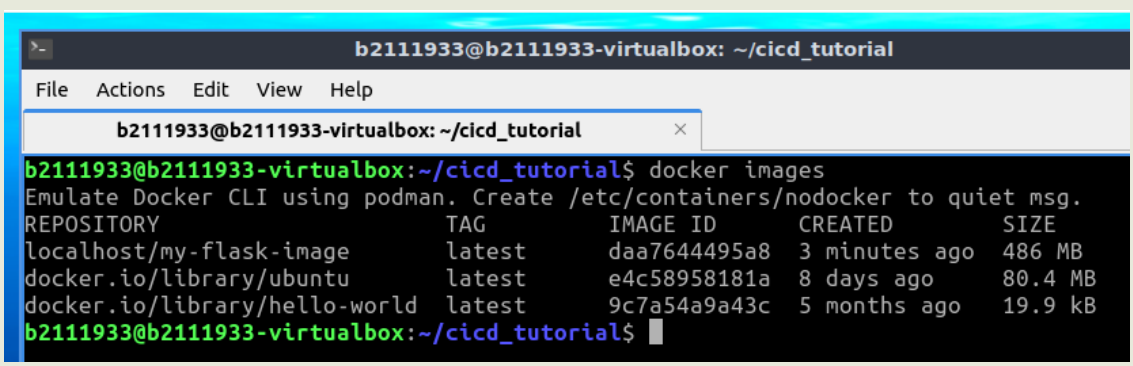




Create a **Docker image** with the name is **my-flask-image:latest**, using the **Dockerfile**

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

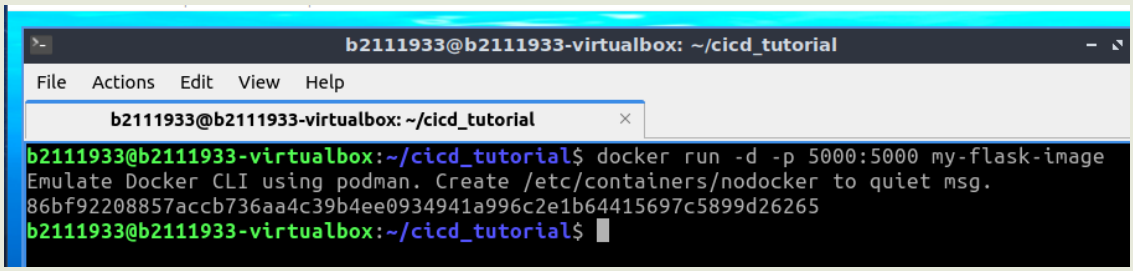
```
$docker images
```



Take a look at the **image** in **Docker**

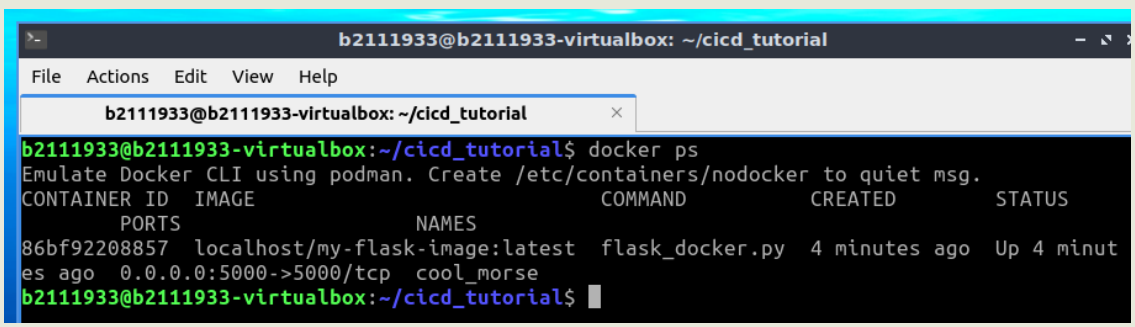
- Run your image (take a screenshot)

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



Run the image (**port 5000 of the image** will run on **port 5000 of the VM**)

```
$docker ps
```

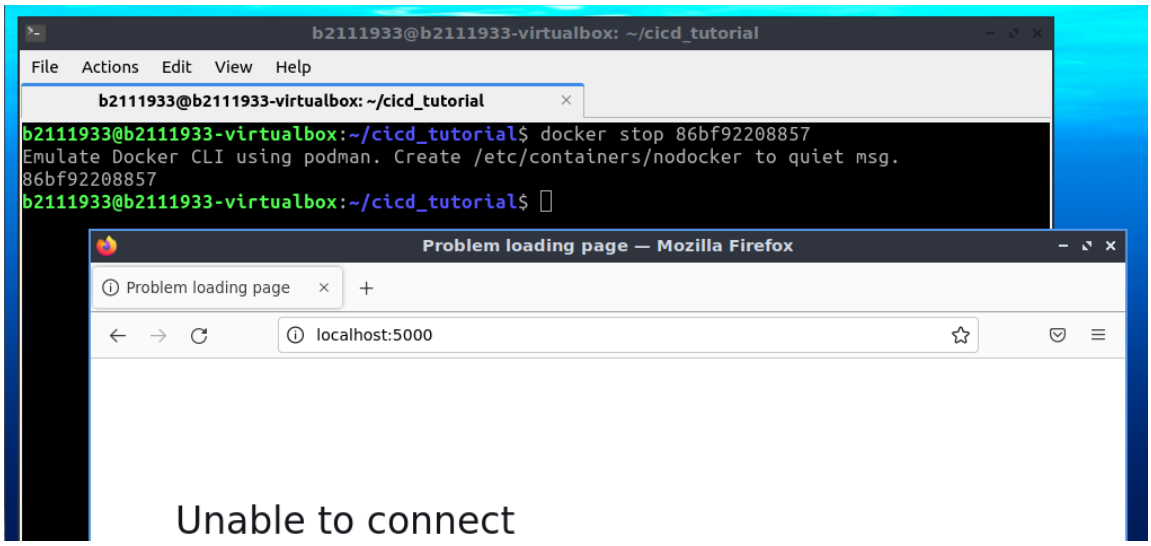


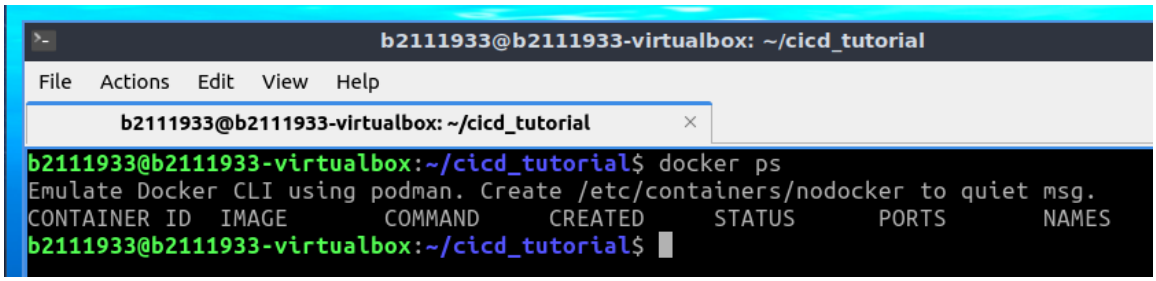
List running containers

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

Access the **Docker file** from **Mozilla Firefox** http://localhost:5000

Let's try to stop the container:



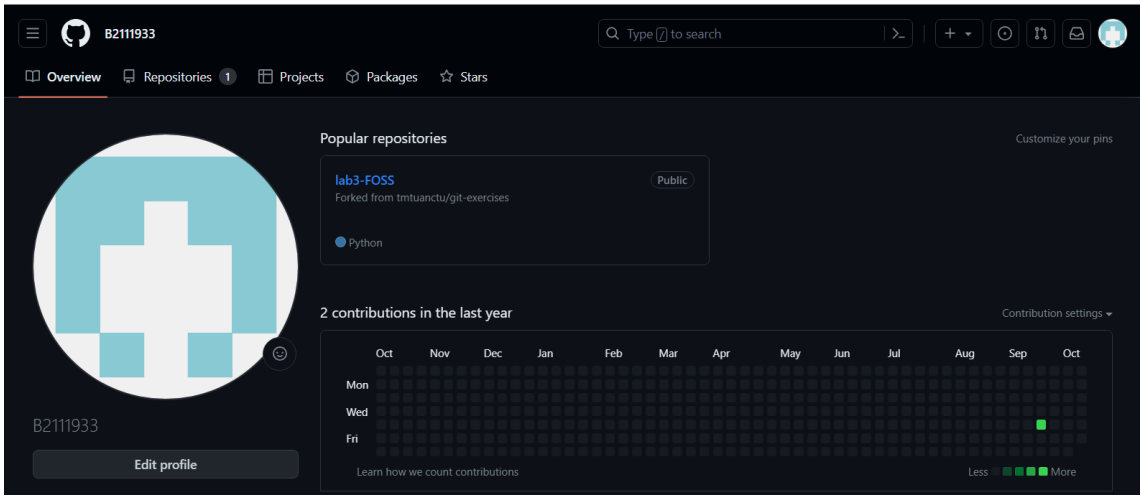


After stop the container

2. Automatically dockerize a Flask project using Jenkins

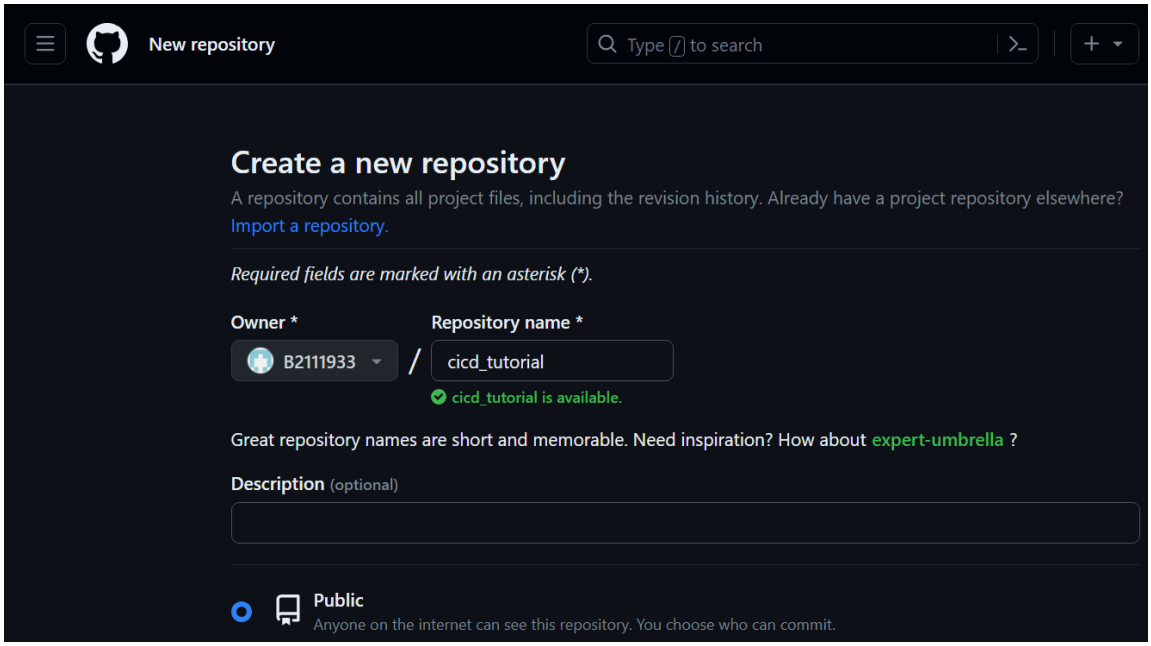
2.1. Push your code to a Github repository

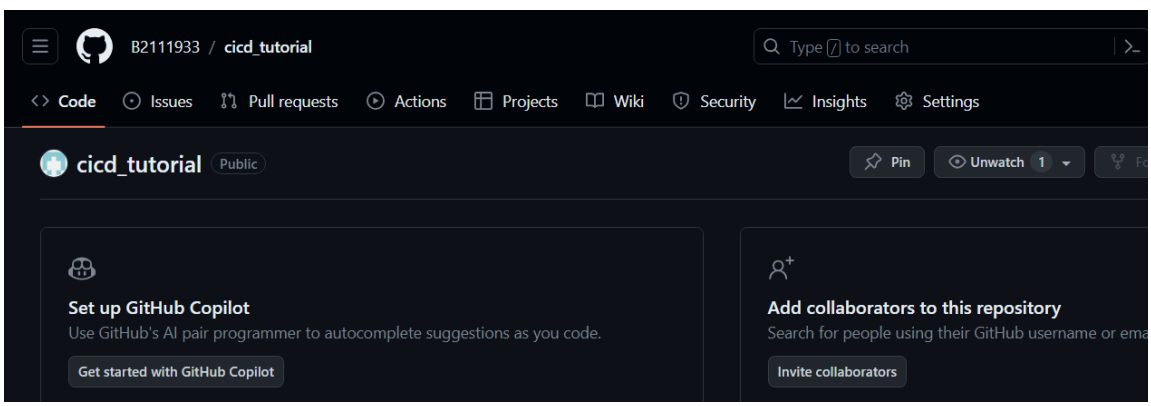
- Create an account (or login) to GitHub at https://github.com



My **GitHub** account

- Create a new repository, name it as "cicd_tutorial". Get the repository URL (for example: https://github.com/TuanThai/cicd_tutorial.git)

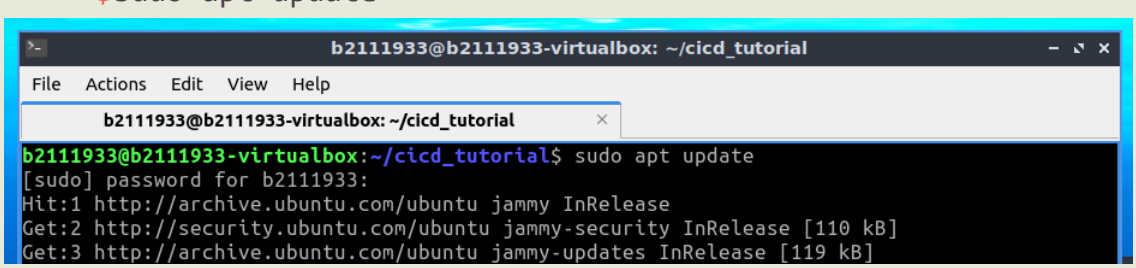




Create a new repository named "**cicd_tutorial**". URL: https://github.com/B2111933/cicd_tutorial

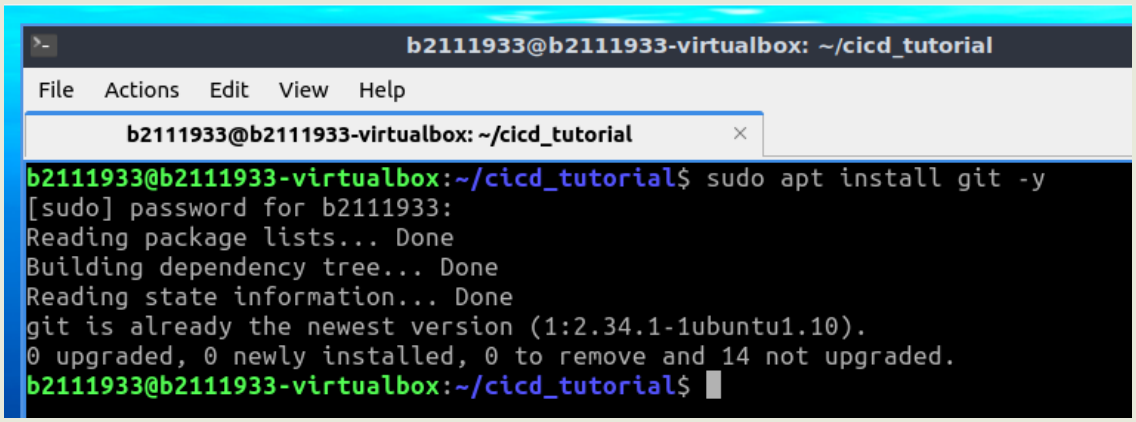
- Install and setup git on your computer (remember to set your name/email)

```
$sudo apt update
```



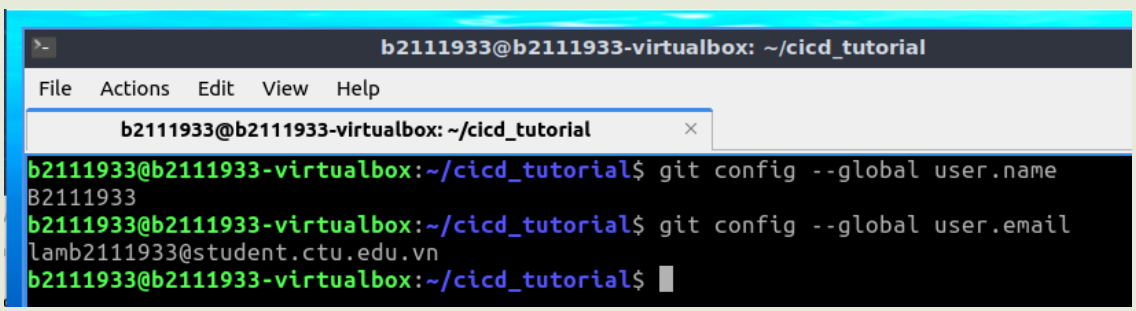
Update packages index

```
$sudo apt install git -y
```



git is already installed in **lab 3**

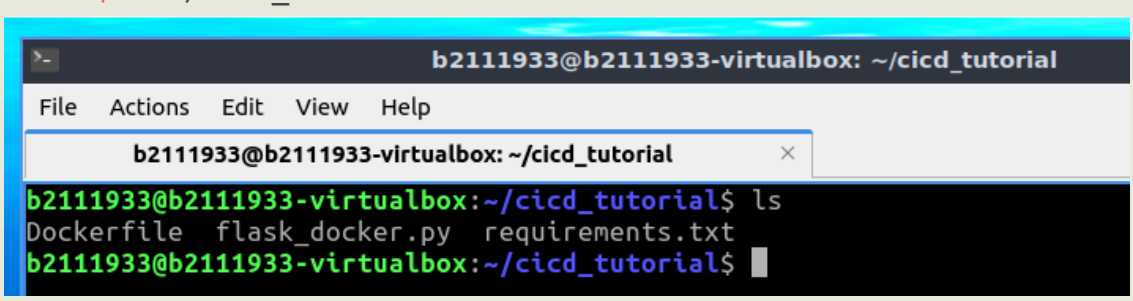
```
$git config --global user.name "Firstname Lastname"
$git config --global user.email "example@.ctu.edu.vn"
```



The **username** and **email** was configured in **lab 3** too

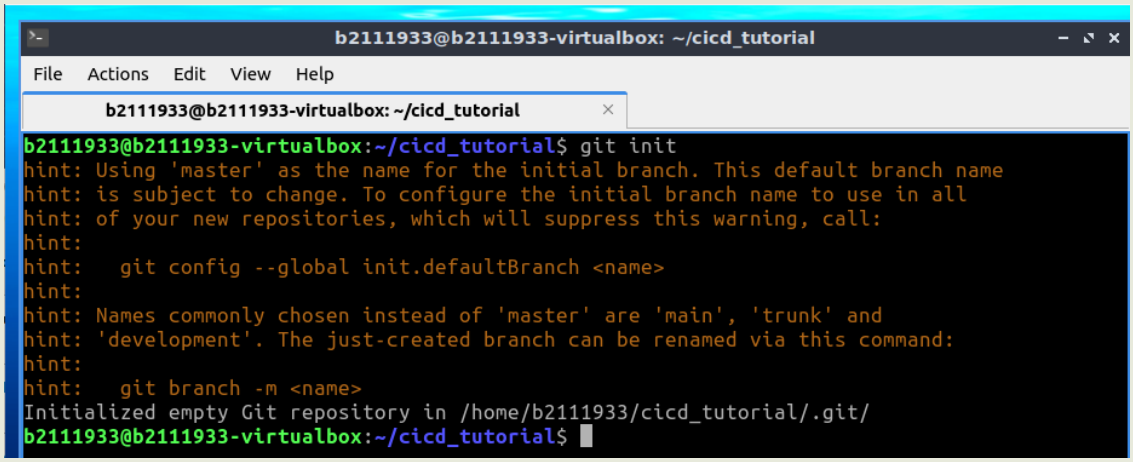
- Initialize git, commit and push your flask project files to Github

```
$mv ~/cicd_tutorial
```



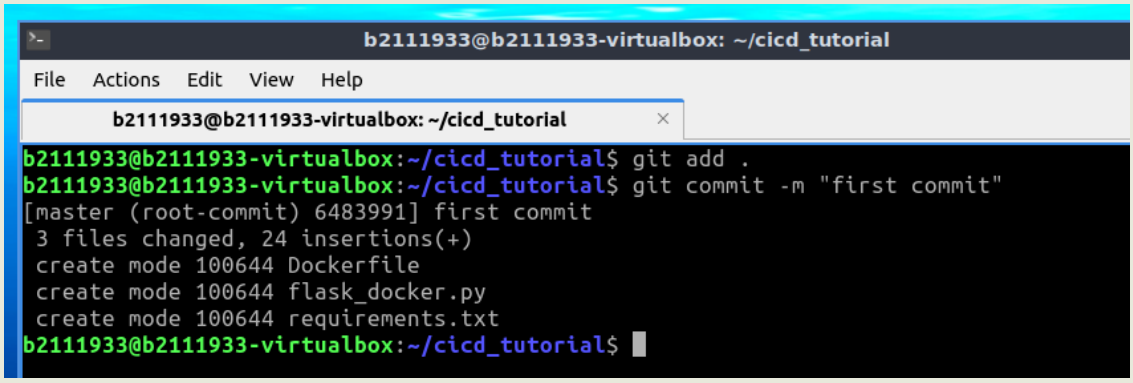
We are already in **~/cicd_tutorial**

```
$git init
```



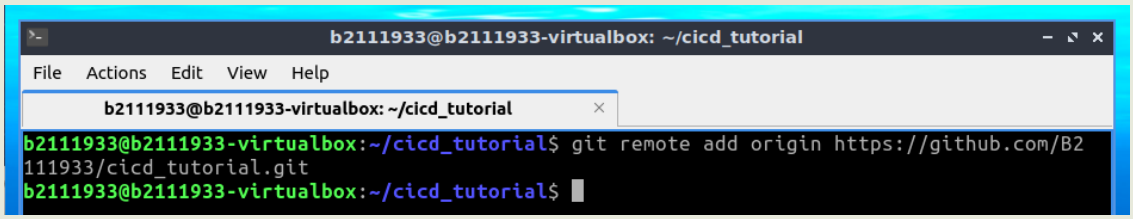
Create a new **Git** repository

```
$git add .
$git commit -m "first commit"
```



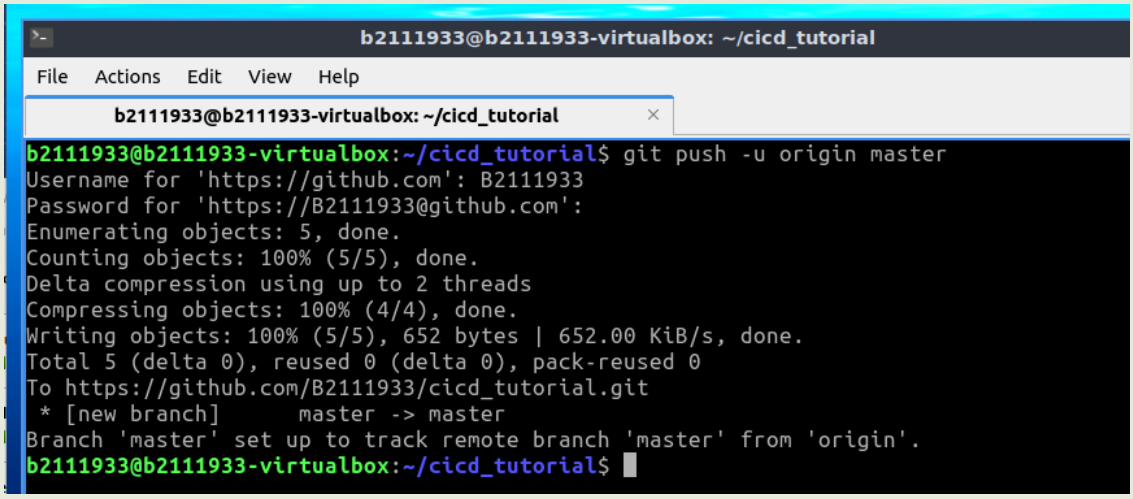
Add and commit all files from the current working repository

```
$git remote add origin <your repository URL>
```

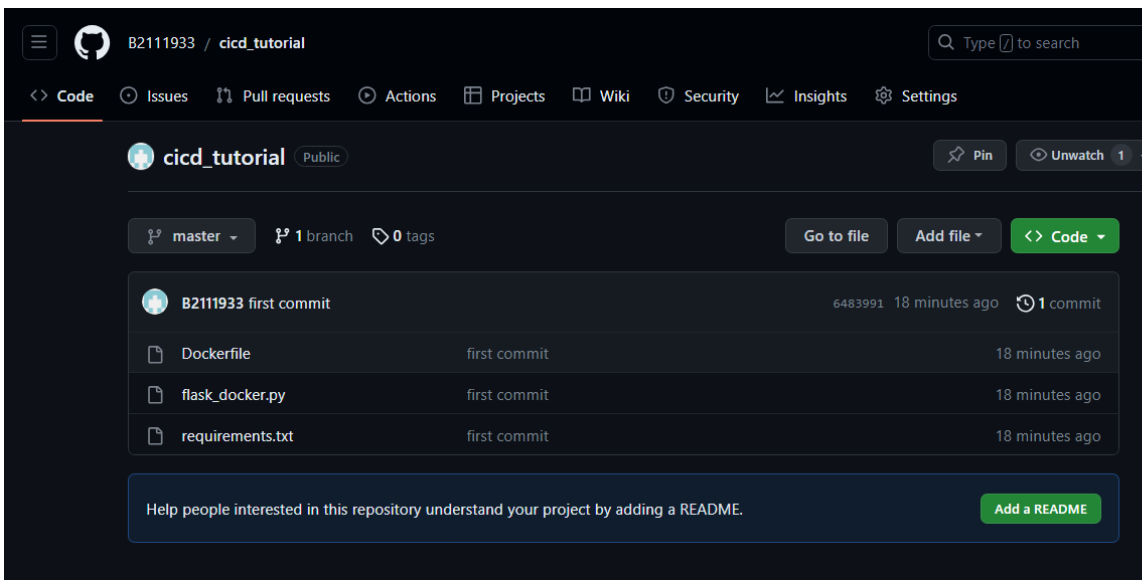


Add the **git** URL: https://github.com/B2111933/cicd_tutorial to our VM

```
$git push -u origin master
```



Push **flask** project files to **Github**

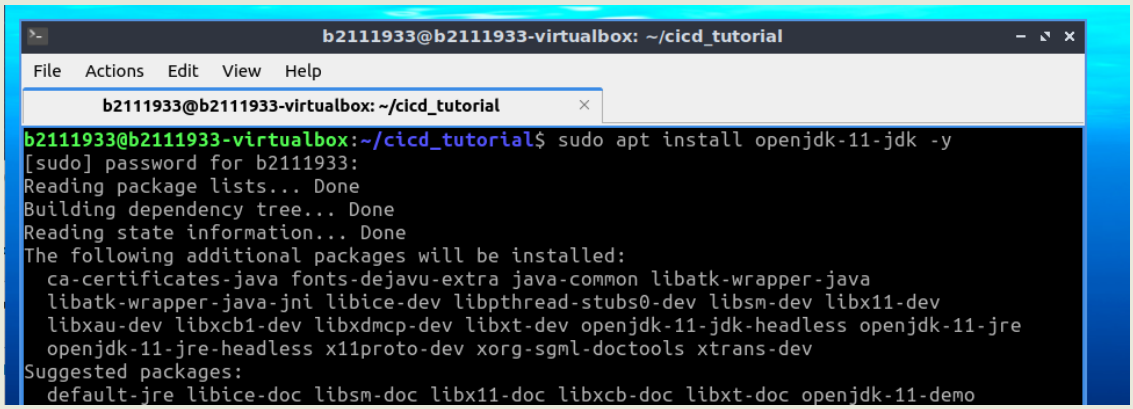


Check the result

2.2. Install and configure Jenkins

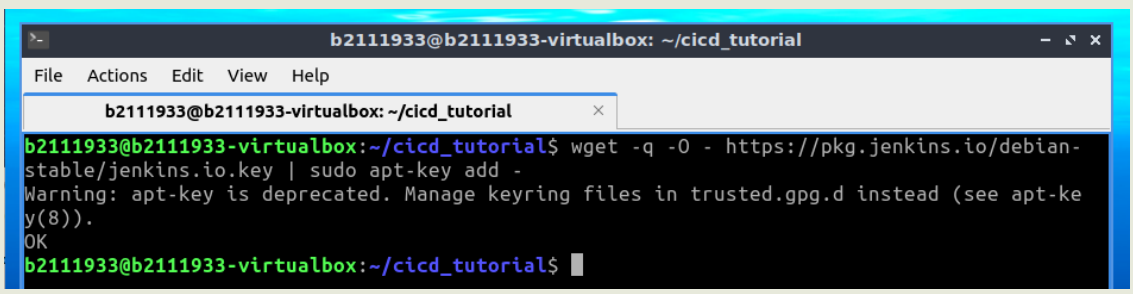
- Install Java and Jenkins

```
$sudo apt install openjdk-11-jdk -y
```



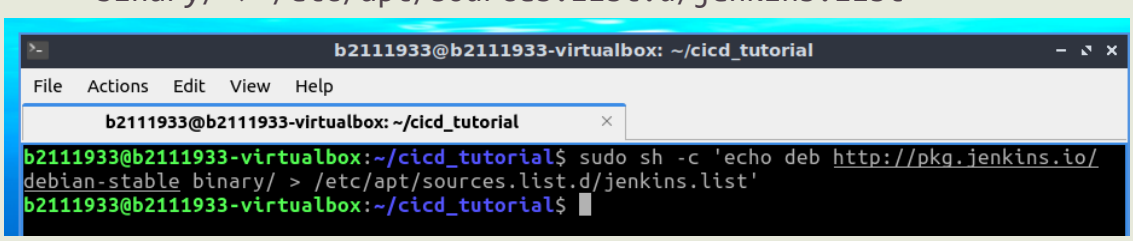
Install **Java 11**

```
$wget -q -O -
https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo
apt-key add -
```



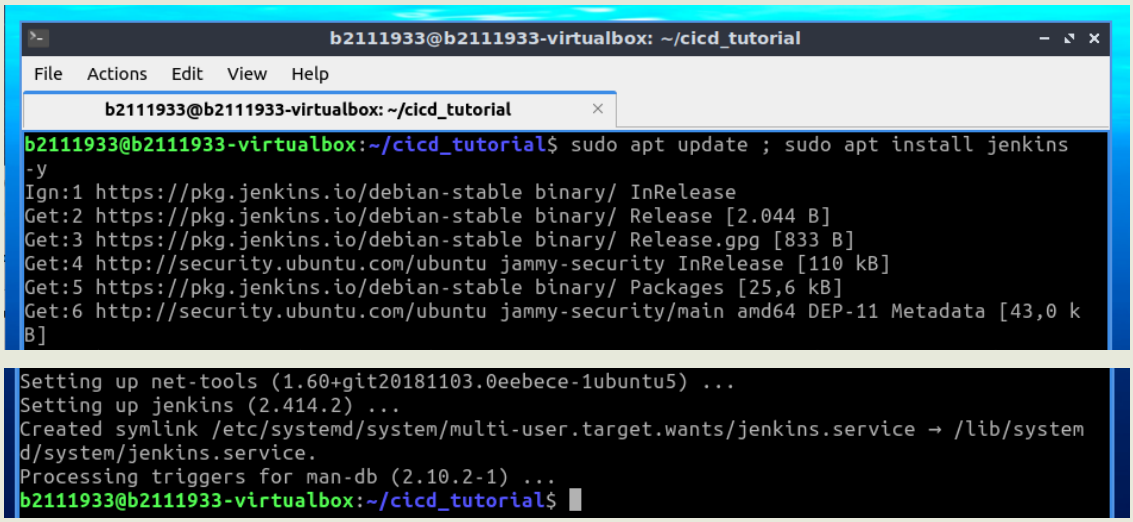
It's warning **apt-key** is deprecated, but the following part still runs for some reason

```
$sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable
binary/ > /etc/apt/sources.list.d/jenkins.list'
```



Update the download link to **apt**

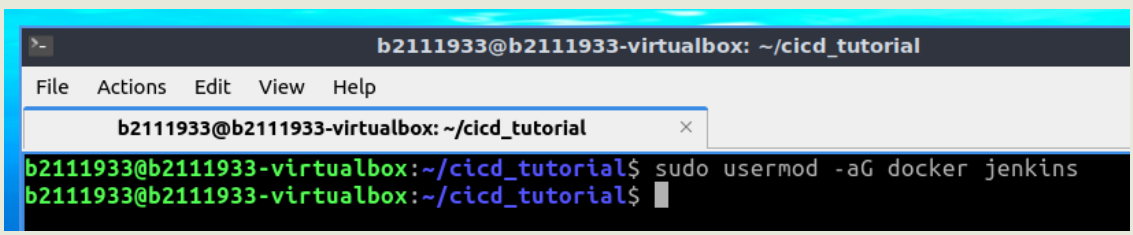
```
$sudo apt update ; sudo apt install jenkins -y
```



Update apt index and install **Jenkins**

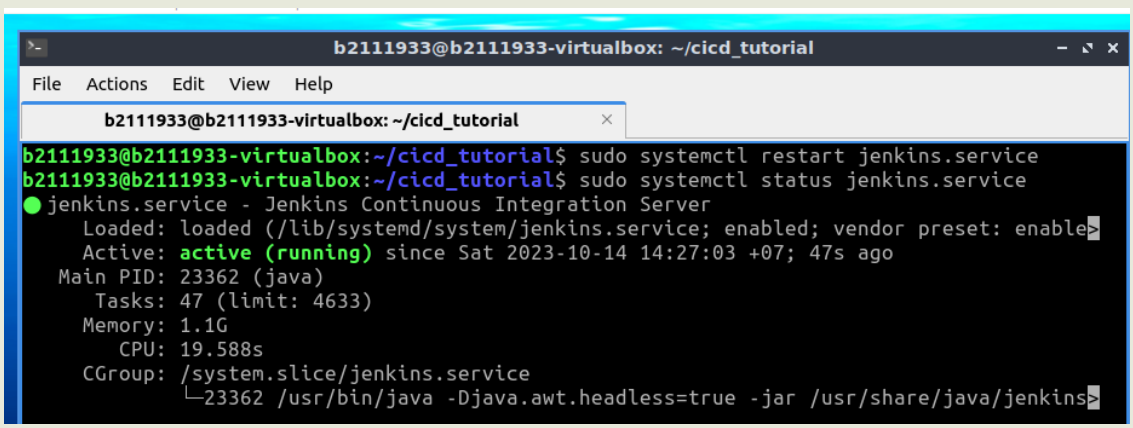
- Launch Jenkins

```
$sudo usermod -aG docker jenkins
```



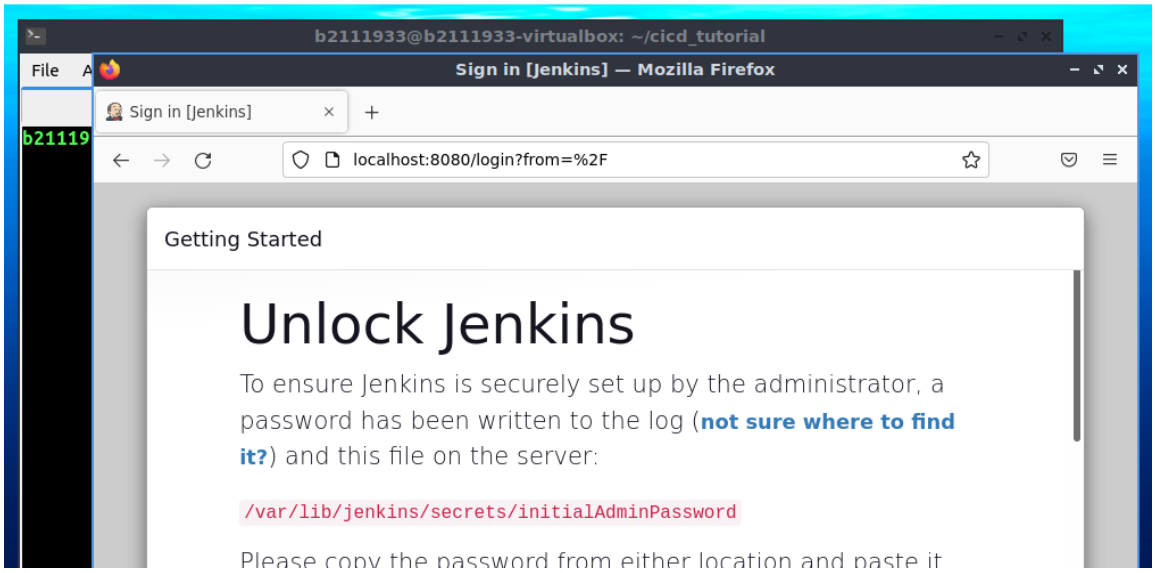
Add user to groups: **docker** & **jenkins** (because **jenkins** will be used to dockerize files)

```
$sudo systemctl restart jenkins.service
```

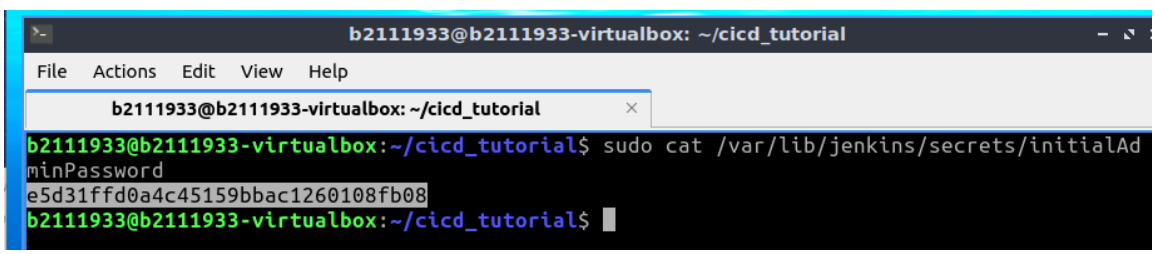


Restart **Jenkins** service and check its status

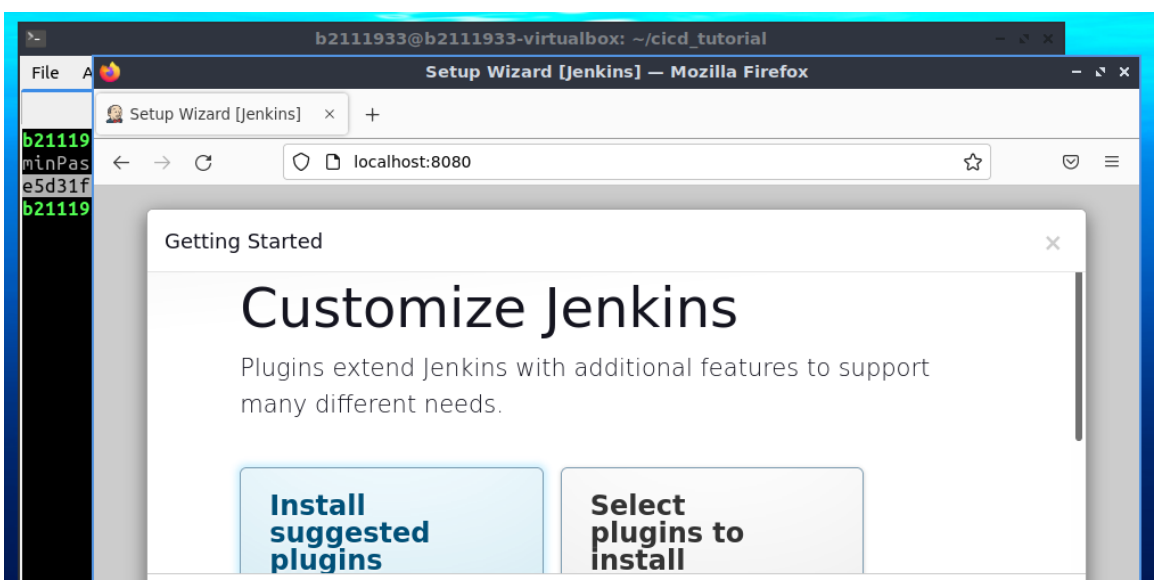
- Access Jenkins using a web browser (http://localhost:8080). Unlock Jenkins, install suggested plugins, create the first admin user.



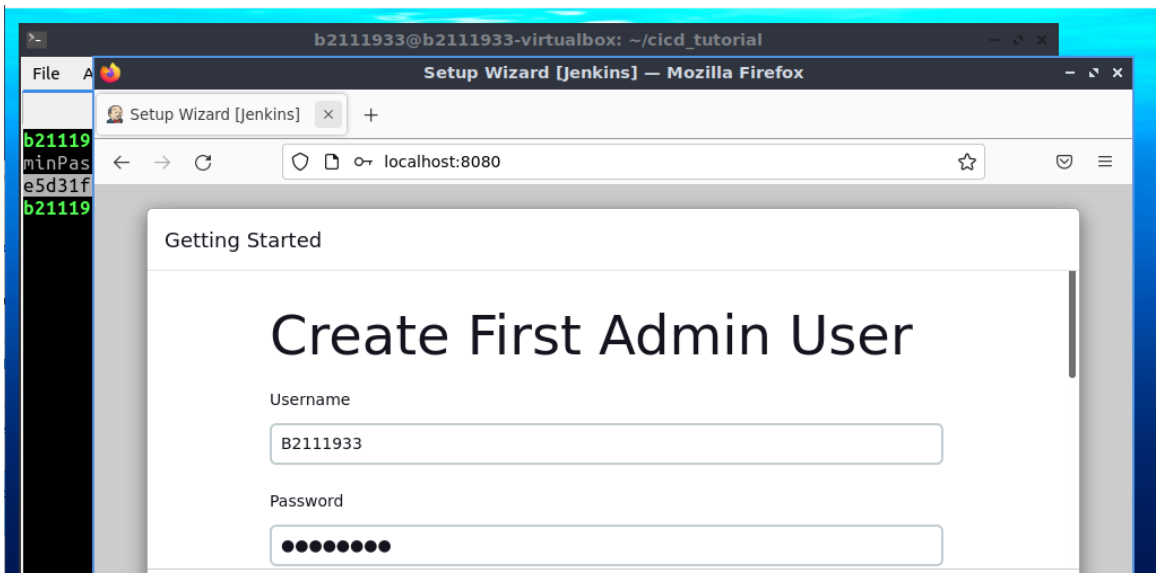
Access Jenkins via http://localhost:8080



Find the password that **Jenkins** request



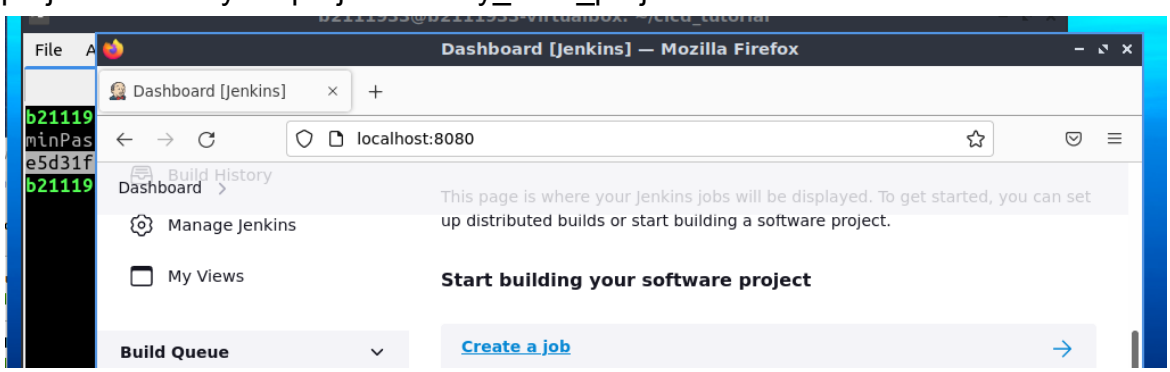
Install suggested plugins

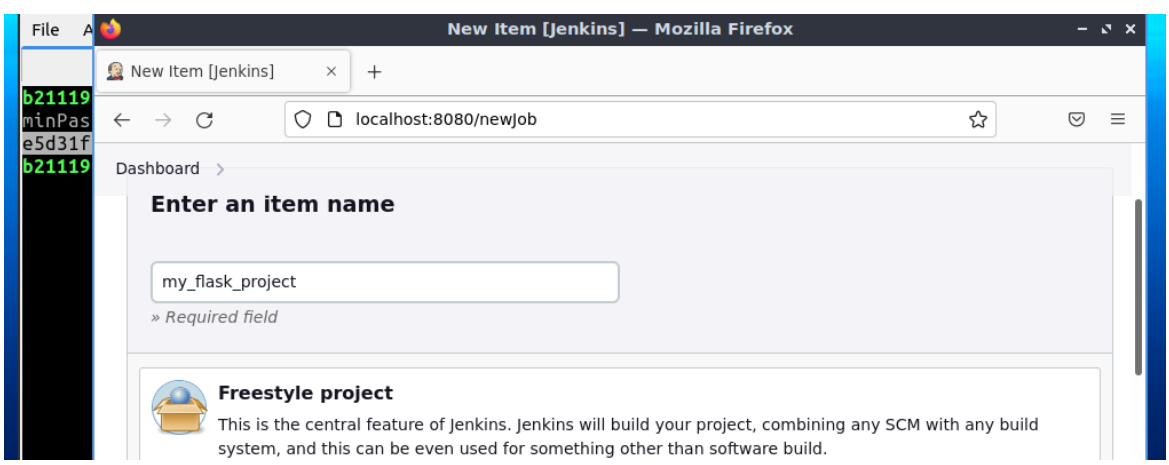


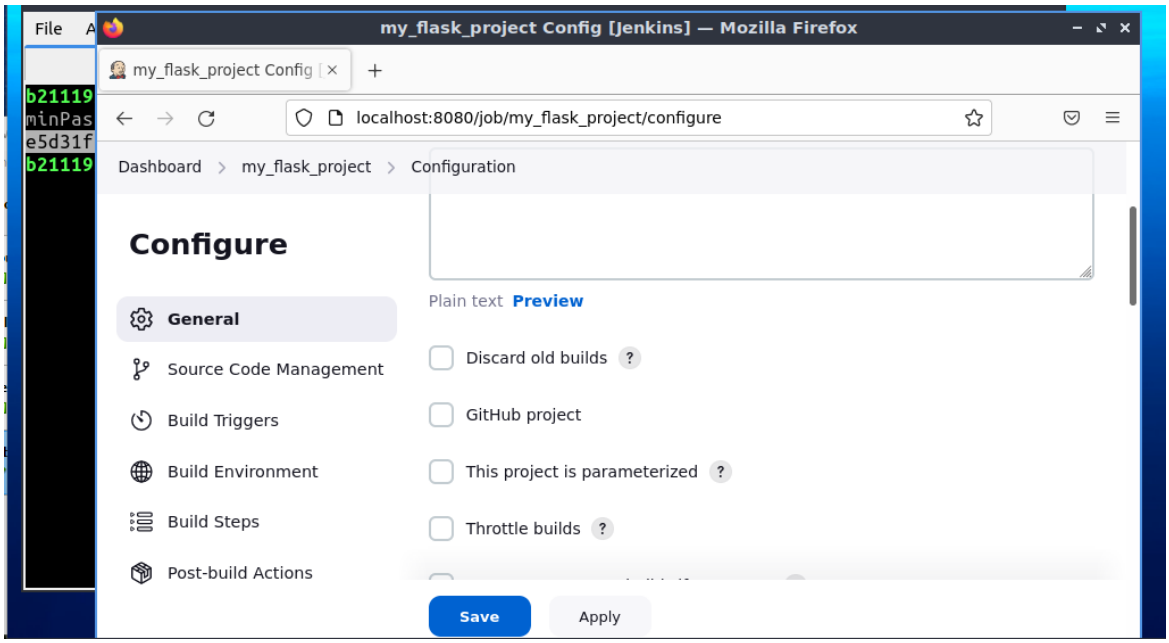
After installing plugins, we create first admin user

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"

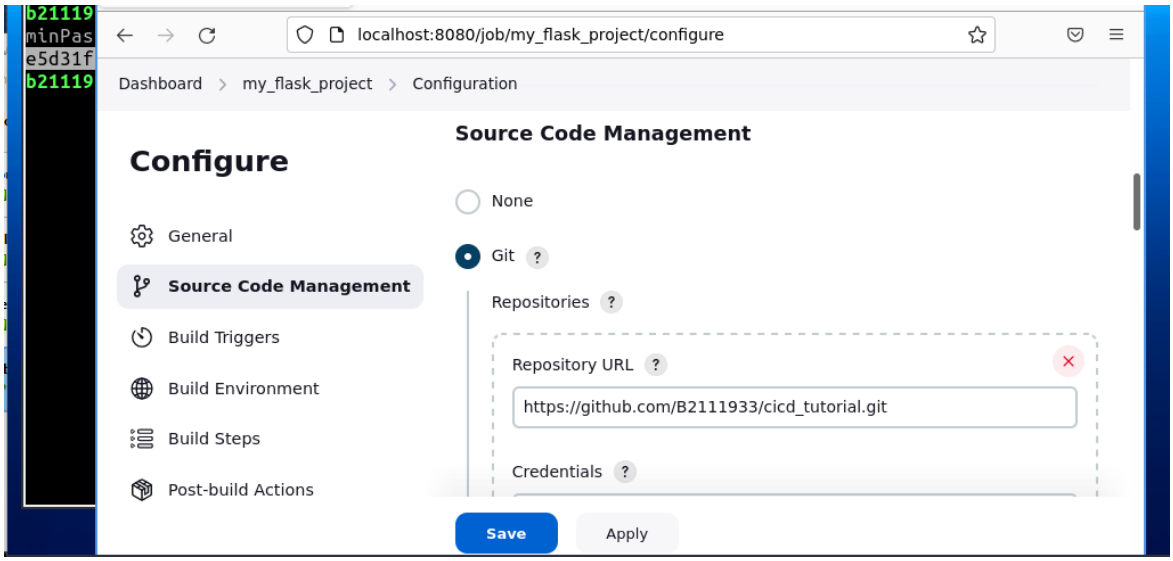






On Jenkins, create a project named "**my_flask_project**"

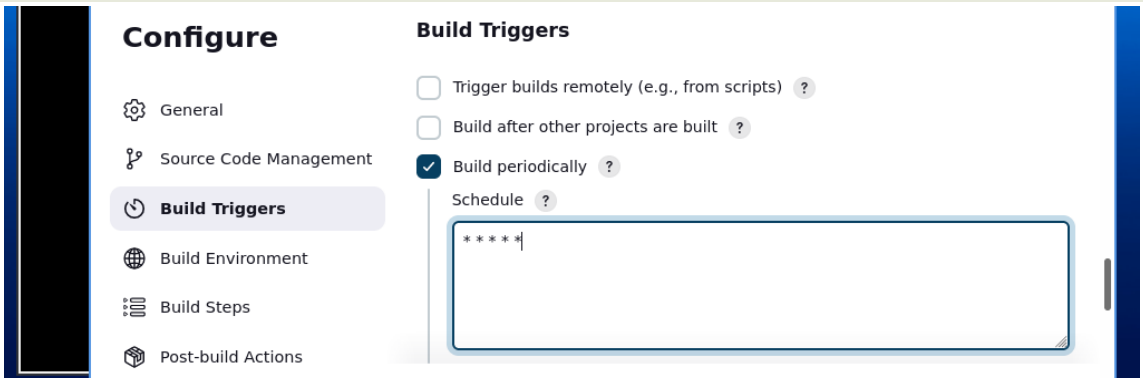
- Under "Source Code Management" choose "Git", fill in your GitHub repository URL



Configure the source code

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

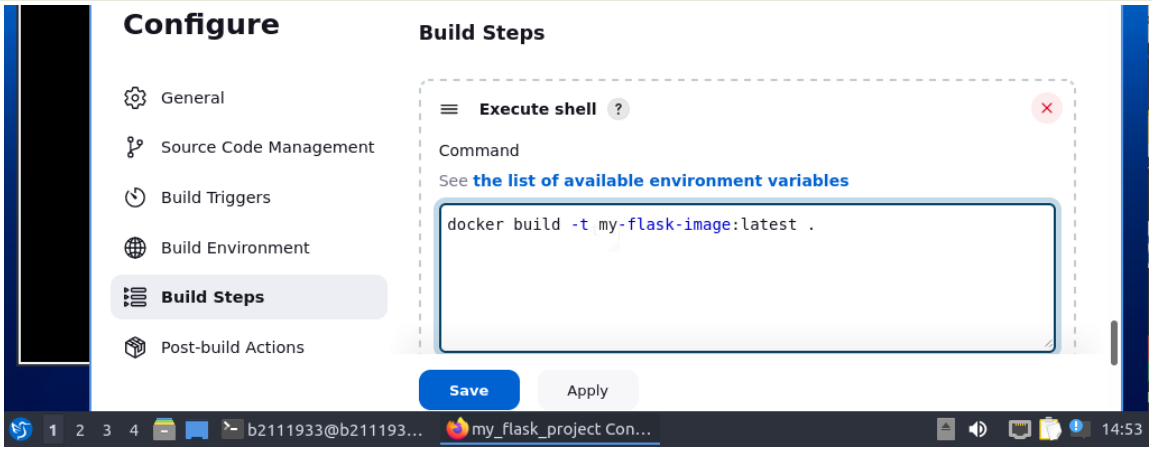
```
* * * * *
```



Select "**Build periodically**", fill in "*** * * * ***". It will build your project every minute

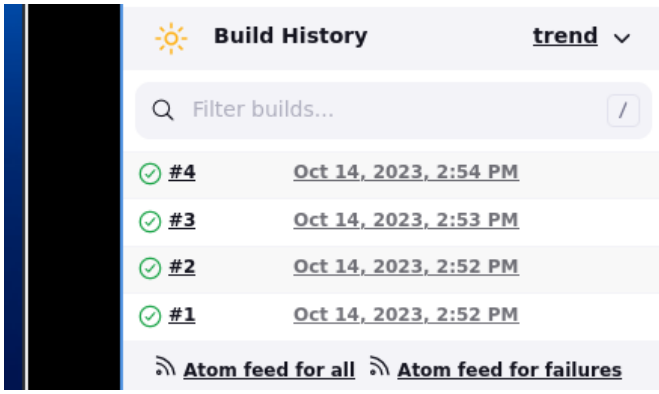
- Under "Build" we will "Add build step", and select "Execute shell". Then fill in "docker build -t my-flask-image:latest ."

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



Configure **Jenkins** to build by **execute shell**

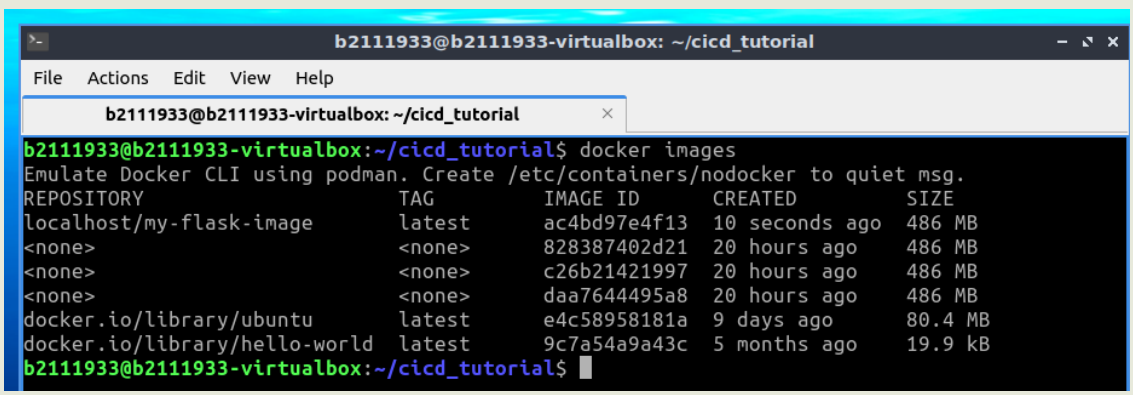
- Save your project. Then look at "Build history" to see that your project is built every minute.



We can see that our project is built **every minute**

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

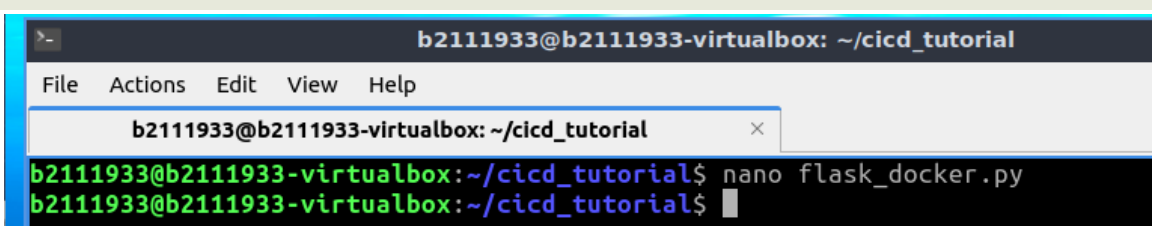
```
$docker images
```



Check for the image in **Docker**

- Modify your Flask application:

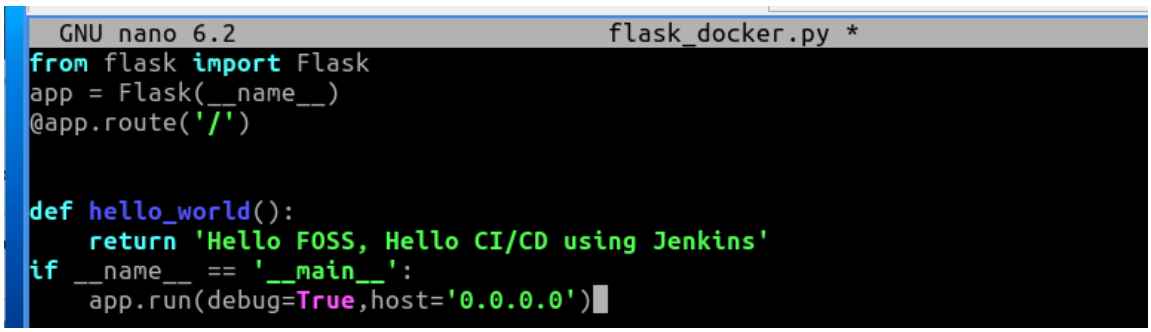
```
$nano flask_docker.py
```



Modify **Flask** application with **nano**

```
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')
```

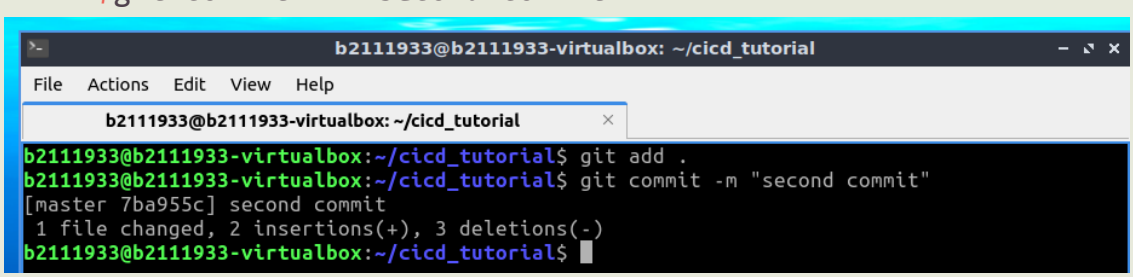
The contents of file:



Modify the contents of file so the website will display: **'Hello FOSS, Hello CI/CD using Jenkins'**

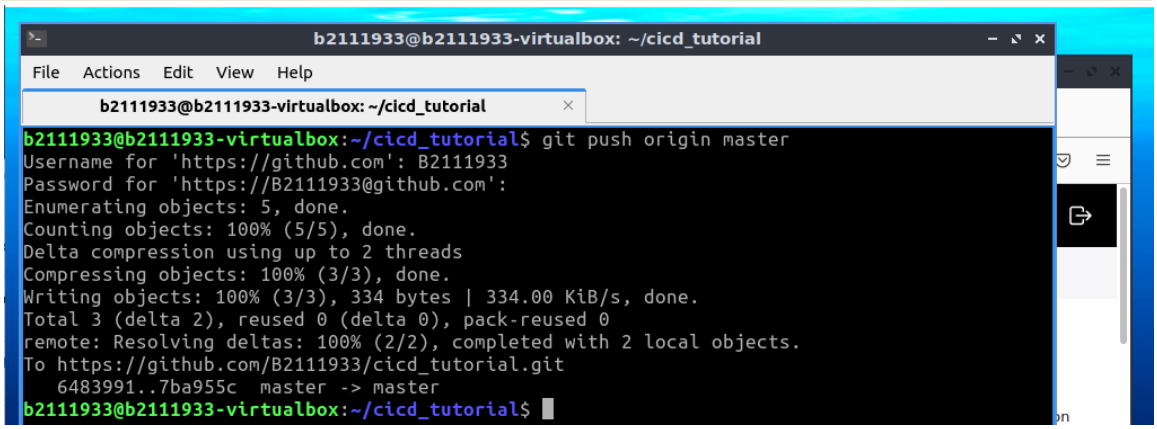
- Commit and push your project files to GitHub

```
$git add .
$git commit -m "second commit"
```

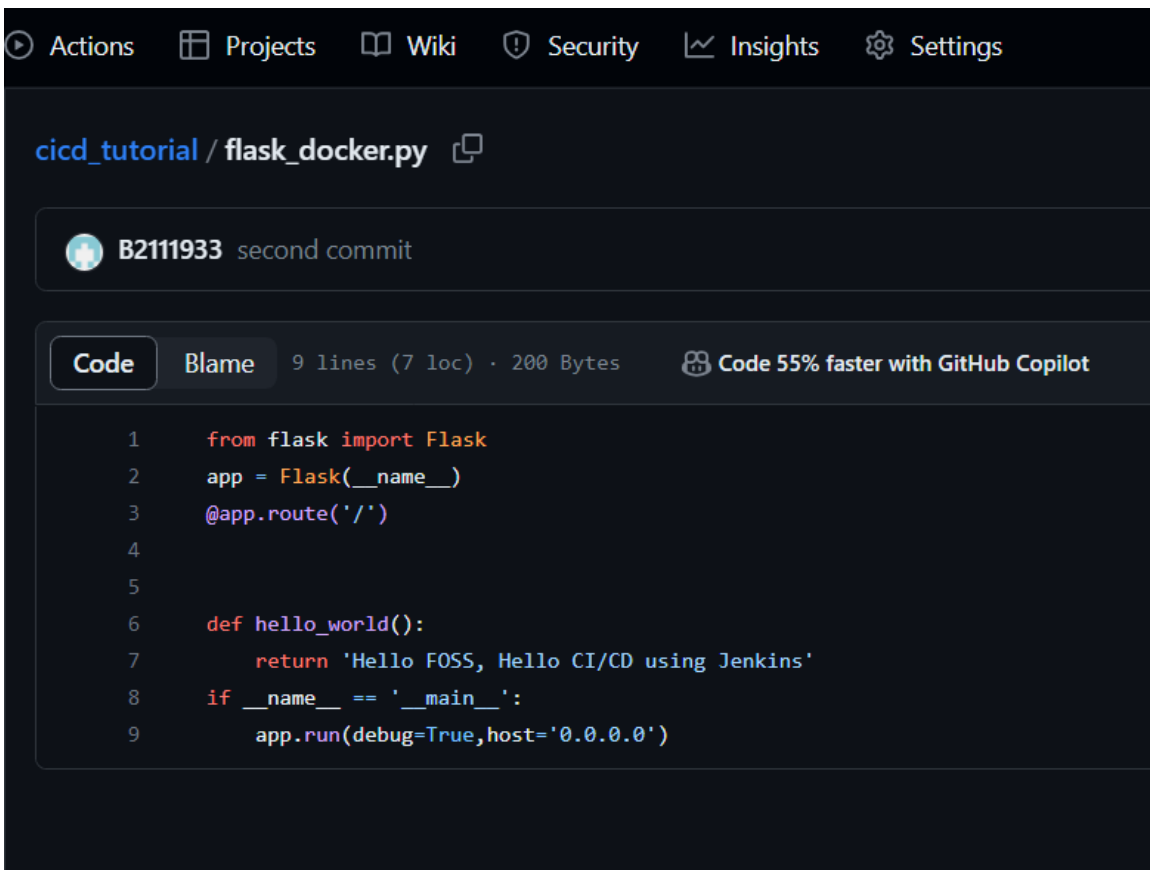


Add and commit all files from the current working repository

```
$git push origin master
```



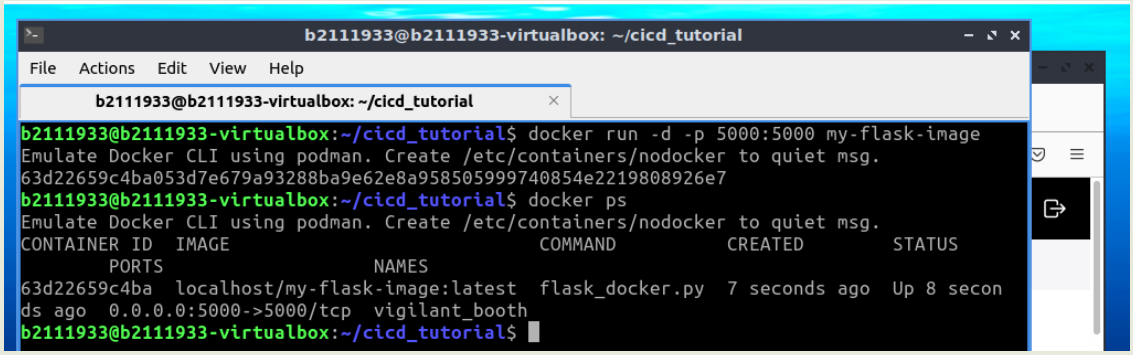
Push the image to **Github**



Check the result

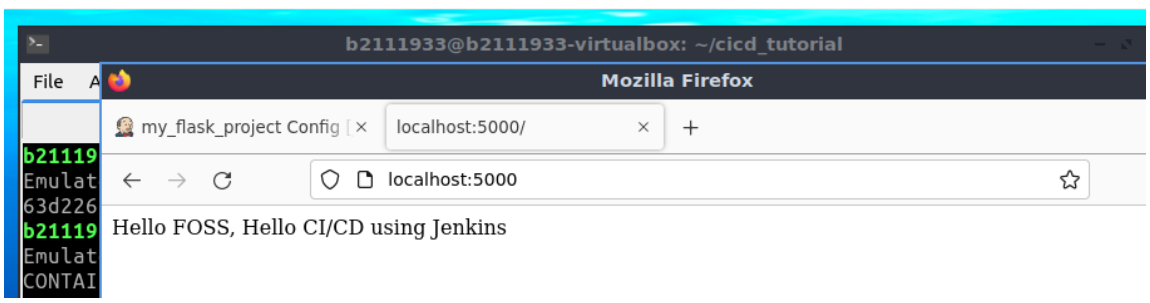
- Wait 1 minute, then run your image

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



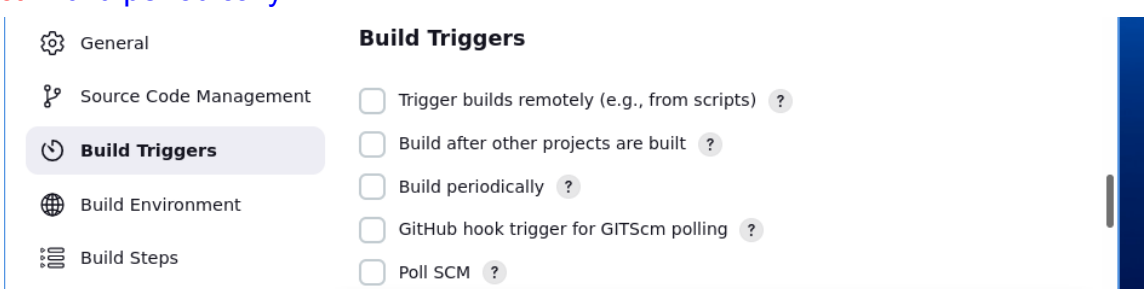
Run the image and check its status

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



Access the application via **http://localhost:5000**

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



Uncheck the **"Build periodically"** to avoid wasting resources

---END---