Docker day 2

In day 1. saw about official docker image. In this see about **custom Docker image**

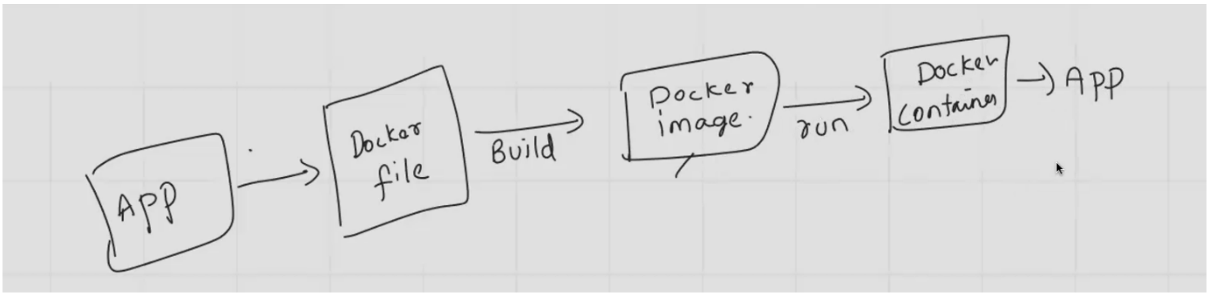
nginx was a running application. Apache 2 was a running application. So these organization they have, do authorize those application, and they have pulled those images. They have pushed those images to the doctor. Hub, registry. Same thing done by ourself. That is **Custom Docker image.**

2. Custom Docker image:

we will be creating our own docker images for our application. Okay, this is what it happens in the real time. So developer gives you an application in the Github Repository. You have to take the application, and you have to convert that application into a docker image. So we that is called as custom Docker image.

Same thing we are going to do here, we should have a running application for that running application. You have to write a docker file. so, to dockerize that application to do authorize this application. You have to write a doctor file. After writing the doctor file, you have to build this docker file. Only if you build this docker file, you'll be able to create the docker image.

Okay? So we now, so far, what we are done. We have used this docker image. We haven't created this docker image. We just use this docker image. So when you run this docker image, you will get the docker container . Okay, via Docker container, you will access your application.

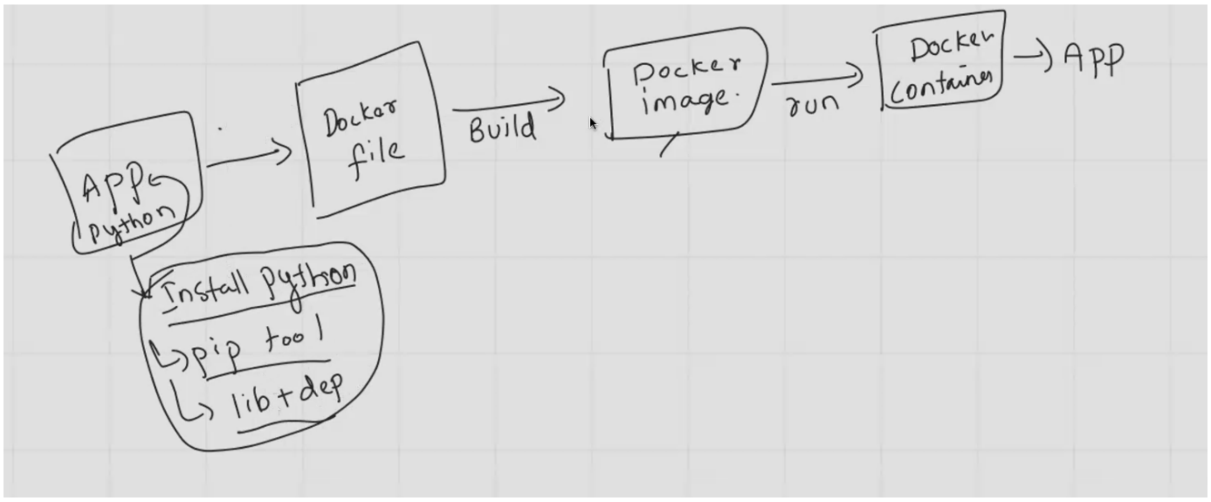


Run python application .okay, because a package management tool for python. So I have to install another tool for pip. Then I have to install all the libraries and the dependencies that are required to access this application. Okay? So now, I don't want to do anything in my ec2.

Okay. So if I have to run this application, and tomorrow I just terminate this Ec 2 and I start the new Ec. 2 .

What I have to do I have to against. I have to again install pip. Then I have to install pip, and I have to install. I have to launch all the tools that I have done in the previous day. I have to install everything in my new Ec2 also, so I don't want to set up this environment again and again.

To avoid this, we use docker.



So what I'll do? I'm just going to package this complete application along with this dependencies into a docker image. So you will be writing a docker file so that this docker image will contain the application and the necessary dependencies. So you have to write the docker file .so that the application will be packaged along with the libraries and the binaries. Okay?

So to write your docker file, you have to write some instructions. So docker file is a normal text file. Okay, which contains multiple instructions.

Developer gives

1. .json file →it denotes write docker file for node.js
2. .txt file → it denotes write docker file for python
3. .jar file → it denotes write docker file for java
4. .xml file → it denotes write docker file for java.

   → But developer does not give .jar file. We can create .jar file using build tool [maven /gradel]

   1. Maven [pom.xml]
   2. Gradel [build.gradle]

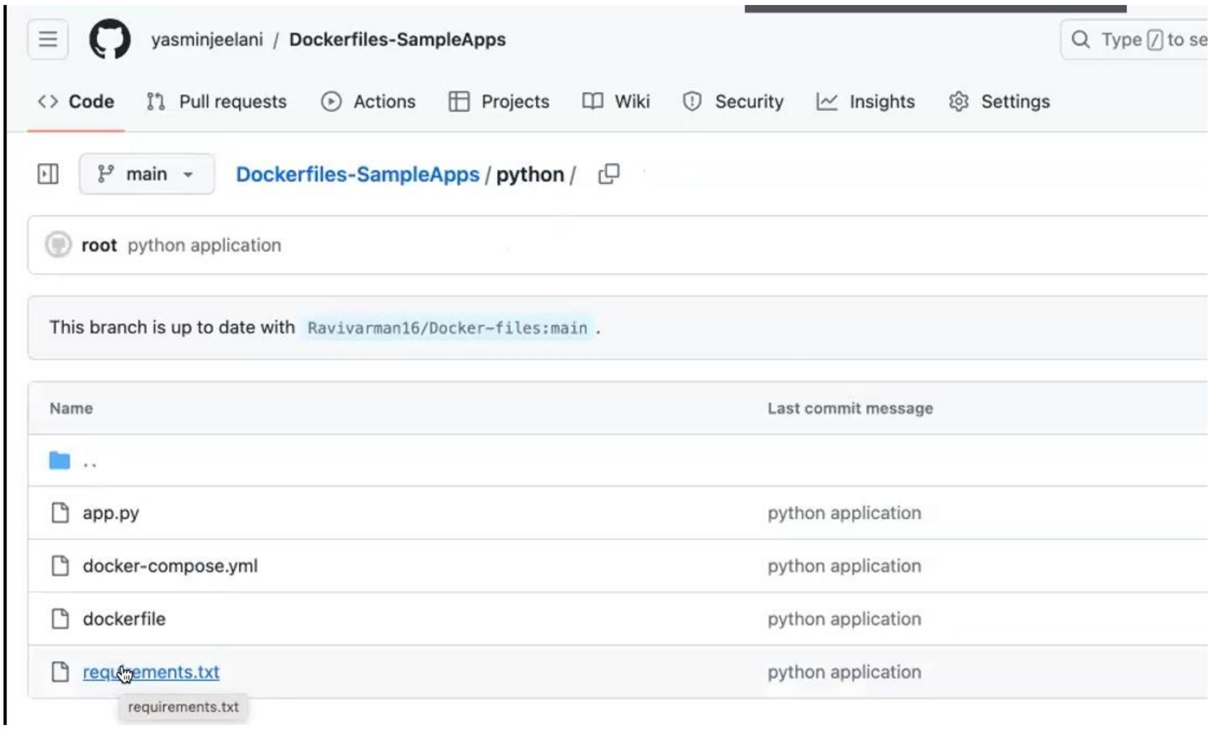
File format	Docker file	Package management tool
package. json	Node.js	Npm [Node Package Manager]
requirement.txt	python	pip
app.jar pom.xml	java	Maven [pom.xml] Gradel [build.gradle]

Suppose package. Json file [default file name] not given in github repo, it means it's an already build application. For run that application, we need web server [nginx or Apache]

docker file:

	Docker file
purpose	Defines how to build a single Docker image.
File type	Text file
syntax	FROM, RUN, COPY, etc
Content	It Contains instructions to set up an environment inside a container

Write docker file for python:



Dockerfiles-SampleApps / python /

root python application

This branch is up to date with Ravivarman16/Docker-files:main .

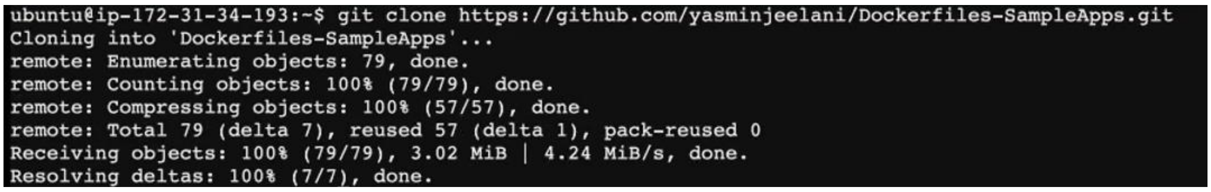
Name	Last commit message
..	
app.py	python application
docker-compose.yml	python application
dockerfile	python application
requirements.txt	python application

.txt file → it denotes write docker file for python

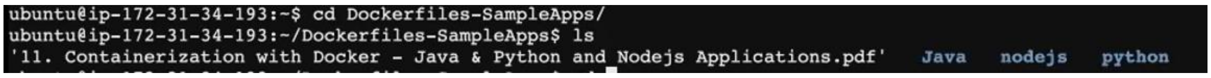
That .txt file inside this github repo

https://github.com/yasminjeelani/Dockerfiles-SampleApps before that you installed git in ec2.

clone that above repo in ec2.



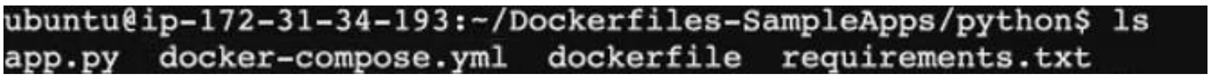
```
ubuntu@ip-172-31-34-193:~$ git clone https://github.com/yasminjeelani/Dockerfiles-SampleApps.git
Cloning into 'Dockerfiles-SampleApps'...
remote: Enumerating objects: 79, done.
remote: Counting objects: 100% (79/79), done.
remote: Compressing objects: 100% (57/57), done.
remote: Total 79 (delta 7), reused 57 (delta 1), pack-reused 0
Receiving objects: 100% (79/79), 3.02 MiB | 4.24 MiB/s, done.
Resolving deltas: 100% (7/7), done.
```



```
ubuntu@ip-172-31-34-193:~$ cd Dockerfiles-SampleApps/
ubuntu@ip-172-31-34-193:~/Dockerfiles-SampleApps$ ls
'11. Containerization with Docker - Java & Python and Nodejs Applications.pdf'   Java   nodejs   python
```

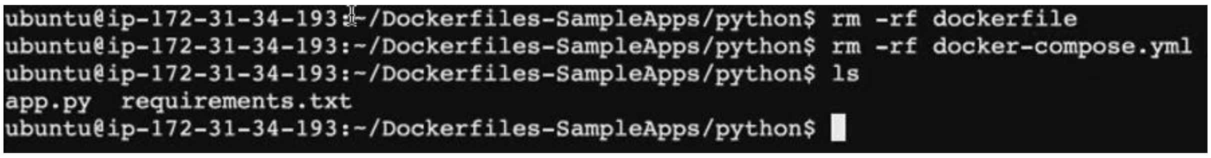
Get into python [cd python]

For this. u want only app.py and requirement.txt files. So u remove remaining 2 files.



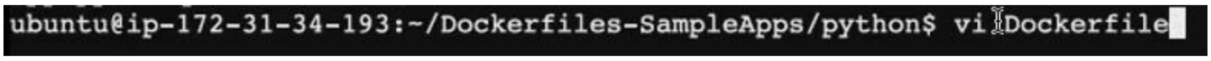
```
ubuntu@ip-172-31-34-193:~/Dockerfiles-SampleApps/python$ ls
app.py  docker-compose.yml  dockerfile  requirements.txt
```

u remove remaining 2 files

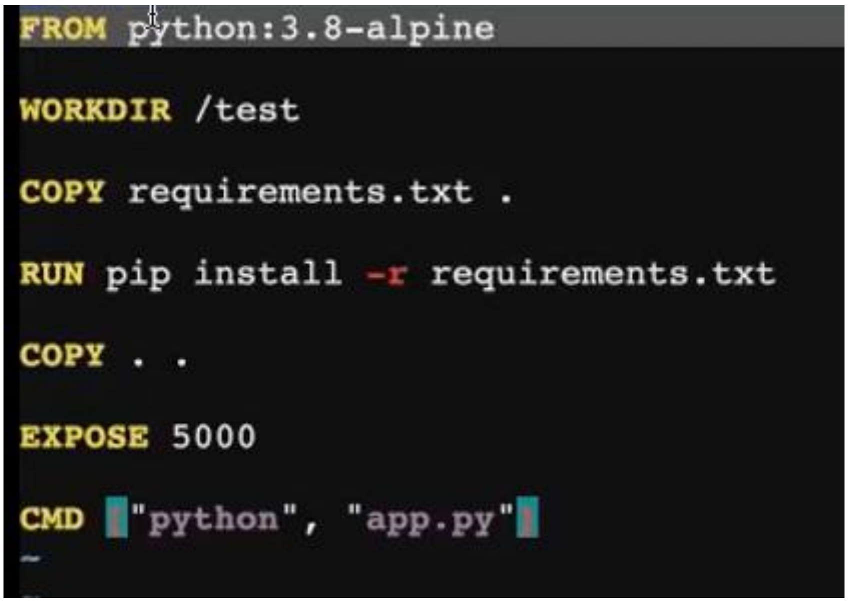


```
ubuntu@ip-172-31-34-193:~/Dockerfiles-SampleApps/python$ rm -rf dockerfile
ubuntu@ip-172-31-34-193:~/Dockerfiles-SampleApps/python$ rm -rf docker-compose.yml
ubuntu@ip-172-31-34-193:~/Dockerfiles-SampleApps/python$ ls
app.py  requirements.txt
ubuntu@ip-172-31-34-193:~/Dockerfiles-SampleApps/python$ 
```

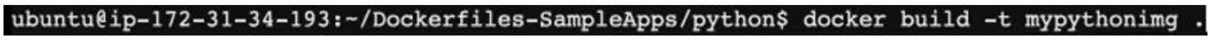
Write docker file for python:



```
ubuntu@ip-172-31-34-193:~/Dockerfiles-SampleApps/python$ vi Dockerfile
```

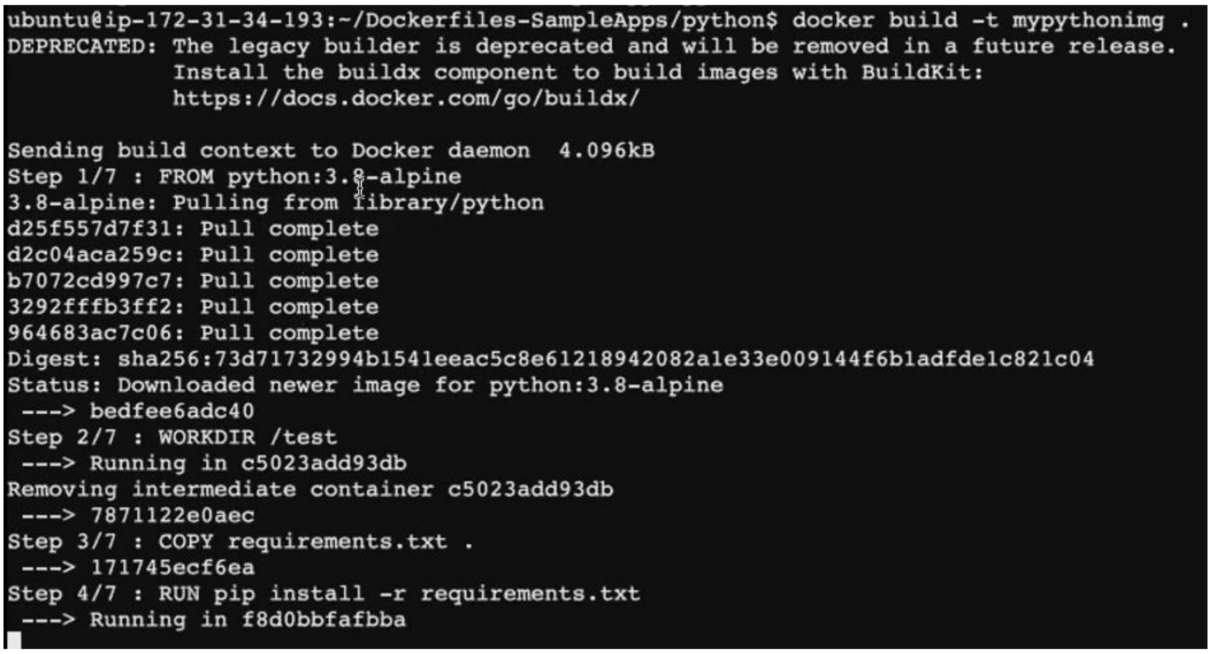


```
FROM python:3.8-alpine

WORKDIR /test

COPY requirements.txt .

RUN pip install -r requirements.txt

COPY . .

EXPOSE 5000

CMD ["python", "app.py"]
~
```

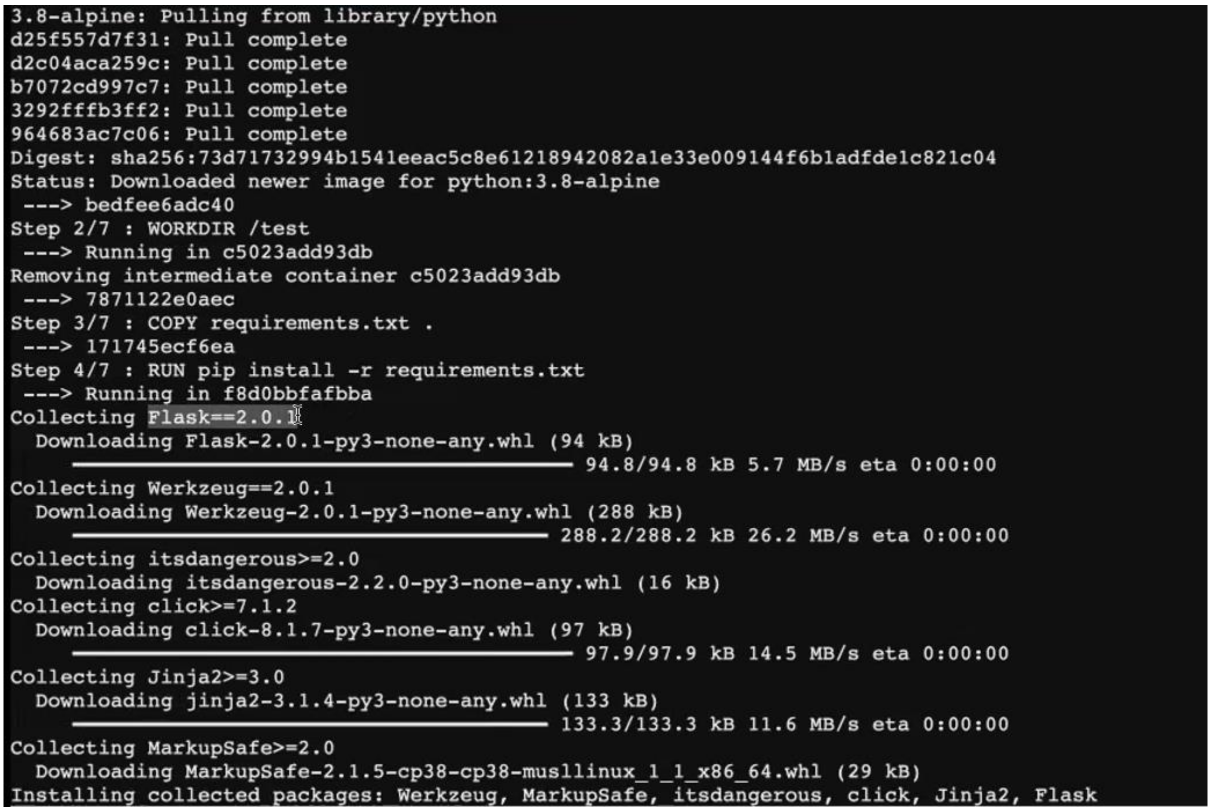


```
ubuntu@ip-172-31-34-193:~/Dockerfiles-SampleApps/python$ docker build -t mypythonimg .
```

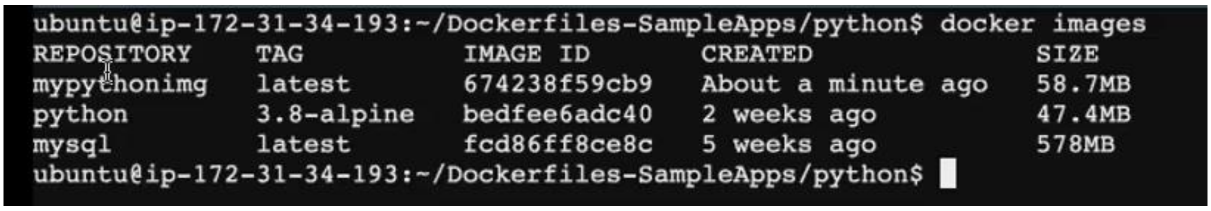
-t tagging docker image name as mypythonimg .



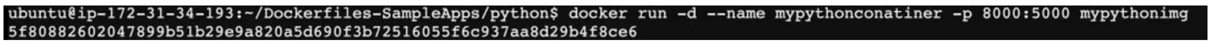
```
ubuntu@ip-172-31-34-193:~/Dockerfiles-SampleApps/python$ docker build -t mypythonimg .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  4.096kB
Step 1/7 : FROM python:3.8-alpine
3.8-alpine: Pulling from library/python
d25f557d7f31: Pull complete
d2c04aca259c: Pull complete
b7072cd997c7: Pull complete
3292fffb3ff2: Pull complete
964683ac7c06: Pull complete
Digest: sha256:73d71732994b1541eeac5c8e61218942082a1e33e009144f6b1adfde1c821c04
Status: Downloaded newer image for python:3.8-alpine
 ---> bedfee6adc40
Step 2/7 : WORKDIR /test
 ---> Running in c5023add93db
Removing intermediate container c5023add93db
 ---> 7871122e0aec
Step 3/7 : COPY requirements.txt .
 ---> 171745ecf6ea
Step 4/7 : RUN pip install -r requirements.txt
 ---> Running in f8d0bbfafbba
```



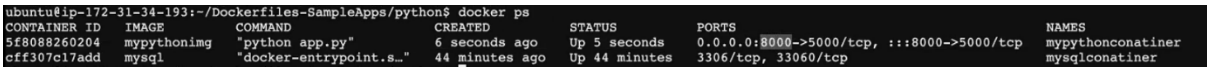
```
3.8-alpine: Pulling from library/python
d25f557d7f31: Pull complete
d2c04aca259c: Pull complete
b7072cd997c7: Pull complete
3292fffb3ff2: Pull complete
964683ac7c06: Pull complete
Digest: sha256:73d71732994b1541eeac5c8e61218942082a1e33e009144f6b1adfde1c821c04
Status: Downloaded newer image for python:3.8-alpine
 ---> bedfee6adc40
Step 2/7 : WORKDIR /test
 ---> Running in c5023add93db
Removing intermediate container c5023add93db
 ---> 7871122e0aec
Step 3/7 : COPY requirements.txt .
 ---> 171745ecf6ea
Step 4/7 : RUN pip install -r requirements.txt
 ---> Running in f8d0bbfafbba
Collecting Flask==2.0.1
  Downloading Flask-2.0.1-py3-none-any.whl (94 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 94.8/94.8 kB 5.7 MB/s eta 0:00:00
Collecting Werkzeug==2.0.1
  Downloading Werkzeug-2.0.1-py3-none-any.whl (288 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 288.2/288.2 kB 26.2 MB/s eta 0:00:00
Collecting itsdangerous>=2.0
  Downloading itsdangerous-2.2.0-py3-none-any.whl (16 kB)
Collecting click>=7.1.2
  Downloading click-8.1.7-py3-none-any.whl (97 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 97.9/97.9 kB 14.5 MB/s eta 0:00:00
Collecting Jinja2>=3.0
  Downloading jinja2-3.1.4-py3-none-any.whl (133 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 133.3/133.3 kB 11.6 MB/s eta 0:00:00
Collecting MarkupSafe>=2.0
  Downloading MarkupSafe-2.1.5-cp38-cp38-musllinux_1_1_x86_64.whl (29 kB)
Installing collected packages: Werkzeug, MarkupSafe, itsdangerous, click, Jinja2, Flask
```



```
ubuntu@ip-172-31-34-193:~/Dockerfiles-SampleApps/python$ docker images
REPOSITORY     TAG           IMAGE ID       CREATED              SIZE
mypythonimg    latest        674238f59cb9   About a minute ago   58.7MB
python         3.8-alpine    bedfee6adc40   2 weeks ago          47.4MB
mysql          latest        fcd86ff8ce8c   5 weeks ago          578MB
ubuntu@ip-172-31-34-193:~/Dockerfiles-SampleApps/python$
```

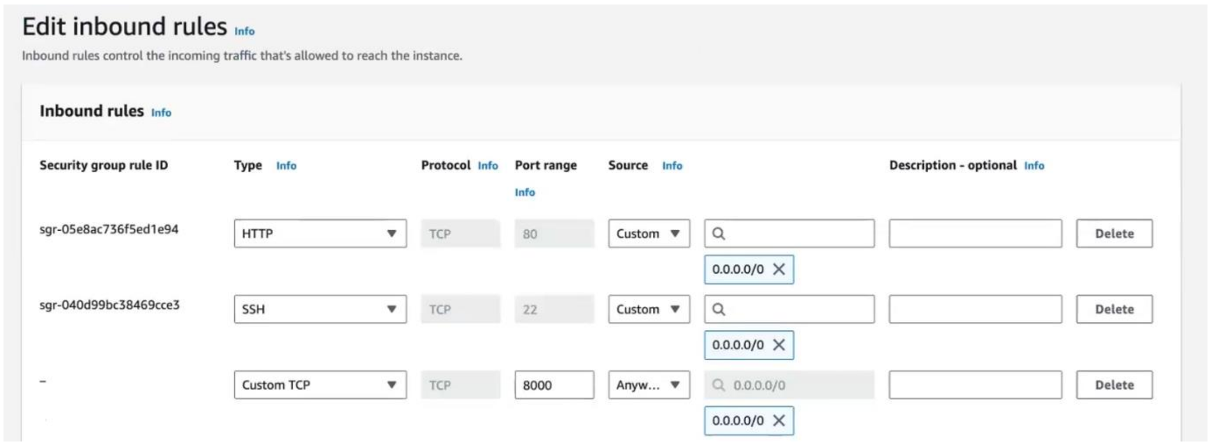


```
ubuntu@ip-172-31-34-193:~/Dockerfiles-SampleApps/python$ docker run -d --name mypythonconatiner -p 8000:5000 mypythonimg
5f80882602047899b51b29e9a820a5d690f3b72516055f6c937aa8d29b4f8ce6
```



```
ubuntu@ip-172-31-34-193:~/Dockerfiles-SampleApps/python$ docker ps
CONTAINER ID   IMAGE         COMMAND                  CREATED          STATUS          PORTS                                       NAMES
5f8088260204   mypythonimg   "python app.py"          6 seconds ago    Up 5 seconds    0.0.0.0:8000->5000/tcp, :::8000->5000/tcp   mypythonconatiner
cff307c17add   mysql         "docker-entrypoint.s…"   44 minutes ago   Up 44 minutes   3306/tcp, 33060/tcp                         mysqlconatiner
```

Open port number 8000 in ec2



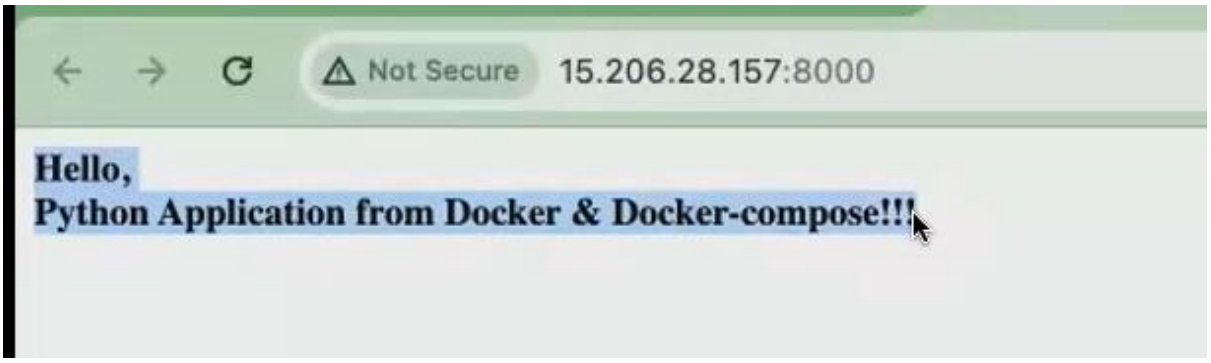
Edit inbound rules Info

Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules Info

Security group rule ID	Type Info	Protocol Info	Port range Info	Source Info		Description - optional Info	
sgr-05e8ac736f5ed1e94	HTTP	TCP	80	Custom	0.0.0.0/0		Delete
sgr-040d99bc38469cce3	SSH	TCP	22	Custom	0.0.0.0/0		Delete
-	Custom TCP	TCP	8000	Anyw...	0.0.0.0/0		Delete

Copy ec2 ip :5000 in browser. Python application run



Not Secure 15.206.28.157:8000

Hello,
Python Application from Docker & Docker-compose!!!