

Lab Report 4 (Docker II) - Antonio Manuel Luque Molina



We will learn about docker images in details.



Build an image with a python application.



Get familiar with Dockerfile.



Learn the layering concepts.



Publicly make available our own container images.

Types of Docker Images:

- Base Images – These images don't have any parent images, for example, ubuntu, Debian, busybox , etc.
- Child Images – These images are built on base images with some added functionality like hello-world.

There is one more way to distinguish images:

- Official – These images are officially maintained and supported by people at docker. Example includes ubuntu, hello-world, etc.
- User - These images are created by users like you. These are built on base images and some functionalities are added. The name format is user/imagename.

We need to make a Dockerfile to build an image with this application;

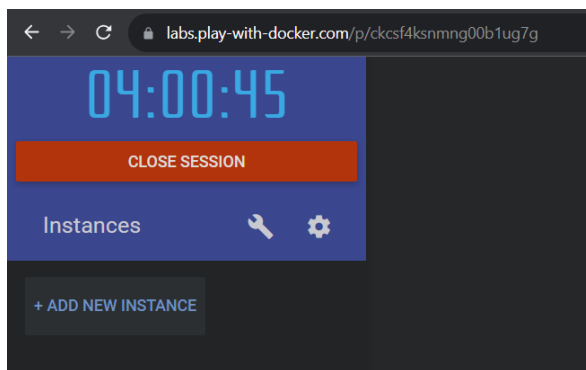
- A Dockerfile is a simple text file that contains a list of commands that the Docker client calls while creating an image.
- The commands are like linux commands, so no need to learn syntax to write a Dockerfile.

Using Docker Playground:

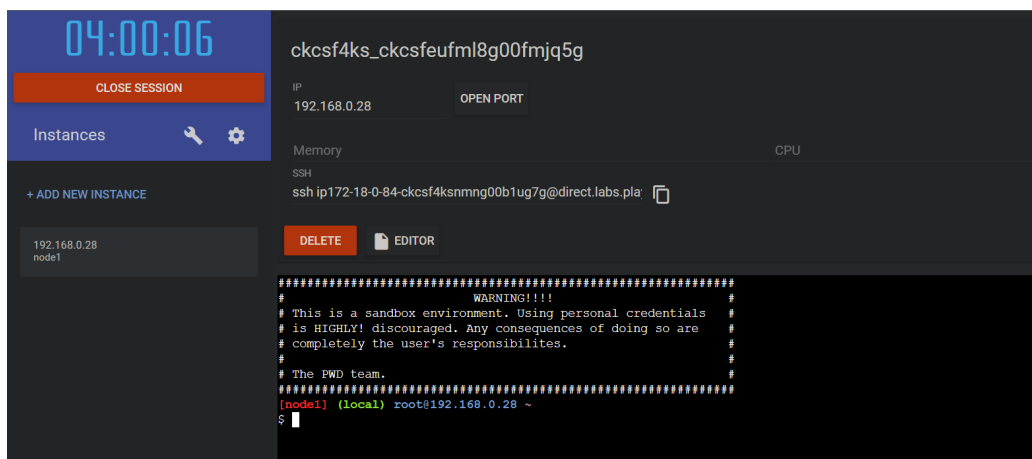
We go to the link: <https://labs.play-with-docker.com/>



We click on Start,



We click on Add New Instance,



This creates a VM instance for us to run docker commands.

Building our first image:

We will clone this link: <https://github.com/Rohit102497/BigDataAndCloudTechnology>

The screenshot shows a Docker container interface for 'cki3i4ss_cki3i84snmng00asnhp0'. It displays the IP address 192.168.0.8, memory usage at 1.15% (46.11MiB / 3.906GiB), and CPU usage at 1.01%. An 'OPEN PORT' button is visible. Below the statistics, an SSH command is provided: 'ssh ip172-18-0-9-cki3i4ssnmng00asnhog@direct.labs.play-'. At the bottom, there are 'DELETE' and 'EDITOR' buttons. A terminal window shows the command 'git clone https://github.com/Rohit102497/BigDataAndCloudTechnology.git' and its output, indicating a successful clone.

Now We have cloned it and We will access it:

```
[node1] (local) root@192.168.0.8 ~
$ ls
BigDataAndCloudTechnology
[node1] (local) root@192.168.0.8 ~
$ cd BigDataAndCloudTechnology/
[node1] (local) root@192.168.0.8 ~/BigDataAndCloudTechnology
$ ls
app.py          requirements.txt  templates
[node1] (local) root@192.168.0.8 ~/BigDataAndCloudTechnology
$
```

We need to make a Dockerfile to build an image with this application, so:

```
[node1] (local) root@192.168.0.8 ~/BigDataAndCloudTechnology
$ vi Dockerfile
```

We press “i” to insert text and We write the following:

The screenshot shows the same Docker container interface as before, but with the 'EDITOR' button clicked. The terminal now displays the content of the Dockerfile, which is being edited in insert mode. The Dockerfile content is as follows:

```
1 FROM python:3
2
3 WORKDIR /usr/src/app
4
5 COPY . .
6
7 RUN pip install --no-cache-dir -r requirements.txt
8
9 EXPOSE 8000
10
11 CMD ["python", "./app.py"]
12
13
```

After that We press esc and We write :wq to save and quit it.

In the next step We will build the image:

```
[node1] (local) root@192.168.0.8 ~/BigDataAndCloudTechnology
$ docker build -t rag004/doggif .
[+] Building 44.4s (9/9) FINISHED
```

Once it is builded We will run it:

OPEN PORT

30000

Memory
27.59% (1.078GiB / 3.906GiB)

CPU
47.33%

SSH
ssh ip172-18-0-9-cki3i4ssnmng00asnhog@direct.labs.play-v 

DELETE

 EDITOR

```
=> => exporting layers 0.3s
=> => writing image sha256:2d63d16240b63a90a138072c4ee5327f01d11869157601a9 0.0s
=> => naming to docker.io/rag004/doggif 0.0s
[node1] (local) root@192.168.0.8 ~/BigDataAndCloudTechnology
$ docker run -p 30000:8000 --name doggif rag004/doggif
```

Now, clicking in the port We are able to see the webpage.

Using Server Access:

```
PowerShell 7 (x64)
[Antonio Luque]--[0master ≠ • • ]
[~]
ssh anluq0157@vs-c2.cs.uit.no
```

This command initiates an SSH (Secure Shell) connection to the server at vs-c2.cs.uit.no.

Building our first image:

We will clone this link: <https://github.com/Rohit102497/BigDataAndCloudTechnology>

```
anluq0157@vs-c2:~$ git clone https://github.com/Rohit102497/
BigDataAndCloudTechnology.git
```

Now We have cloned it and We will access it:

```
anluq0157@vs-c2:~$ ls
BigDataAndCloudTechnology bin
anluq0157@vs-c2:~$ cd BigDataAndCloudTechnology/
anluq0157@vs-c2:~/BigDataAndCloudTechnology$ ls
app.py Dockerfile requirements.txt templates
anluq0157@vs-c2:~/BigDataAndCloudTechnology$
```

We need to make a Dockerfile to build an image with this application, so:

```
anluq0157@vs-c2:~/BigDataAndCloudTechnology$ vi Dockerfile
```

```
anluq0157@vs-c2: ~/BigData/
FROM python:3
WORKDIR /usr/src/app
COPY . .
RUN pip install --no-cache-dir -r requirements.txt
EXPOSE 8000
CMD ["python", "./app.py"]
~
~
~
```

After write that we will type :wq in order to save and quit.

In the next step We will build the image;

```

anluq0157@vs-c2:~/BigDataAndCloudTechnology$ docker build -t rag004/doggif .
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   72.7kB
Step 1/6 : FROM python:3
--> b94d01b49295
Step 2/6 : WORKDIR /usr/src/app
--> Using cache
--> 16583913156b
Step 3/6 : COPY . .
--> Using cache
--> c6dd0e11ef7a
Step 4/6 : RUN pip install --no-cache-dir -r requirements.txt
--> Using cache
--> 461441a13aee
Step 5/6 : EXPOSE 8000
--> Using cache
--> cd516ded899e
Step 6/6 : CMD ["python", "./app.py"]
--> Using cache
--> 459badd4dfb0
Successfully built 459badd4dfb0
Successfully tagged rag004/doggif:latest

```

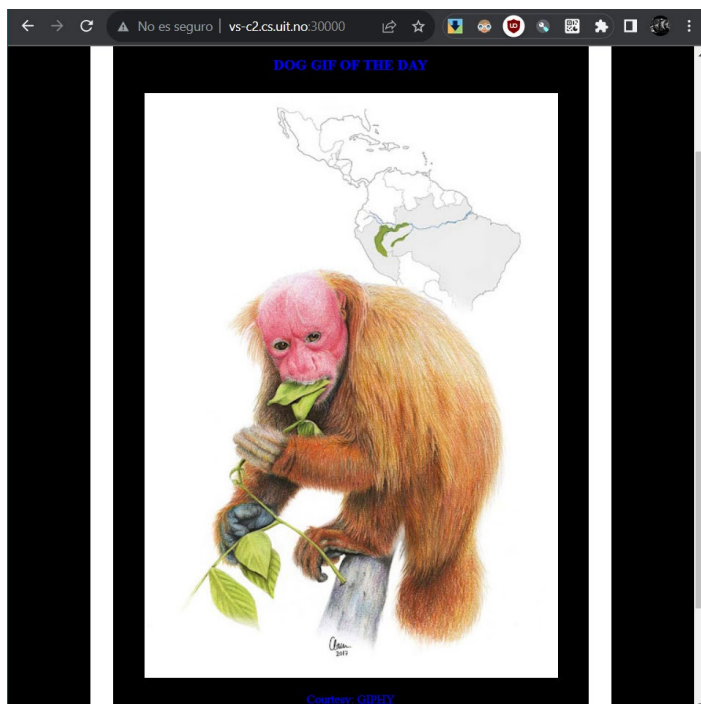
Once it is built We will run it:

```

anluq0157@vs-c2:~/BigDataAndCloudTechnology$ docker run -p 30000:8000 --name
doggif rag004/doggif
* Serving Flask app 'app' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production depl
oyment.
  Use a production WSGI server instead.
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deploy
ment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:8000
* Running on http://172.17.0.2:8000
Press CTRL+C to quit

```

Now, We can go to the webpage to see it in the web browser:

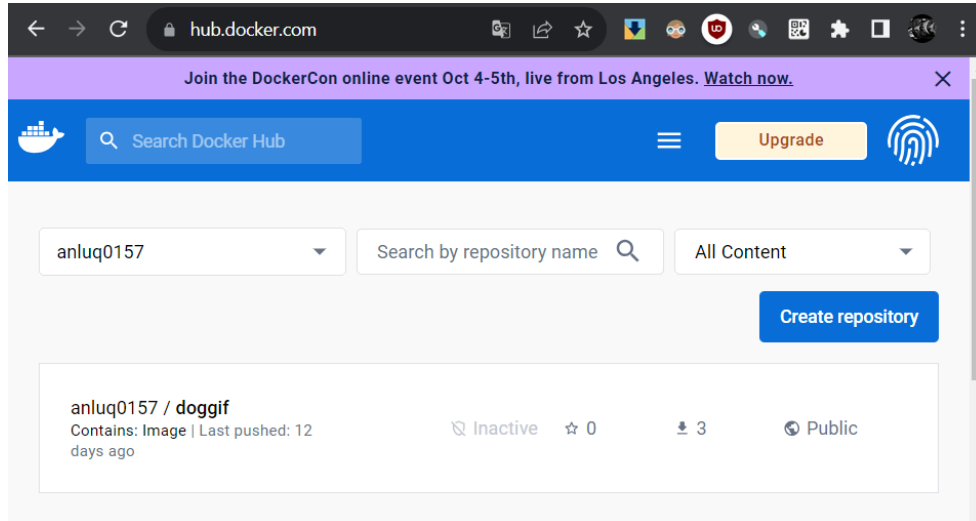


This view is with my update in the application app.py.

Now We will save our image to our docker hub.

Firs of all We will do **docker login** in the console to login in dockerhub, and after that We will write docker **push rag004/doggif** to uplading our image to our docker hub.

Finally, We can see our image there:



We can pull our image and run it in our system and We can pull our peers images too.

Thank you for reading!