

TALLER FASTAPI

ESTUDIANTE:
BREINER ANDRES JAIMES MIRANDA

PROFESOR:
SALOMON CADENA



INGENIERIA DE SISTEMAS. FACULTAD DE INGENIERIA Y TECNOLOGIAS
UNIVERSIDAD POPULAR DEL CESAR
VALLEDUPAR
2023

Creamos las carpetas:

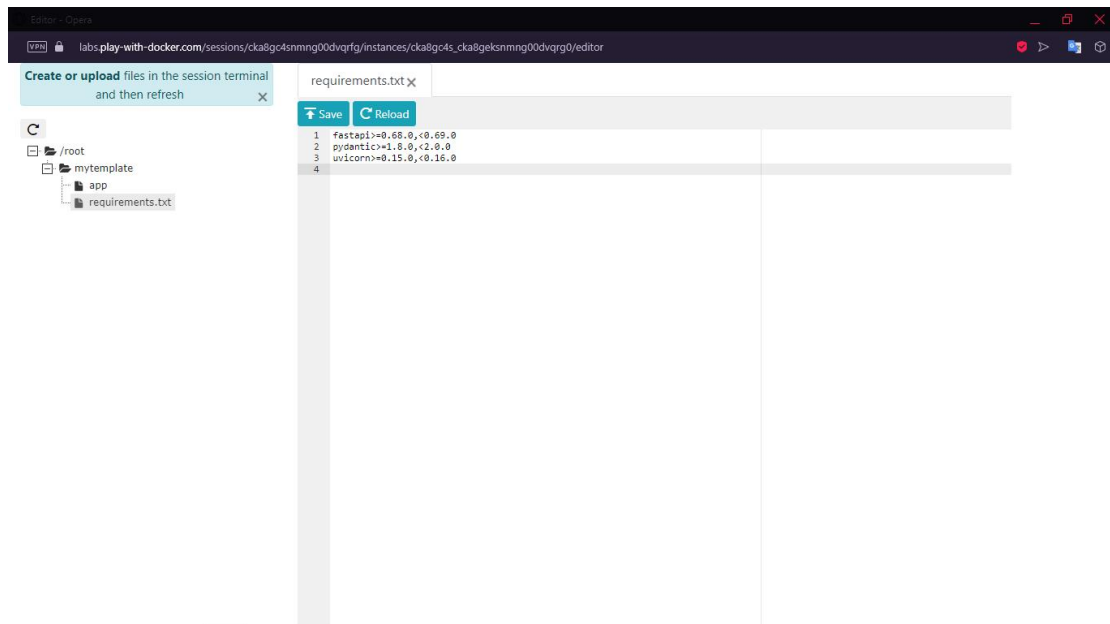
The screenshot shows the Docker Playground interface. On the left, a sidebar contains a clock showing 03:57:02, a 'CLOSE SESSION' button, and an 'Instances' section with a list of instances including '192.168.0.8 node1'. The main panel displays details for instance 'cka8gc4s_cka8geksnmng00dvqrg0', including its IP (192.168.0.8), memory usage (1.17%), CPU usage (0.56%), and an SSH command. Below this, a terminal window shows the following commands and output:

```
# completely the user's responsibilities.
#
# The FWD team.
#####
(node1) (local) root@192.168.0.8 ~
$ pwd
/root
(node1) (local) root@192.168.0.8 ~
$ mkdir mytemplate
(node1) (local) root@192.168.0.8 ~
$ cd mytemplate
(node1) (local) root@192.168.0.8 ~/mytemplate
$ mkdir app
(node1) (local) root@192.168.0.8 ~/mytemplate
$ ls
app
(node1) (local) root@192.168.0.8 ~/mytemplate
$ pwd
/root/mytemplate
(node1) (local) root@192.168.0.8 ~/mytemplate
$
```

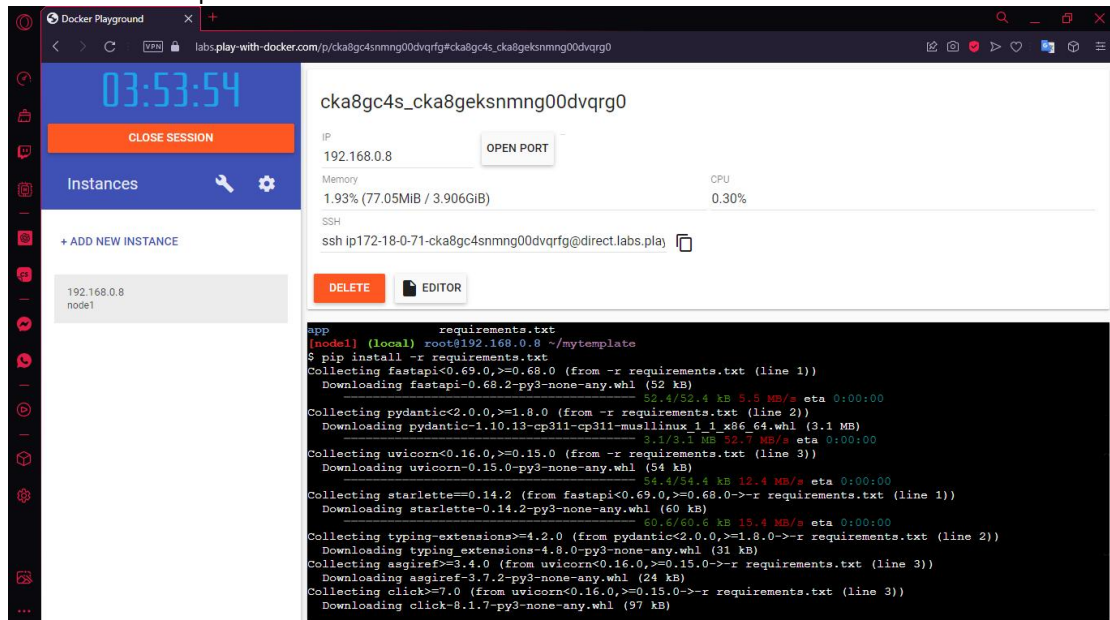
Creacion del archivo requirements.txt:

The screenshot shows the Docker Playground interface. On the left, a sidebar contains a clock showing 03:55:58, a 'CLOSE SESSION' button, and an 'Instances' section with a list of instances including '192.168.0.8 node1'. The main panel displays details for instance 'cka8gc4s_cka8geksnmng00dvqrg0', including its IP (192.168.0.8), memory usage (1.23%), CPU usage (0.43%), and an SSH command. Below this, a terminal window shows the following commands and output:

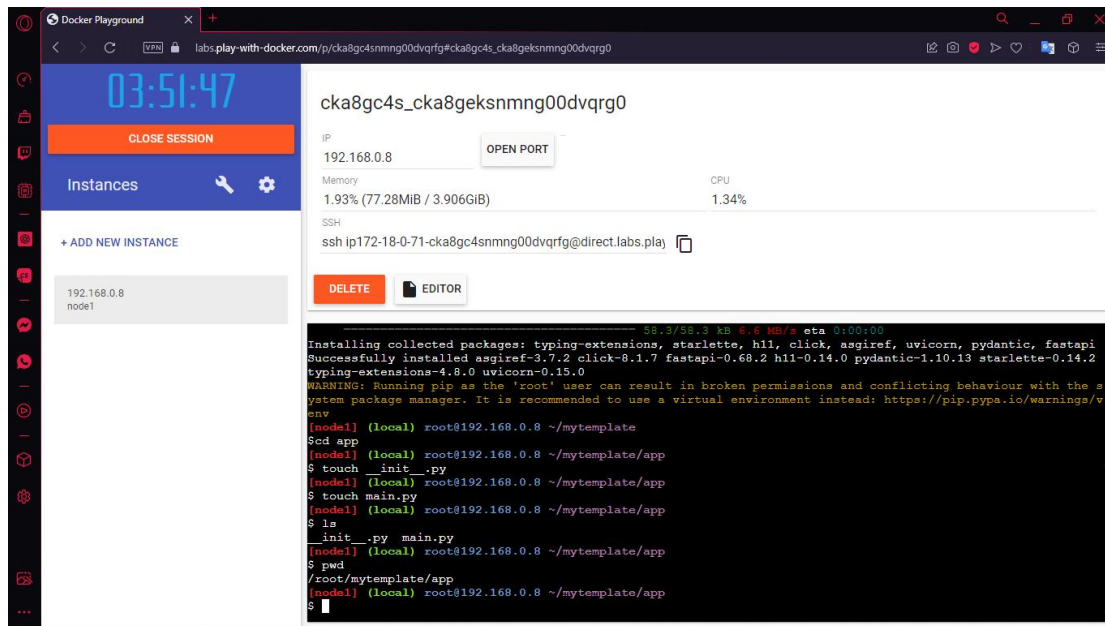
```
$ pwd
/root
(node1) (local) root@192.168.0.8 ~
$ mkdir mytemplate
(node1) (local) root@192.168.0.8 ~
$ cd mytemplate
(node1) (local) root@192.168.0.8 ~/mytemplate
$ mkdir app
(node1) (local) root@192.168.0.8 ~/mytemplate
$ ls
app
(node1) (local) root@192.168.0.8 ~/mytemplate
$ pwd
/root/mytemplate
(node1) (local) root@192.168.0.8 ~/mytemplate
$ touch requirements.txt
(node1) (local) root@192.168.0.8 ~/mytemplate
$ ls
app
requirements.txt
(node1) (local) root@192.168.0.8 ~/mytemplate
$
```



Instalamos los requerimientos indicados en el archivo:

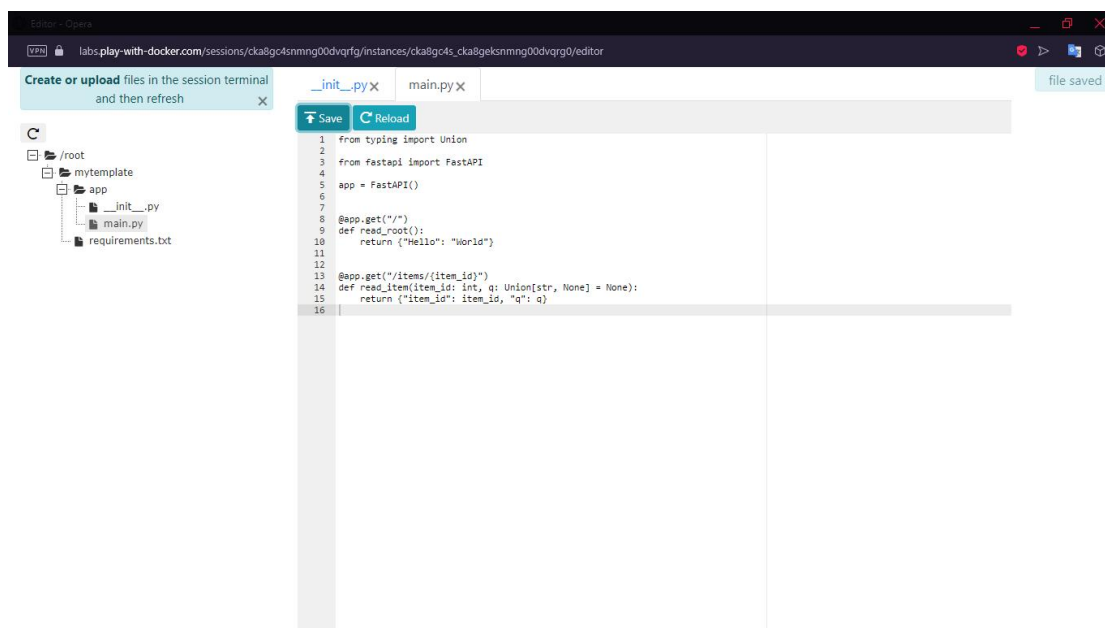


Ingresamos a la carpeta app y creamos los archivos `__init__.py` y `main.py`:



The screenshot shows the Docker Playground interface. On the left, there's a sidebar with a clock showing 03:51:47 and a 'CLOSE SESSION' button. Below that, the 'Instances' section shows a single instance named 'node1' with IP 192.168.0.8. The main panel displays details for the instance 'cka8gc4s_cka8geksnmng00dvqrg0'. It shows the IP 192.168.0.8, memory usage at 1.93% (77.28MiB / 3.906GiB), and CPU usage at 1.34%. There's an 'OPEN PORT' button and an SSH command: `ssh ip172-18-0-71-cka8gc4snmng00dvqrg@direct.labs.play`. Below this, there's a 'DELETE' button and an 'EDITOR' button. The terminal window shows the following commands and output:

```
58.3/38.3 KB 6.8 MB/s eta 0:00:00
Installing collected packages: typing-extensions, starlette, h11, click, asgiref, uvicorn, pydantic, fastapi
Successfully installed asgiref-3.7.2 click-8.1.7 fastapi-0.68.2 h11-0.14.0 pydantic-1.10.13 starlette-0.14.2
typing-extensions-4.8.0 uvicorn-0.15.0
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the s
ystem package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/v
env
[node1] (local) root@192.168.0.8 ~/mytemplate
$ cd app
[node1] (local) root@192.168.0.8 ~/mytemplate/app
$ touch __init__.py
[node1] (local) root@192.168.0.8 ~/mytemplate/app
$ touch main.py
[node1] (local) root@192.168.0.8 ~/mytemplate/app
$ ls
__init__.py  main.py
[node1] (local) root@192.168.0.8 ~/mytemplate/app
$ pwd
/root/mytemplate/app
[node1] (local) root@192.168.0.8 ~/mytemplate/app
$
```

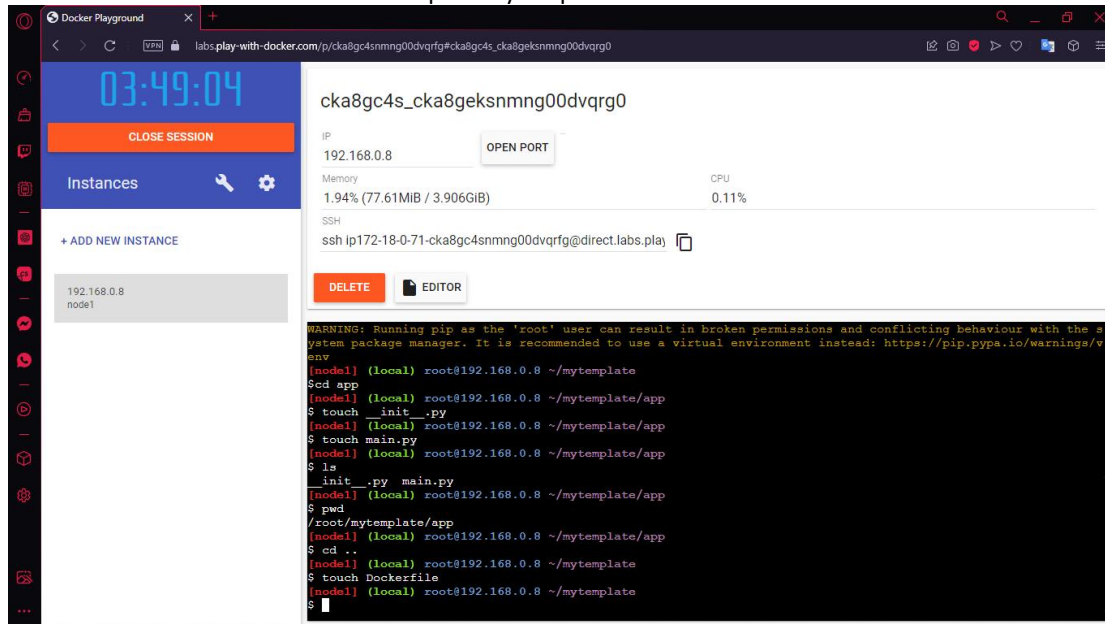


The screenshot shows the Docker Playground editor interface. On the left, there's a file explorer showing the directory structure: `/root`, `mytemplate`, `app`, `__init__.py`, `main.py`, and `requirements.txt`. The main panel shows the content of the `main.py` file, which is being edited. The code is as follows:

```
1 from typing import Union
2
3 from fastapi import FastAPI
4
5 app = FastAPI()
6
7
8 @app.get("/")
9 def read_root():
10     return {"Hello": "World"}
11
12
13 @app.get("/items/{item_id}")
14 def read_item(item_id: int, q: Union[str, None] = None):
15     return {"item_id": item_id, "q": q}
16
```

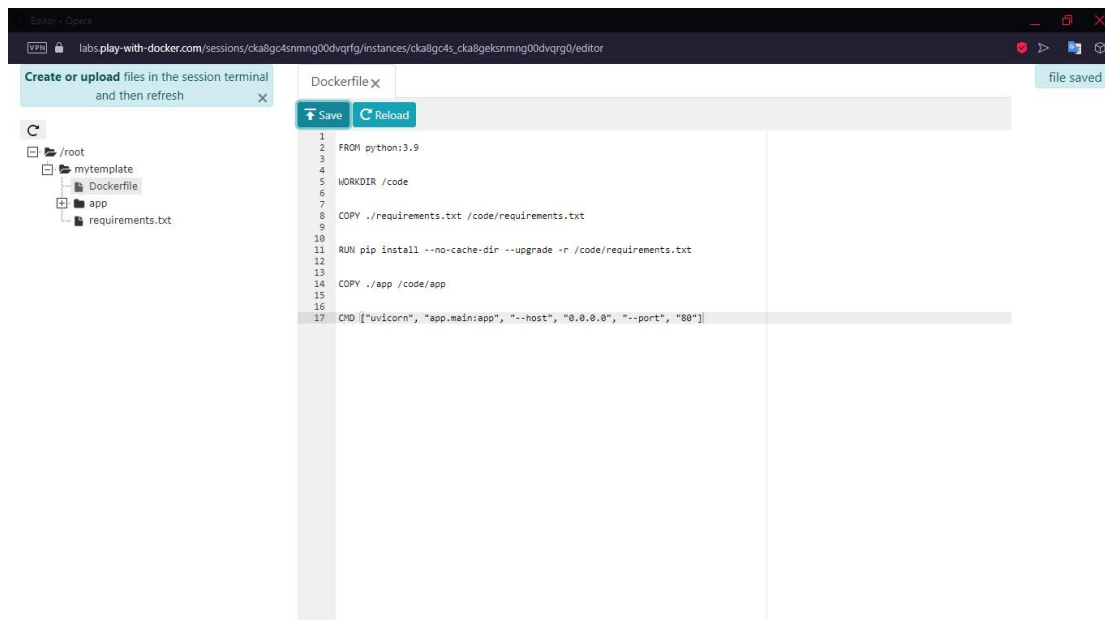
Rellenamos el archivo `main.py` con el código indicado en el tutorial

Creamos el archivo Dockerfile en la carpeta mytemplate:



The screenshot shows the Docker Playground interface. On the left, there's a sidebar with a clock showing 03:49:04 and a 'CLOSE SESSION' button. Below that, the 'Instances' section shows a single instance named 'node1' with IP 192.168.0.8. The main panel displays the instance details for 'cka8gc4s_cka8geksnmng00dvqrg0', including its IP (192.168.0.8), memory usage (1.94%), and CPU usage (0.11%). Below the details is a terminal window showing the following commands and output:

```
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the s
ystem package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/v
env
(node1) (local) root@192.168.0.8 ~/mytemplate
$ cd app
(node1) (local) root@192.168.0.8 ~/mytemplate/app
$ touch __init__.py
(node1) (local) root@192.168.0.8 ~/mytemplate/app
$ touch main.py
(node1) (local) root@192.168.0.8 ~/mytemplate/app
$ ls
__init__.py  main.py
(node1) (local) root@192.168.0.8 ~/mytemplate/app
$ pwd
/root/mytemplate/app
(node1) (local) root@192.168.0.8 ~/mytemplate/app
$ cd ..
(node1) (local) root@192.168.0.8 ~/mytemplate
$ touch Dockerfile
(node1) (local) root@192.168.0.8 ~/mytemplate
$
```



The screenshot shows the Dockerfile editor in Docker Playground. The left sidebar shows a file tree with the following structure:

- /root
 - mytemplate
 - Dockerfile
 - app
 - requirements.txt

The main editor area shows the Dockerfile content:

```
1 FROM python:3.9
2
3
4 WORKDIR /code
5
6
7 COPY ./requirements.txt /code/requirements.txt
8
9
10 RUN pip install --no-cache-dir --upgrade -r /code/requirements.txt
11
12
13 COPY ./app /code/app
14
15
16
17 CMD ["uvicorn", "app.main:app", "--host", "0.0.0.0", "--port", "80"]
```

Buttons for 'Save' and 'Reload' are visible at the top of the editor. A 'file saved' notification is shown in the top right corner.

Añadimos la información al archivo Dockerfile

Creamos la imagen:

The screenshot shows the Docker Playground interface. On the left, there's a sidebar with a clock showing 03:45:43, a 'CLOSE SESSION' button, and an 'Instances' section with a list of instances (192.168.0.8, node1). The main area displays the instance details for 'cka8gc4s_cka8geksnmng00dvqrg0'. It shows the IP address 192.168.0.8, memory usage (13.44% / 537.4MiB / 3.906GiB), and CPU usage (155.41%). There's an 'OPEN PORT' button and an SSH command: 'ssh ip172-18-0-71-cka8gc4snmng00dvqrg0@direct.labs.play'. Below this, there's a 'DELETE' button and an 'EDITOR' button. The terminal output shows the process of building a Docker image named 'miimagen' from a Dockerfile. The build process includes steps like 'Building 6.3s (4/9)', 'transferring dockerfile: 304B', 'load metadata for docker.io/library/python:3.9', and 'extracting sha256:167b8a53ca4504bcca3182e336fa96f4ef76875d158c1933d3e2fa19c57e0c3'.

Corremos la imagen

The screenshot shows the Docker Playground interface. On the left, there's a sidebar with a clock showing 03:43:28, a 'CLOSE SESSION' button, and an 'Instances' section with a list of instances (192.168.0.8, node1). The main area displays the instance details for 'cka8gc4s_cka8geksnmng00dvqrg0'. It shows the IP address 192.168.0.8, memory usage (30.51% / 1.192GiB / 3.906GiB), and CPU usage (1.03%). There's an 'OPEN PORT' button with a dropdown menu showing '80'. There's an SSH command: 'ssh ip172-18-0-71-cka8gc4snmng00dvqrg0@direct.labs.play'. Below this, there's a 'DELETE' button and an 'EDITOR' button. The terminal output shows the process of running the Docker image. It includes steps like 'extracting sha256:3d8bce5f9fa9709885f7e2ad3cf41f036a3b57b406b27ba3a88392815787042', 'RUN pip install --no-cache-dir --upgrade -r /code/requirements.txt', and 'COPY ./app /code/app'. The final output shows the image being exported and named 'miimagen'.

Verificamos la creacion de la imagen:

The screenshot shows the Docker Playground interface. On the left, there's a sidebar with a clock showing 03:42:57, a 'CLOSE SESSION' button, and a list of instances with one instance named 'node1' at IP 192.168.0.8. The main panel displays details for the instance 'cka8gc4s_cka8geksnmng00dvqrg0' with IP 192.168.0.8. It shows memory usage at 30.52% (1.192GiB / 3.906GiB) and CPU usage at 4.05%. An SSH command is provided: `ssh ip172-18-0-71-cka8gc4snmng00dvqrg@direct.labs.play`. Below this are 'DELETE' and 'EDITOR' buttons. The terminal window shows the following output:

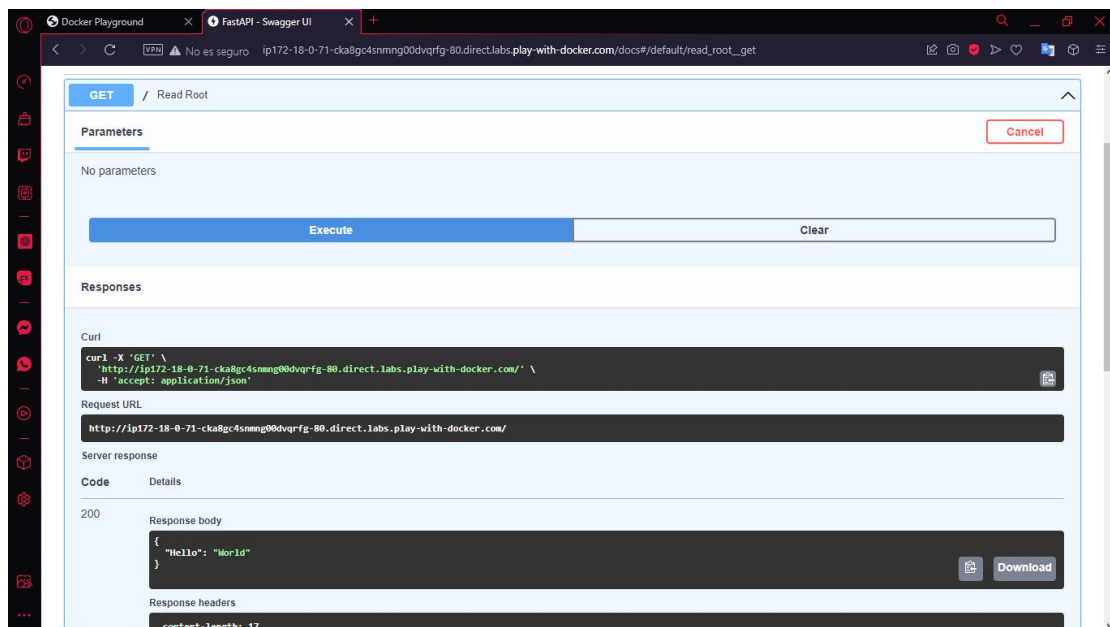
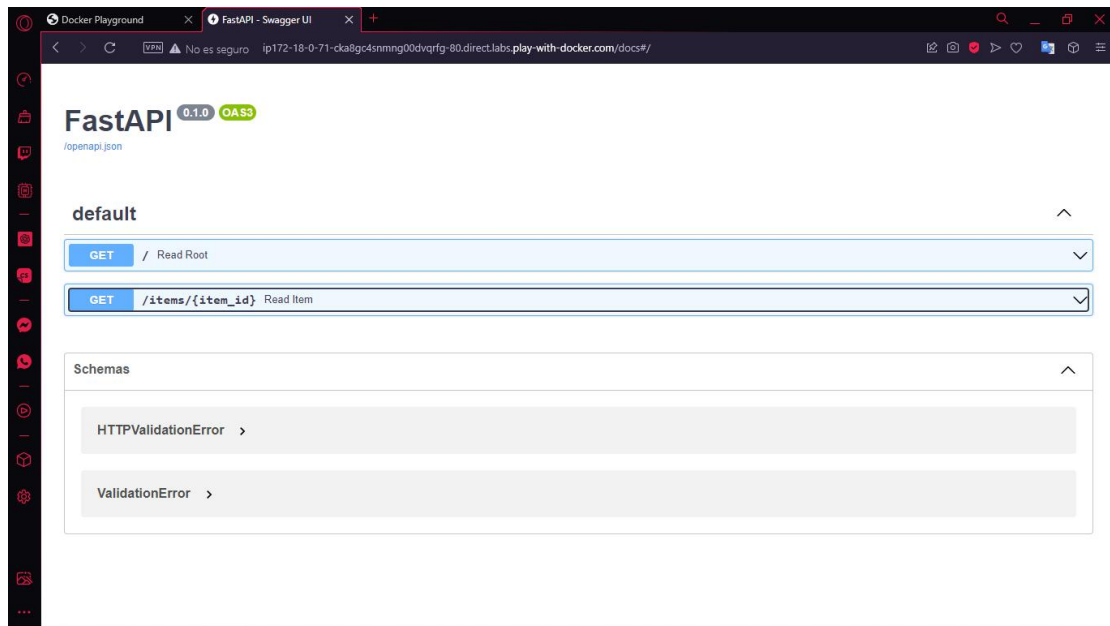
```
=> => transferring context: 499B 0.0s
=> [2/5] WORKDIR /code 0.0s
=> [3/5] COPY ./requirements.txt /code/requirements.txt 0.1s
=> [4/5] RUN pip install --no-cache-dir --upgrade -r /code/requirements.txt 6.4s
=> [5/5] COPY ./app /code/app 0.1s
=> exporting to image 0.3s
=> => exporting layers 0.3s
=> => writing image sha256:f8a72b3d68cd48508adea04b7fa9bc811169dd8e41af47e9cce597c31d796b15 0.0s
=> => naming to docker.io/library/miimagen 0.0s
(node1) (local) root@192.168.0.8 ~/mytemplate
$ docker run -d --name micontenedor -p 80:80 miimagen
21b06bd0ae2de0f5719e034c38f678b3e3a3f42cf71b70f8bc63ae4bdb9a0ce6
(node1) (local) root@192.168.0.8 ~/mytemplate
$ Docker images
bash: Docker: command not found
(node1) (local) root@192.168.0.8 ~/mytemplate
$ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
miimagen latest f8a72b3d68cd About a minute ago 1.02GB
(node1) (local) root@192.168.0.8 ~/mytemplate
$
```

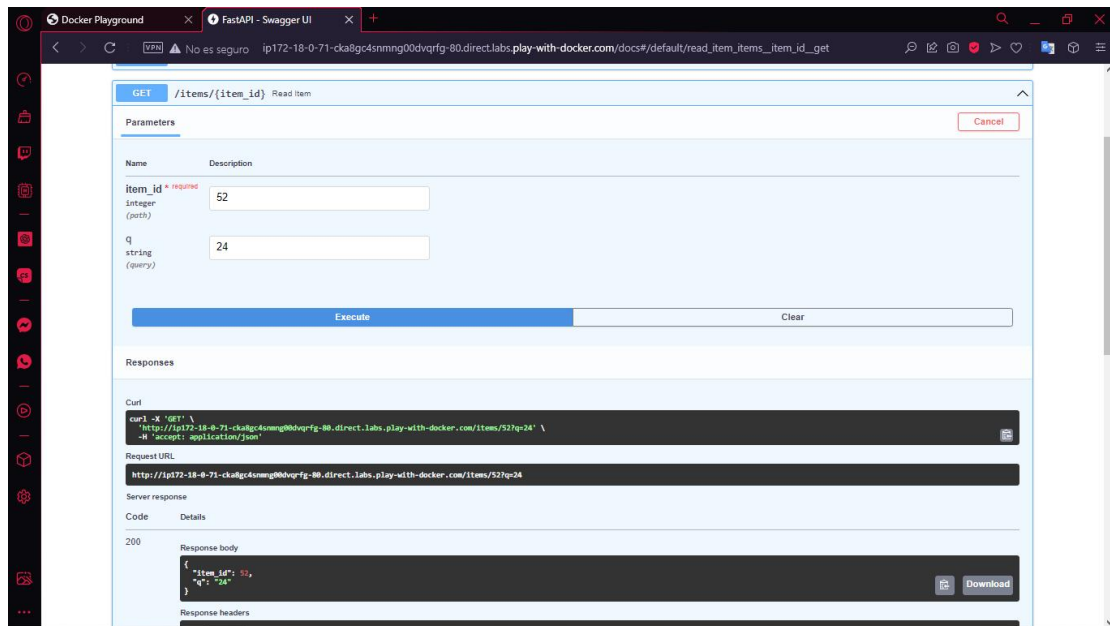
Abrimos el puerto:

The screenshot shows the Docker Playground interface with a terminal window open. The terminal displays the output of a curl command, showing a successful response from the API:

```
{ "Hello": "World" }
```

Accedemos al api:





Repositorio GuitHub: <https://github.com/BreinerAndresJaimesMiranda/TallerFastApi>