Curso: Docker

Homework 3 - Building Images

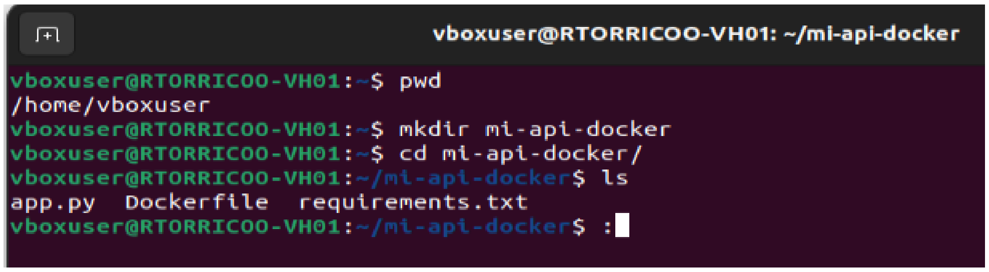
Instructor: Caleb Espinoza Gutierrez

Estudiante: Ronald Torrico Ovando

Exercise 1: Build and Containerize an API (Back-End)

Develop a simple API using a programming language of your choice (e.g., Node.js, Python, Go).

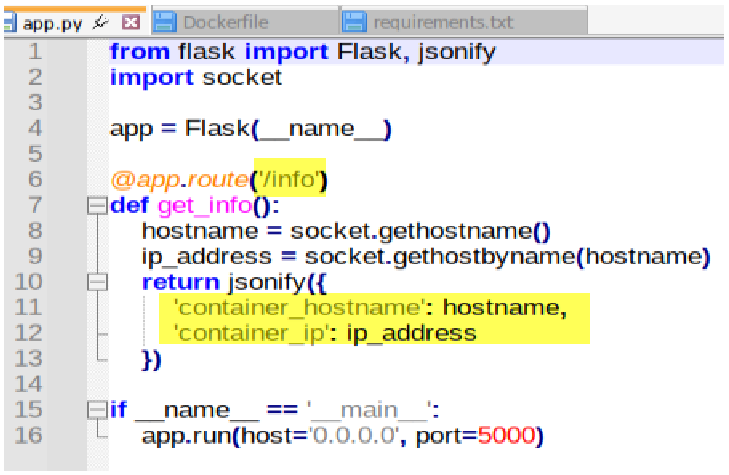
- Create basic structure Project



vboxuser@RTORRICOO-VH01: ~/mi-api-docker

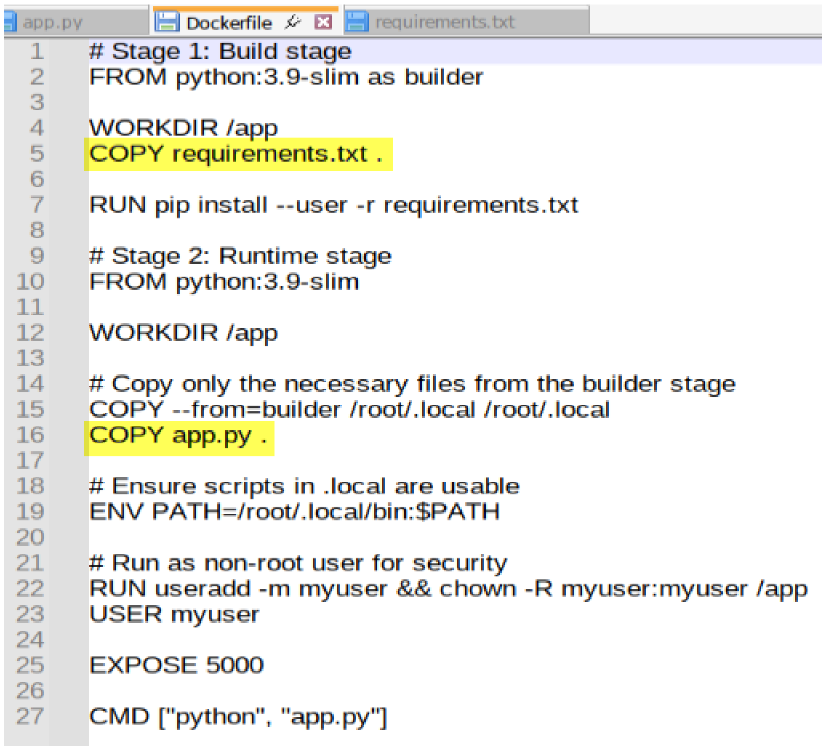
```
vboxuser@RTORRICOO-VH01:~$ pwd
/home/vboxuser
vboxuser@RTORRICOO-VH01:~$ mkdir mi-api-docker
vboxuser@RTORRICOO-VH01:~$ cd mi-api-docker/
vboxuser@RTORRICOO-VH01:~/mi-api-docker$ ls
app.py  Dockerfile  requirements.txt
vboxuser@RTORRICOO-VH01:~/mi-api-docker$ :
```

- The API must expose an endpoint (e.g., /info) that returns:
- The container's hostname.
- The container's IP address.



```python
from flask import Flask, jsonify
import socket

app = Flask(__name__)

@app.route('/info')
def get_info():
    hostname = socket.gethostname()
    ip_address = socket.gethostbyname(hostname)
    return jsonify({
        'container_hostname': hostname,
        'container_ip': ip_address
    })

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

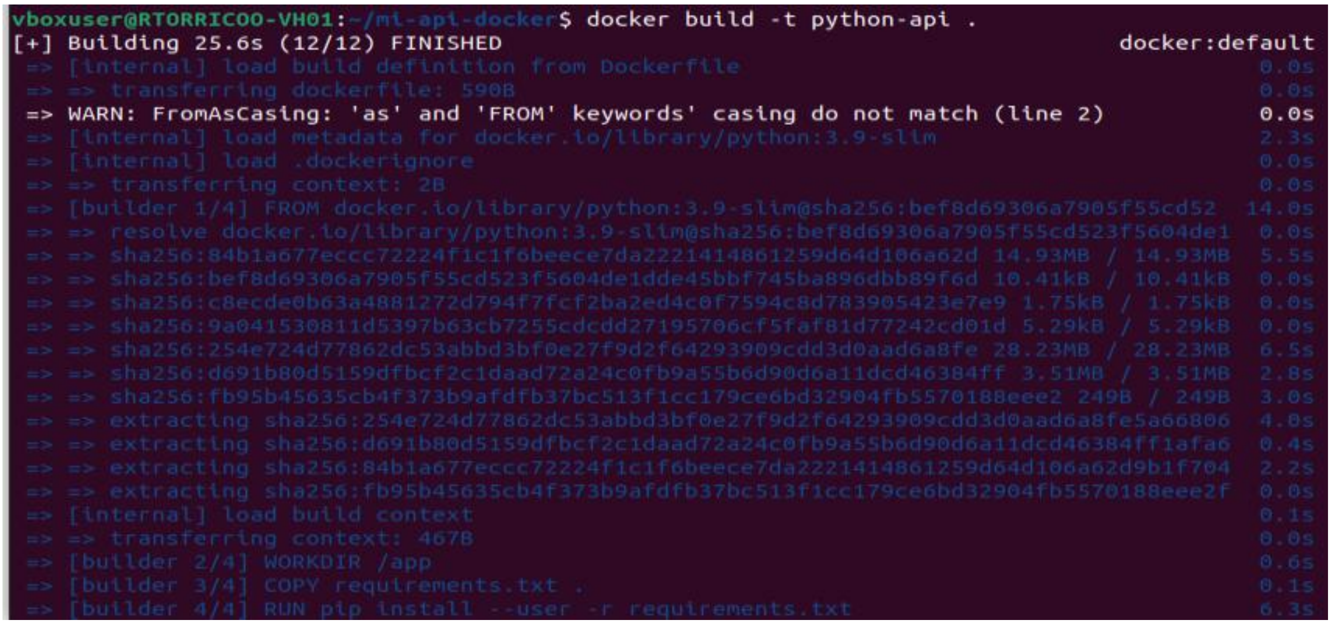
- Write a Dockerfile using *multistage build* to containerize the API.



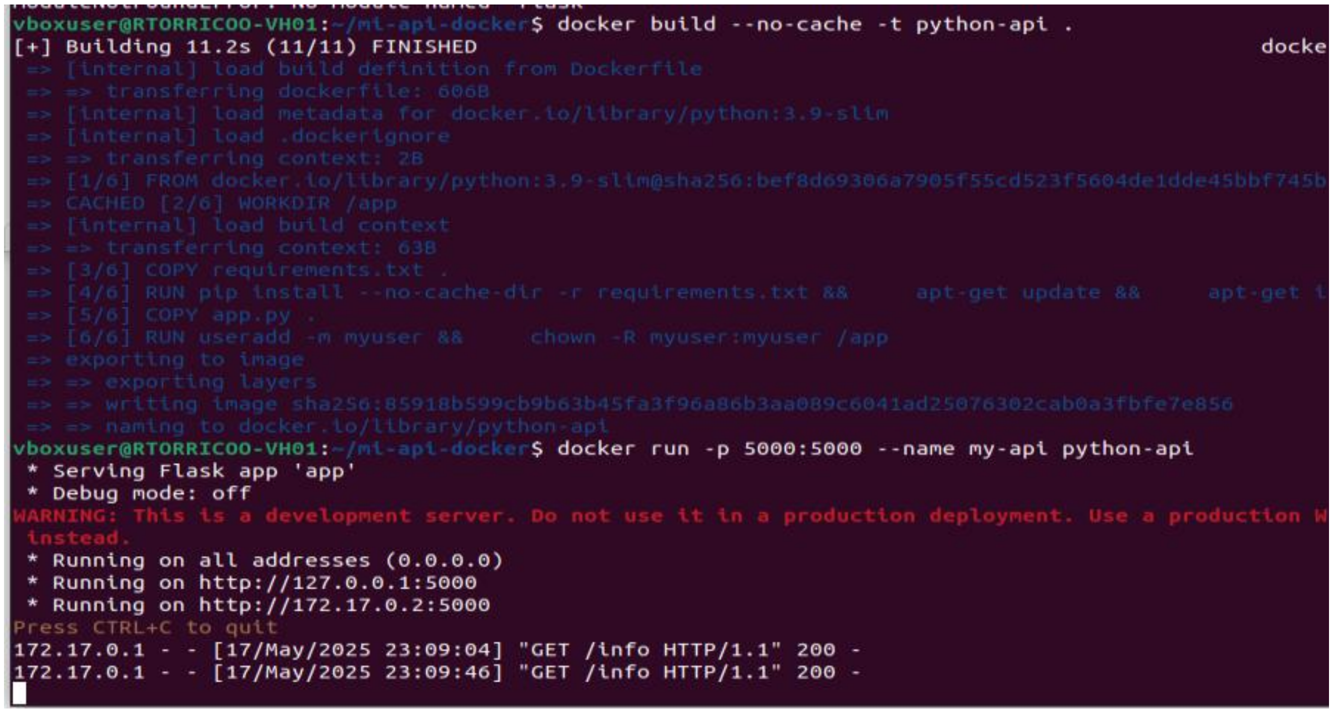
```dockerfile
# Stage 1: Build stage
FROM python:3.9-slim as builder

WORKDIR /app
COPY requirements.txt .

RUN pip install --user -r requirements.txt

# Stage 2: Runtime stage
FROM python:3.9-slim

WORKDIR /app

# Copy only the necessary files from the builder stage
COPY --from=builder /root/.local /root/.local
COPY app.py .

# Ensure scripts in .local are usable
ENV PATH=/root/.local/bin:$PATH

# Run as non-root user for security
RUN useradd -m myuser && chown -R myuser:myuser /app
USER myuser

EXPOSE 5000

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

- Build the image and run the container.

  docker build -t python-api .
  docker run -p 5000:5000 --name my-api python-api

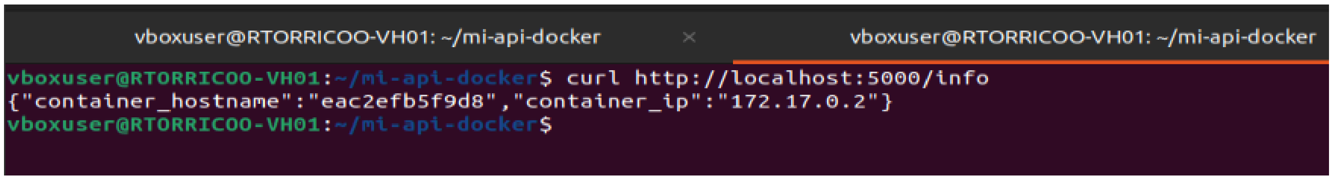


```
vboxuser@RTORRICOO-VH01:~/ml-api-docker$ docker build -t python-api .
[+] Building 25.6s (12/12) FINISHED                                      docker:default
 => [internal] load build definition from Dockerfile                              0.0s
 => => transferring dockerfile: 590B                                              0.0s
 => WARN: FromAsCasing: 'as' and 'FROM' keywords' casing do not match (line 2)    0.0s
 => [internal] load metadata for docker.io/library/python:3.9-slim                2.3s
 => [internal] load .dockerignore                                                 0.0s
 => => transferring context: 2B                                                   0.0s
 => [builder 1/4] FROM docker.io/library/python:3.9-slim@sha256:bef8d69306a7905f55cd52  14.0s
 => => resolve docker.io/library/python:3.9-slim@sha256:bef8d69306a7905f55cd523f5604de1  0.0s
 => => sha256:84b1a677eccc72224f1c1f6beece7da2221414861259d64d106a62d 14.93MB / 14.93MB  5.5s
 => => sha256:bef8d69306a7905f55cd523f5604de1dde45bbf745ba896dbb89f6d 10.41kB / 10.41kB  0.0s
 => => sha256:c8ecde0b63a4881272d794f7fcf2ba2ed4c0f7594c8d783905423e7e9 1.75kB / 1.75kB  0.0s
 => => sha256:9a041530811d5397b63cb7255cdcdd27195706cf5faf81d77242cd01d 5.29kB / 5.29kB  0.0s
 => => sha256:254e724d77862dc53abbd3bf0e27f9d2f64293909cdd3d0aad6a8fe 28.23MB / 28.23MB  6.5s
 => => sha256:d691b80d5159dfbcf2c1daad72a24c0fb9a55b6d90d6a11dcd46384ff 3.51MB / 3.51MB  2.8s
 => => sha256:fb95b45635cb4f373b9afdfb37bc513f1cc179ce6bd32904fb5570188eee2 249B / 249B  3.0s
 => => extracting sha256:254e724d77862dc53abbd3bf0e27f9d2f64293909cdd3d0aad6a8fe5a66806  4.0s
 => => extracting sha256:d691b80d5159dfbcf2c1daad72a24c0fb9a55b6d90d6a11dcd46384ff1afa6  0.4s
 => => extracting sha256:84b1a677eccc72224f1c1f6beece7da2221414861259d64d106a62d9b1f704  2.2s
 => => extracting sha256:fb95b45635cb4f373b9afdfb37bc513f1cc179ce6bd32904fb5570188eee2f  0.0s
 => [internal] load build context                                                 0.1s
 => => transferring context: 467B                                                 0.0s
 => [builder 2/4] WORKDIR /app                                                    0.6s
 => [builder 3/4] COPY requirements.txt .                                         0.1s
 => [builder 4/4] RUN pip install --user -r requirements.txt                      6.3s
```



```
vboxuser@RTORRICOO-VH01:~/mi-api-docker$ docker build --no-cache -t python-api .
[+] Building 11.2s (11/11) FINISHED                                                      docke
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 606B
 => [internal] load metadata for docker.io/library/python:3.9-slim
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/6] FROM docker.io/library/python:3.9-slim@sha256:bef8d69306a7905f55cd523f5604de1dde45bbf745b
 => CACHED [2/6] WORKDIR /app
 => [internal] load build context
 => => transferring context: 63B
 => [3/6] COPY requirements.txt .
 => [4/6] RUN pip install --no-cache-dir -r requirements.txt &&     apt-get update &&     apt-get i
 => [5/6] COPY app.py .
 => [6/6] RUN useradd -m myuser &&     chown -R myuser:myuser /app
 => exporting to image
 => => exporting layers
 => => writing image sha256:85918b599cb9b63b45fa3f96a86b3aa089c6041ad25076302cab0a3fbfe7e856
 => => naming to docker.io/library/python-api
vboxuser@RTORRICOO-VH01:~/mi-api-docker$ docker run -p 5000:5000 --name my-api python-api
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production W
 instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.17.0.2:5000
Press CTRL+C to quit
172.17.0.1 - - [17/May/2025 23:09:04] "GET /info HTTP/1.1" 200 -
172.17.0.1 - - [17/May/2025 23:09:46] "GET /info HTTP/1.1" 200 -
```

- Test the endpoint with curl to verify that it returns the correct information.


vboxuser@RTORRICOO-VH01: ~/mi-api-docker
vboxuser@RTORRICOO-VH01: ~/mi-api-docker

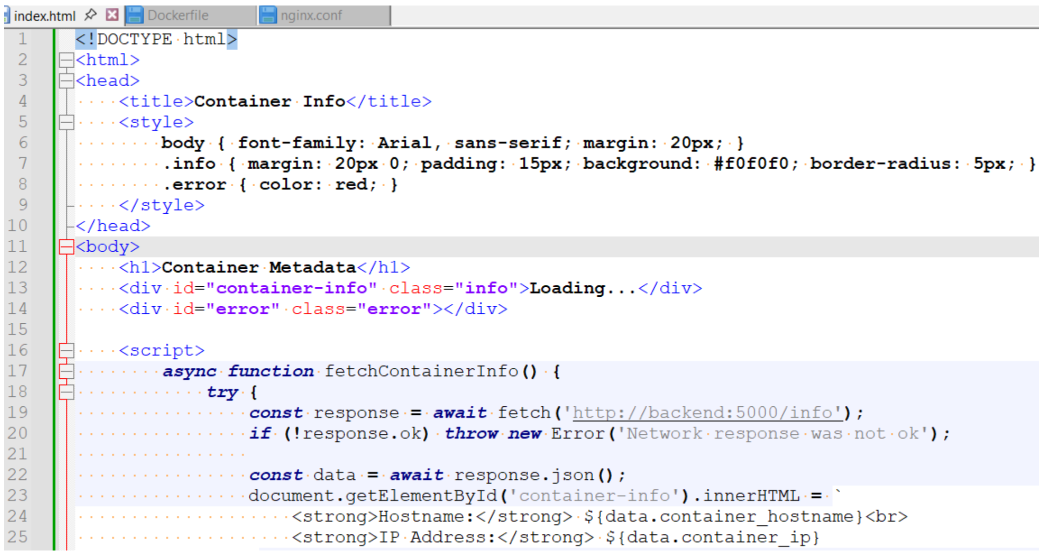
```
vboxuser@RTORRICOO-VH01:~/mi-api-docker$ curl http://localhost:5000/info
{"container_hostname":"eac2efb5f9d8","container_ip":"172.17.0.2"}
vboxuser@RTORRICOO-VH01:~/mi-api-docker$
```

- Ensure the API is not exposed to the host.

Exercise 2: Build and Containerize a Front-End Application

```
mi-proyecto-docker/
├── backend/
│   ├── app.py
│   ├── requirements.txt
│   └── Dockerfile          # Dockerfile para el backend
└── frontend/
    ├── index.html
    ├── nginx.conf
    └── Dockerfile          # Dockerfile para el frontend
```

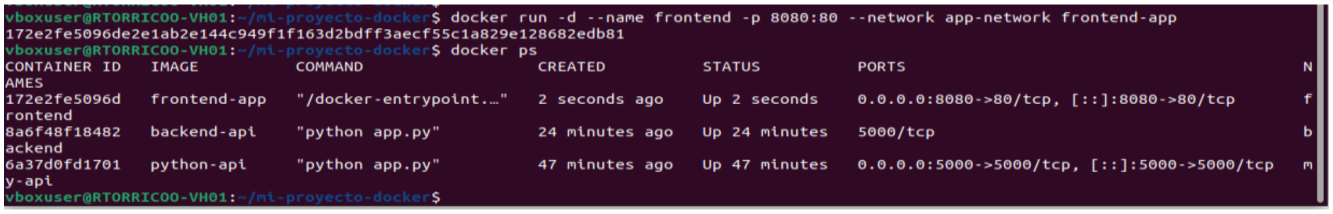
- Create a front-end application using HTML/JavaScript or a framework of your choice.



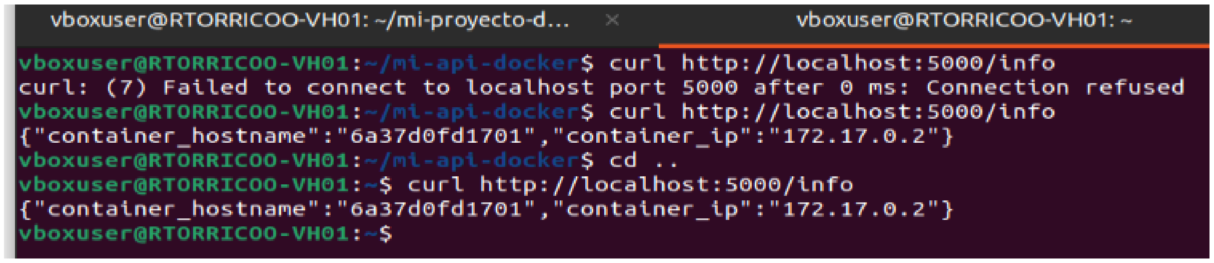
```html
<!DOCTYPE html>
<html>
<head>
    <title>Container Info</title>
    <style>
        body { font-family: Arial, sans-serif; margin: 20px; }
        .info { margin: 20px 0; padding: 15px; background: #f0f0f0; border-radius: 5px; }
        .error { color: red; }
    </style>
</head>
<body>
    <h1>Container Metadata</h1>
    <div id="container-info" class="info">Loading...</div>
    <div id="error" class="error"></div>

    <script>
        async function fetchContainerInfo() {
            try {
                const response = await fetch('http://backend:5000/info');
                if (!response.ok) throw new Error('Network response was not ok');

                const data = await response.json();
                document.getElementById('container-info').innerHTML = `
                    <strong>Hostname:</strong> ${data.container_hostname}<br>
                    <strong>IP Address:</strong> ${data.container_ip}
```

- The app must fetch the /info endpoint from the backend API and display the hostname and IP address.

```
# Construir la imagen del backend
docker build -t backend-api -f backend/Dockerfile backend/

# Construir la imagen del frontend
docker build -t frontend-app -f frontend/Dockerfile frontend/
```



```
vboxuser@RTORRICOO-VH01:~/mi-proyecto-docker$ docker run -d --name frontend -p 8080:80 --network app-network frontend-app
172e2fe5096de2e1ab2e144c949f1f163d2bdff3aecf55c1a829e128682edb81
vboxuser@RTORRICOO-VH01:~/mi-proyecto-docker$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                                         N
AMES
172e2fe5096d   frontend-app   "/docker-entrypoint.…"   2 seconds ago    Up 2 seconds    0.0.0.0:8080->80/tcp, [::]:8080->80/tcp       f
rontend
8a6f48f18482   backend-api    "python app.py"          24 minutes ago   Up 24 minutes   5000/tcp                                      b
ackend
6a37d0fd1701   python-api     "python app.py"          47 minutes ago   Up 47 minutes   0.0.0.0:5000->5000/tcp, [::]:5000->5000/tcp   m
y-api
vboxuser@RTORRICOO-VH01:~/mi-proyecto-docker$
```

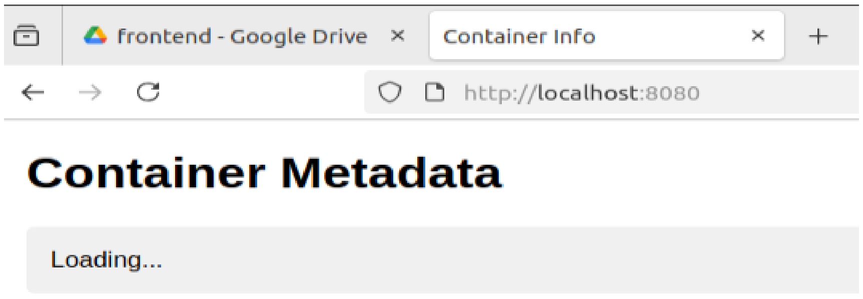

vboxuser@RTORRICOO-VH01: ~/mi-proyecto-d...
vboxuser@RTORRICOO-VH01: ~

```
vboxuser@RTORRICOO-VH01:~/mi-api-docker$ curl http://localhost:5000/info
curl: (7) Failed to connect to localhost port 5000 after 0 ms: Connection refused
vboxuser@RTORRICOO-VH01:~/mi-api-docker$ curl http://localhost:5000/info
{"container_hostname":"6a37d0fd1701","container_ip":"172.17.0.2"}
vboxuser@RTORRICOO-VH01:~/mi-api-docker$ cd ..
vboxuser@RTORRICOO-VH01:~$ curl http://localhost:5000/info
{"container_hostname":"6a37d0fd1701","container_ip":"172.17.0.2"}
vboxuser@RTORRICOO-VH01:~$
```

- Write a Dockerfile using multistage build to containerize and minimize the final image.
- Create a user-defined Docker network and run both frontend and backend containers within it.

```
# Crear la red Docker
docker network create app-network

# Ejecutar el backend en la red
docker run -d --name backend --network app-network backend-api

# Ejecutar el frontend (mapeando puerto 8080 del host al 80 del contenedor)
docker run -d --name frontend -p 8080:80 --network app-network frontend-app
```

- Verify in the browser that the frontend correctly shows the container metadata served by the Back End.

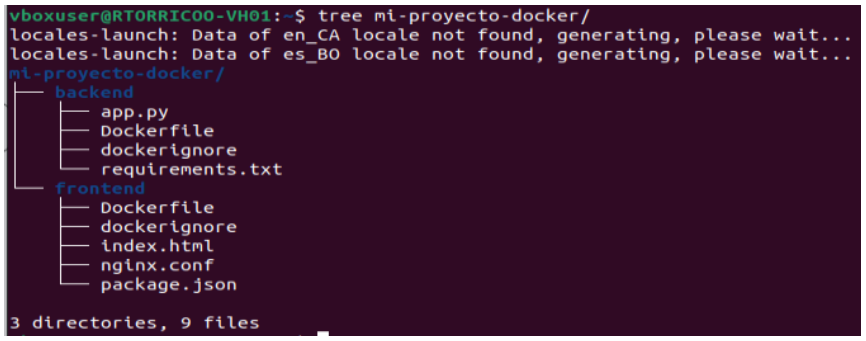


Container Metadata

Loading...

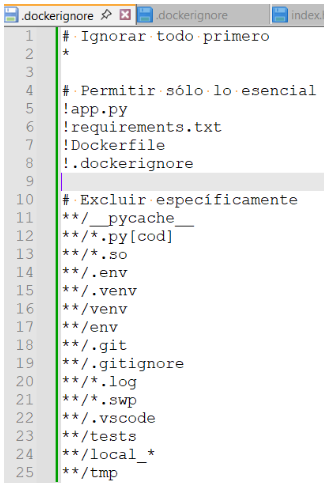
Exercise 3: The .dockerignore File

- Create a *.dockerignore* file in both Back End and Front End repos to exclude all *unnecessary* files and directories when building.



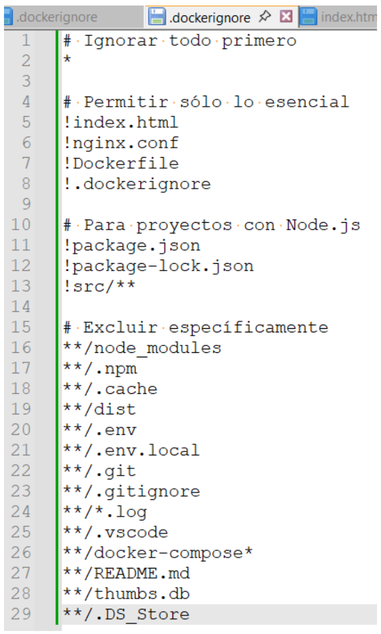
```
vboxuser@RTORRICO0-VH01:~$ tree mi-proyecto-docker/
locales-launch: Data of en_CA locale not found, generating, please wait...
locales-launch: Data of es_BO locale not found, generating, please wait...
mi-proyecto-docker/
├── backend
│   ├── app.py
│   ├── Dockerfile
│   ├── dockerignore
│   └── requirements.txt
└── frontend
    ├── Dockerfile
    ├── dockerignore
    ├── index.html
    ├── nginx.conf
    └── package.json

3 directories, 9 files
```

Backend file



```
# Ignorar todo primero
*

# Permitir sólo lo esencial
!app.py
!requirements.txt
!Dockerfile
!.dockerignore

# Excluir especificamente
**/__pycache__
**/*.py[cod]
**/*.so
**/.env
**/.venv
**/venv
**/env
**/.git
**/.gitignore
**/*.log
**/*.swp
**/.vscode
**/tests
**/local_*
**/tmp
```

Frontend file



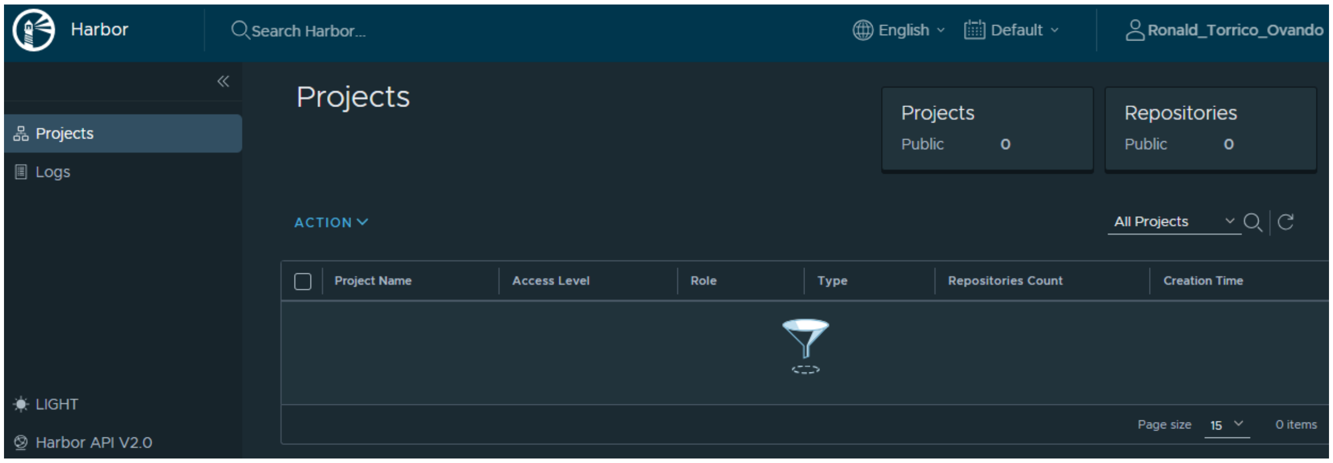
.dockerignore .dockerignore index.htm

```
# Ignorar todo primero
*

# Permitir sólo lo esencial
!index.html
!nginx.conf
!Dockerfile
!.dockerignore

# Para proyectos con Node.js
!package.json
!package-lock.json
!src/**

# Excluir específicamente
**/node_modules
**/.npm
**/.cache
**/dist
**/.env
**/.env.local
**/.git
**/.gitignore
**/*.log
**/.vscode
**/docker-compose*
**/README.md
**/thumbs.db
**/.DS_Store
```

Exercise 4: Private Registry

- Push the previously built Back End and Front End images to the private registry at *docker.jala.pro*
- Tag your images: *docker.jala.pro/docker-training/[CONTAINER-NAME=BackEnd || FrontEnd]:[TAG=FullName]*
- For Instance: *docker.jala.pro/docker-training/backend:calebespinoza*

Tuve algunos problemas con los permisos en el sitio web del repo



Projects

Projects	Repositories
Public 0	Public 0

Project Name	Access Level	Role	Type	Repositories Count	Creation Time

Page size 15 0 items