



Carrera:

Ingeniería en software

Integrantes:

Juan Daza
Julio Vincas
Christian Saraguro

Materia:

Practicas Graficas y Visualizaciones

Tema:

Contenedores

Docente:

Ing. Guillermo Cevallos

Nivel:

Tercer Semestre

Periodo: 2023-05

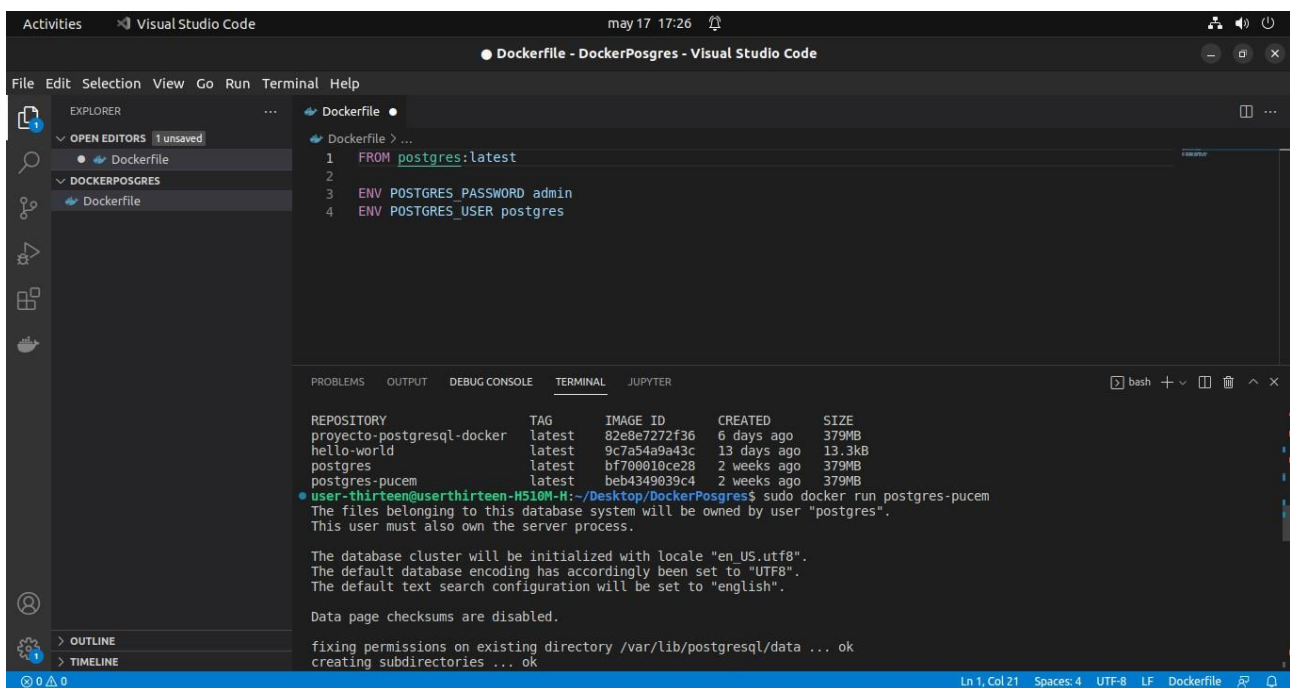
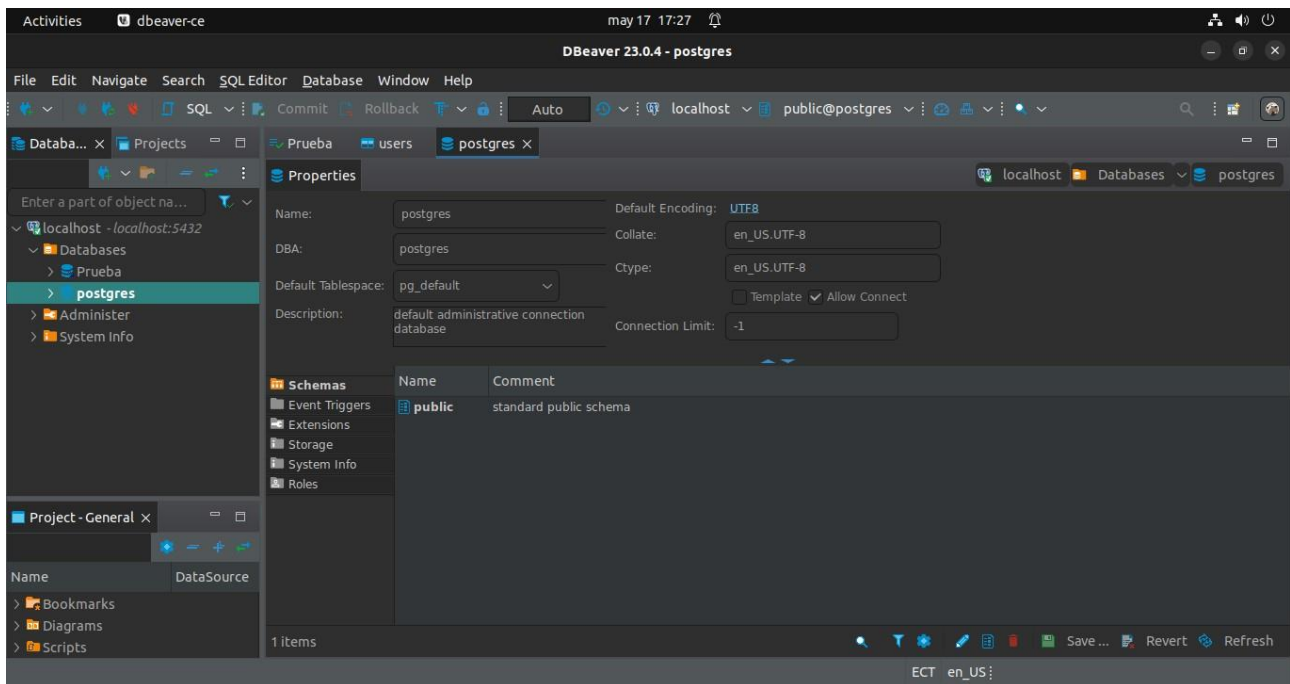
Contenedor de Posgrestr

Pasos:

1. Crear una carpeta nueva.
2. Dentro del Visual studio code abrir la carpeta y crear un. dockerfile.
3. Dentro del dockerfile definir las variables de entorno con la palabra determinada ENV.
4. En la terminal del Visual ejecutar los debidos comandos, uno para crear la imagen donde se creará el contenedor y luego el comando para crear el contenedor como tal.

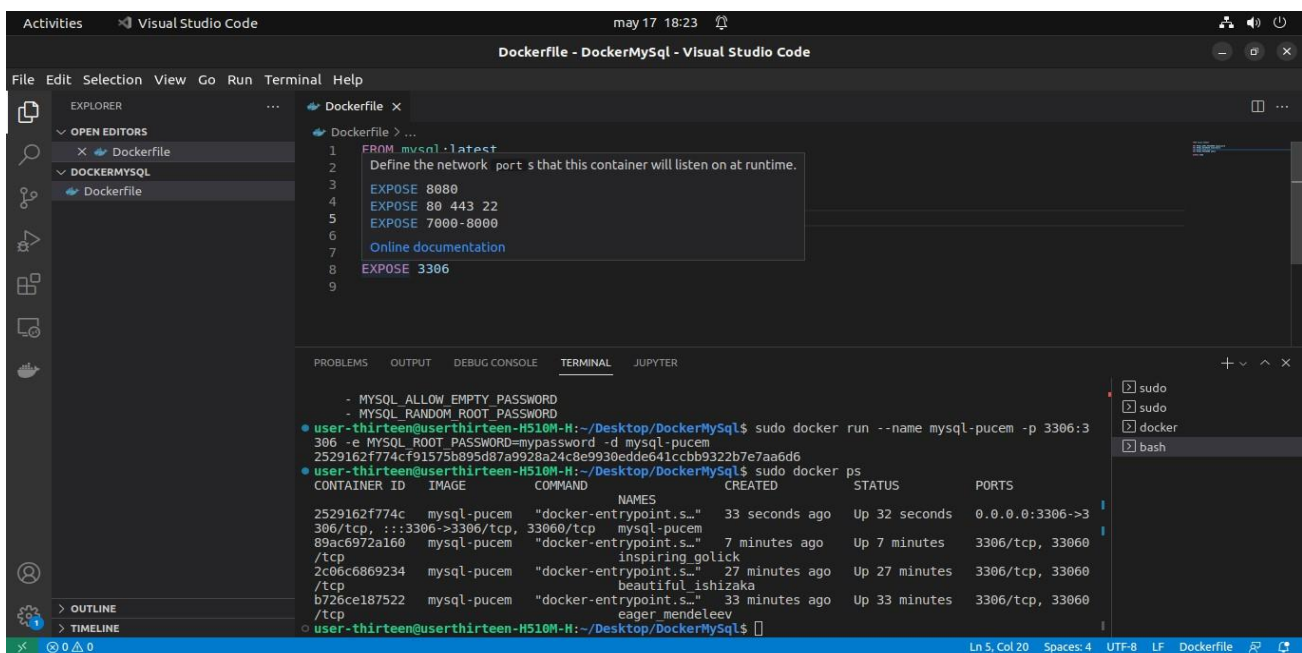
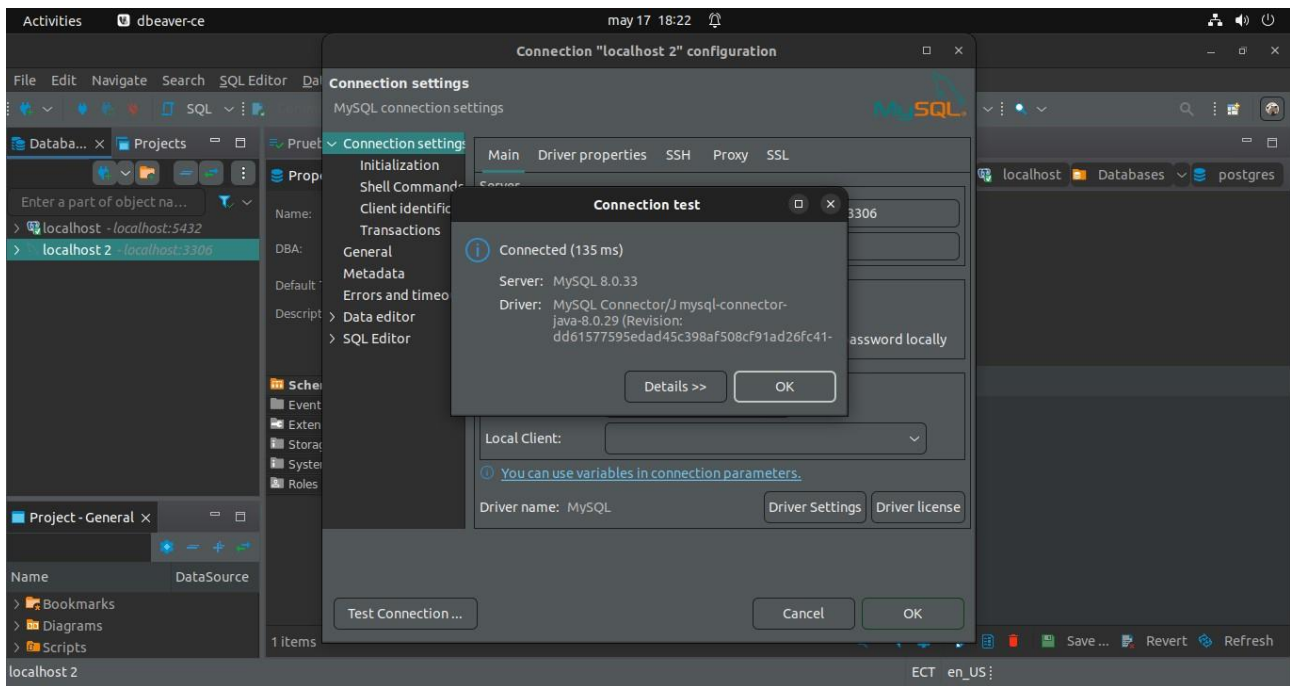
5. Luego de esto conectar con el gestor de base de datos de Dveaber.

Evidencias:



Contenedor de MySQL Pasos:

1. Crear una carpeta nueva.
2. Dentro del Visual studio code abrir la carpeta y crear un .dockerfile.
3. Dentro del dockerfile definir las variables de entorno con la palabra determinada ENV.
4. En la terminal del Visual ejecutar los debidos commands, uno para crear la imagen donde se creará el contendedor y luego el commando para crear el contendedor como tal.
5. Luego de esto conenctar con el gestor de base de datos de Dveaber. **Evidencias:**

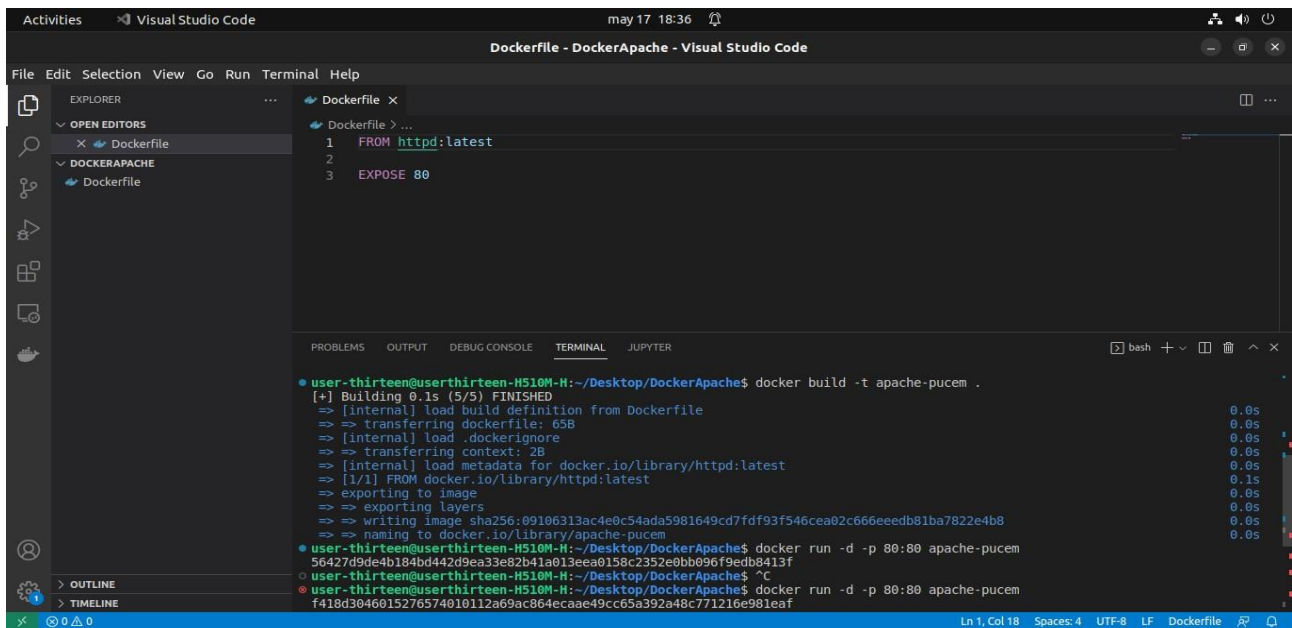


Contenedor de Apache

Pasos:

1. Crear una carpeta nueva.
2. Dentro del Visual studio code abrir la carpeta y crear un .dockerfile.
3. Dentro del dockerfile definir las variables de entorno con la palabra determinada ENV.
4. En la terminal del Visual ejecutar los debidos comandos, uno para crear la imagen donde se creará el contendedor y luego el comando para crear el contendedor como tal.
5. Crear un archive html para ejecutar el hola mundo.
6. Una vez todo construido ejecutar el contendedor y entrar en la ruta para verificar su funcionamiento.

Evidencias:

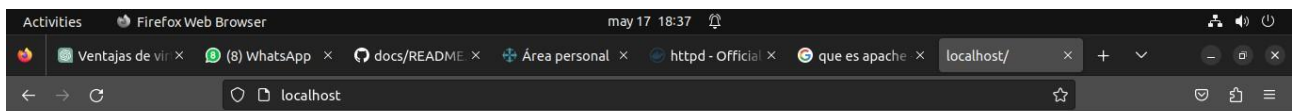


The screenshot shows the Visual Studio Code interface with a Dockerfile open in the editor. The Dockerfile contains the following instructions:

```
1 FROM httpd:latest
2
3 EXPOSE 80
```

The terminal window shows the output of the `docker build` and `docker run` commands. The build process is successful, and the container is running on port 80.

```
user-thirteen@userthirteen-H510M-H: ~/Desktop/DockerApache$ docker build -t apache-pucem .
[+] Building 0.1s (5/5) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 658B 0.0s
=> [internal] load .dockerignore
=> => transferring context: 2B 0.0s
=> [internal] load metadata for docker.io/library/httpd:latest 0.0s
=> [1/1] FROM docker.io/library/httpd:latest 0.1s
=> exporting to image 0.0s
=> => exporting layers 0.0s
=> => writing image sha256:09106313ac4e0c54ada5981649cd7fdf93f546cea02c666eedb81ba7822e4b8 0.0s
=> => naming to docker.io/library/apache-pucem 0.0s
user-thirteen@userthirteen-H510M-H: ~/Desktop/DockerApache$ docker run -d -p 80:80 apache-pucem
56427d9de4b184bd442d9ea35e82b41a013eea0158c2352e0bb096f9edb8413f
user-thirteen@userthirteen-H510M-H: ~/Desktop/DockerApache$ ^C
user-thirteen@userthirteen-H510M-H: ~/Desktop/DockerApache$ docker run -d -p 80:80 apache-pucem
f418d3046015276574010112a69ac864ecaae49cc65a392a48c771216e981eaf
```



It works!

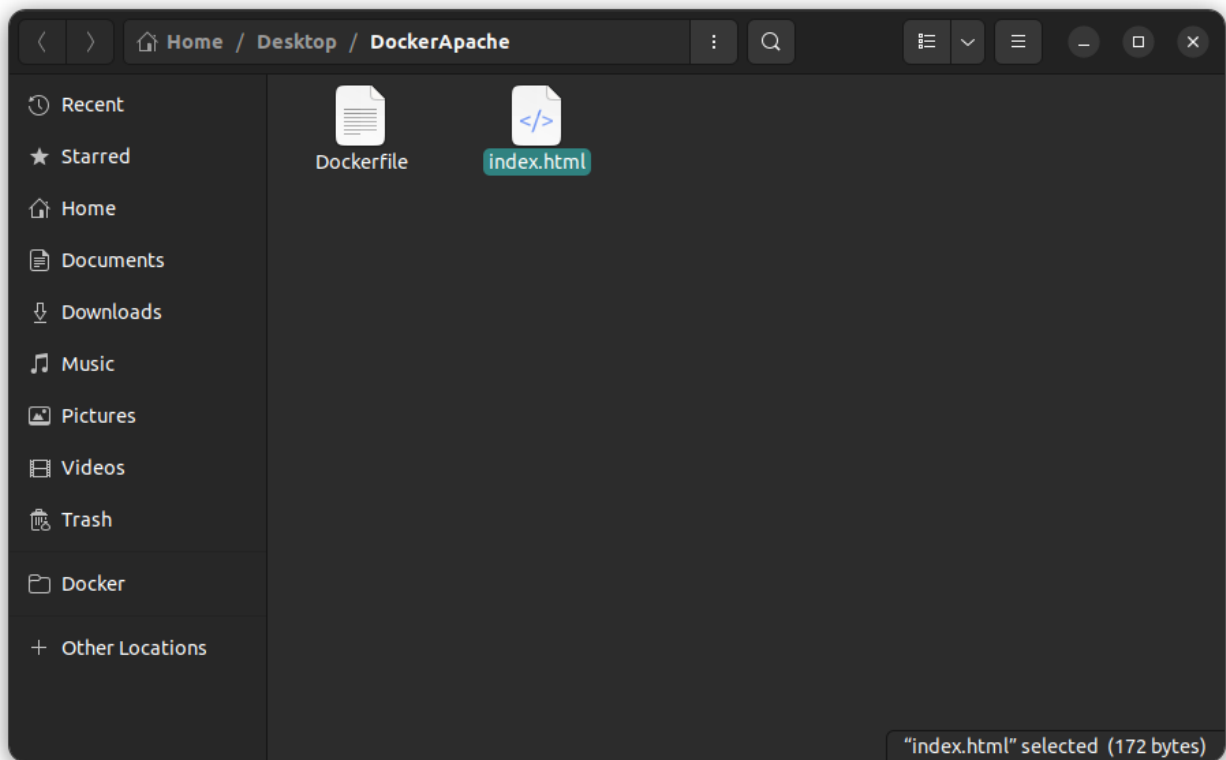
Contendor pagina web de apache “Hola Mundo”

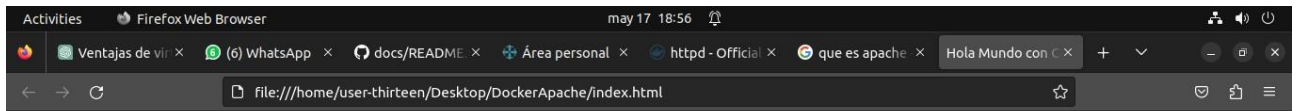
Visual Studio Code interface showing the Dockerfile and index.html files. The terminal output shows the Docker run command and the resulting container logs.

```
index.html - DockerApache - Visual Studio Code

1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Hola Mundo con Cambio de Color</title>
5 </head>
6 <body>
7   <h1>Hola Mundo, Grupo Juan Daza y Julio Vines
8   </h1>
9 </body>
10 </html>
11
```

```
bash
c9aed1a0936d apache-pucem "httpd-foreground" 4 minutes ago Up 4 minutes 0.0.0.0:80->80/tcp, :::80->80/tcp
2529162f774c mysql-pucem "docker-entrypoint.s..." 40 minutes ago Up 40 minutes 0.0.0.0:3306->3306/tcp, :::3306->
3306/tcp, 33060/tcp mysql-pucem
89ac6972a160 mysql-pucem "docker-entrypoint.s..." 47 minutes ago Up 47 minutes 3306/tcp, 33060/tcp
2c06c0869234 mysql-pucem "docker-entrypoint.s..." About an hour ago Up About an hour 3306/tcp, 33060/tcp
b726ce187522 mysql-pucem "docker-entrypoint.s..." About an hour ago Up About an hour 3306/tcp, 33060/tcp
eager_mendelev
user-thirteen@userthirteen-H510M-H:~/Desktop/DockerApache$ docker run -d -p 80:80 apache-pucem
6e0657f3fec4eb3954e79e861ad0c6a1a1217c96c2d9c479e0c4b7c60336cd7
docker: Error response from daemon: driver failed programming external connectivity on endpoint eager_murdock (92d7a0fdd5227692
2f4959464e7373e281432a2afad0d12552912e7ee0fb0258): Bind for 0.0.0.0:80 failed: port is already allocated.
user-thirteen@userthirteen-H510M-H:~/Desktop/DockerApache$ docker cp index.html 56427d9de4b1:/usr/local/apache2/htdocs/
Successfully copied 2.05kB to 56427d9de4b1:/usr/local/apache2/htdocs/
user-thirteen@userthirteen-H510M-H:~/Desktop/DockerApache$
```



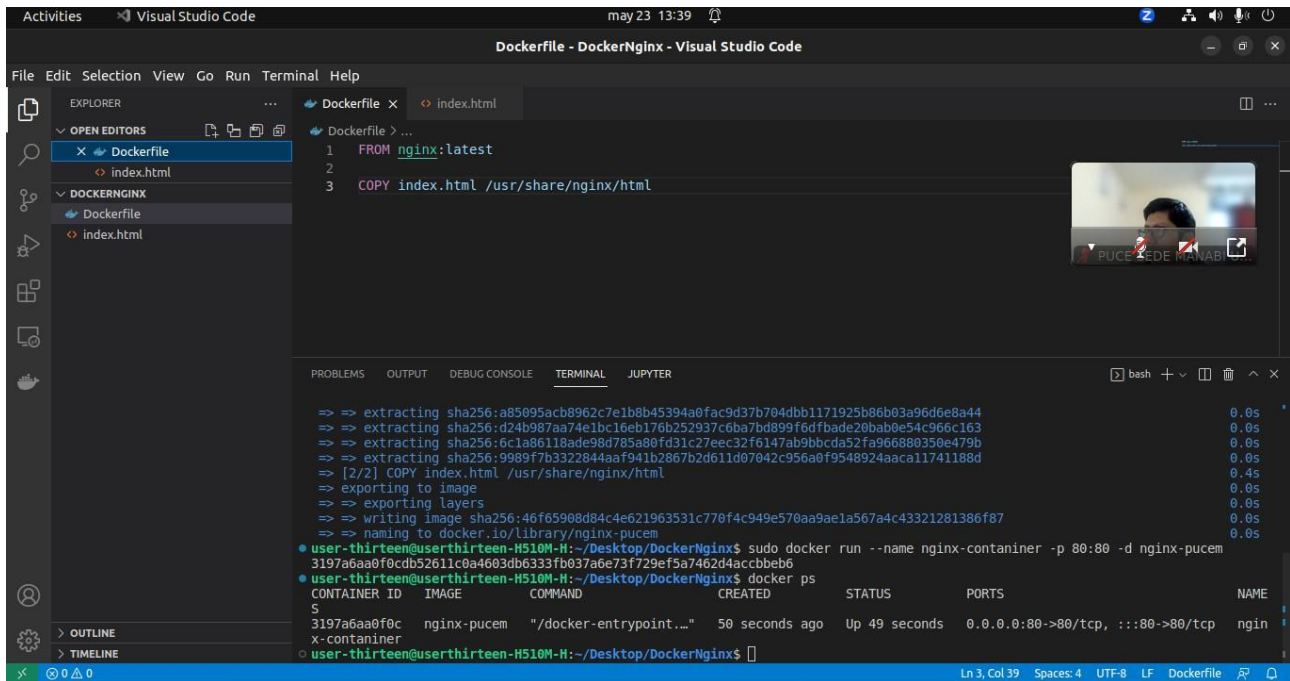


Hola Mundo, Grupo Juan Daza y Julio Vincés

Contenedor de Nginx Pasos:

1. Crear una carpeta nueva.
2. Dentro del Visual studio code abrir la carpeta y crear un .dockerfile.
3. Dentro del dockerfile definir las variables de entorno con la palabra determinada ENV.
4. En la terminal del Visual ejecutar los debidos comandos, uno para crear la imagen donde se creará el contendedor y luego el comando para crear el contenedor como tal.
5. Luego crear un archive html y editarlo para poder evidenciar que el funcionamiento del contendedor es óptimo.

Evidencias:

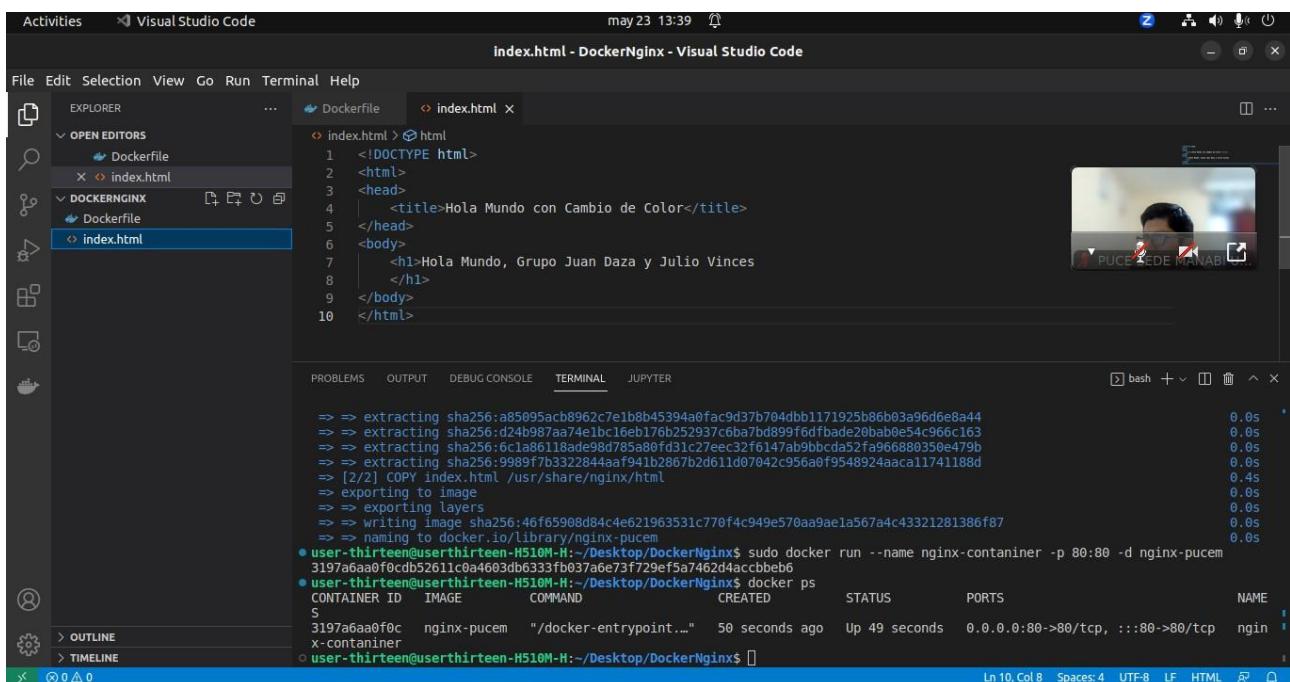


The screenshot shows the Visual Studio Code interface with a Dockerfile being edited. The Dockerfile contains the following commands:

```
1 FROM nginx:latest
2
3 COPY index.html /usr/share/nginx/html
```

The terminal output shows the process of building the image and running the container:

```
=> extracting sha256:a85095acb8962c7e1b8b45394a0fac9d37b704ddb1171925b86b03a96d6e8a44 0.0s
=> extracting sha256:d24b987aa74e1bc16eb176b252937c6ba7bd899f6dfbade20bab0e54c966c163 0.0s
=> extracting sha256:6c1a86118ade98d785a80fd31c27eec32f6147ab9bbcd52fa966880350e479b 0.0s
=> extracting sha256:9989f7b3322844aaf941b2867b2d611d07042c956a0f9548924aaca11741188d 0.0s
=> [2/2] COPY index.html /usr/share/nginx/html 0.4s
=> exporting to image 0.0s
=> exporting layers 0.0s
=> writing image sha256:46f65908d84c4e621963531c770f4c949e570aa9aela567a4c43321281386f87 0.0s
=> naming to docker.io/library/nginx-pucem 0.0s
user-thirteen@userthirteen-H510M-H:~/Desktop/DockerNginx$ sudo docker run --name nginx-container -p 80:80 -d nginx-pucem
3197a6aa0f0c52611c0a4603db6333fb037a6e73f729ef5a7462d4accbbeb6
user-thirteen@userthirteen-H510M-H:~/Desktop/DockerNginx$ docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS                               NAME
3197a6aa0f0c   nginx-pucem "/docker-entrypoint..." 50 seconds ago Up 49 seconds 0.0.0.0:80->80/tcp, :::80->80/tcp   nginx-container
user-thirteen@userthirteen-H510M-H:~/Desktop/DockerNginx$
```



The screenshot shows the Visual Studio Code interface with the index.html file being edited. The index.html file contains the following HTML code:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Hola Mundo con Cambio de Color</title>
5 </head>
6 <body>
7   <h1>Hola Mundo, Grupo Juan Daza y Julio Vincas
8 </h1>
9 </body>
10 </html>
```

The terminal output shows the process of building the image and running the container:

```
=> extracting sha256:a85095acb8962c7e1b8b45394a0fac9d37b704ddb1171925b86b03a96d6e8a44 0.0s
=> extracting sha256:d24b987aa74e1bc16eb176b252937c6ba7bd899f6dfbade20bab0e54c966c163 0.0s
=> extracting sha256:6c1a86118ade98d785a80fd31c27eec32f6147ab9bbcd52fa966880350e479b 0.0s
=> extracting sha256:9989f7b3322844aaf941b2867b2d611d07042c956a0f9548924aaca11741188d 0.0s
=> [2/2] COPY index.html /usr/share/nginx/html 0.4s
=> exporting to image 0.0s
=> exporting layers 0.0s
=> writing image sha256:46f65908d84c4e621963531c770f4c949e570aa9aela567a4c43321281386f87 0.0s
=> naming to docker.io/library/nginx-pucem 0.0s
user-thirteen@userthirteen-H510M-H:~/Desktop/DockerNginx$ sudo docker run --name nginx-container -p 80:80 -d nginx-pucem
3197a6aa0f0c52611c0a4603db6333fb037a6e73f729ef5a7462d4accbbeb6
user-thirteen@userthirteen-H510M-H:~/Desktop/DockerNginx$ docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS                               NAME
3197a6aa0f0c   nginx-pucem "/docker-entrypoint..." 50 seconds ago Up 49 seconds 0.0.0.0:80->80/tcp, :::80->80/tcp   nginx-container
user-thirteen@userthirteen-H510M-H:~/Desktop/DockerNginx$
```

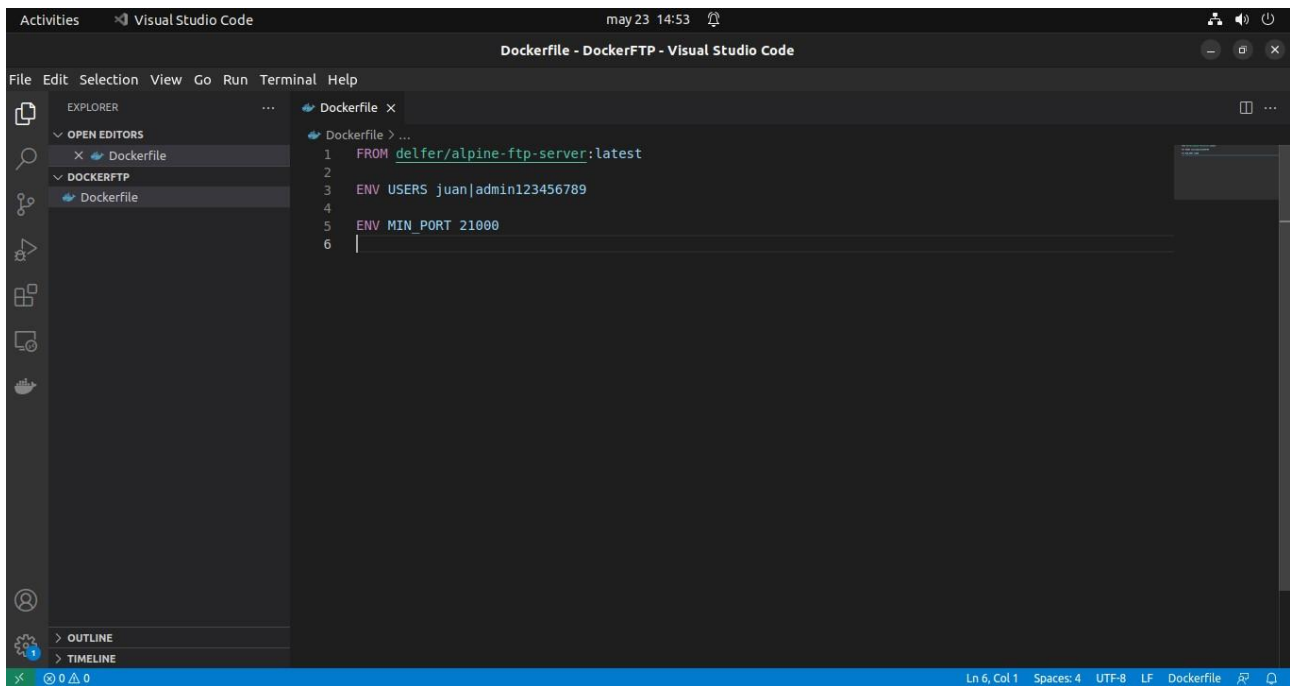
Hola Mundo, Grupo Juan Daza y Julio Vines



Contenedor de FTP

Pasos:

1. Crear una carpeta nueva.
2. Dentro del Visual studio code abrir la carpeta y crear un .dockerfile.
3. Dentro del dockerfile definir las variables de entorno con la palabra determinada ENV.
4. En la terminal del Visual ejecutar los debidos comandos, uno para crear la imagen donde se creará el contendedor y luego el comando para crear el contenedor como tal.
5. Mediante un comando logramos acceder a la ip del contenedor.
6. Teniendo acceso a la ip nos conectamos al localhost de FTP.
7. Al conectarnos al servidor, nos pedirá ingresar usuario y contraseña los cuales ya asignamos en el dockerfile.
8. Luego nos mostrará un mensaje de verificación y con eso evidenciamos la conexión.



The screenshot shows the Visual Studio Code interface with a Dockerfile open in the editor. The Dockerfile contains the following content:

```
1 FROM delfer/alpine-ftp-server:latest
2
3 ENV USERS juan|admin123456789
4
5 ENV MIN_PORT 21000
6 |
```

The interface includes a sidebar with 'EXPLORER' and 'DOCKERTFTP' sections, and a bottom status bar indicating 'Ln 6, Col 1', 'Spaces: 4', 'UTF-8', 'LF', and 'Dockerfile'.

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

ftp + - [] ^ x

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
8e0c9e5ba783	delfer/alpine-ftp-server:latest	"/sbin/tini -- /bin/_"	5 minutes ago	Up 5 minutes	0.0.0.0:20-21->20-21/tcp, :::20-21->20-21/tcp, 0.0.0.0:21000-21010->21000-21010/tcp, :::21000-21010->21000-21010/tcp
					inspiring_borg

```
user-thirteen@userthirteen-H510M-H:~/Desktop/DockerFTP$ docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' 8e0c9e5ba783
172.17.0.2
user-thirteen@userthirteen-H510M-H:~/Desktop/DockerFTP$ ftp 172.17.0.2 21
Connected to 172.17.0.2.
220 Welcome Alpine ftp server https://hub.docker.com/r/delfer/alpine-ftp-server/
Name (172.17.0.2:user-thirteen): juan
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

Ln 2, Col 1 Spaces: 4 UTF-8 LF Dockerfile AP Q