

Contenido

Introducción	2
Recomendaciones	2
1 etapa: construcción y levantamiento del proyecto	2
Descargar el código	2
Acceder a la carpeta raíz.....	3
Ejecutar archivos docker en la carpeta raíz	3
Construcción de la imagen	4
Levantamiento del contenedor	4
2 etapa: Edición de código en entorno aislado	5
Abrir la carpeta raíz en vsc	5
Extensión Dev Container	5
Abrir proyecto en un Dev Container	6
Selección del intérprete de Python	6

Introducción

Para ejecutar el proyecto por primera vez, el usuario debe completar dos etapas que constan de un total de nueve pasos.

La primera etapa consiste en la construcción y despliegue del **Docker Compose** del proyecto, lo que permitirá acceder a la plataforma o a la vista inicial de la página.

La segunda etapa corresponde a la ejecución del **Dev Container**, que le permitirá al usuario editar el código según sea necesario. Todo esto se ejecutará en un entorno aislado, sin requerir configuraciones adicionales en el dispositivo local.

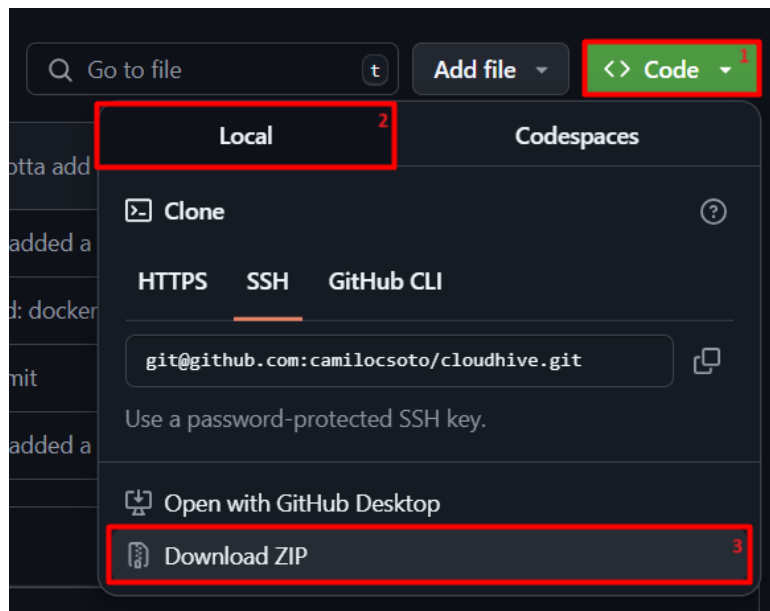
Recomendaciones

Antes de comenzar, es fundamental contar con dos herramientas instaladas en el entorno local: **Docker Desktop**, **Visual Studio Code** y una terminal de comandos **Shell** u otra. Son fáciles de instalar y esenciales para el correcto funcionamiento del proyecto.

1 etapa: construcción y levantamiento del proyecto

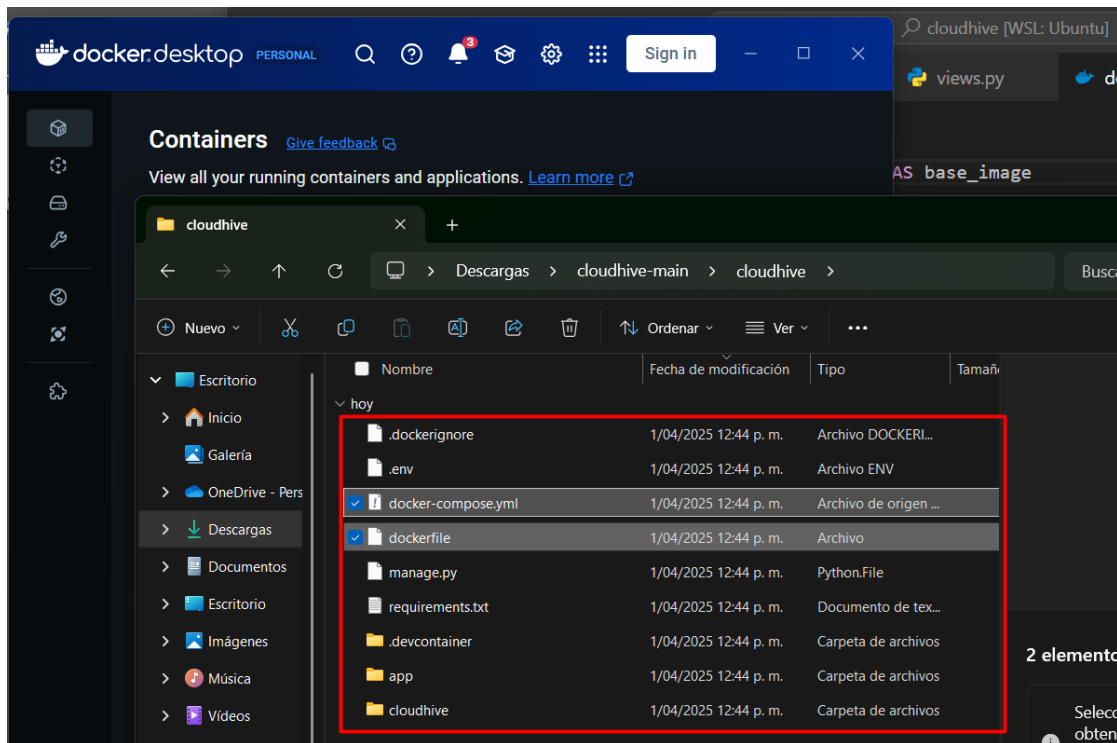
Durante esta primera etapa, usted entenderá el proceso para desplegar el proyecto en su entorno local (*localhost:8181*)

Descargar el código: Para obtener el código, diríjase al repositorio de **GitHub**, donde se encuentra alojado el proyecto, y descargue el archivo **.zip** en su PC, siguiendo los pasos que se muestran a continuación.



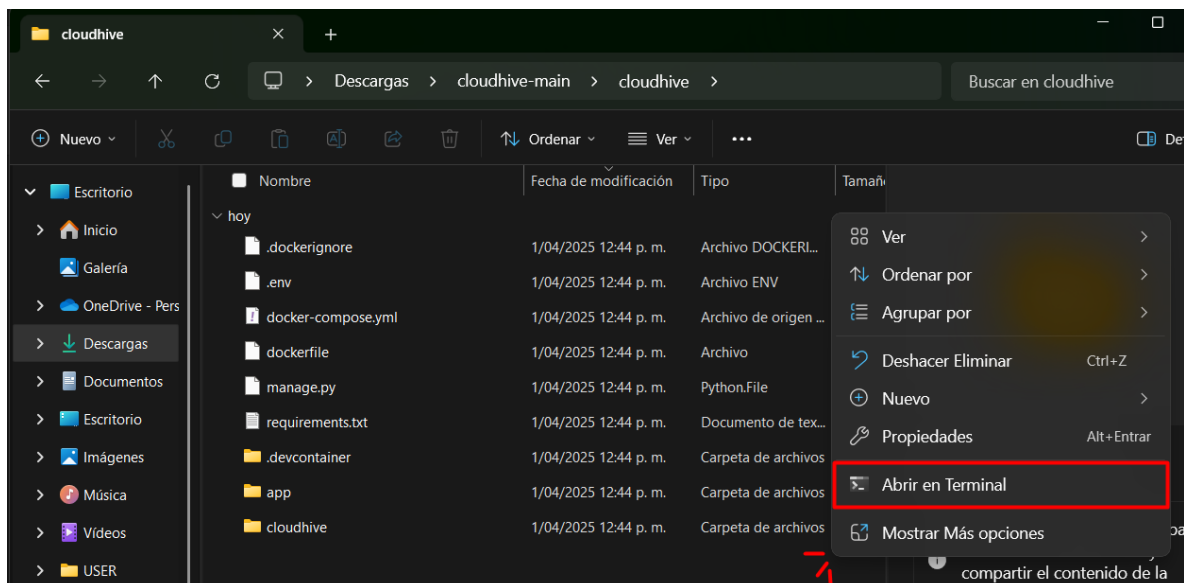
Acceder a la carpeta raíz: Descomprima el archivo `.zip` descargado y acceda a la carpeta resultante. Dentro de ella, debería encontrar archivos como `docker-compose.yml` o `Dockerfile`. Si estos archivos están visibles, significa que está en la ubicación correcta.

Nota: Asegúrese que `Docker Desktop` esté en ejecución en segundo plano; simplemente debe mantener la aplicación abierta.



Ejecutar archivos docker en la carpeta raíz: En el directorio raíz, haga clic derecho y seleccione la opción `Abrir en terminal`.

Nota: Si esta opción no está disponible, copie la ruta desde el explorador de archivos y abra la terminal de su preferencia, luego navegue hasta la ruta copiada.



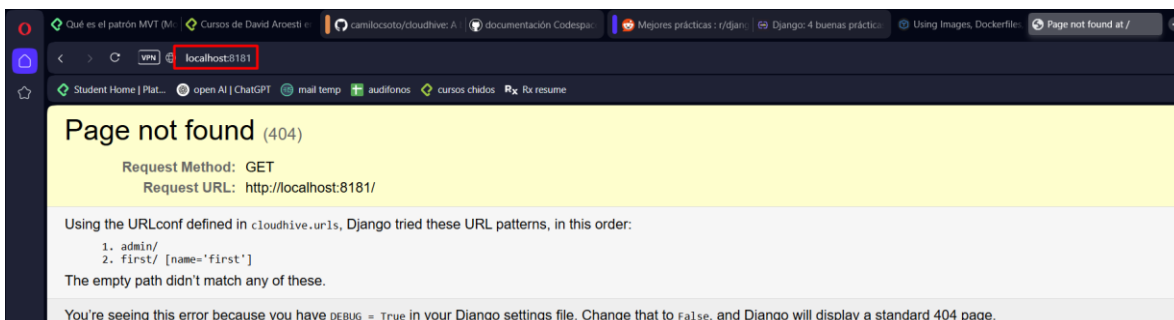
Construcción de la imagen: Para construir la imagen de docker que contiene Django y PostgreSQL, ejecute el siguiente comando en la terminal: `docker-compose build`, este proceso puede tardar entre 1 y 5 minutos.

```
nodens@DESKTOP-J9NHF2Q: /mnt/c/Users/USER/Downloads/cloudhive-main/cloudhive$ docker-compose build
[+] Building 0/1
[+] Building 0/10-web Building
[+] Building 33.0s (15/15) FINISHED
=> [django-web internal] load build definition from dockerfile
=> transferring dockerfile: 1.32kB
=> [django-web internal] load metadata for docker.io/library/python:3.12-slim
=> [django-web internal] load .dockerignore
=> transferring context: 257B
=> [django-web internal] load build context
=> transferring context: 879B
=> [django-web base_image 1/5] FROM docker.io/library/python:3.12-slim@sha256:a866731a6b71c4a194a845d86e06568725
=> CACHED [django-web base_image 2/5] RUN apt-get update && apt-get install -y libpq-dev gcc
=> CACHED [django-web base_image 3/5] RUN python3 -m venv /opt/venv
=> CACHED [django-web base_image 4/5] COPY requirements.txt /requirements.txt
=> [django-web build_image 5/5] RUN pip install --no-cache-dir -r requirements.txt
=> CACHED [django-web build_image 2/5] RUN apt-get update && apt-get install -y libpq-dev python3-dev
=> [django-web build_image 3/5] COPY --from=base_image /opt/venv /opt/venv
=> [django-web build_image 4/5] COPY . /usr/src/app
=> [django-web build_image 5/5] WORKDIR /usr/src/app
=> [django-web] exporting to image
=> exporting layers
=> writing image sha256:962f9ed6b3bf19107f9d13740e3c1ab088f07537df53a6760e3ea054368eb394b
[+] Building 1/1 docker.io/library/cloudhive-django-web
Service django-web Built
```

Levantamiento del contenedor: Para iniciar el contenedor de Docker que gestionará los servicios de Django y PostgreSQL, ejecute el siguiente comando en la terminal: `docker-compose up`. Este proceso puede tardar entre 1 y 3 minutos.

```
nodens@DESKTOP-J9NHF2Q: /mnt/c/Users/USER/Downloads/cloudhive-main/cloudhive$ docker-compose up
[+] Running 2/2
✔ Container cloudhive-db-1 Created
✔ Container django-docker Recreated
Attaching to db-1, django-docker
db-1 | PostgreSQL Database directory appears to contain a database; Skipping initialization
db-1 |
db-1 | 2025-04-01 18:19:35.288 UTC [1] LOG: starting PostgreSQL 17.4 (Debian 17.4-1.pgdg120+2) on x86_64-pc-
linux-gnu, compiled by gcc (Debian 12.2.0-14) 12.2.0, 64-bit
db-1 | 2025-04-01 18:19:35.301 UTC [1] LOG: listening on IPv4 address "0.0.0.0", port 5432
db-1 | 2025-04-01 18:19:35.301 UTC [1] LOG: listening on IPv6 address ":::", port 5432
db-1 | 2025-04-01 18:19:35.310 UTC [1] LOG: listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
db-1 | 2025-04-01 18:19:35.321 UTC [29] LOG: database system was interrupted; last known up at 2025-04-01 17
:26:32 UTC
db-1 | 2025-04-01 18:19:35.674 UTC [29] LOG: database system was not properly shut down; automatic recovery
in progress
db-1 | 2025-04-01 18:19:35.680 UTC [29] LOG: redo starts at 0/1909270
db-1 | 2025-04-01 18:19:35.680 UTC [29] LOG: invalid record length at 0/1909378: expected at least 24, got 0
db-1 | 2025-04-01 18:19:35.680 UTC [29] LOG: redo done at 0/1909340 system usage: CPU: user: 0.00 s, system:
0.00 s, elapsed: 0.00 s
db-1 | 2025-04-01 18:19:35.688 UTC [27] LOG: checkpoint starting: end-of-recovery immediate wait
db-1 | 2025-04-01 18:19:35.717 UTC [27] LOG: checkpoint complete: wrote 3 buffers (0.0%); 0 WAL file(s) adde
d, 0 removed, 0 recycled; write=0.008 s, sync=0.004 s, total=0.033 s; sync files=2, longest=0.002 s, average=0.002 s; d
istance=0 kB, estimate=0 kB; lsn=0/1909378, redo lsn=0/1909378
db-1 | 2025-04-01 18:19:35.736 UTC [1] LOG: database system is ready to accept connections
django-docker | [2025-04-01 18:19:36 +0000] [1] [INFO] Starting gunicorn 20.1.0
django-docker | [2025-04-01 18:19:36 +0000] [1] [INFO] Listening at: http://0.0.0.0:8181 (1)
django-docker | [2025-04-01 18:19:36 +0000] [1] [INFO] Using worker: sync
```

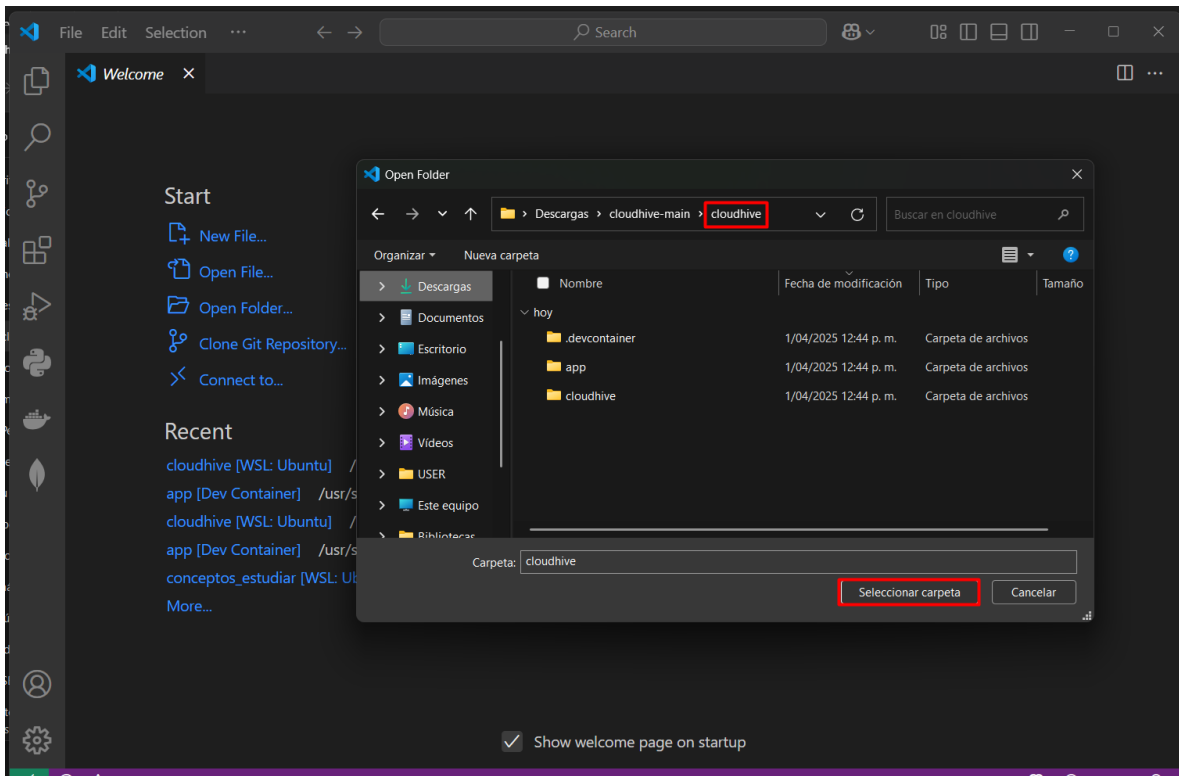
Si en un navegador accede a `localhost:8181` usted podrá ver la interfaz de django:



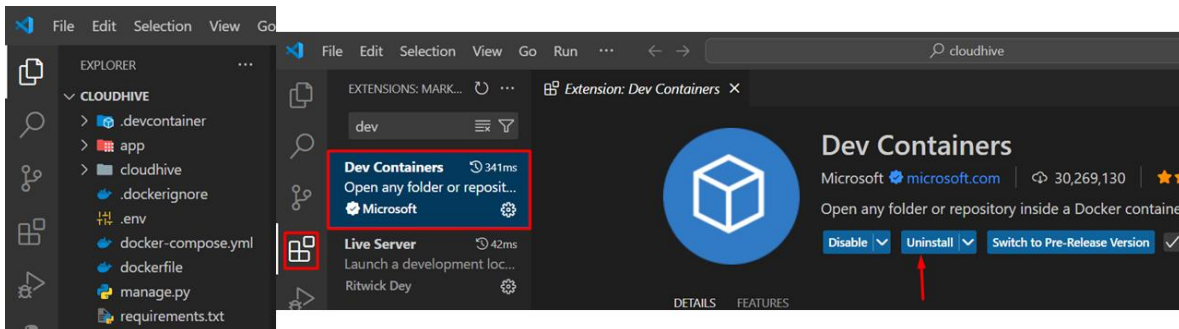
2 etapa: Edición de código en entorno aislado

En esta segunda etapa, aprenderá a modificar el código del proyecto en un entorno aislado, sin necesidad de instalar SDKs, librerías u otras dependencias, ya que todo está previamente configurado.

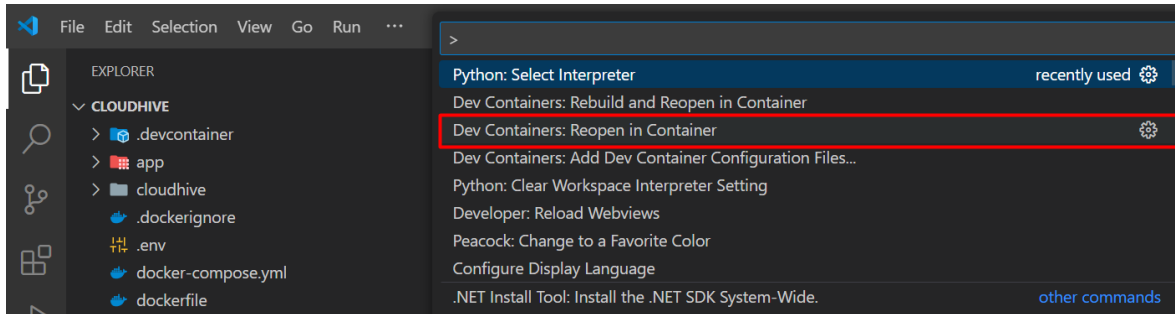
Abrir la carpeta raíz en vsc: Para ello, debe abrir su visual studio code y en la zona superior izquierda seleccionar File > Open Folder. Debe ubicar la carpeta raíz que contiene los directorios clouddhive, app y .devcontainer como se observa a continuación



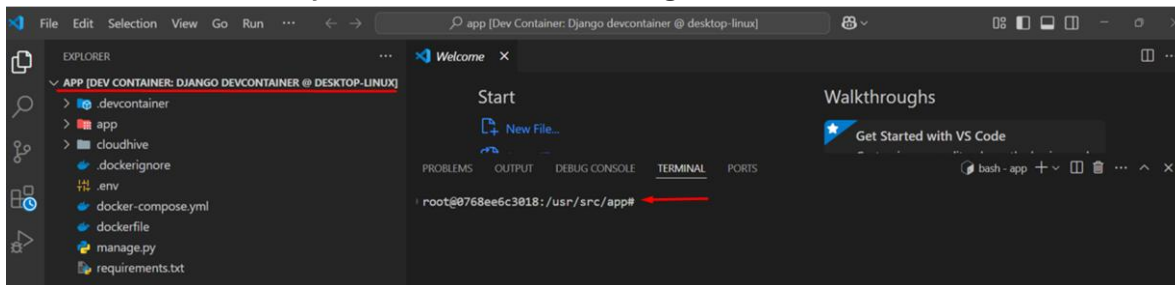
Extensión Dev Container: Una vez abierto el directorio, podrá visualizar una serie de archivos en la parte izquierda de la pantalla. A continuación, diríjase al menú izquierdo, seleccione la opción **Extensiones** y busque **Dev Containers**. Asegúrese de que esta extensión esté instalada ya que es el único requisito adicional.



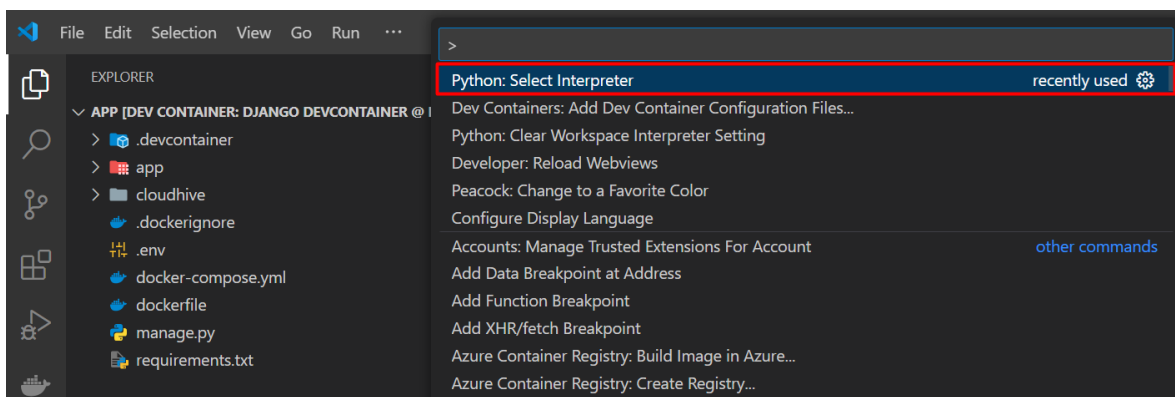
Abrir proyecto en un Dev Container: Para iniciar el entorno aislado debe presionar las teclas: **Ctrl + Shift + P**, para abrir la barra de comandos. Luego, busque y seleccione **Dev Containers: Reopen in container**. Si esta opción no aparece, puede elegir **Dev Containers: Rebuild and reopen in container** aunque este proceso puede ser más lento, ya que reconstruye la imagen de Docker y la conexión con el contenedor puede tardar un poco más en establecerse.



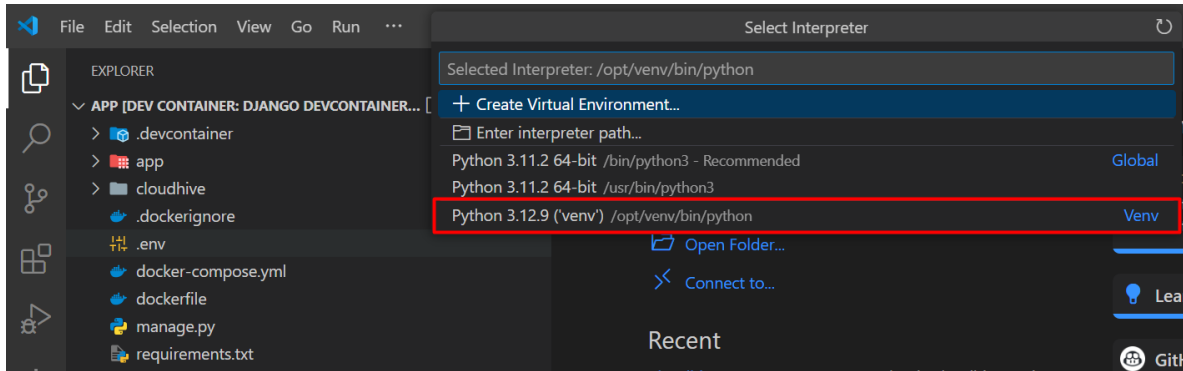
Sabrás que la conexión con el **Dev Container** se ha completado cuando el nombre del directorio raíz cambie y la terminal muestre la siguiente ruta:



Selección del intérprete de Python: Seleccione el entorno virtual donde se encuentran todas las librerías del proyecto. Para ello, presione las teclas: **Ctrl + Shift + P**, para abrir la barra de comandos, luego busque y seleccione la opción **Python: Select interpreter**



A continuación, aparecerá una ventana que indicará el entorno de Python a utilizar en el proyecto. Seleccione `Python 3.12.9 ('venv') /opt/venv/bin/python`, el entorno virtual que opera dentro del contenedor:



Al haber llegado hasta este punto, usted podrá editar lo que considere necesario en el proyecto de django + postgresQL.

Ficha del documento

Fecha	Revisión	Autor	Revisado por
Marzo 2025	1	Juan Camilo Suárez Soto	Edward Chaparro Andrés Gil Giovanni Pinto