To initialise the docker container with a postgres image:

~/Documents/Data Engineering/Learning/Data Zoomcamp 2025/Week 1/2\_docker\_sql docker run -it \

*--*name ny\_taxi\_container \

  -e POSTGRES\_USER="root" \

  -e POSTGRES\_PASSWORD="root" \

  -e POSTGRES\_DB="ny\_taxi" \

  -v "/c/Users/Utilizador/Documents/Data Engineering/Learning/Data Zoomcamp 2025/Week 1/2\_docker\_sql/ny\_taxi\_postgres:/var/lib/postgresql/data" \

  -p 5433:5432 \

  postgres:17.4

running ingest.py locally (using git bash)

PYTHONUNBUFFERED=1 python ingest\_data.py \

--user root \

--password root \

--host localhost \

--port 5433 \

--db ny\_taxi \

--table\_name yellow\_taxi\_data \

--url <https://d37ci6vzurychx.cloudfront.net/trip-data/yellow_tripdata_2025-01.parquet>

Building docker image for python ingest file

docker build -t ny\_taxi\_ingest -f ingest.Dockerfile .

then running it

docker run --rm -e PYTHONUNBUFFERED=1 ny\_taxi\_ingest \

--user root \

--password root \

--host host.docker.internal \

--port 5433 \

--db ny\_taxi \

--table\_name yellow\_taxi\_data \

--url <https://d37ci6vzurychx.cloudfront.net/trip-data/yellow_tripdata_2025-01.parquet>

Explaining using py ingest in a container:

it is inserting the rows now. If I want to close this and re start it how would I do it? do I ahve two containers now?

**Dito pelo ChatGPT:**

Great that it’s inserting rows now! 🎉

**About stopping and restarting:**

* **If you used docker run --rm ...** (with --rm flag), the container **is removed automatically after it finishes** running. So it doesn’t persist.
* If you want to **stop a running container** (without --rm), you can do:

bash

CopiarEditar

docker stop <container\_name\_or\_id>

* And to **restart** that stopped container:

bash

CopiarEditar

docker start <container\_name\_or\_id>

**Do you have two containers now?**

* Every time you run docker run without --name or with a different --name, **Docker creates a new container** based on the image.
* So if you run the command multiple times, yes, you may have multiple containers.

You can check all containers (running and stopped) with:

bash

CopiarEditar

docker ps -a

**What should you do?**

* For your ingest job (which runs once and exits), using --rm is good because it cleans up containers automatically.
* For your **Postgres database container**, you want it to keep running and data to persist, so you run it with a name and volume mapping (no --rm).

**Quick workflow for your case:**

1. **Start Postgres container** (with a name and volume, no --rm):

bash

CopiarEditar

docker start ny\_taxi\_container

(or if not created yet, create with docker run --name ny\_taxi\_container ...)

1. **Run ingest container** (with --rm):

bash

CopiarEditar

docker run --rm ny\_taxi\_ingest ...

This runs ingest and deletes container after.

1. When you want to rerun ingest, just run that same ingest command again (new container each time).
2. Check containers with docker ps -a to see all containers and their states.

If you want, I can help you write exact commands based on what containers you have now! Want to check your current containers first?

4.1-mini

**Dito por si:**

wait I don't get it... wouldn't I want to run the ingest python from within my first nytaxi\_container?

**Dito pelo ChatGPT:**

Ah, that’s a really common question! Here’s the key:

**Your ny\_taxi\_container is running Postgres, which is a database server — it does not have Python or your ingest script installed inside it.**

**Why you don’t run the ingest Python script inside the Postgres container:**

* Official Postgres Docker images are built just to run the database server.
* They **don’t include Python, pandas, or any of your code**.
* They also don’t have your ingest script files inside them.

**How it works instead:**

1. **You run the Postgres container** (ny\_taxi\_container) — this just runs the database server.
2. **You run a separate container** based on your ingest Docker image (built from your ingest.Dockerfile) — this container has Python, pandas, your ingest script, etc.
3. The ingest container connects to the Postgres container **over the network** (using host and port) and writes data to the database.

**So to recap:**

* Postgres container: just the database server.
* Ingest container: runs your Python code that downloads the data and loads it into the database.

They are separate containers talking to each other.

**How to connect between them:**

* If you’re running both containers on the same machine and using Docker Desktop, you can use host.docker.internal as the Postgres host inside the ingest container.
* Your ingest container connects to the Postgres container like this:

bash

CopiarEditar

postgresql://root:root@host.docker.internal:5433/ny\_taxi

**TL;DR:**

* **Don’t run your Python ingest script inside the Postgres container**.
* **Build a separate ingest container with your Python code**.
* **Run that ingest container to load data into your running Postgres container**.

Let me know if you want help with the exact commands or Dockerfiles!

4.1-mini

**Dito por si:**

should they not be in the same container or do they ahve to be seperate containers?

**Dito pelo ChatGPT:**

Great question! Here's the deal:

**Should Postgres and your Python ingest script be in the same container?**

**Usually, no — they should be separate containers.**

**Why separate containers?**

1. **Single Responsibility Principle:**  
   Each container does one thing and does it well.
   * Postgres container runs the database server.
   * Python ingest container runs your data ingestion code.
2. **Isolation & Flexibility:**
   * You can upgrade or restart the Python app without touching the database container.
   * You can scale or manage each independently (e.g., run multiple ingest containers if needed).
3. **Official images & ecosystem:**
   * Official Postgres image is optimized for running the database only.
   * If you mix everything in one container, you have to maintain that custom image yourself (install Python, dependencies, etc.).

**When might you combine them?**

* For very simple projects or demos, sometimes people put everything in one container for convenience.
* But it’s not recommended for production or scalable setups.

**Docker philosophy in a nutshell:**

* **One service per container.**
* Compose multiple containers (services) to build the full app.

**How to run multiple containers together?**

* Use **Docker Compose** — it lets you define and run multi-container apps easily.
* For example, one service for Postgres, another for the ingest app, networked together.

If you want, I can help you set up a Docker Compose file for this! Would you like that?

4.1-mini

**Dito por si:**

and why are we removing the ingest container right after creating it is it standard

**Dito pelo ChatGPT:**

Great point! The --rm flag in docker run means:

**Remove the container automatically once it stops.**

**Why use --rm for the ingest container?**

* The ingest container’s job is usually **short-lived**:  
  It runs the Python script to ingest data, then it’s done.
* You don’t need to keep the container around after it finishes, so --rm cleans it up automatically.

**Is this standard?**

* **Yes, for one-off or batch jobs like data ingestion, testing, or scripts that exit quickly.**
* It keeps your environment clean without leftover stopped containers.

**What about the Postgres container?**

* You **don’t** use --rm there because you want the database container to **keep running persistently** to serve data continuously.

**If you want to keep the ingest container for debugging or rerunning:**

* Just omit --rm so the container stays around after running.
* You can restart or inspect it later.

**Summary:**  
--rm is standard for short-run containers (like your ingest job) to avoid clutter. For long-running services (like databases), you keep containers persistent.

Want me to help you with commands to run the ingest container both ways?

4.1-mini

**Dito por si:**

so next time I open the container do I need to keep ingesting the data every time? it takes a long time because it has a lot of rows. or will the data be already ingested in the ny\_taxi\_contianer?

**Dito pelo ChatGPT:**

Great question!

Here’s how it works:

**The Postgres container (ny\_taxi\_container) stores the data persistently if you’ve mapped a volume**

* You used -v "/your/local/path:/var/lib/postgresql/data"
* This **mounts a local folder** as the database’s storage inside the container
* So, even if you stop or restart the Postgres container, your data **stays saved** on your host machine

**What about the ingest step?**

* **You only need to run the ingest Python script once** (or whenever you want to load new data)
* After ingesting, the data is inside the Postgres database on your volume
* You do **not** need to re-ingest every time you start the Postgres container unless the database was reset or the data deleted

**Summary:**

* Starting/stopping the Postgres container: **data stays intact** because of the volume mount
* Running the ingest container/script: **loads data into the database** (only run when needed)
* If you want to keep the data permanently, always use volume mounts with your Postgres container!