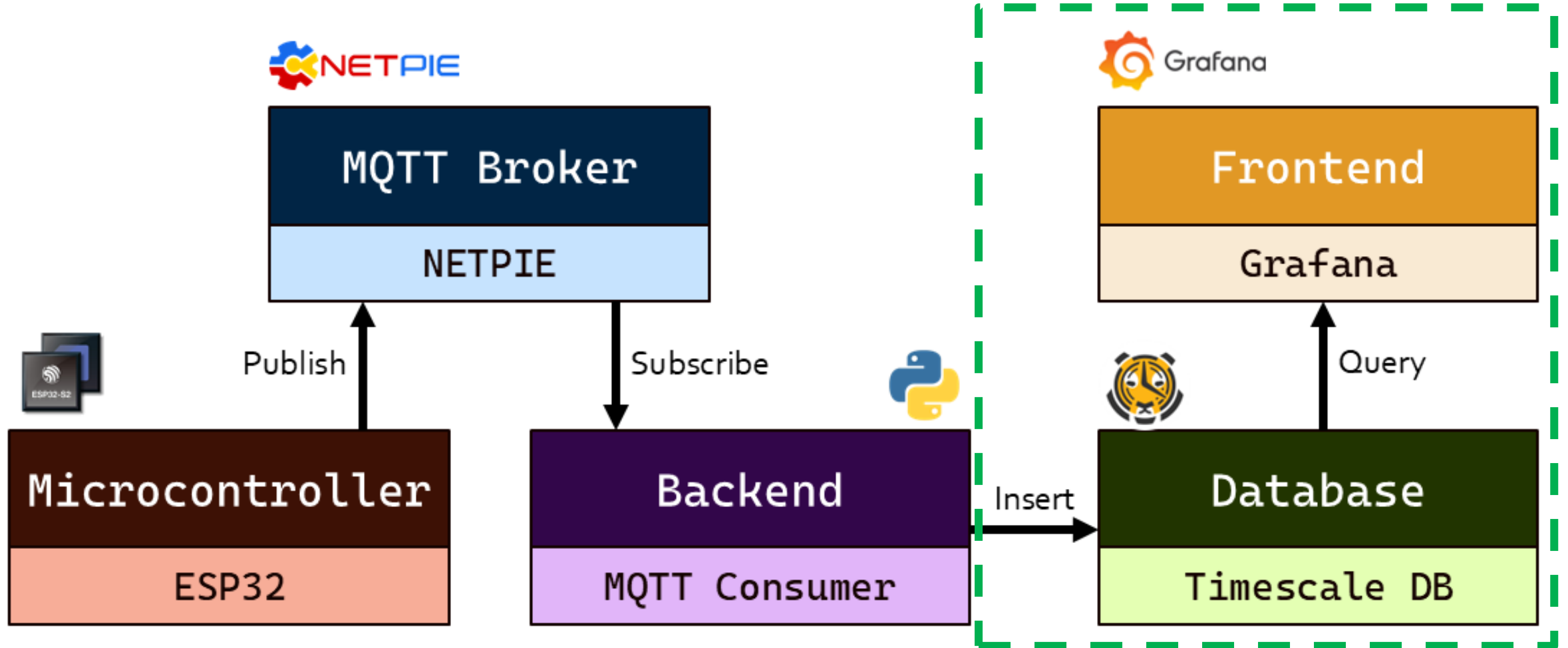# DATABASE
&
DASHBOARD

IoT CEPT Internship

28 May 2025

System Overview

What is database ?

A collection of data organized for efficient

- Storage
- Management
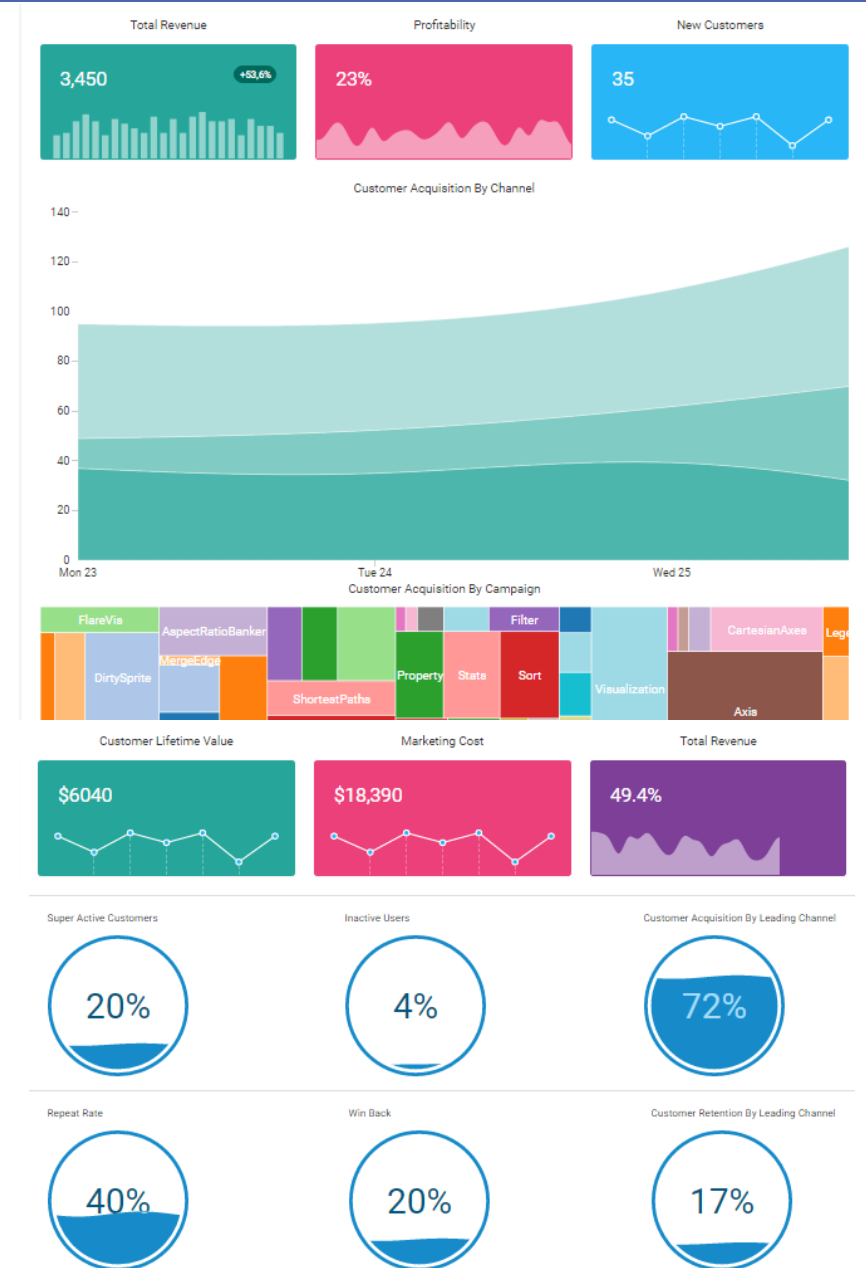- Update
- Retrieve

Why you need database ?

- Fast, organized data storage and access
- Secure, accurate data handling
- Scales easily, supports multiple users
- Reliable backup and automation

4

What is dashboard ?

A dashboard is a visual interface that displays key information and data in an organized and easy-to-digest format
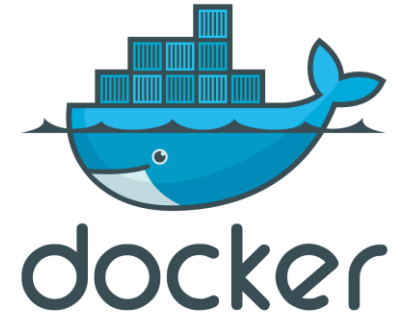
Why you need dashboard ?

- Turn raw data into visual formats

- Highlights trends and patterns

- Improves communication for non-technical audiences

- Enable real-time monitoring

Docker

- An open platform for developing, shipping, and running applications.

- Docker provides the ability to package and run an application in a loosely isolated environment called a **container.**

- **Containers** are lightweight and contain everything needed to run the application

To be continue…

https://docs.docker.com/get-started/overview/

What is a Container and how to run it? Hands-on Guide

Let's follow the guide from

https://docs.docker.com/guides/walkthroughs/what-is-a-container/
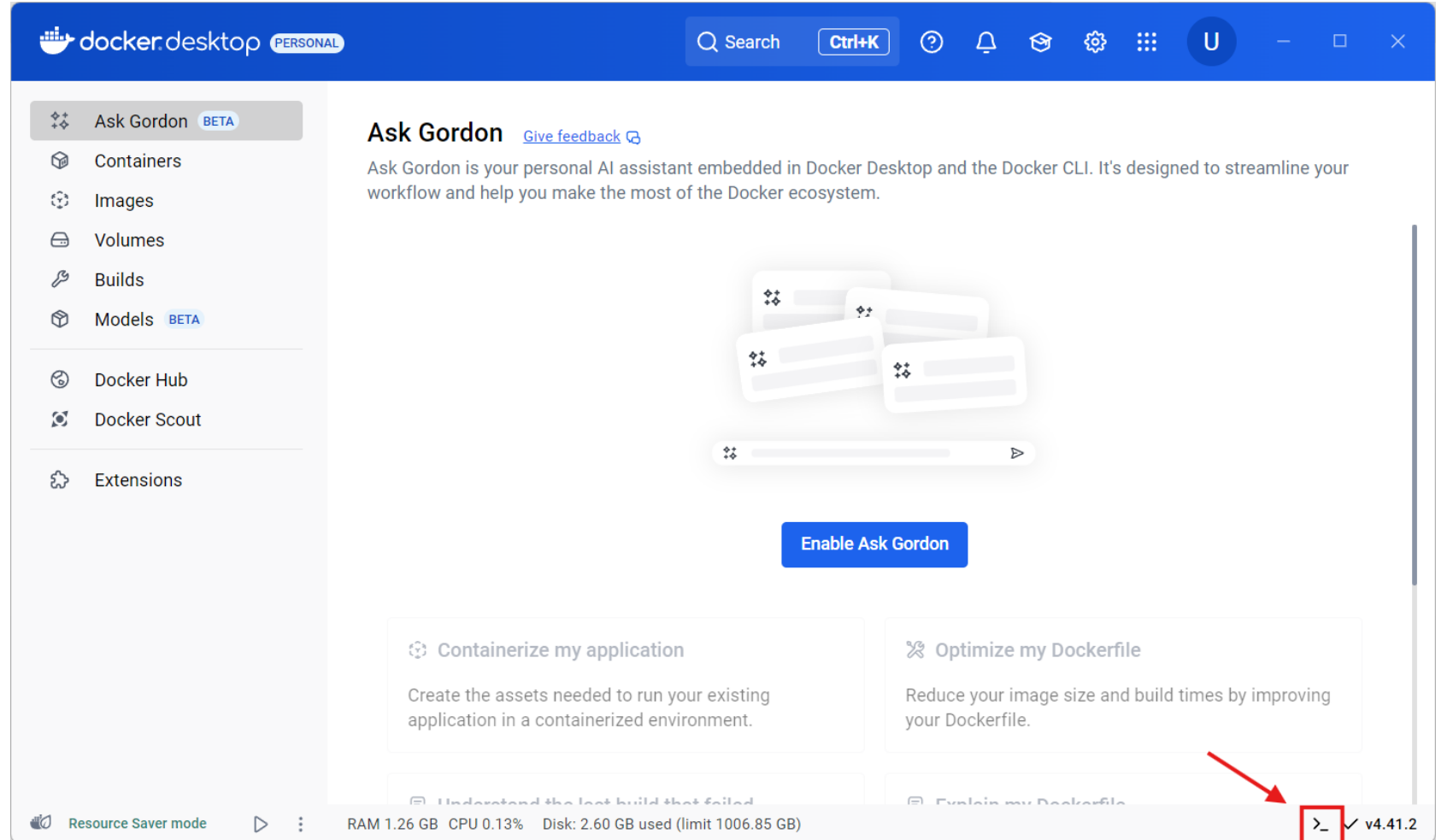
https://docs.docker.com/guides/walkthroughs/run-a-container/

https://docs.docker.com/guides/walkthroughs/run-hub-images/

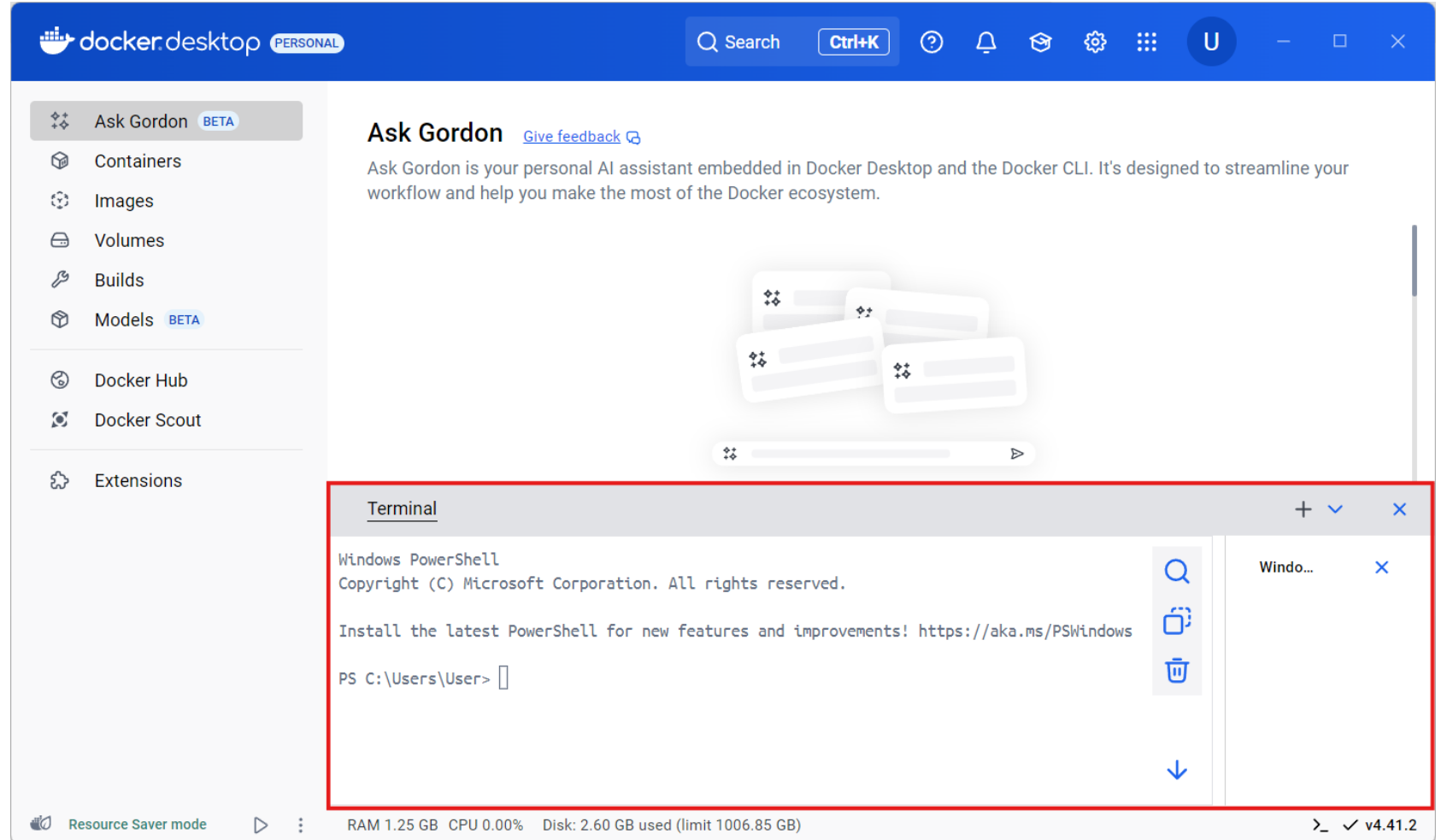https://docs.docker.com/guides/walkthroughs/publish-your-image/

How to run database and dashboard in Docker?

Step 1:

Open the terminal

How to run database and dashboard in Docker?

Step 1:

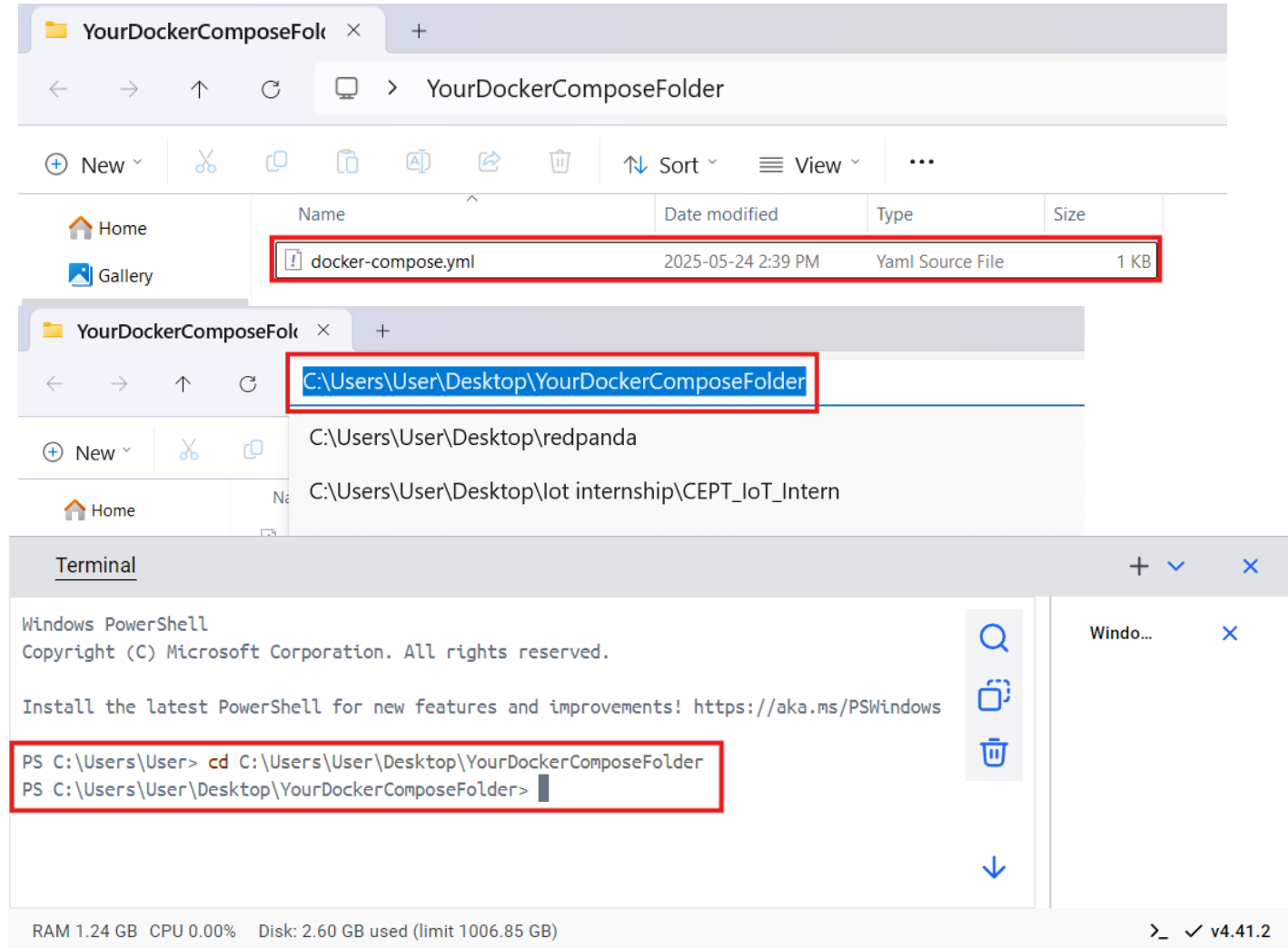## Open the terminal

How to run database and dashboard in Docker?
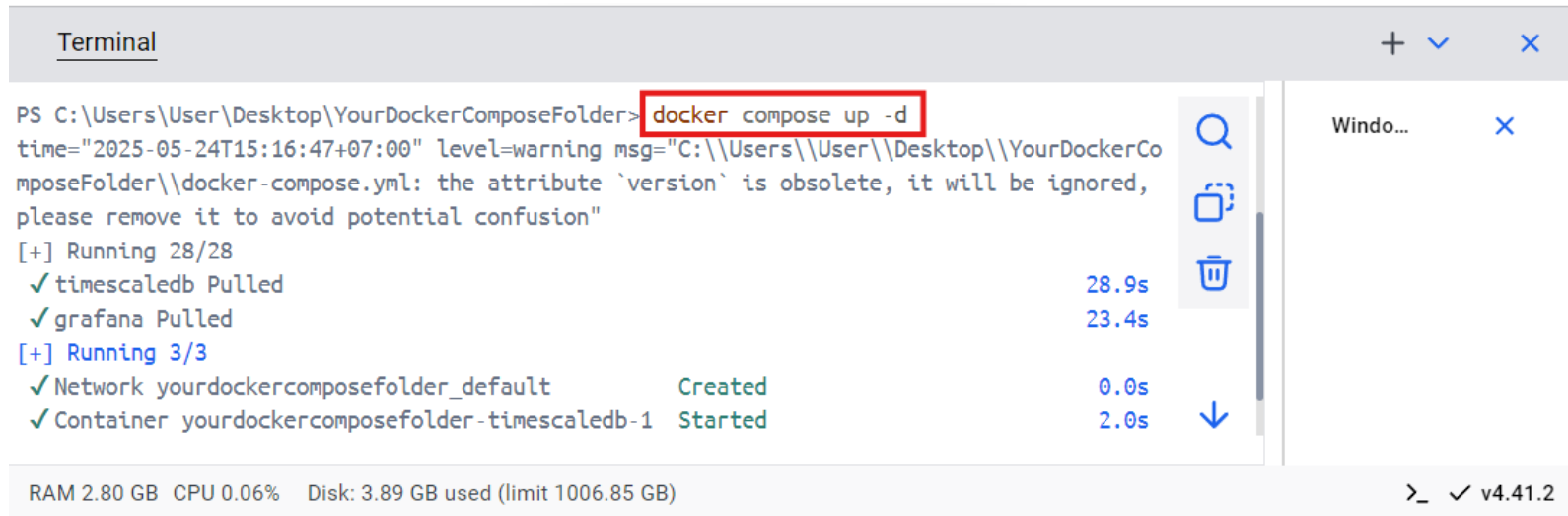
Step 2:

Navigate to the folder with

`docker-compose.yml`

by running following command in the terminal

`cd <YourDockerComposeFolder>`

How to run database and dashboard in Docker?
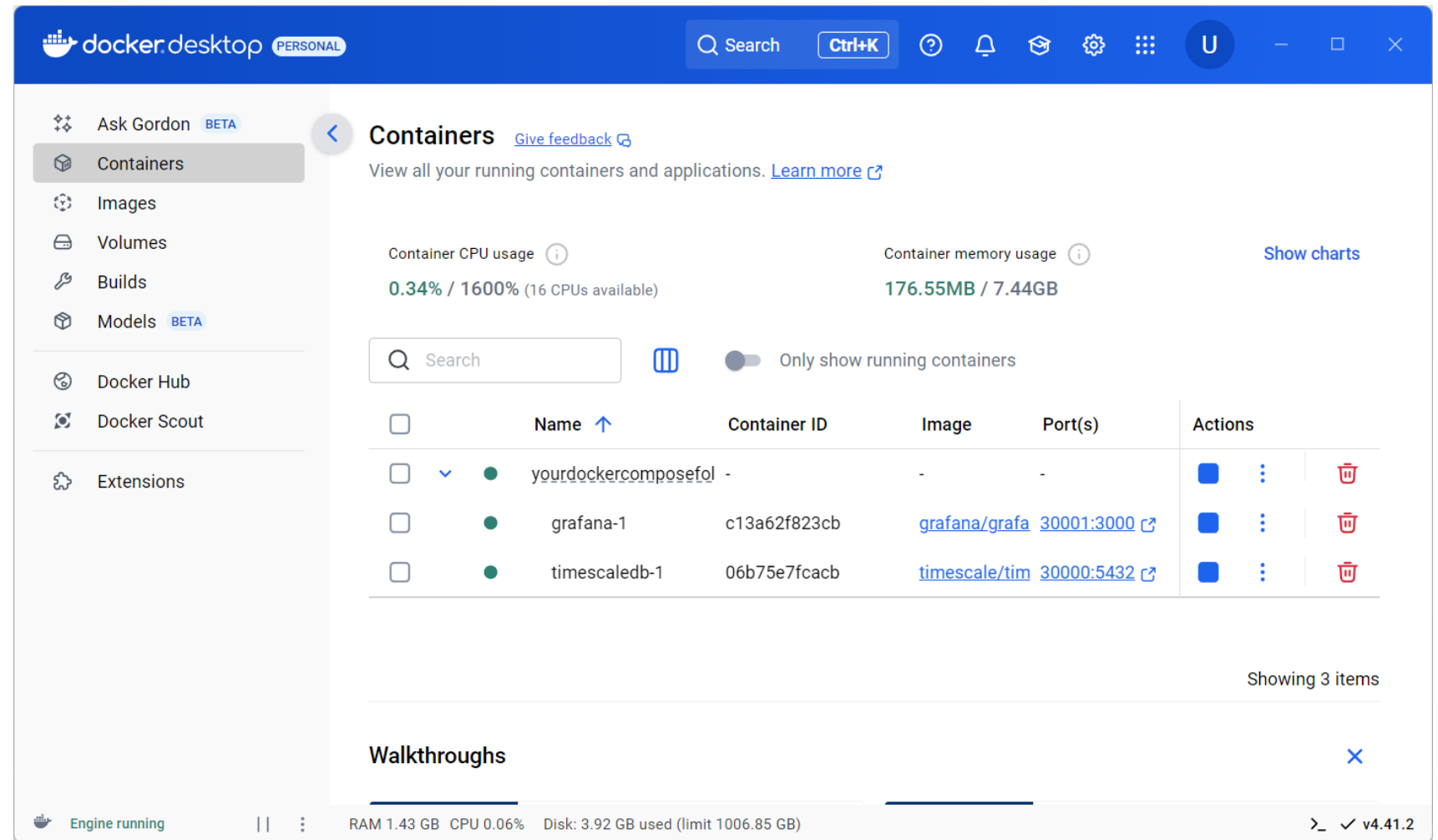
Step 3: Run `docker compose up -d`

How to run database and dashboard in Docker?

If it works, you will see 2 containers running in the Containers tab
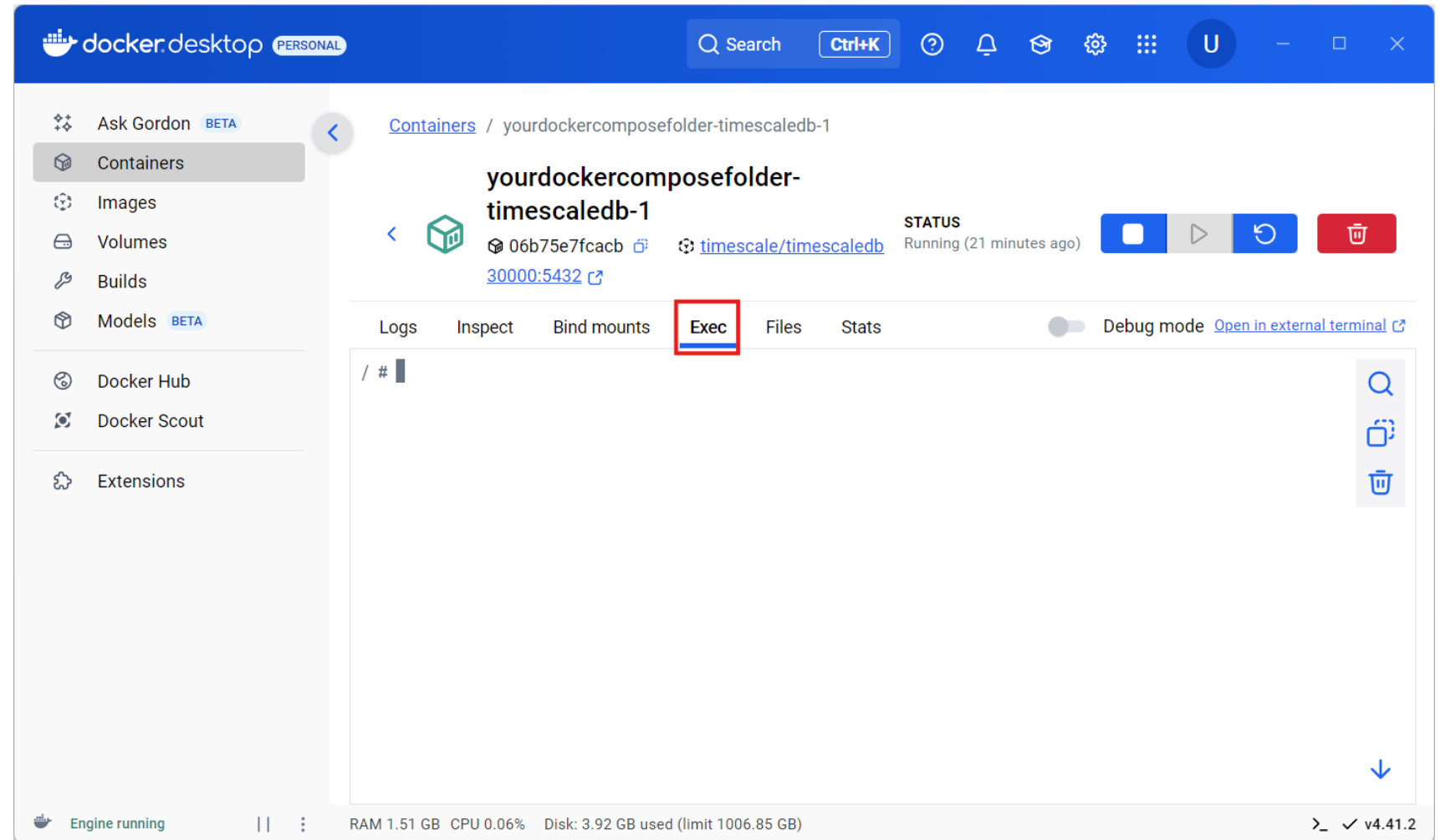
Check if database can be access ?

Step 1:

Open timescaledb container in the Containers tab

Check if database can be access ?

Step 2:

Go to Exec tab

Check if database can be access ?

/ yourdockercomposefolder-timescaledb-1

yourdockercomposefolder-timescaledb-1

06b75e7fcacb   timescale/timescaledb

30000:5432

STATUS
Running (21 minutes ago)

Logs   Inspect   Bind mounts   **Exec**   Files   Stats   Debug mode Open in external terminal

```
/ # psql -d "postgres://postgres:password@localhost/postgres"
psql (17.5)
Type "help" for help.

postgres=#
```

RAM 1.51 GB   CPU 0.06%   Disk: 3.92 GB used (limit 1006.85 GB)   >_ ✓ v4.41.2

Step 3:
Run

psql -d "postgres://postgres:password@localhost/postgres"

to connect to database

16

Check if dashboard can be access ?

Step 1:

In the grafana container row, click the number at the Port(s) column

Check if dashboard can be access ?

Grafana User Interface (UI)
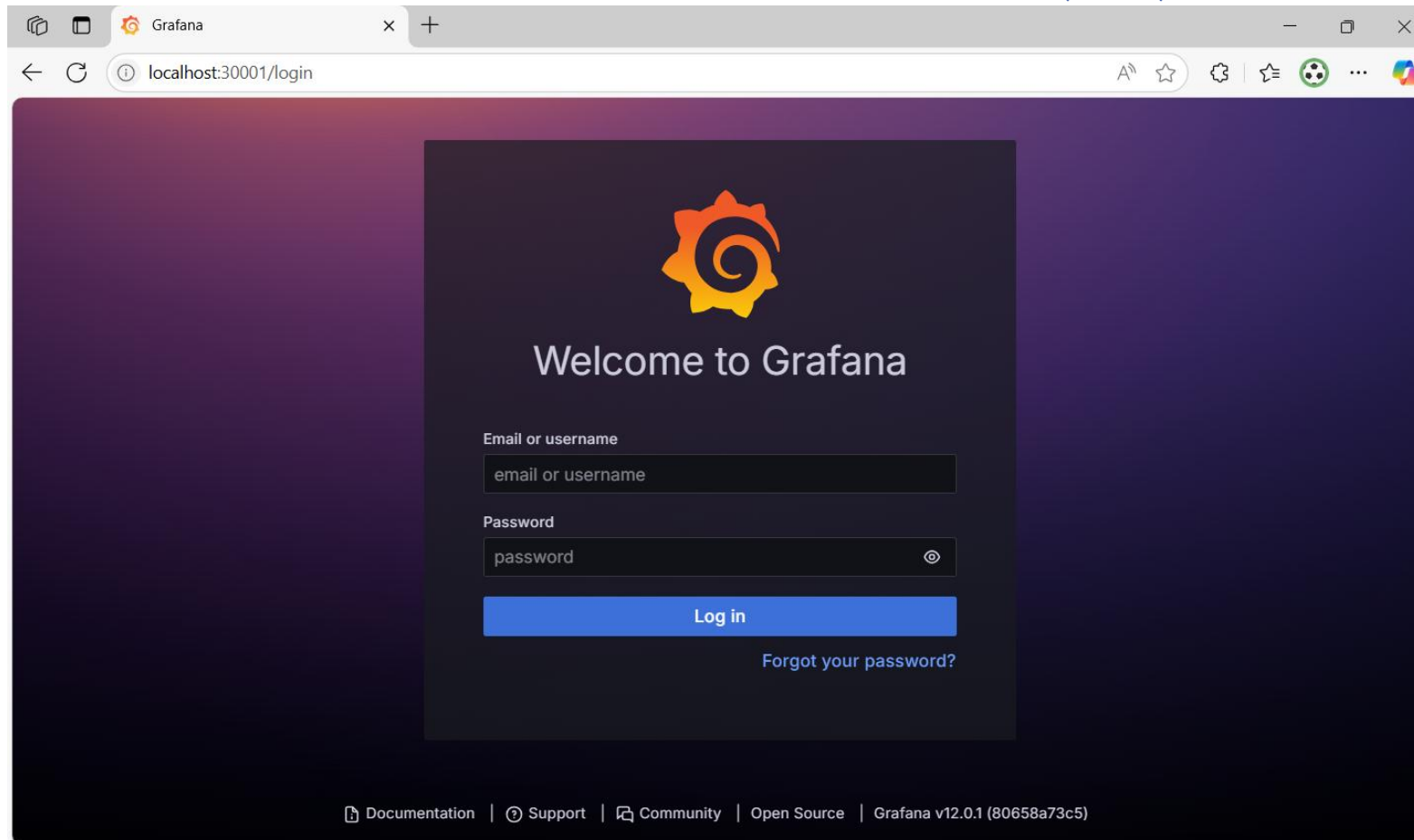
Done !
Now you have database and dashboard running on your computer

Next !
we'll take a quick look at database and dashboard

Relational Database Overview

- Relational Database is a type of database that organizes data into tables

- A database can have multiple tables

students

student_id	name	birthday
6630000121	John	1/1/2004
6630000221	Mary	2/2/2004

Row

Column

students

student_id

name

birthday

enrolments

enrolment_id

student_id

course_id

courses

course_id

course_name
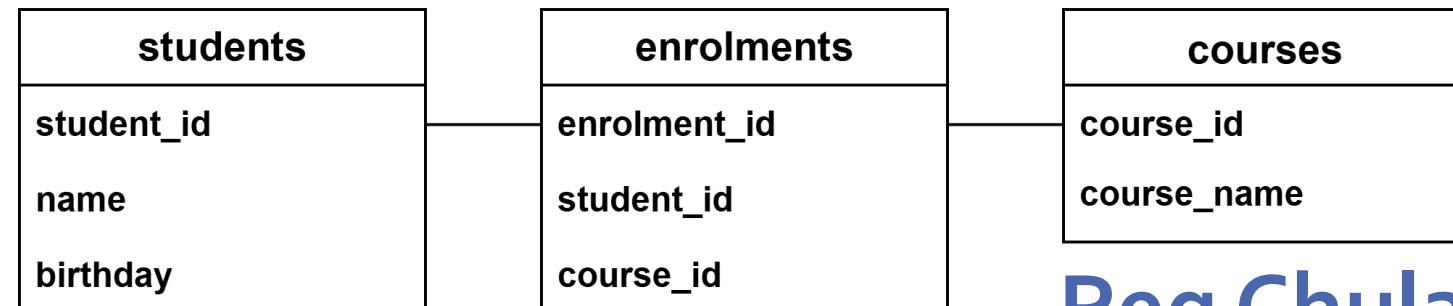
Reg Chula

Postgres Database Command

\dt   list tables

No table exists

1 table named students

What is SQL ?

SQL (Structured Query Language) is the standard language used to communicate with a relational database

Today SQL command

- CREATE
- INSERT
- SELECT
- DELETE
- DROP

SQL CREATE table

sensor_data			
time	sensor_id	temperature	humidity
'2025-05-25 08:30:00+07'	1	22.5	45.2
'2025-05-25 08:35:00+07'	1	23.1	47.8
…	…	…	…

sensor_data	
Column	Data type
time	TIMESTAMPTZ
sensor_id	INTEGER
temperature	FLOAT
humidity	FLOAT

Data type

- Character: Dave → **VARCHAR(n)**
- Numeric: 15 → **INTEGER**, 3.1415 → **REAL**
- Date/Time: '2004-10-19 10:23:54' → **TIMESTAMP**
  '2004-10-19 10:23:54+02' → **TIMESTAMPTZ**
- Boolean : false → **BOOLEAN**

SQL CREATE table

```
CREATE TABLE table name (

        column_1 datatype_1,

        column_2 datatype_2,

        ...,

        column_n datatype_n

); Don't forget to end SQL command with semicolon (;)
```

sensor_data			
time	sensor_id	temperature	humidity

❖Try create sensor_data table with following column name and data type

sensor_data	
Column	Data type
time	TIMESTAMPTZ
sensor_id	INTEGER
temperature	FLOAT
humidity	FLOAT

Check created table by **\dt** command

```
postgres=# \dt
                List of relations
 Schema |     Name    | Type  | Owner
--------+-------------+-------+---------
 public | sensor_data | table | postgres
(1 row)
```

24

SQL INSERT data

```
INSERT INTO table name (column_1, column_2, ..., column_n) VALUES

(data_column_1, data_column_2, ..., data_column_n),

(data_column_1, data_column_2, ..., data_column_n),

...,

(data_column_1, data_column_2, ..., data_column_n);
```

Don't forget to end SQL command with semicolon (;)

❖ Try insert following sample data

```
INSERT INTO sensor_data (time, sensor_id, temperature, humidity) VALUES

('2025-05-25 08:30:00+07', 1, 22.5, 45.2),

('2025-05-25 08:35:00+07', 1, 23.1, 47.8),

('2025-05-25 08:40:00+07', 1, 22.8, 46.0),

('2025-05-25 08:45:00+07', 1, 24.3, 44.5),

('2025-05-25 08:50:00+07', 1, 23.7, 48.1);
```

SQL SELECT data

To select specific column

```
SELECT column_1, column_2, ..., column_n FROM table name;
```

To select all column

```
SELECT * FROM table name;
```

Don't forget to end SQL command with semicolon (;)

❖ Select time and temperature columns

```
postgres=# select time, temperature from sensor_data ;
          time          | temperature
------------------------+-------------
 2025-05-25 01:30:00+00 |        22.5
 2025-05-25 01:35:00+00 |        23.1
 2025-05-25 01:40:00+00 |        22.8
 2025-05-25 01:45:00+00 |        24.3
 2025-05-25 01:50:00+00 |        23.7
(5 rows)
```

❖ Select all columns

```
postgres=# select * from sensor_data ;
          time          | sensor_id | temperature | humidity
------------------------+-----------+-------------+---------
 2025-05-25 01:30:00+00 |         1 |        22.5 |     45.2
 2025-05-25 01:35:00+00 |         1 |        23.1 |     47.8
 2025-05-25 01:40:00+00 |         1 |        22.8 |       46
 2025-05-25 01:45:00+00 |         1 |        24.3 |     44.5
 2025-05-25 01:50:00+00 |         1 |        23.7 |     48.1
(5 rows)
```
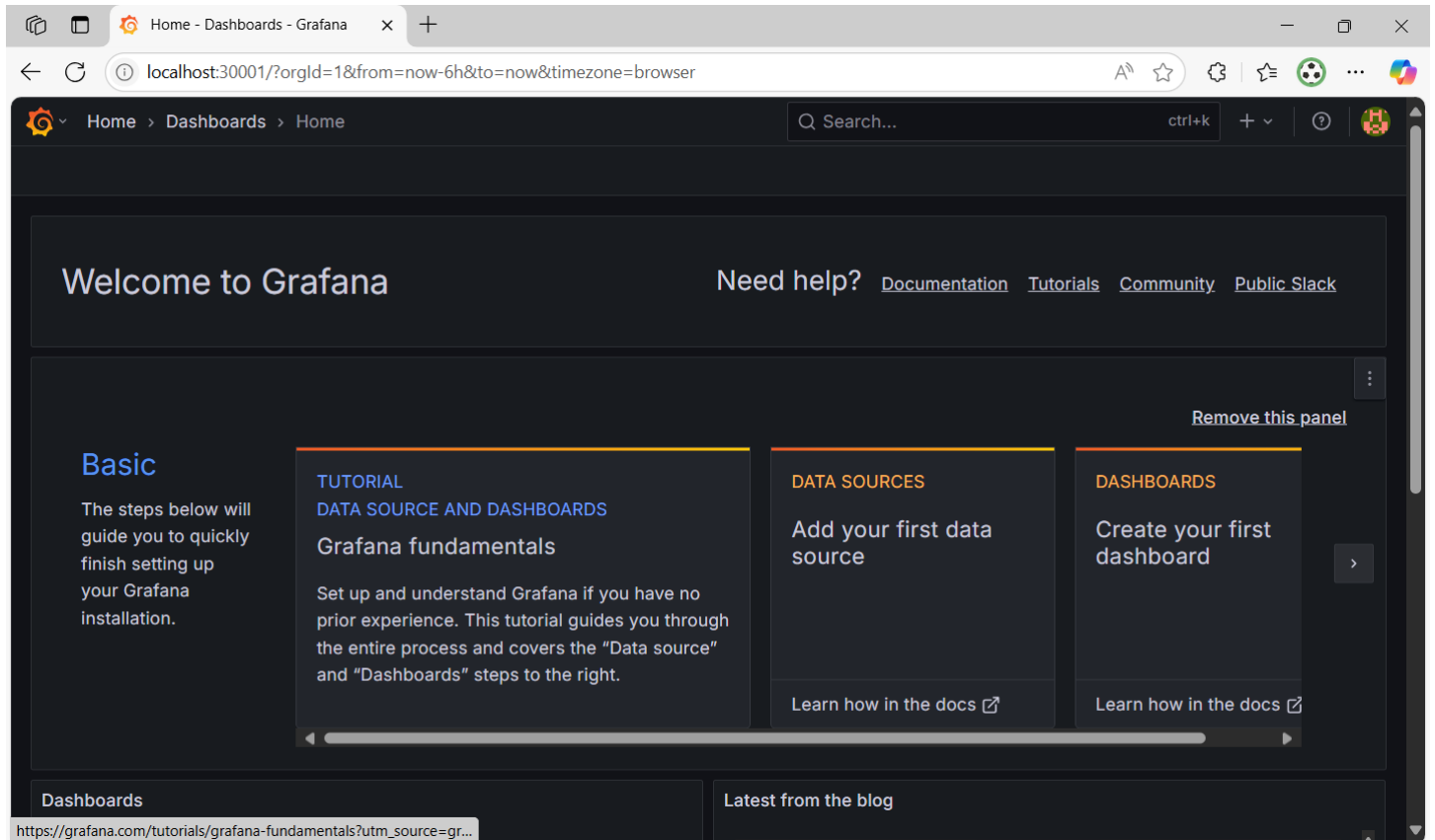
Visualize data in dashboard

Step 1:

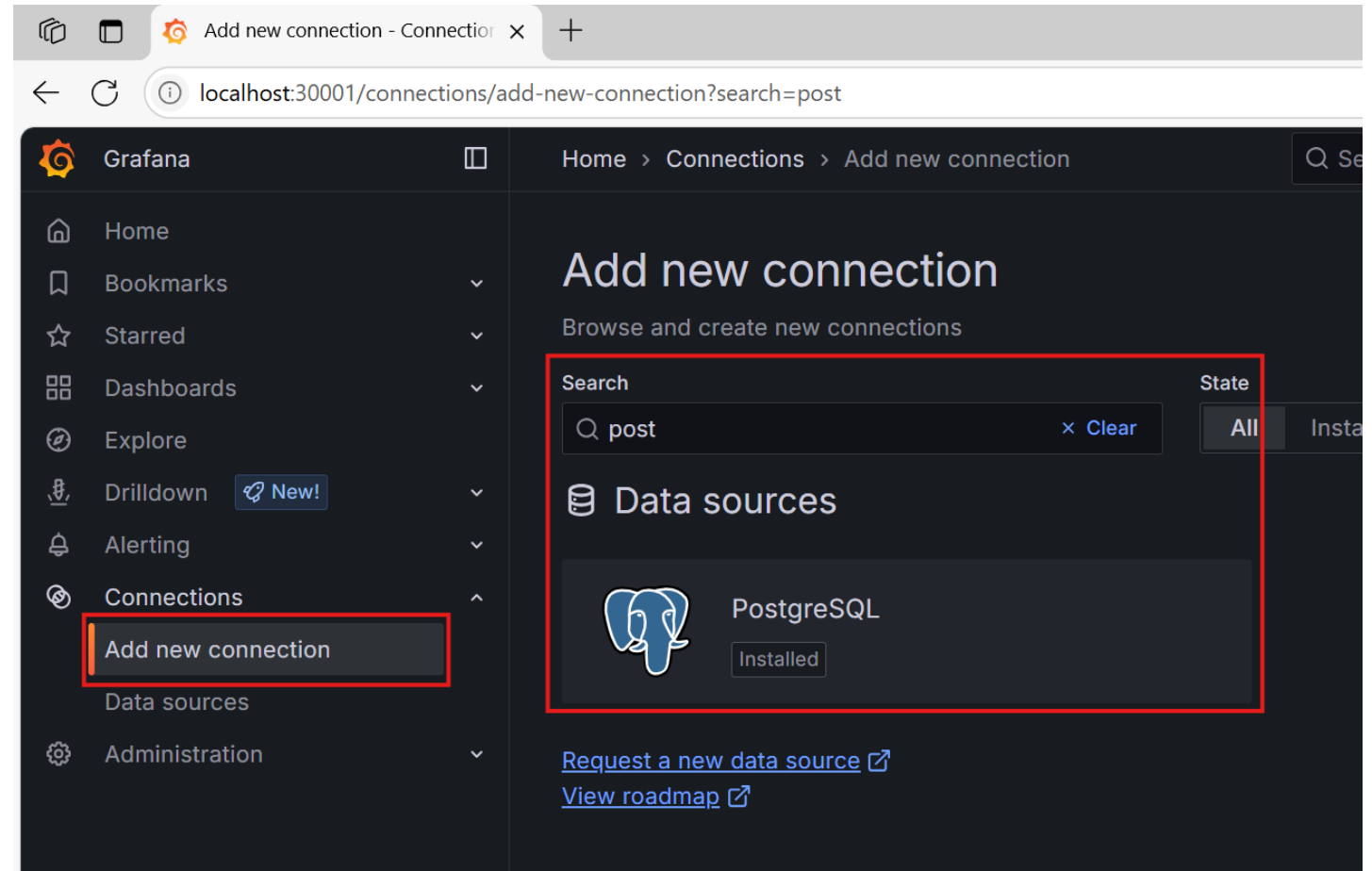Access Grafana UI, then log in using default user and password:
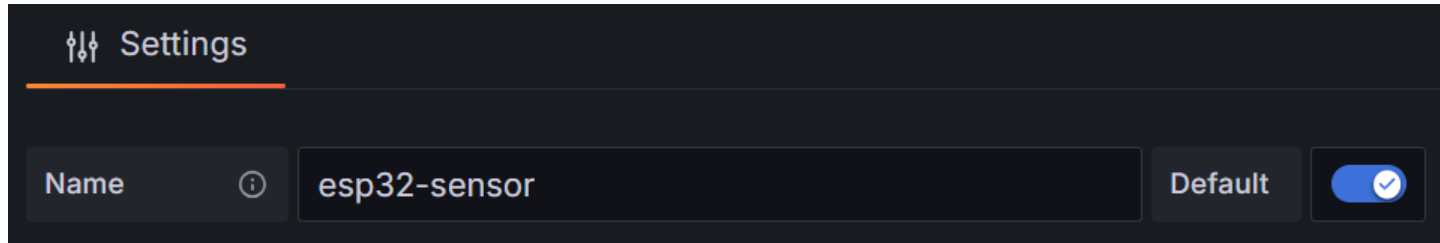
User: admin

Pass: admin

Visualize data in dashboard

Step 2:

Go to Add new connection in the Connections drop down tab, then search for PostgreSQL and Add new data source
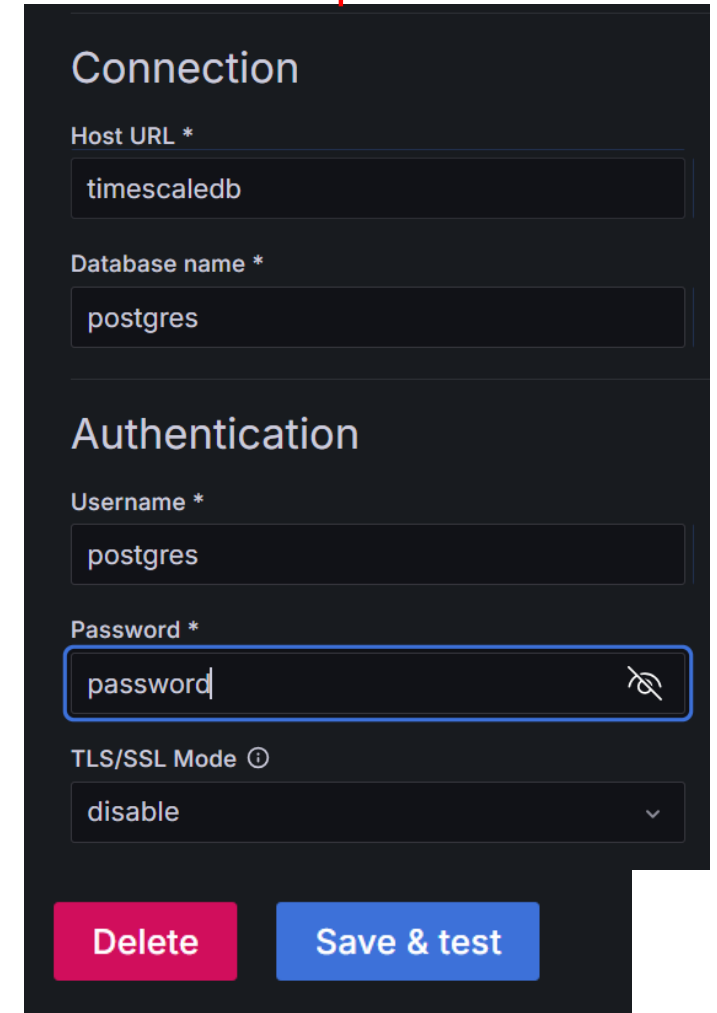
Visualize data in dashboard

Settings

Name ⓘ | esp32-sensor | Default | ⬤

Step 3:

Enter Name, Connection and Authentication of the database then Save & test

Connection

Host URL *
timescaledb

Database name *
postgres

Authentication

Username *
postgres

Password *
password

TLS/SSL Mode ⓘ
disable

Delete   Save & test
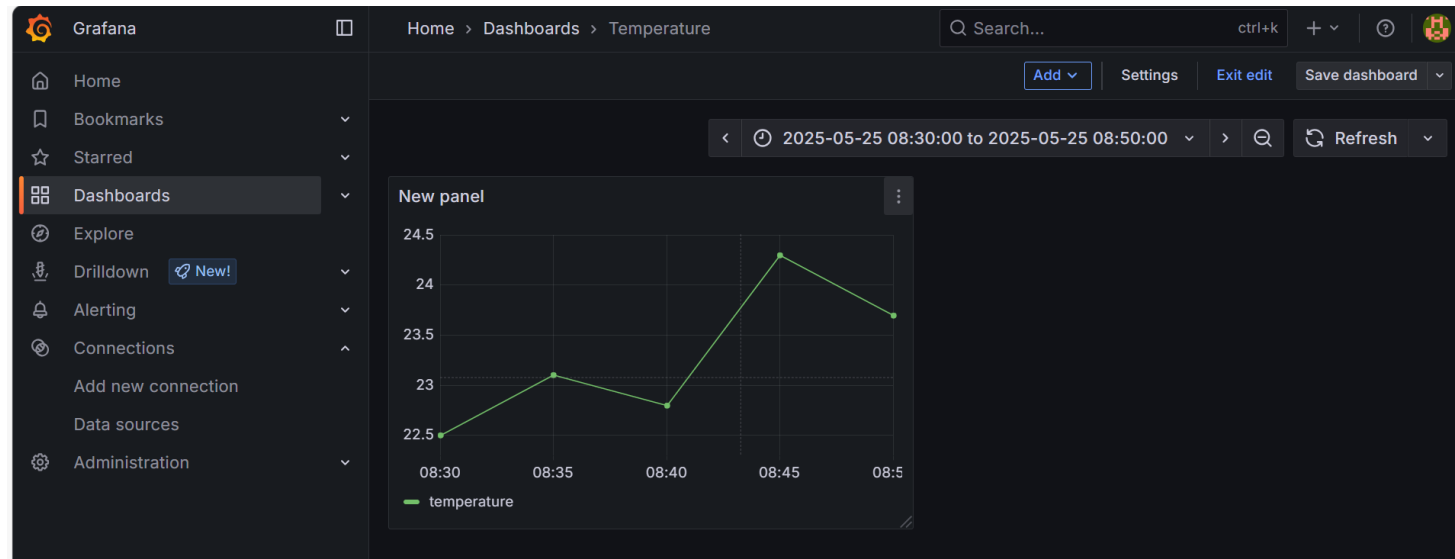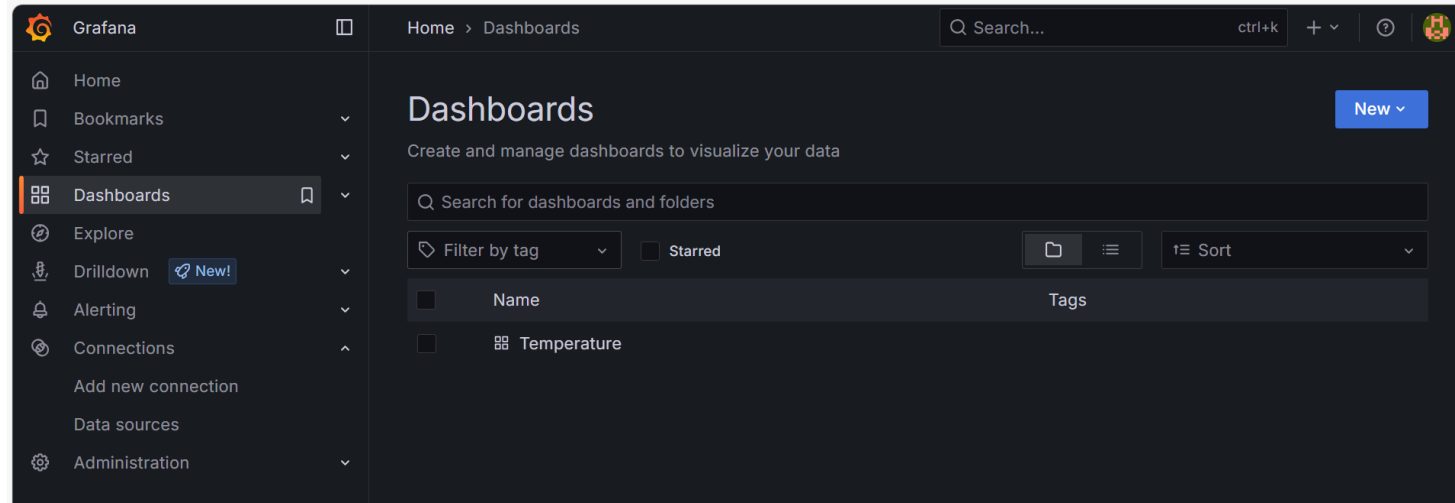
Visualize data in dashboard

Step 4:

Go to Dashboards tab, Add visualization and select our data source then query the data in code section and save dashboard

Visualize data in dashboard

How to connect database using python

Use **connect_postgres.py** in **database-and-dashboard** folder

```python
import psycopg2

                                        Connection parameters
CONNECTION = "postgres://postgres:password@localhost:30000/postgres"

with psycopg2.connect(CONNECTION) as conn:

    with conn.cursor() as cur: # use the cursor to interact with your database

        cur.execute("SELECT version();")

        version = cur.fetchone()

        print(version)
```

Successfully Connected

How to insert data in database using python

```python
import psycopg2
# Data to be inserted
data = ('2025-05-25 08:55:00+07', 1, 30, 60)
# SQL INSERT statement
insert_query = """INSERT INTO sensor_data
                    (time, sensor_id, temperature, humidity)
                    VALUES (%s, %s, %s, %s);"""
CONNECTION = "postgres://postgres:password@localhost:30000/postgres"
with psycopg2.connect(CONNECTION) as conn:
    with conn.cursor() as cur:
        cur.execute(insert_query, data)
    conn.commit()
print(f"Inserted row: {data}")
```

Use **insert_postgres.py** in **database-and-dashboard** folder

❖ Try check the inserted data using select command or dashboard

```
PS C:\Users\User> & C:/Users/User/AppData/Local/Programs/Python/Python312/python.exe c:/Users/User/Desktop/timescaledb/insert_postgres
Inserted row: ('2025-05-25 08:55:00+07', 1, 30, 60)
```

SQL DELETE data

```
DELETE FROM table_name;
```
Don't forget to end SQL command with semicolon (;)

❖ Delete sensor_data table and see what happen with the dashboard
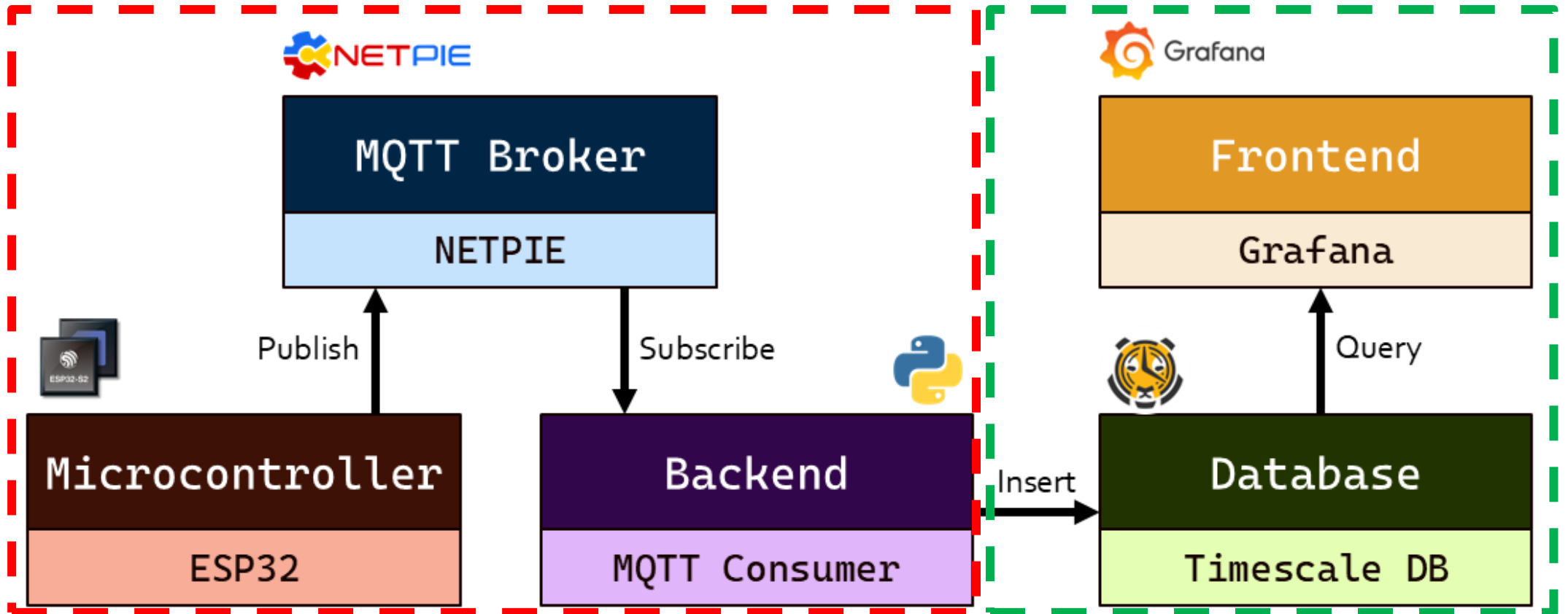
SQL DROP table

```
DROP TABLE table_name;
```
Don't forget to end SQL command with semicolon (;)

❖ Drop sensor_data table and see what happen with table in the database **\dt**

To receive data from ESP32, recreate the sensor_data table

```sql
CREATE TABLE sensor_data(

    time TIMESTAMPTZ,

    sensor_id INTEGER,

    temperature REAL,

    humidity REAL

);
```

Combine
ESP32(subscribe_mqtt.py)
and
Database & Dashboard (insert_postgres.py)

Which one is your project ?