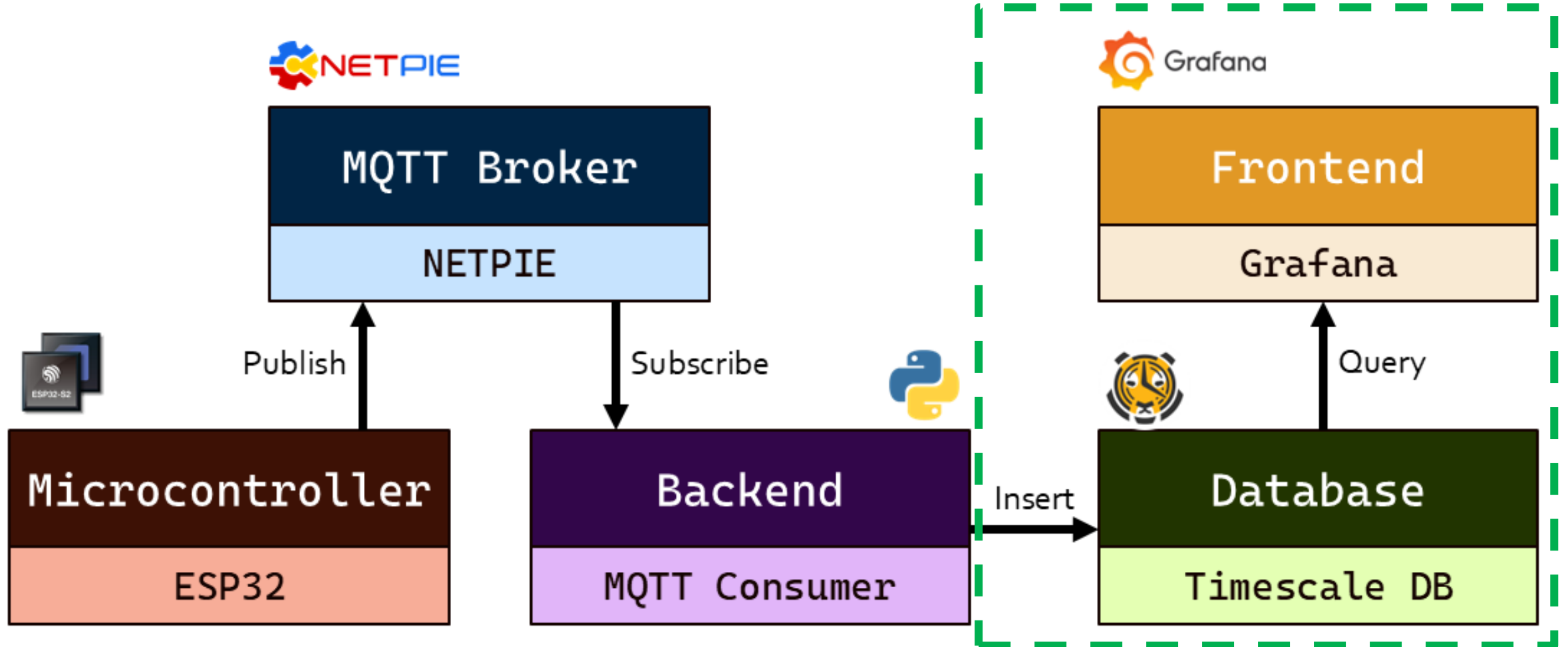


DATABASE & DASHBOARD

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 ?

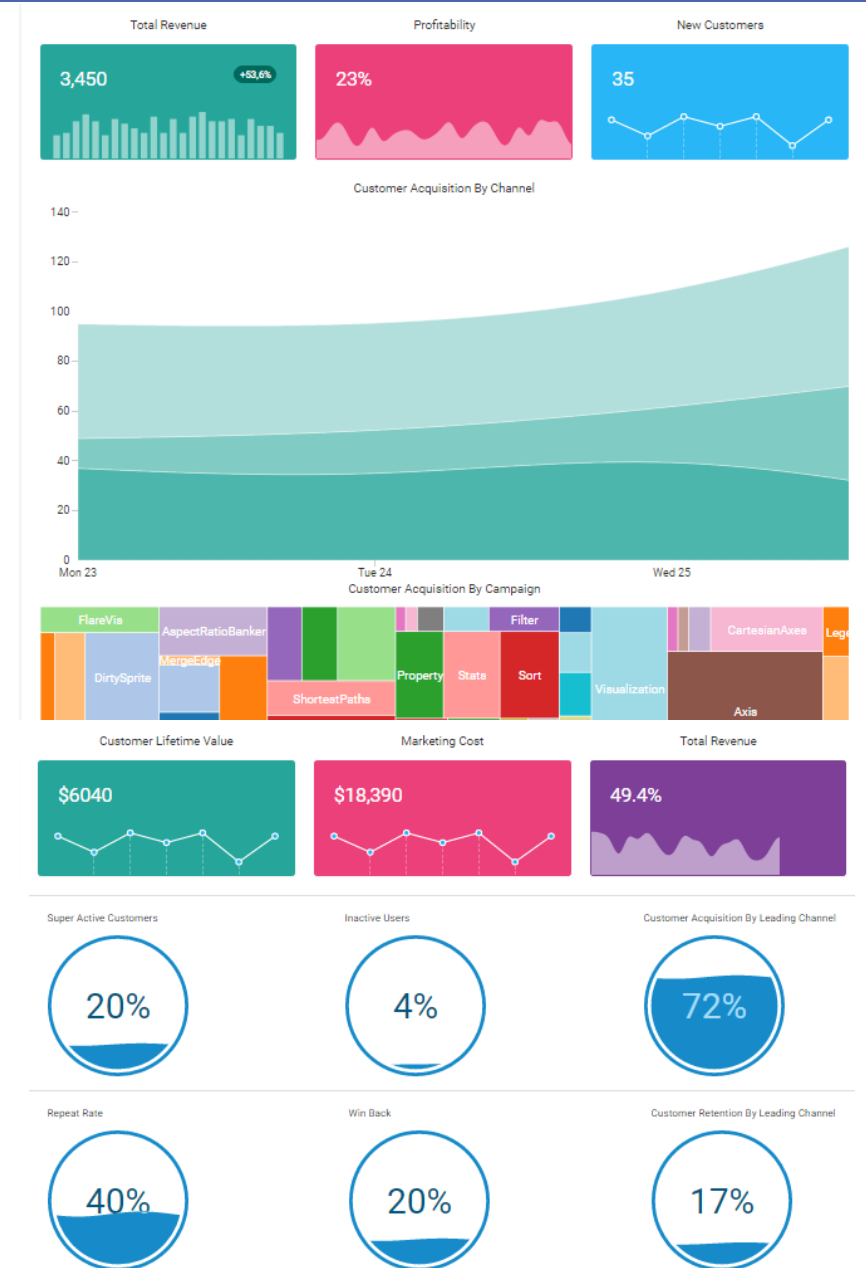


- Fast, organized data storage and access
- Secure, accurate data handling
- Scales easily, supports multiple users
- Reliable backup and automation



What is dashboard ?

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



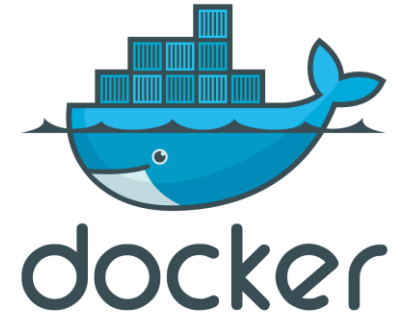
Why you need dashboard ?



- Turn raw data into visual formats
- Highlights trends and patterns
- Improves communication for non-technical audiences
- Enable real-time monitoring



Docker



[This Photo](#) by Unknown Author is licensed under [CC BY-SA](#)

- 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...

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/>

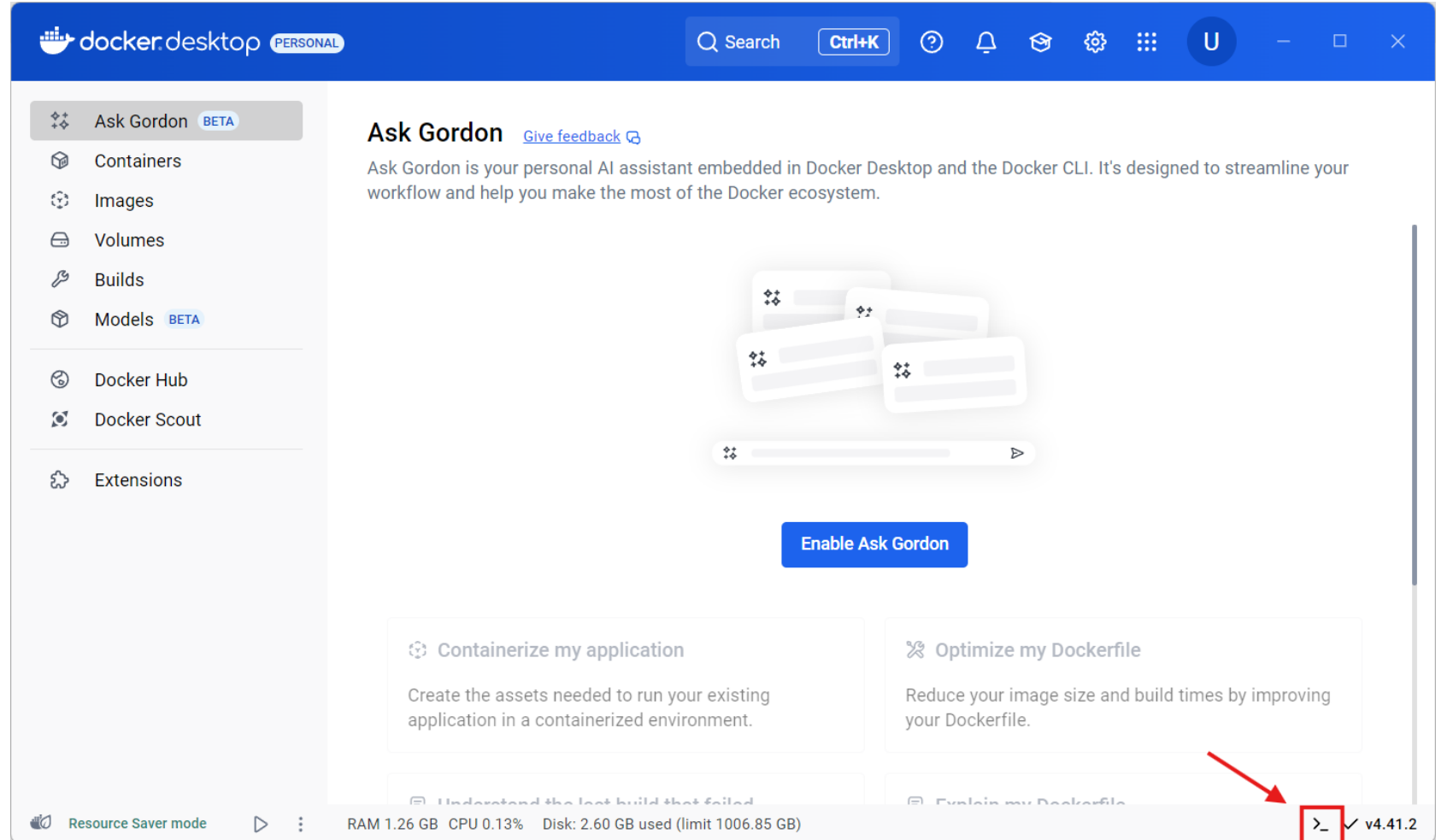
<https://docs.docker.com/guides/walkthroughs/run-a-container/>

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

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

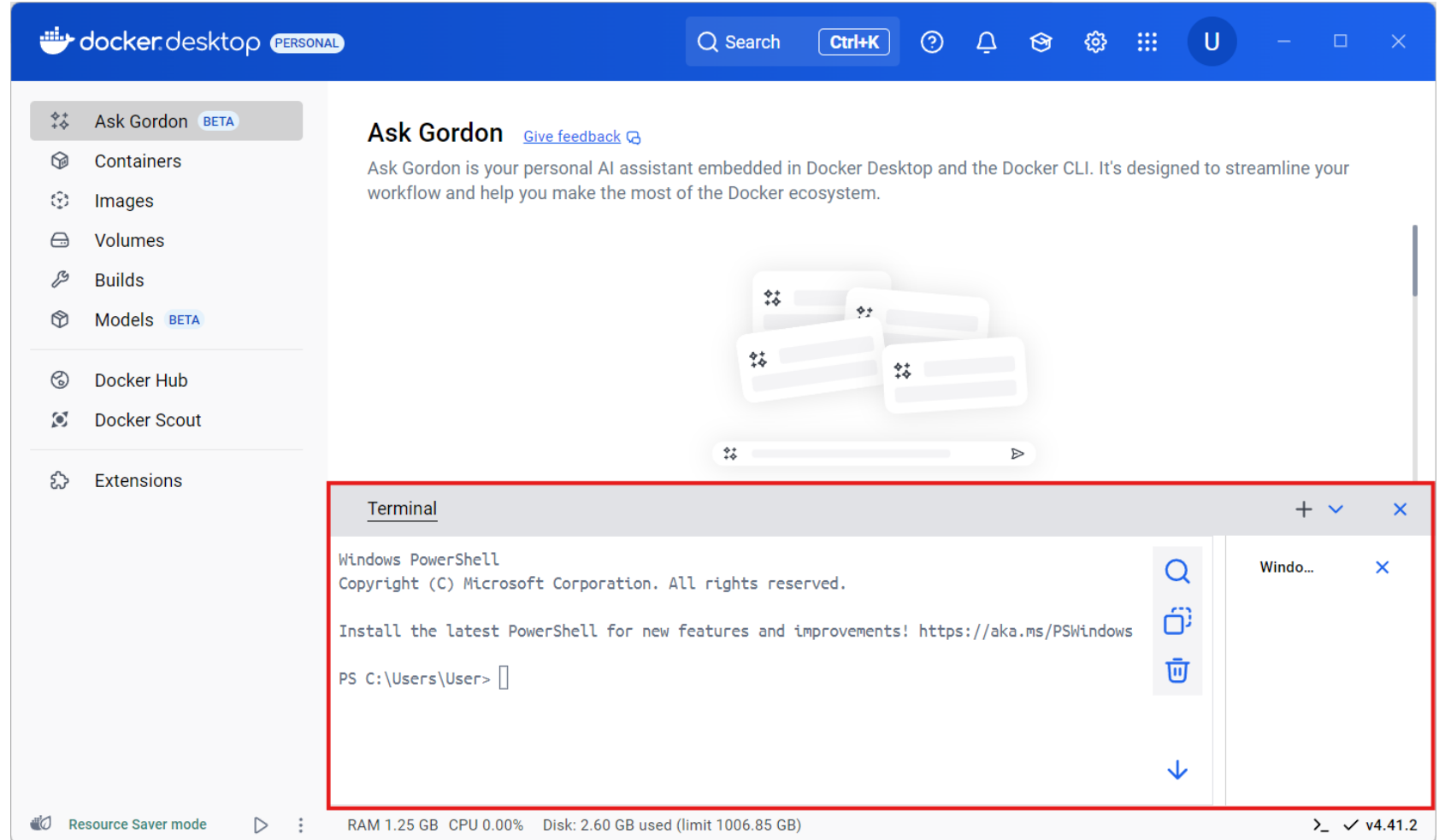
How to run database and dashboard in Docker?

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?

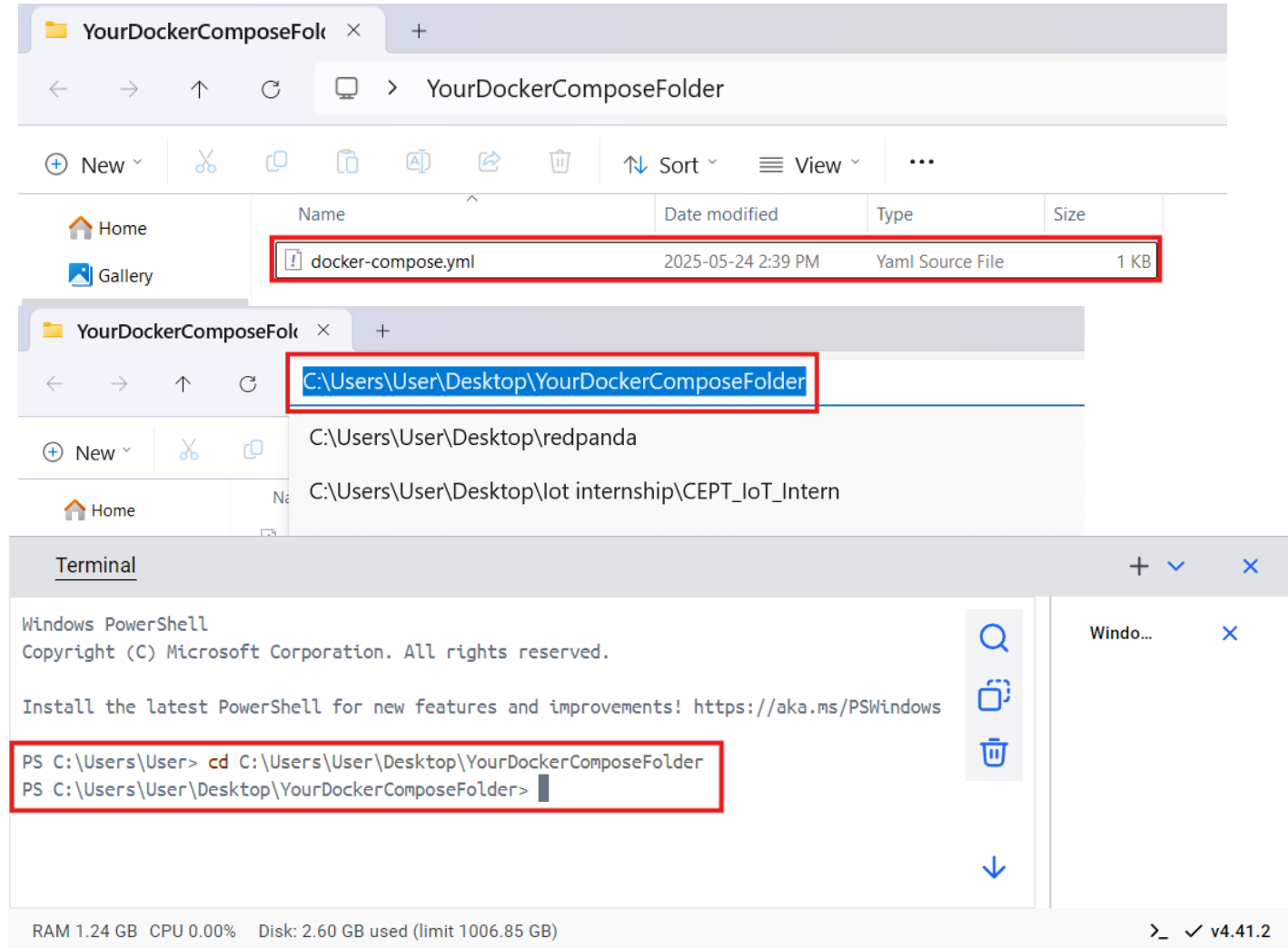
Step 2:

Navigate to the folder with

docker-compose.yml

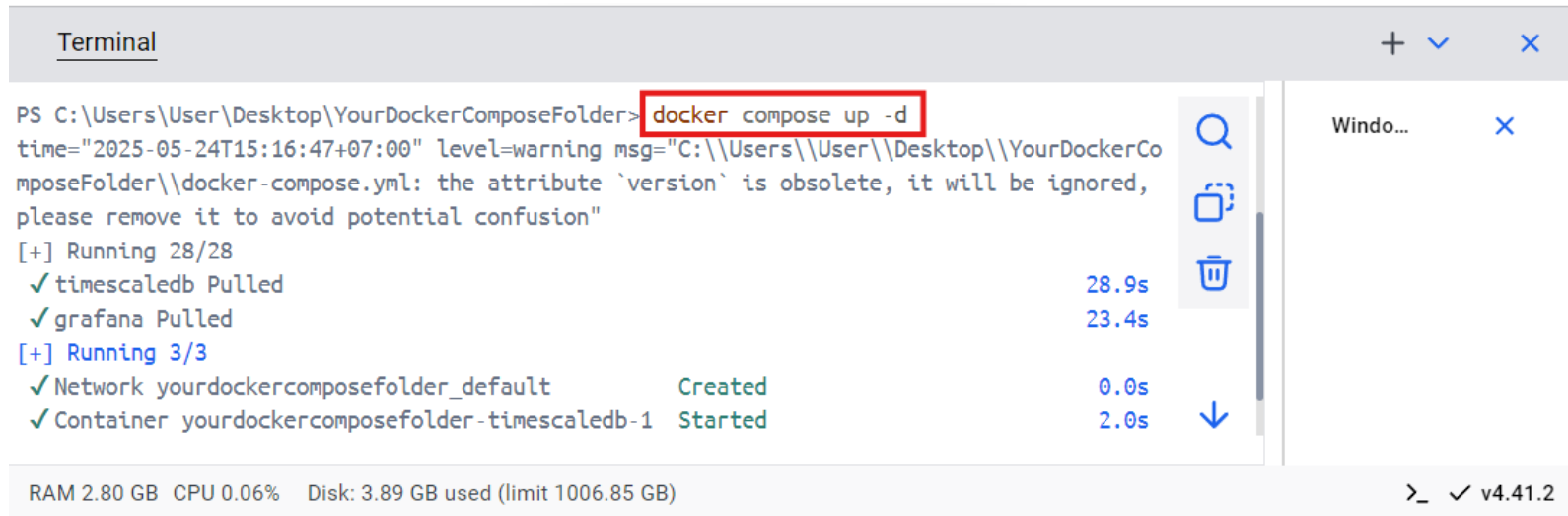
by running following command in the terminal

cd <YourDockerComposeFolder>



How to run database and dashboard in Docker?

Step 3: Run `docker compose up -d`

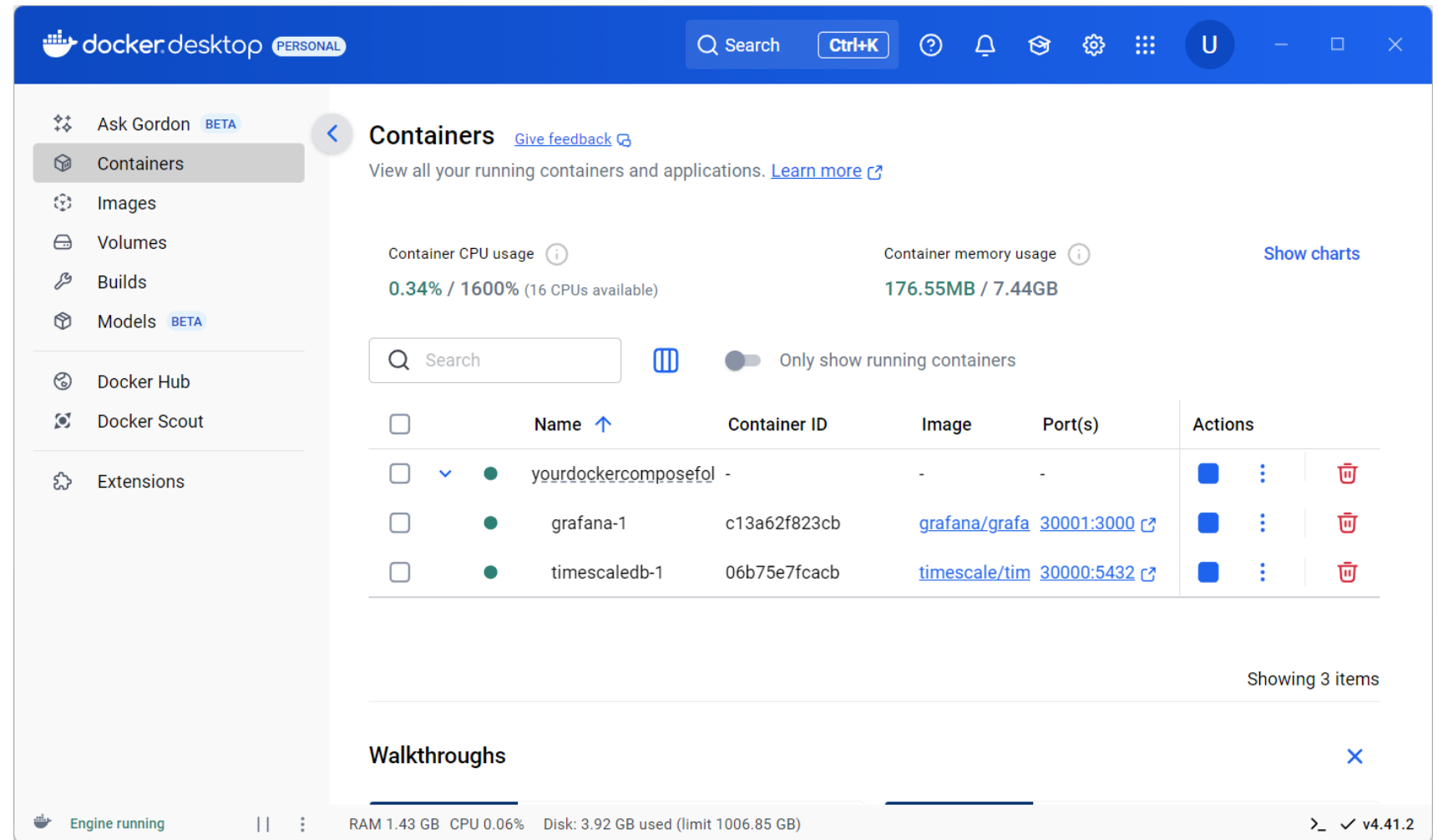


```
Terminal
PS C:\Users\User\Desktop\YourDockerComposeFolder> docker compose up -d
time="2025-05-24T15:16:47+07:00" level=warning msg="C:\\Users\\User\\Desktop\\YourDockerCo
mposeFolder\\docker-compose.yml: the attribute `version` is obsolete, it will be ignored,
please remove it to avoid potential confusion"
[+] Running 28/28
  ✓ timescaledb Pulled                                28.9s
  ✓ grafana Pulled                                    23.4s
[+] Running 3/3
  ✓ Network yourdockercomposefolder_default           0.0s Created
  ✓ Container yourdockercomposefolder-timescaledb-1   2.0s Started

RAM 2.80 GB CPU 0.06% Disk: 3.89 GB used (limit 1006.85 GB) v4.41.2
```

How to run database and dashboard in Docker?

If it works, you will see 2 containers running in the Containers tab



The screenshot shows the Docker Desktop interface. The left sidebar contains navigation options: Ask Gordon (BETA), Containers (selected), Images, Volumes, Builds, Models (BETA), Docker Hub, Docker Scout, and Extensions. The main area is titled 'Containers' and shows system usage: Container CPU usage is 0.34% / 1600% (16 CPUs available) and Container memory usage is 176.55MB / 7.44GB. A table lists three running containers:

	Name ↑	Container ID	Image	Port(s)	Actions
<input type="checkbox"/>	yourdockercomposefol -	-	-	-	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
<input type="checkbox"/>	grafana-1	c13a62f823cb	grafana/grafana	30001:3000	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
<input type="checkbox"/>	timescaledb-1	06b75e7fcac	timescale/timescaledb	30000:5432	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>

Showing 3 items

At the bottom, a status bar indicates 'Engine running', 'RAM 1.43 GB', 'CPU 0.06%', 'Disk: 3.92 GB used (limit 1006.85 GB)', and 'v4.41.2'.

Check if database can be access ?

Step 1:
Open
timescaledb
container in
the Containers
tab

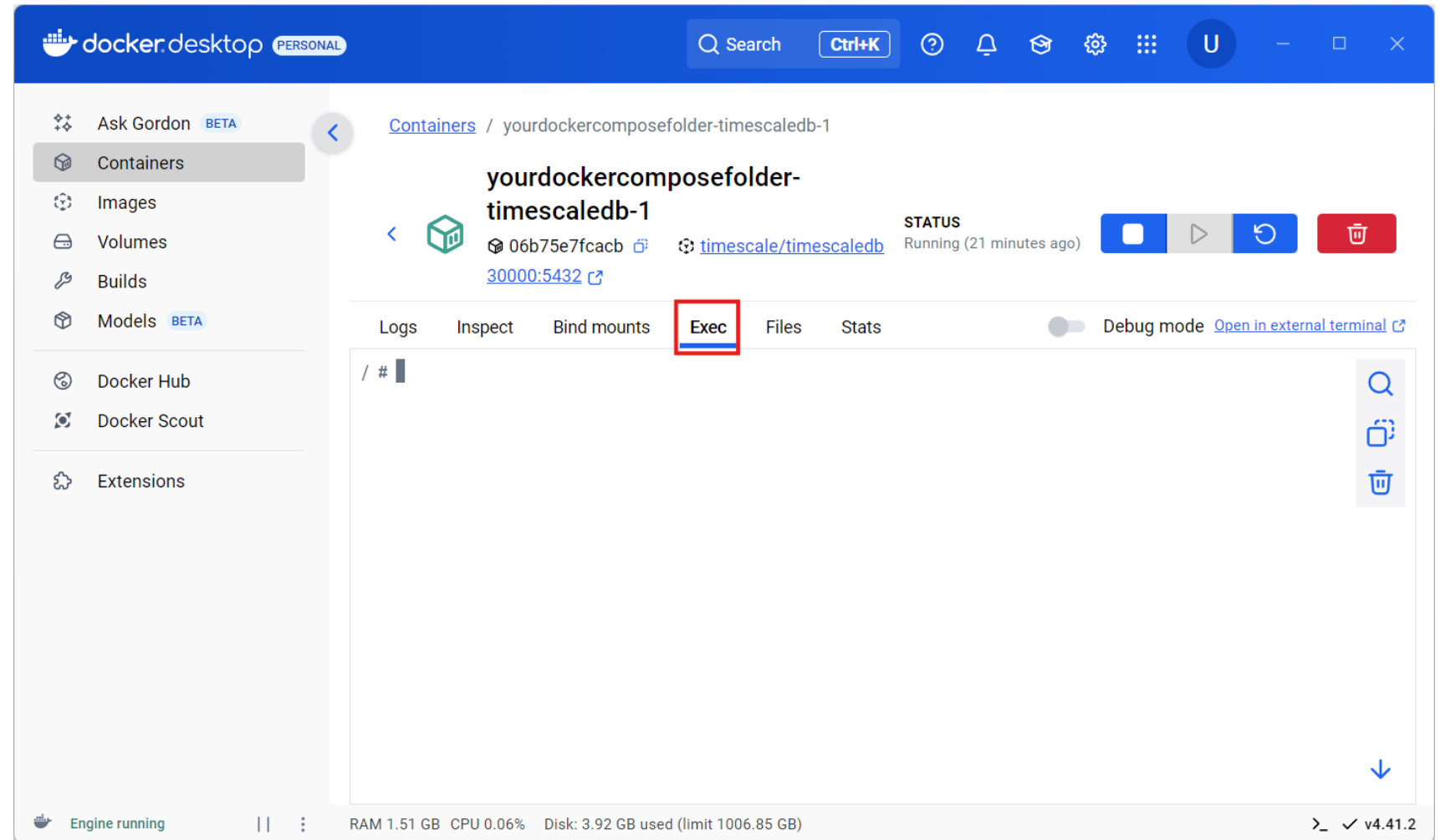
The screenshot shows the Docker Desktop interface. On the left is a sidebar with navigation options: Ask Gordon (BETA), Containers (selected), Images, Volumes, Builds, Models (BETA), Docker Hub, Docker Scout, and Extensions. The main area is titled 'Containers' and shows system usage: Container CPU usage at 0.47% / 1600% (16 CPUs available) and Container memory usage at 179.74MB / 7.44GB. Below this is a search bar and a toggle for 'Only show running containers'. A table lists three containers:

	Name ↑	Container ID	Image	Port(s)	Actions
<input type="checkbox"/>	yourdockercomposefol -	-	-	-	<input type="checkbox"/> ⋮ 🗑️
<input type="checkbox"/>	grafana-1	c13a62f823cb	grafana/grafana	30001:3000	<input type="checkbox"/> ⋮ 🗑️
<input type="checkbox"/>	timescaledb-1	06b75e7fcacb	timescale/tim	30000:5432	<input type="checkbox"/> ⋮ 🗑️

The 'timescaledb-1' container row is highlighted with a red box, and a red arrow points to its checkbox. The status bar at the bottom indicates 'Engine running', 'RAM 1.51 GB', 'CPU 0.00%', 'Disk: 3.92 GB used (limit 1006.85 GB)', and version 'v4.41.2'.

Check if database can be access ?

Step 2:
Go to Exec tab









Check if database can be access ?


Step 3:
Run


psql -d "postgres://postgres:password@localhost/postgres"
to connect to database

[Containers](#) / yourdockercomposefolder-timescaledb-1

yourdockercomposefolder-timescaledb-1



 06b75e7fcacb  [timescale/timescaledb](#) **STATUS** Running (21 minutes ago)    

[30000:5432](#) 

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

Check if dashboard can be access ?

Step 1:

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

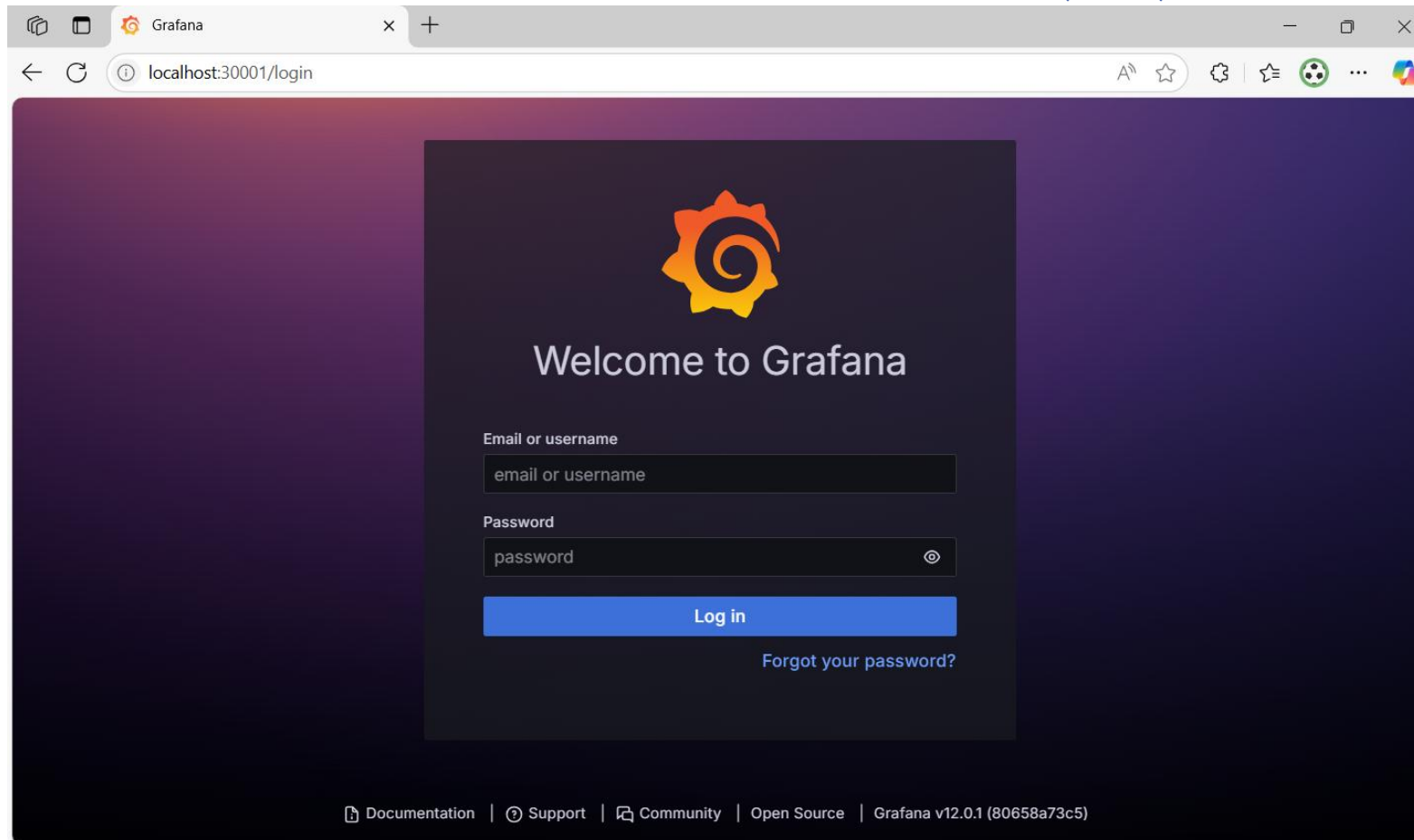
The screenshot shows the Docker Desktop interface. The left sidebar contains navigation options: Ask Gordon (BETA), Containers (selected), Images, Volumes, Builds, Models (BETA), Docker Hub, Docker Scout, and Extensions. The main area displays the 'Containers' tab with a search bar and a toggle for 'Only show running containers'. Below this is a table of running containers:

	Name ↑	Container ID	Image	Port(s)	Actions
<input type="checkbox"/>	yourdockercomposefol -	-	-	-	<input type="checkbox"/> ⋮ 🗑️
<input type="checkbox"/>	grafana-1	c13a62f823cb	grafana/grafana	30001:3000	<input type="checkbox"/> ⋮ 🗑️
<input type="checkbox"/>	timescaledb-1	06b75e7fcac	timescale/tim	30000:5432	<input type="checkbox"/> ⋮ 🗑️

A red arrow points to the '30001:3000' port mapping in the 'grafana-1' row. The status bar at the bottom indicates 'Engine running', 'RAM 1.50 GB', 'CPU 0.00%', 'Disk: 3.92 GB used (limit 1006.85 GB)', and 'v4.41.2'.

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

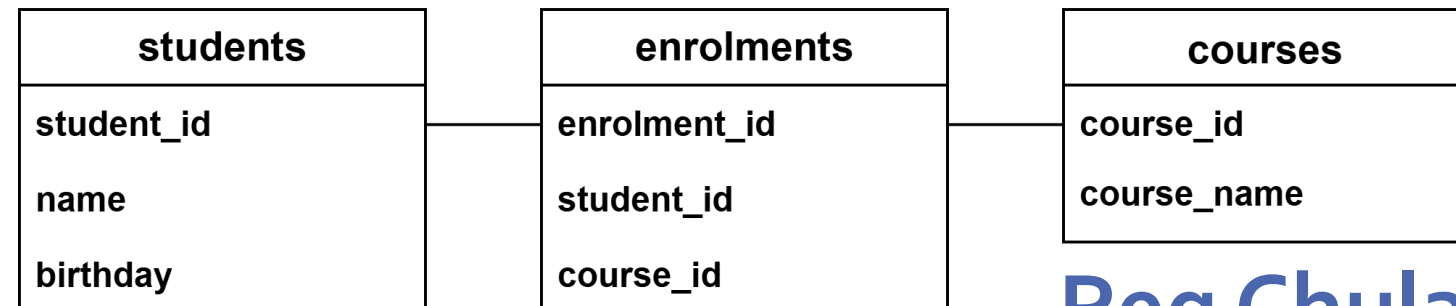


- Relational Database is a type of database that organizes data into tables
- A database can have multiple tables

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

Row

Column



Reg Chula

Postgres Database Command

\dt list tables

```
Logs    Inspect  Bind mounts  Exec  Files  Stats
postgres=# \dt
Did not find any relations.
postgres=#
```

No table exists

RAM 1.57 GB CPU 0.37% Disk: 3.92 GB used (limit 1006.85 GB)

```
Logs    Inspect  Bind mounts  Exec  Files  Stats
postgres=# \dt
List of relations
Schema | Name   | Type | Owner
-----+-----+-----+-----
public | students | table | postgres
(1 row)

postgres=#
```

1 table named students

RAM 1.57 GB CPU 0.00% Disk: 3.92 GB used (limit 1006.85 GB)

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	REAL
humidity	REAL

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	REAL
humidity	REAL

Check created table by **\dt** command

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


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

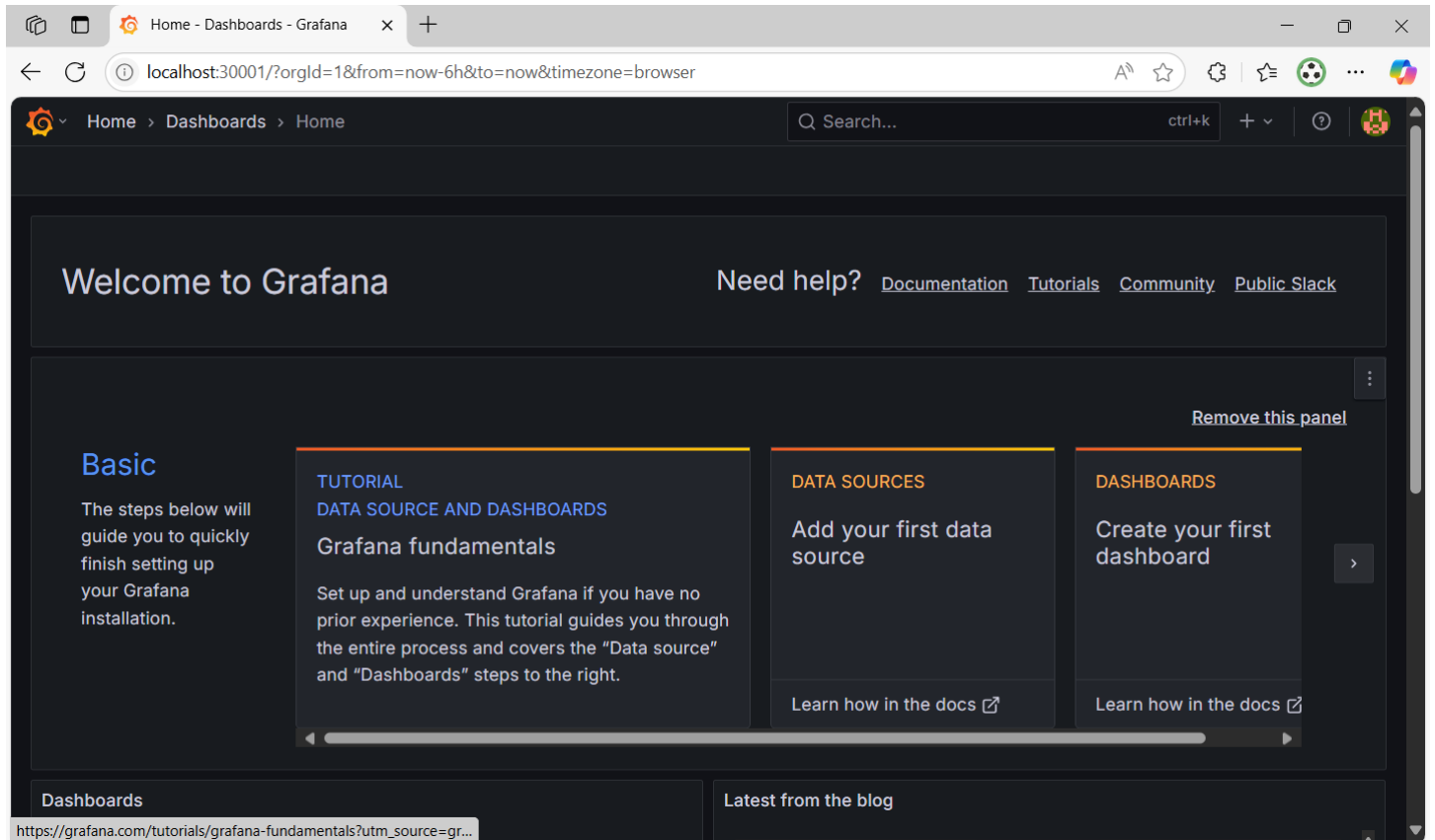


Step 1:

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

User: admin

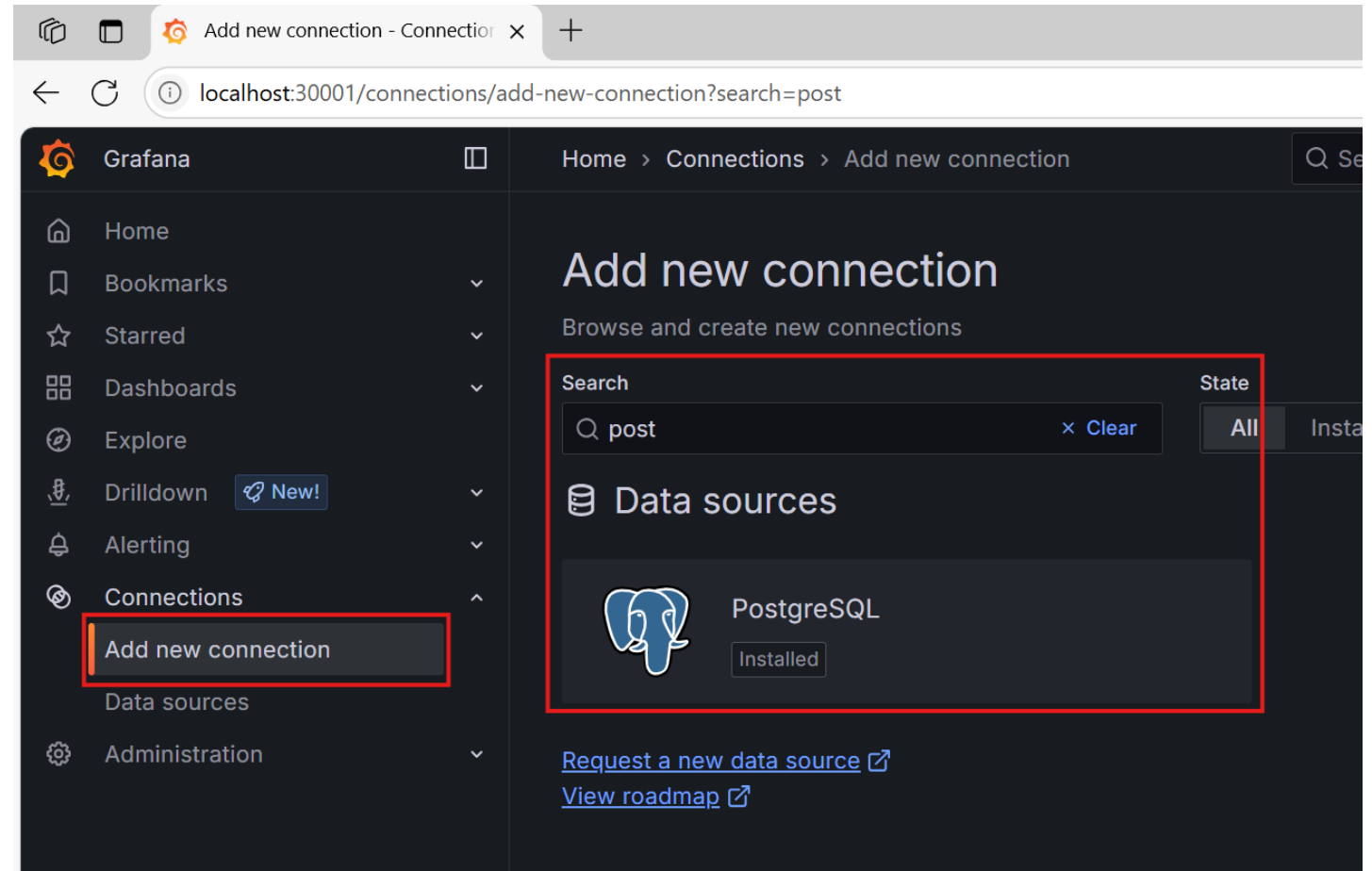
Pass: admin



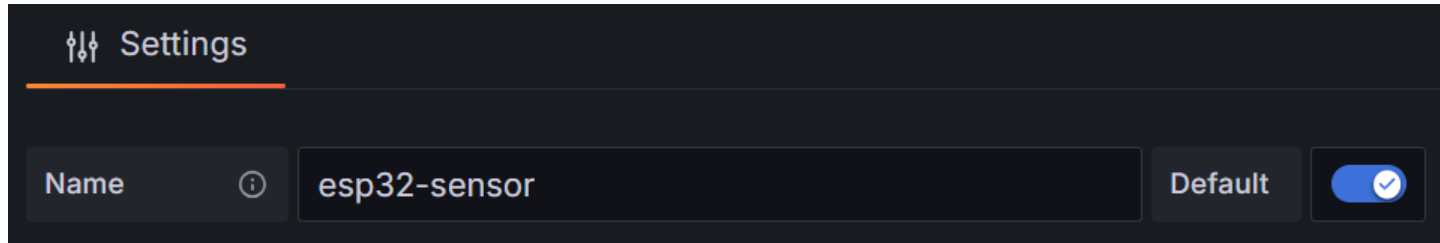
Visualize data in dashboard

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

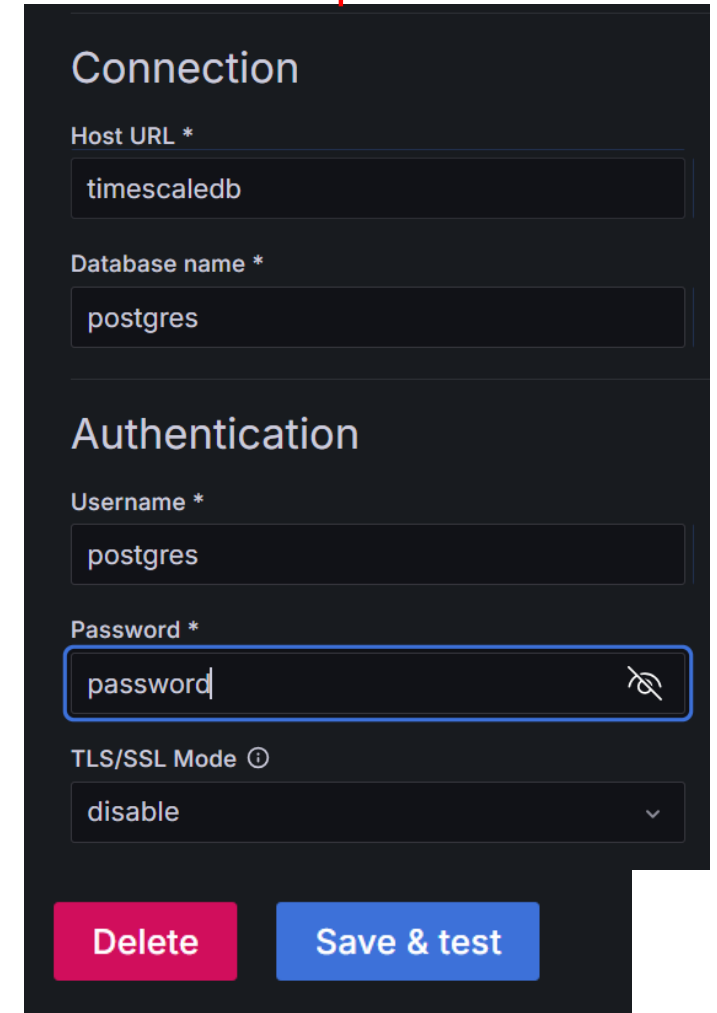


Name	Default	
esp32-sensor	Default	<input checked="" type="checkbox"/>

Step 3:

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

Connection parameters



Connection

Host URL *
timescaledb

Database name *
postgres

Authentication

Username *
postgres

Password *
password

TLS/SSL Mode ⓘ
disable

Delete Save & test

Visualize data in dashboard

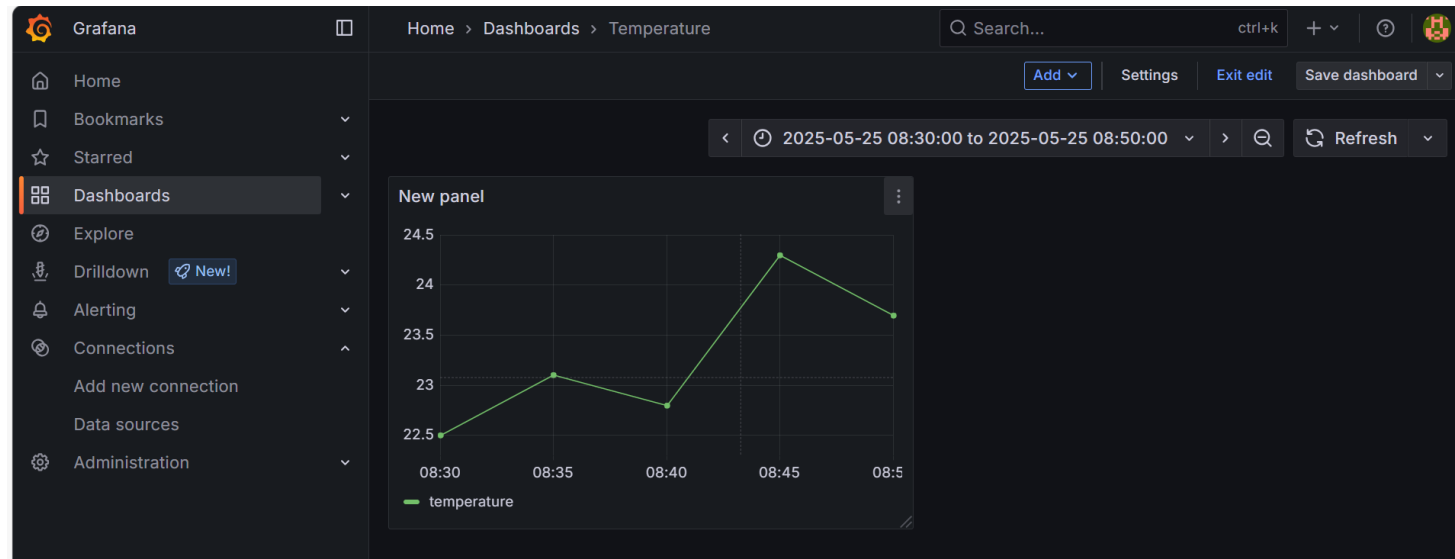
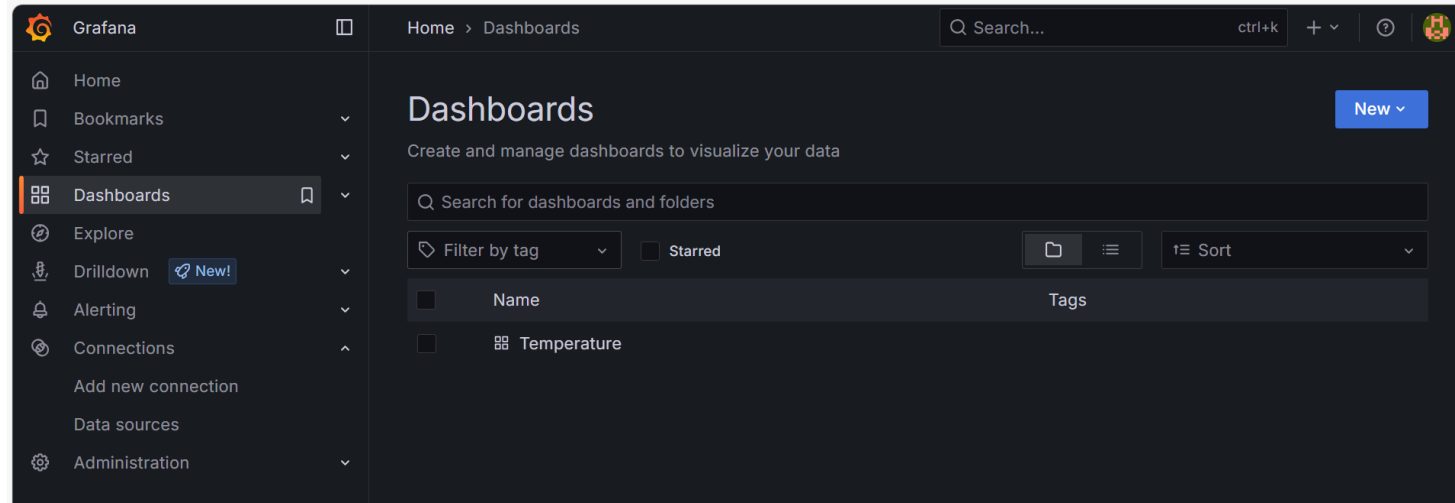
Step 4:

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

The screenshot displays the InfluxDB dashboard interface. The top navigation bar includes 'Queries' (1), 'Transformations' (0), and 'Alert' (0). The 'Data source' is set to 'esp32-sensor'. The 'Query o...' button is visible, along with 'MD = auto = 500' and 'Interval = 1m'. The 'Query inspector' tab is active. Below the query inspector, the 'Format' is set to 'Table', and the 'Preview' toggle is checked. The 'Builder' and 'Code' tabs are shown at the bottom, with 'Code' highlighted. The main area shows a line graph titled 'New panel' with the y-axis labeled 'temperature' ranging from 22.5 to 24.5. The x-axis shows time points: 08:30, 08:35, 08:40, 08:45, and 08:5. The graph shows a line with points at these times. The bottom section shows the 'Code' tab with the query: `1 SELECT "time",temperature FROM sensor_data ;`. The 'Run query' button is visible next to the code editor.

time	temperature
08:30	22.5
08:35	23.25
08:40	22.75
08:45	24.25
08:5	23.75

Visualize data in dashboard



How to connect database using python

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

```
import psycopg2
```

Connection parameters

```
CONNECTION = "postgres://postgres:password@localhost:30000/postgres"
```

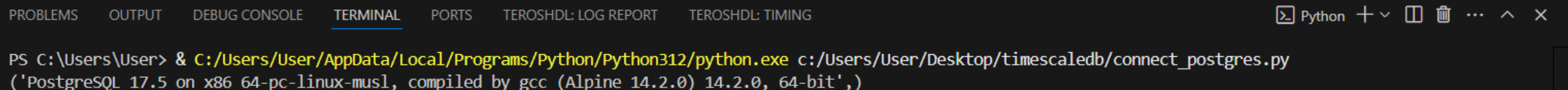
```
with psycopg2.connect(CONNECTION) as conn:
```

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

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

```
        version = cur.fetchone()
```

```
        print(version)
```



The screenshot shows a terminal window with a dark background. At the top, there are tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is active), 'PORTS', 'TEROSHDL: LOG REPORT', and 'TEROSHDL: TIMING'. On the right side of the terminal, there are icons for running a command, a search icon, a window icon, a trash icon, and a close icon. The terminal text shows a command prompt 'PS C:\Users\User>' followed by the command '& C:/Users/User/AppData/Local/Programs/Python/Python312/python.exe c:/Users/User/Desktop/timescaledb/connect_postgres.py'. The output of the script is displayed on the next line: "('PostgreSQL 17.5 on x86_64-pc-linux-musl, compiled by gcc (Alpine 14.2.0) 14.2.0, 64-bit',)'. The command prompt is at the end of the line.

```
PS C:\Users\User> & C:/Users/User/AppData/Local/Programs/Python/Python312/python.exe c:/Users/User/Desktop/timescaledb/connect_postgres.py
('PostgreSQL 17.5 on x86_64-pc-linux-musl, compiled by gcc (Alpine 14.2.0) 14.2.0, 64-bit',)
```

Successfully Connected

How to insert data in database using 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;
```

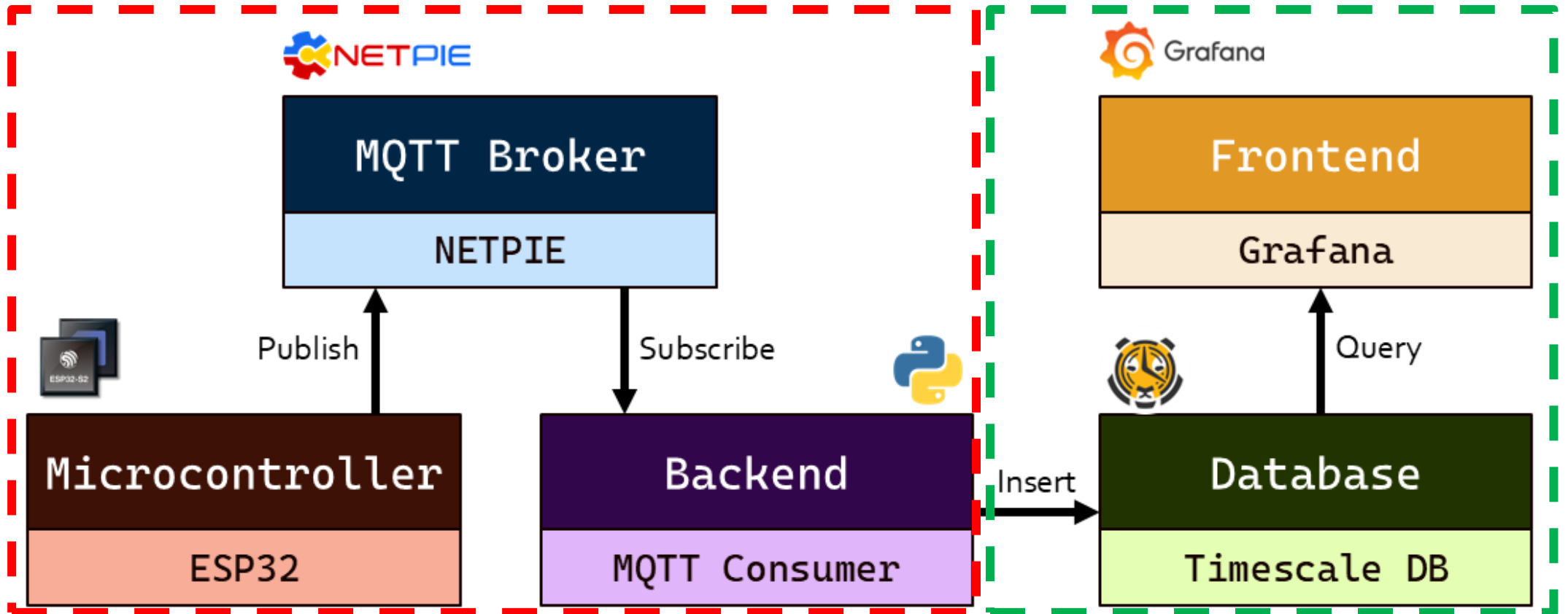
Don't forget to end SQL command with semicolon (;)

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

To receive data from ESP32, recreate the sensor_data table

```
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 ?

