End-to-End Setup Guide: PostgreSQL, pgAdmin 4, and Docker (with Troubleshooting)

Covers: local install, server registration in pgAdmin, Dockerized setup with docker-compose, and common fixes for connection/authentication issues.

Quick Outline

1■■ Install PostgreSQL

Windows (EnterpriseDB installer)

- Download the Windows installer from the official PostgreSQL downloads page (EnterpriseDB).
- Run the installer and select: PostgreSQL Server, pgAdmin (optional), Command Line Tools.
- Set a strong password for the default superuser account: postgres.
- Keep default port 5432 unless you have a conflict. Note the data directory.
- Finish and ensure the Windows service 'postgresql-x64-XX' is set to start automatically.

Verify service:
```
Win+R → services.msc → 'postgresql-x64-…' should be Running
```

macOS

- Option A: Postgres.app → drag to Applications → open to initialize a cluster.
- Option B: Homebrew → brew install postgresql@16 → brew services start postgresql@16.
- Set/confirm password for role 'postgres' if needed; server listens on localhost:5432.

Check status (Homebrew):
```
brew services list | grep postgres
```

Ubuntu/Debian

- Install: sudo apt update && sudo apt install -y postgresql postgresql-contrib
- Verify: systemctl status postgresql (Active: running)
- Open psql: sudo -u postgres psql
- Optionally set password for 'postgres':

```
ALTER USER postgres WITH PASSWORD 'YourStrongPassword';
\q to quit psql.
```

2■■ Verify PostgreSQL and connect with psql

Basic connection to local server:
```
psql -h 127.0.0.1 -p 5432 -U postgres
```
Create and test a database:
```
CREATE DATABASE demo; \c demo; SELECT version();
```

3■■ Install pgAdmin 4 and Register a Server

- Install pgAdmin 4 (standalone or bundled). On first launch, set the master password (for saved connections).
- Register server: Object → Register → Server…
- General tab → Name: Local PostgreSQL

- Connection tab → Host: 127.0.0.1, Port: 5432, Maintenance DB: postgres, Username: postgres, Password: , Save Password: ✓
- Save. Expand 'Servers' → your server → Databases to verify access.

4■■ Install Docker and run services with docker-compose

- Install Docker (Docker Desktop on Windows/macOS; docker-ce on Linux). The compose plugin is included in recent versions.
- Verify: docker --version and docker compose version
- From your project directory (contains docker-compose.yml): start services in background:

```
docker compose up -d
```

Check running containers:

```
docker ps
```

Stop & remove:

```
docker compose down
```

Example docker-compose (PostgreSQL + pgAdmin):

```
services:
  db:
    image: postgres:16
    restart: unless-stopped
    environment:
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: your_password_here
      POSTGRES_DB: appdb
    ports:
      - "5432:5432"
    volumes:
      - pg_data:/var/lib/postgresql/data

  pgadmin:
    image: dpage/pgadmin4
    restart: unless-stopped
    environment:
      PGADMIN_DEFAULT_EMAIL: admin@example.com
      PGADMIN_DEFAULT_PASSWORD: adminpass
    ports:
      - "8080:80"
    depends_on:
      - db

volumes:
  pg_data:
```

5■■ PostgreSQL in Docker (Networking & Connection Strings)

- Inside the compose network, other services connect to the database using host=db and port=5432.
- From your host OS, use the mapped port: psql -h 127.0.0.1 -U postgres -p 5432 -d appdb
- Exec into the container: docker exec -it bash → psql -U postgres
- Typical Python DSN: postgresql://postgres:your_password_here@db:5432/appdb

6■■ Troubleshooting (Common Errors & Fixes)

A) FATAL: password authentication failed for user 'postgres'

- Confirm the exact password used during installation or via POSTGRES_PASSWORD in Docker.
- If Docker: print effective env with `docker compose config` and check postgres section.
- Reset password in psql: ALTER USER postgres WITH PASSWORD 'NewStrongPass';
- Ensure your app uses the same network/host (db) and database name as the container.

Reset inside container:

```
docker exec -it psql -U postgres -c "ALTER USER postgres WITH PASSWORD
'NewStrongPass';"
```

B) could not connect / connection refused

- Service not running: check systemctl (Linux) or Services (Windows) or Docker container status.
- Port not mapped/open: ensure '5432:5432' mapping and no firewall rules block it.
- Wrong host: from host use 127.0.0.1; from another container use service name db.

C) FATAL: role "postgres" does not exist

- If POSTGRES_USER was changed on first run, the superuser may not be 'postgres'.
- Check roles: psql -U -c "\du"
- Create the role if really needed: CREATE ROLE postgres WITH LOGIN SUPERUSER PASSWORD 'Pass';

D) no pg_hba.conf entry / auth method mismatch

- Local development only: ensure pg_hba.conf allows host 127.0.0.1/32 with method 'scram-sha-256' or 'md5'.
- Reload config after edits: SELECT pg_reload_conf(); (or restart the service/container).
- Avoid 'trust' except for throwaway dev environments.

7■■ Useful Maintenance Commands

- Server version: psql -c 'SELECT version();'
- List DBs: \l | List roles: \du | Connect: \c dbname
- Container logs: docker logs -f
- Backup (custom format): pg_dump -h 127.0.0.1 -U postgres -d appdb -F c -f backup.dump
- Restore (custom): pg_restore -h 127.0.0.1 -U postgres -d appdb backup.dump
- Plain SQL backup: pg_dump -h 127.0.0.1 -U postgres -d appdb > backup.sql
- Restore plain SQL: psql -h 127.0.0.1 -U postgres -d appdb -f backup.sql

Note: changing POSTGRES_* env vars on an existing data directory will not reset credentials. To reinitialize, stop containers, delete the volume (e.g., docker volume rm), then start again.