■ PostgreSQL - Docker on Mac Install

Tagline:

"Spin Up PostgreSQL on Mac in Minutes — Docker-Powered, Developer-Ready."

Table of Contents

1. Introduction

Installing PostgreSQL natively on a Mac can introduce system dependencies, conflicts, or bloated environments. With Docker, you get a clean, portable, isolated setup — ideal for local dev, testing, and learning.

2. Why Use Docker for PostgreSQL on Mac

- ■ No need to pollute your host OS
- ■ Version-controlled environments
- ■ Easily replicable for teams
- ■ Containers simulate production-like setups
- ■ Integrates seamlessly with CI/CD and dev tools

3. Prerequisites

- Install [Docker Desktop for Mac](https://www.docker.com/products/docker-desktop)

- Ensure `docker` CLI is working:

docker --version

4. Basic Docker Setup for PostgreSQL

docker run --name postgres-dev -e POSTGRES_PASSWORD=secretpass -e POSTGRES_USER=devuser -e POSTGRES_DB=devdb -p 5432:5432 -d postgres:15

5. Using Docker Volumes for Persistent Storage

docker volume create pgdata

docker run --name postgres-dev -e POSTGRES_PASSWORD=secretpass -v pgdata:/var/lib/postgresql/data -p 5432:5432 -d postgres:15

6. Configuring PostgreSQL Users and Passwords

Use environment variables:

- `POSTGRES_USER`: custom user
- `POSTGRES_PASSWORD`: password
- `POSTGRES_DB`: default database

7. Accessing the Database

CLI:

psql -h localhost -U devuser -d devdb

GUI: Use `localhost:5432`, user `devuser`, db `devdb`.

8. Managing Your Container

docker ps # List running containers

docker stop postgres-dev # Stop container

docker start postgres-dev # Start again

docker logs postgres-dev # View logs

9. Using Docker Compose

Create `docker-compose.yml`:

version: '3.8'

services:

postgres:

image: postgres:15

environment:

POSTGRES_USER: devuser

POSTGRES_PASSWORD: secretpass

POSTGRES_DB: devdb

ports:

- "5432:5432"

volumes:

- pgdata:/var/lib/postgresql/data

volumes:

pgdata:

Run it:

docker-compose up -d

10. Security Considerations

- Use `.env` files for secrets
- Use custom networks
- Lock down port exposure in production
- Apply role-based access controls inside Postgres

11. Backup and Restore

Backup:

docker exec -t postgres-dev pg_dumpall -c -U devuser > backup.sql

Restore:

cat backup.sql | docker exec -i postgres-dev psql -U devuser

12. Integrating with Tools

pgAdmin, DBeaver, DataGrip:

- Host: `localhost`
- Port: `5432`

- DB: `devdb`
- User: `devuser`

13. Troubleshooting

- Port conflicts? Use `lsof -i :5432`
- Logs not showing? `docker logs postgres-dev`
- Container not starting? Check `docker inspect` for environment setup

14. Final Thoughts

Docker gives developers a clean, fast, and flexible way to work with PostgreSQL. Especially on macOS, where installing multiple versions can be painful, Docker provides a version-controlled, portable database stack with full lifecycle control.