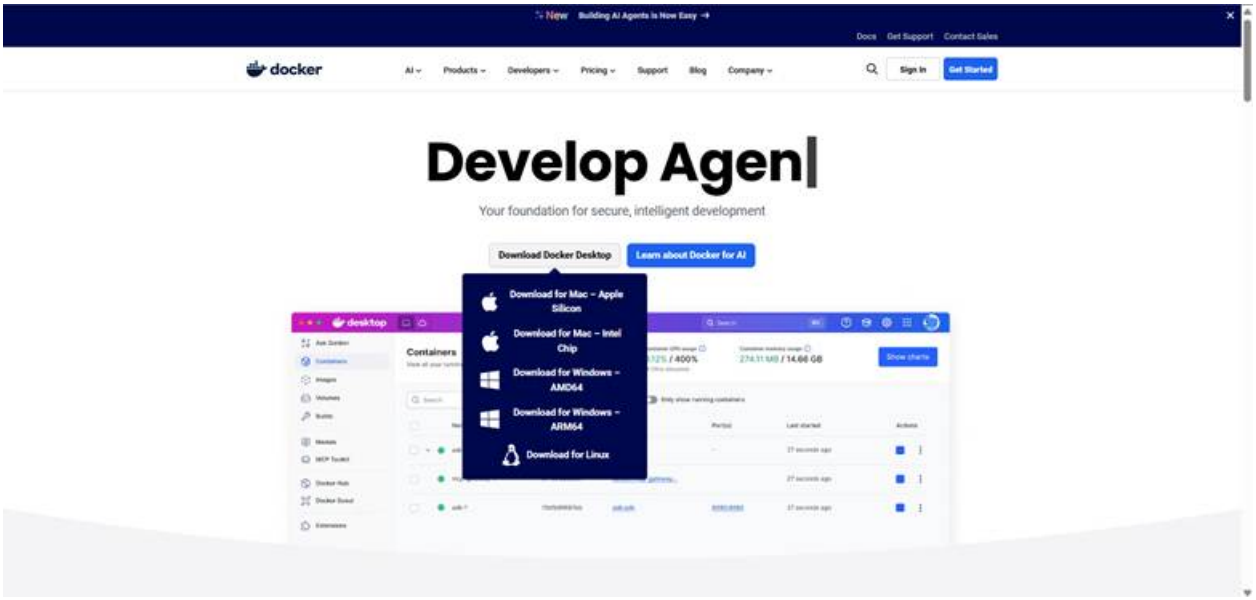
Setting Up Supabase Using Docker Desktop

If you’re new to self-hosting or want a simple, visual setup, Docker Desktop offers the easiest way to run Supabase locally. It provides a graphical interface for managing containers, configuring ports, and ensuring your data persists across restarts—no command-line work required.

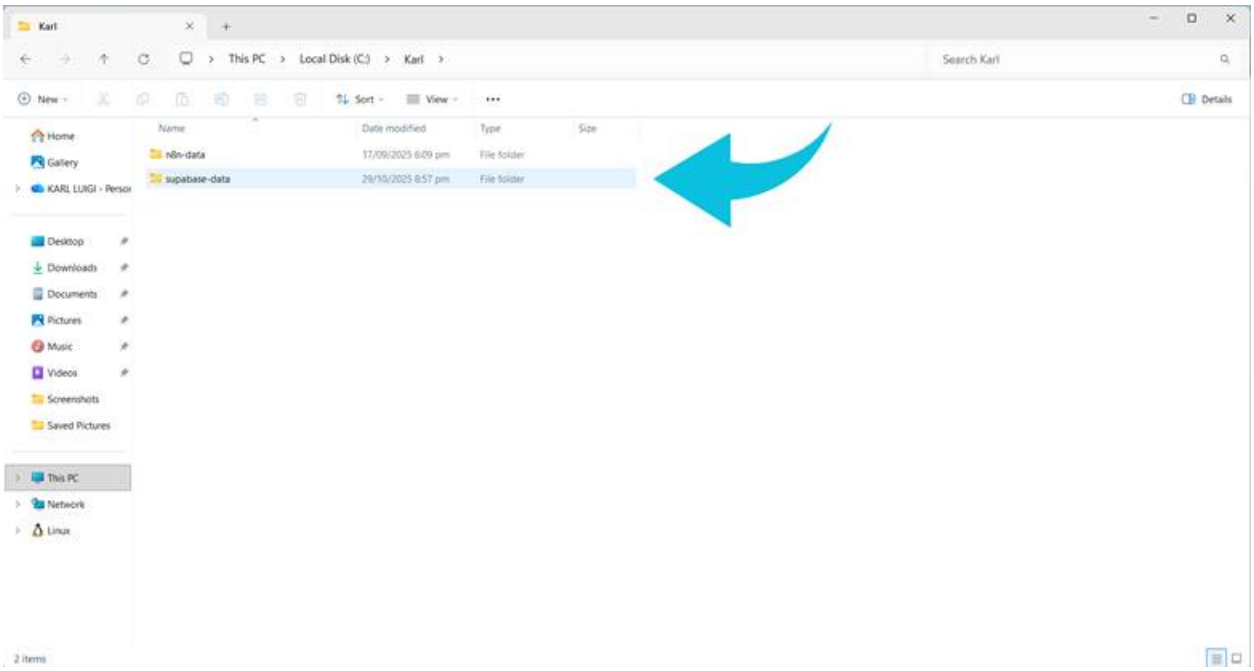
Step 1. Install Docker Desktop

- [Download Docker Desktop] for your operating system.
- Install it following the installer instructions.
  Windows users: Enable **WSL2** during installation and restart if prompted.
- Open Docker Desktop and make sure it’s running. You should see the Docker whale icon in your system tray or menu bar.



Step 2. Prepare a Folder for Data

- Supabase stores data (PostgreSQL database, file uploads) in volumes. To make your data persistent:
  - **macOS / Linux:** ~/supabase-data
  - **Windows:** C:\supabase-data

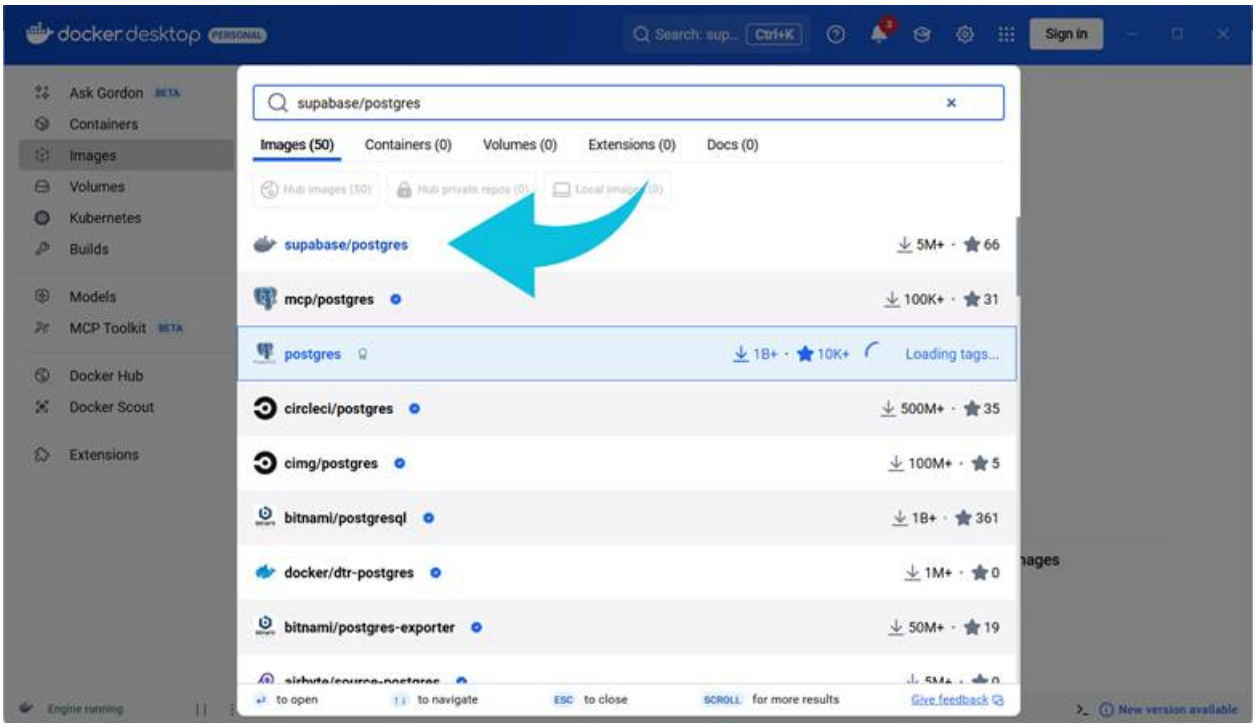


This folder ensures your data is safe even if you remove or update containers.

Step 3. Pull the Official Supabase Images

Supabase uses multiple Docker images for its services: PostgreSQL, Auth, Realtime, Storage, API, and Studio. You can pull them directly in Docker Desktop:

1. Open Docker Desktop and go to the **Images** tab.
2. In the search bar, enter the official image names:
    - supabase/postgres
    - supabase/gotrue
    - supabase/realtime
    - supabase/storage-api
    - supabase/postgrest
    - supabase/studio
3. Click **Pull** for each image to download it to your machine. Repeat for all images.



Step 4. Create Containers for Each Service

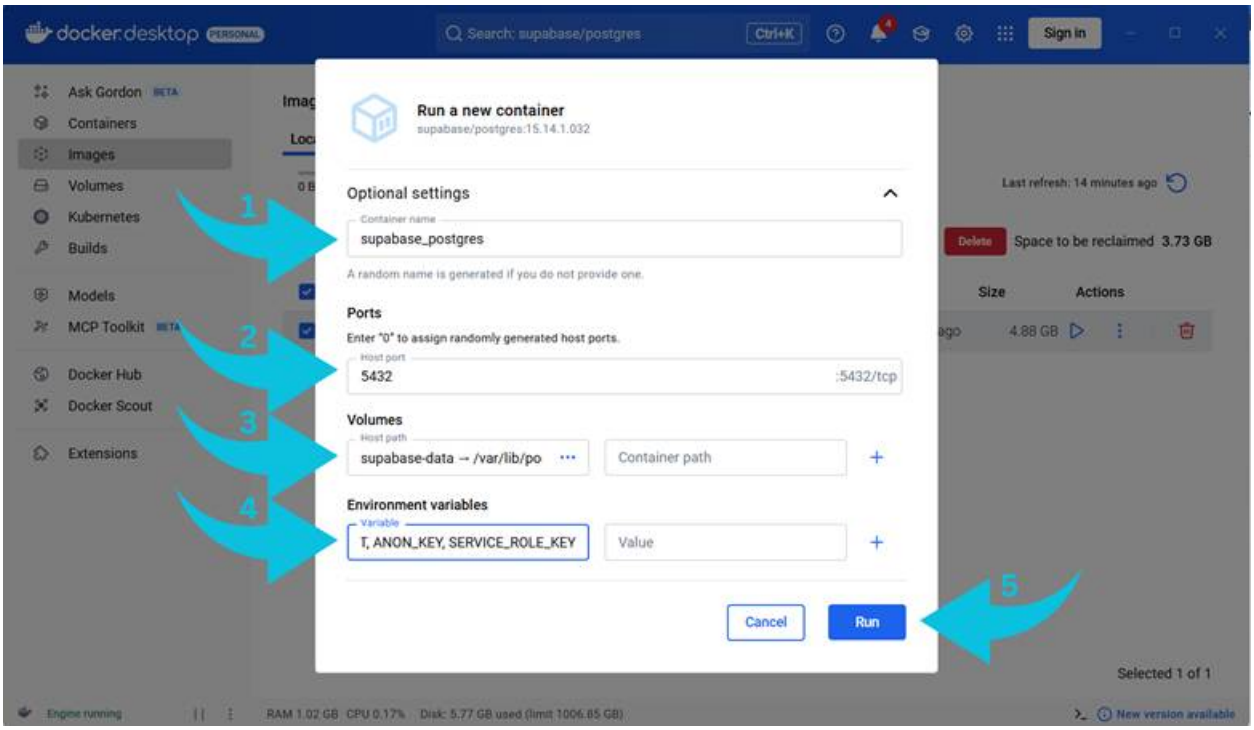
Once the images are downloaded, you need to create a container for each service:

1. Go to the **Containers/Apps** tab in Docker Desktop and click **+ Add Container**.
2. Select the image you just pulled.
3. Configure settings for each container:
    - **Name:** Give it a meaningful name (e.g., supabase_postgres, supabase_auth).
    - **Ports:** Map the container ports to host ports:
        - PostgreSQL: 5432
        - Supabase Studio: 3000
        - Others use default ports (check each service documentation).
    - **Volumes:** Bind your host folder to the container for persistence:
        - Example: ~/supabase-data → /var/lib/postgresql/data for PostgreSQL
    - **Environment Variables:** Set essential keys:

- POSTGRES_PASSWORD, JWT_SECRET, ANON_KEY, SERVICE_ROLE_KEY

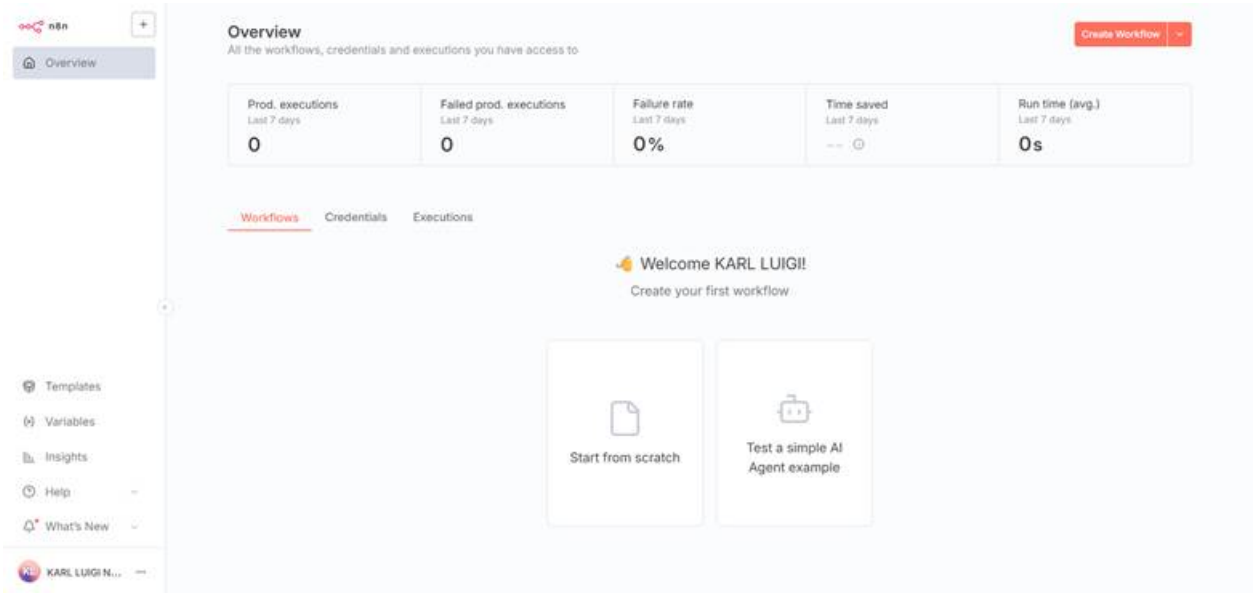
- **Restart Policy:** Always or Unless Stopped

4. Click **Run** to launch the container. Repeat for all Supabase services.



Step 5. Access Supabase Studio

- Once all containers are running, open your browser and go to: http://localhost:3000
- You now have full access to Supabase Studio and can manage your database, authentication, and storage visually.



Step 6. Updating Supabase

When new Supabase releases are available:

1. Stop and remove the containers using Docker Desktop.
2. Pull the updated images (Images tab → Pull).
3. Recreate containers using the same settings and volume mappings.

Because your data is stored in persistent volumes, it **remains safe** during updates.

Step 7. Maintenance and Monitoring

- Docker Desktop lets you monitor resource usage, logs, and container health.
- Regularly check logs for errors and ensure your data folder has sufficient disk space.
- For production or long-term local usage, consider automating container updates using **Watchtower**.

Revision #1
Created 2026-06-17 11:04:50 CEST by DotRoll Knowledge Base
Updated 2026-06-17 11:18:58 CEST by DotRoll Knowledge Base