**Docker Services**

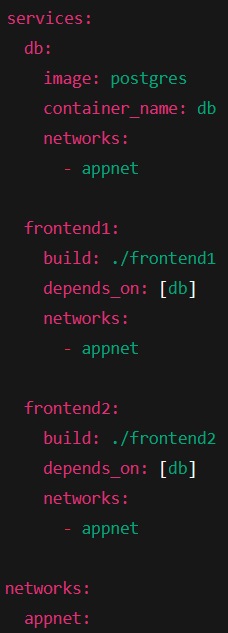
**🧠 What is Docker DNS Resolution?**

When containers are on the **same Docker network**, Docker **automatically provides internal DNS resolution**. This means:

* Containers can **talk to each other using service names** defined in docker-compose.yml.
* No need to hardcode IP addresses — Docker keeps track of IPs and handles name-to-IP resolution dynamically.

**✅ How It Helps Here**

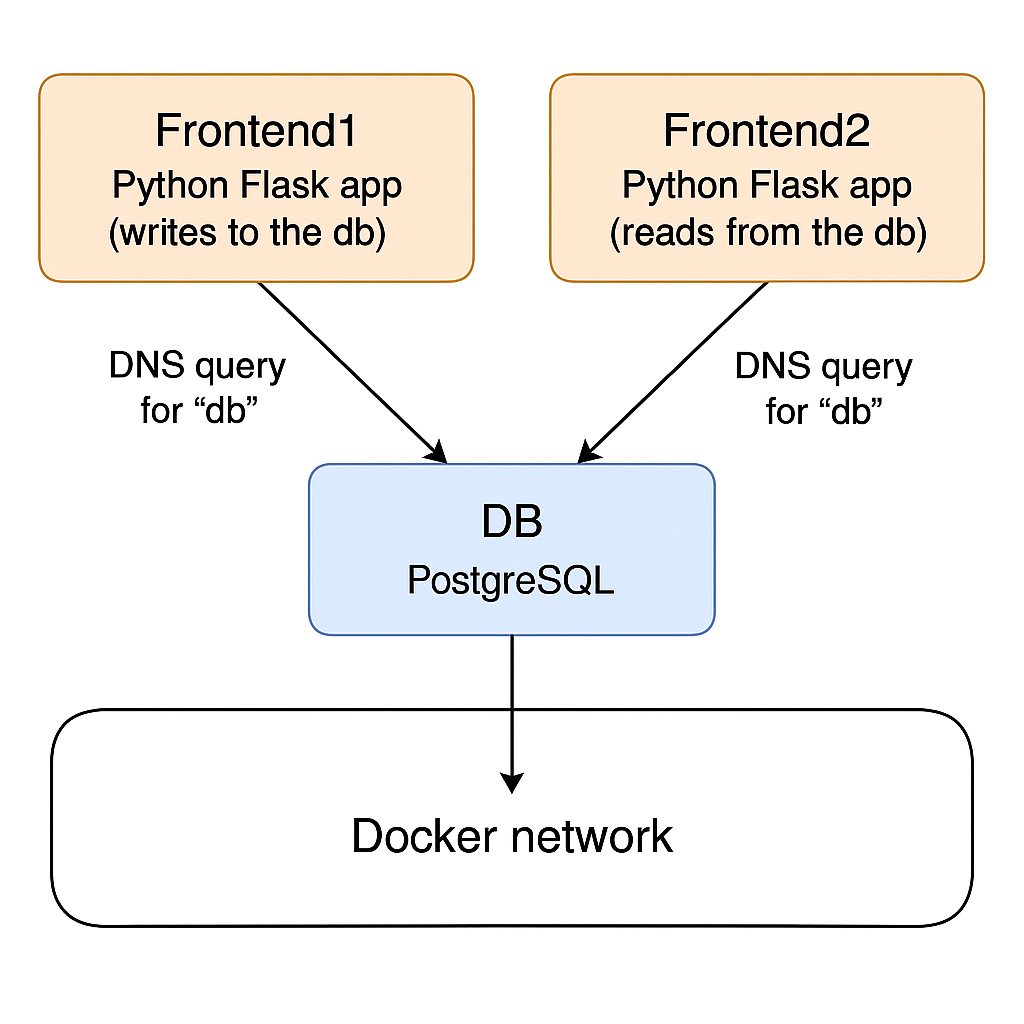
Let’s say our docker-compose.yml includes:



Now, **inside frontend1 and frontend2**, you can connect to PostgreSQL using:

A screen shot of a computer

AI-generated content may be incorrect.

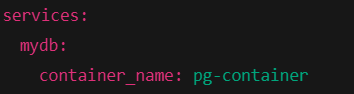


🏆 Benefits of Using Service Name DNS

| **Feature** | **Benefit** |
| --- | --- |
| **Automatic name resolution** | No need to find out IPs manually |
| **Simpler networking** | Containers can talk by name if on same network |
| **Portable config** | Move to any machine or setup without changing DB hostnames |
| **Dynamic IP handling** | If DB container restarts and gets a new IP, the name db still works |

**📌 Service Name ≠ Container Name**

Service names (from docker-compose.yml) act as **hostnames** for DNS resolution — not necessarily the container name.



* pg-container is the container name
* But **mydb is the DNS hostname** you must use inside other containers

**🛑 Don’t Use localhost to Access DB**

Inside frontend1 or frontend2, avoid this:



Why? Because localhost means the **current container**, not another one. Always use the **service name** (e.g., db).

**📄 Check DNS Resolution Inside a Container**

You can verify DNS resolution manually:



This should return responses showing that db is being resolved to an internal IP like 172.x.x.x.

Note: Need to install Ping inside the container (if required)