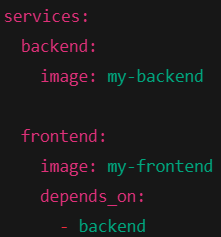
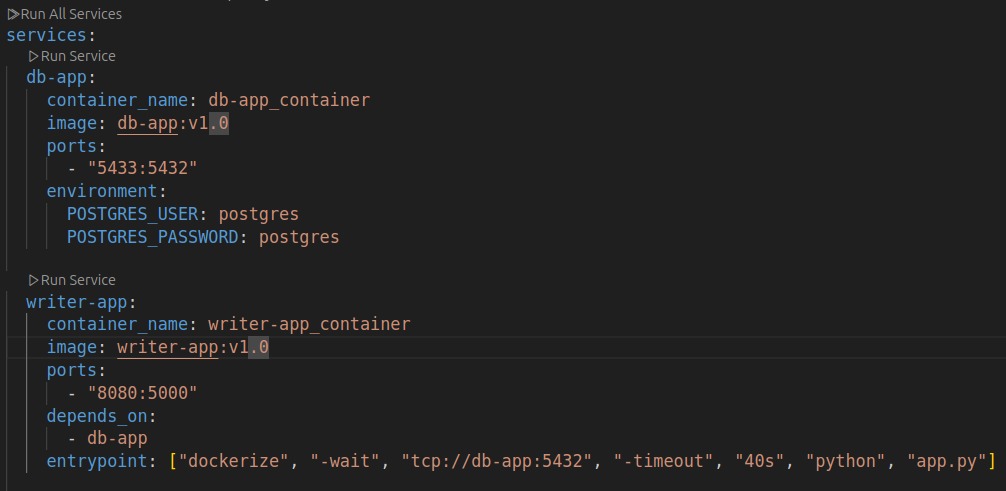
**Docker Compose – depends\_on**

* In a Docker Compose file, the depends\_on key is used to declare **service startup order**.  
    
  
* **This means:**
  + Docker Compose will start the backend container before starting the frontend container.
  + It controls the order of startup only at the container level — not at the application level (i.e., whether it’s fully ready or not).
  + If wer backend takes 10 seconds to boot, but the frontend tries to fetch data in 2 seconds → **❌ error or timeout**.
* **When to use?**
  + You want basic container startup ordering.
  + You don't need to wait for the application inside the container to be ready.
* **Don't rely on it if?**
  + Your app startup requires that another container's services are up and running (e.g., database, API).
* **Workarounds for Readiness (here will see only in v3 version)?**
  + To ensure that a service only starts when another one is ready, we can:
    - wait-for-it.sh
    - dockerize
  + These are the startup wait logic, used to wait for a service/port to become available.

1. **wait-for-it.sh**
   * Download the script from: <https://github.com/vishnubob/wait-for-it>  
     and use below command to make it executable inside Dockerfile  
     *chmod +x wait-for-it.sh*  
     A screen shot of a computer program

     AI-generated content may be incorrect.
   * docker-compose.yml  
     A screenshot of a computer screen

     AI-generated content may be incorrect.
   * **./wait-for-it.sh**: A shell script that waits until a service becomes reachable on a network port.
   * **db:5432**: This means wait until the PostgreSQL container (db) is reachable on port 5432.
   * **--**: This is a standard way to tell wait-for-it.sh: *“End of arguments for wait-for-it. Everything after this will be executed once the port is open.”*
   * **python app.py**: This is wer actual Flask app. Once PostgreSQL is reachable, this command runs to start wer app.
   * Run the docker-compose.yml: *docker-compose up –build*
2. **Dockerize**
   * docker-compose.yml  
     
   * Dockerfile  
     A computer screen shot of a program code

     AI-generated content may be incorrect.  
       
     It means:
     + Wait up to 40 seconds for db-app:5432 to become reachable.
     + Once it's available, run python app.py to start the Flask app.

| **Tool** | **Best For** | **Production Use?** | **Power** | **Simplicity** |
| --- | --- | --- | --- | --- |
| wait-for-it.sh | Simple port checks (TCP) | widely used | ⭐⭐ | ⭐⭐⭐⭐ |
| dockerize | Advanced needs (HTTP, file, TCP) | more powerful | ⭐⭐⭐⭐ | ⭐⭐ |

| **Feature** | **wait-for-it.sh** | **dockerize** |
| --- | --- | --- |
| Size | Lightweight (~10 KB) | Heavier (binary ~1 MB) |
| Wait for TCP ports | ✅ Yes | ✅ Yes |
| Wait for HTTP endpoints | ❌ No | ✅ Yes (-wait http://...) |
| Wait for files | ❌ No | ✅ Yes (-wait file://...) |
| Extensibility | Moderate (we can modify script) | Low (compiled binary) |

✅ **Use wait-for-it.sh** if we just need to wait for a TCP port like db-app:5432.

✅ **Use dockerize** if we need:

* HTTP checks (e.g., wait until a web API returns 200 OK)
* File existence checks (e.g., wait for a config file or socket)

**Real Production Advice:**

In **large-scale deployments**, readiness is usually handled **by the orchestrator** (e.g., Kubernetes readinessProbe) — not the app itself.

But in **Docker Compose or small Docker setups**, especially in CI/CD environments:

* ✅ wait-for-it.sh is **perfect** for quick-and-simple TCP checks.
* ✅ dockerize gives we **more power** if we need HTTP or file readiness.