**Docker Multi-Stage Build**

**🎯 What is a Docker Multi-Stage Build?**

**A multi-stage build allows you to use multiple FROM statements in your Dockerfile to create multiple stages. This helps you:**

* **Build your application in one stage (with all necessary build tools)**
* **Copy only the required artifacts into a clean, minimal final image**

✅ **Benefits:**

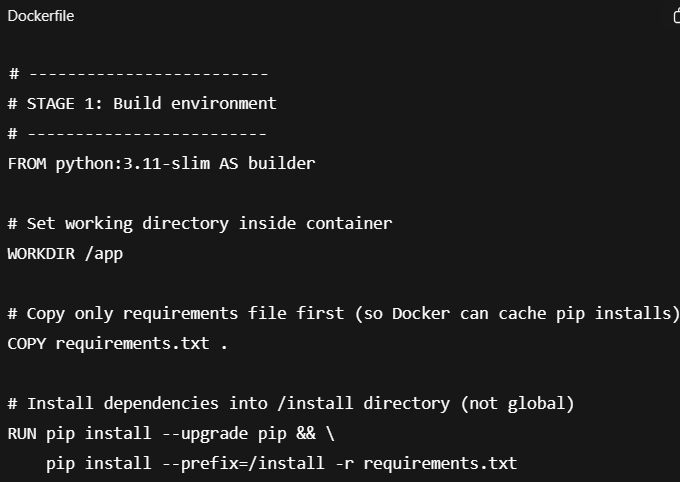
* Smaller final image size
* Clean separation between build and runtime
* Better security (no compilers/tools in final image)

💡 Why Use Multi-Stage Builds?

| **Feature** | **Traditional Dockerfile** | **Multi-Stage Build** |
| --- | --- | --- |
| Image Size | Larger (includes dev tools) | Smaller |
| Layers | Single-stage, more bloated | Clean separation |

**🐳 Dockerfile Breakdown**

Let’s look at the Dockerfile used for a Flask app:



**🔍 Explanation of Stage 1:**

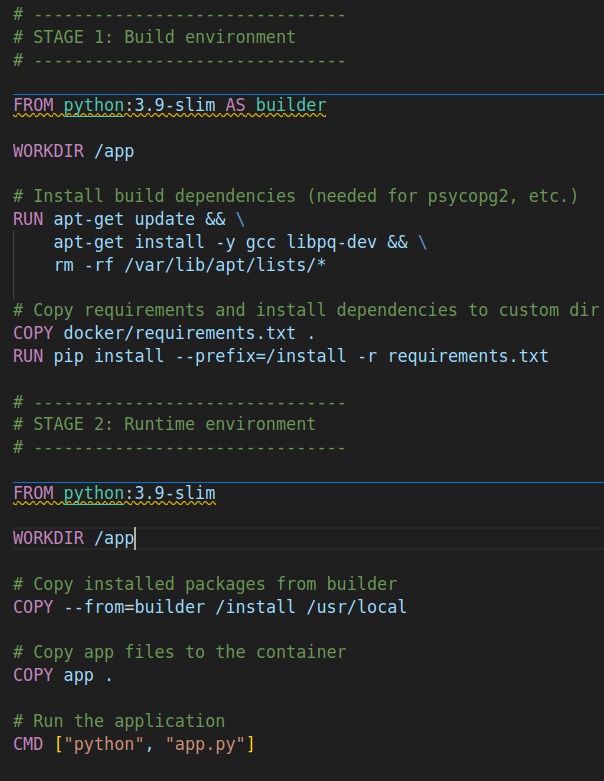
* We use a full Python image (python:3.11-slim) for installing dependencies.
* WORKDIR /app sets a folder in the container to work from.
* COPY requirements.txt . copies the requirements.txt into that folder.
* pip install --prefix=/install ... installs all Python packages into /install so we can later move only this folder, and not unnecessary files.

A screenshot of a computer program

AI-generated content may be incorrect.

**🔍 Explanation of Stage 2:**

* Again, we use a clean python:3.11-slim, but now it has **nothing else** installed.
* COPY --from=builder ... brings only installed packages from the build stage.
* We also copy the app.py file (your Flask application).
* Finally, we run the app using: python app.py.



Practise: For windows image