# Dockerfile Explanation

This Dockerfile is a \*\*multi-stage Dockerfile\*\* for a Python Flask application. Below is a detailed breakdown of each stage and instruction.  
  
------------------------------------------------------------  
Stage 1: Base Stage  
------------------------------------------------------------  
FROM python:3.11-slim AS base  
- Uses the lightweight official Python 3.11 slim image.  
- "AS base" allows this stage to be referenced later.  
  
RUN apt-get update && apt-get install -y --no-install-recommends curl && rm -rf /var/lib/apt/lists/\*  
- Installs curl (for health checks and debugging).  
- Cleans up apt cache to reduce image size.  
  
WORKDIR /usr/local/app  
- Sets the working directory inside the container.  
  
COPY requirements.txt ./requirements.txt  
RUN pip install --no-cache-dir -r requirements.txt  
- Copies requirements.txt first to leverage Docker caching.  
- Installs Python dependencies.  
  
------------------------------------------------------------  
Stage 2: Development Stage  
------------------------------------------------------------  
FROM base AS dev  
RUN pip install watchdog  
ENV FLASK\_ENV=development  
CMD ["python", "app.py"]  
- Inherits from base.  
- Installs watchdog (watches for file changes → auto-reloads app).  
- Sets FLASK\_ENV=development for debug mode.  
- Runs app using "python app.py".  
  
Purpose: For development use only (not production).  
  
------------------------------------------------------------  
Stage 3: Production Stage  
------------------------------------------------------------  
FROM base AS final  
- Starts again from base (avoids unnecessary dev tools).  
  
COPY . .  
- Copies all application code into the container.  
  
EXPOSE 80  
- Declares that the container listens on port 80.  
  
CMD ["gunicorn", "app:app", "-b", "0.0.0.0:80", "--log-file", "-", "--access-logfile", "-", "--workers", "4", "--keep-alive", "0"]  
- Runs Gunicorn (production-grade WSGI server).  
- app:app → import app.py and use the "app" instance.  
- Binds to port 80 and logs to stdout.  
- Uses 4 workers for concurrency.  
- Disables keep-alive connections.  
  
------------------------------------------------------------  
Summary  
------------------------------------------------------------  
- base → Common setup (Python + dependencies).  
- dev → Development stage (hot reload, Flask server).  
- final → Production stage (Gunicorn, optimized).