Dockerfile Image build recipe

CONCEPTS

Each instruction adds a layer, and a changed layer invalidates every layer after it. Put rarely changing steps first.

Base image FROM
↓ RUN / COPY
Layers one per step
↓ cache
Final stage what ships
↓ CMD
Container running process

RUN, COPY and ADD happen at build time. CMD and ENTRYPOINT only describe what runs when a container starts.

BASE AND STAGES

FROM *<image>*:*<tag>*
Start from a base image

FROM *<image>* **AS** *<name>*
Name a build stage

COPY --from=*<stage>* *<src>* *<dest>*
Copy files out of another stage

FROM scratch
Start from an empty image

FROM --platform=$BUILDPLATFORM *<image>*
Run the stage on the build host

docker build --target *<stage>* **.**
Build only up to one stage

ARG *<name>*=*<default>*
Before FROM: parametrize the base

BUILD STEPS

RUN *<command>*
Run a command in a new layer

COPY *<src>* *<dest>*
Copy files from the build context

COPY --chown=*<user>*:*<group>* *<src>* *<dest>*
Copy and set owner

COPY --chmod=*<mode>* *<src>* *<dest>*
Copy and set permissions

BUILD STEPS (cont.)

COPY --link *<src>* *<dest>*
Copy into an independent layer

ADD *<url>* *<dest>*
Download a remote file

ADD *<archive>***.tar.gz** *<dest>*
Copy and auto-extract a local tar

RUN --mount=type=cache, target=*<dir>* *<command>*
Reuse a package cache between builds

RUN --mount=type=secret,id=*<id>* *<command>*
Use a secret without baking it in

RUN a && b && c
Chain steps in one layer

CONFIGURATION

ENV *<key>*=*<value>*
Set a variable for build and runtime

ARG *<name>*[=*<default>*]
Declare a build-time variable

WORKDIR *<path>*
Set the directory for later steps

USER *<user>*[:*<group>*]
Run later steps as this user

LABEL *<key>*=*<value>*
Attach image metadata

RUN useradd -r -u 10001 *<user>*
Create an unprivileged user

RUNTIME

EXPOSE *<port>*[/tcp|/udp]
Document a port

CMD ["exe", "arg"]
Default command (exec form)

ENTRYPOINT ["exe"]
Fixed executable for the container

CMD ["arg1", "arg2"]
Default arguments for ENTRYPOINT

CMD *<command>* *<arg>*
Default command via /bin/sh -c

VOLUME ["*<path>***"]**
Declare a mount point

HEALTHCHECK CMD *<command>*
Probe container health

RUNTIME (cont.)

ENTRYPOINT and CMD

Dockerfile	Runs
`CMD ["a"]`	a
`ENTRYPOINT ["e"]`	e
`ENTRYPOINT ["e"] + CMD ["a"]`	e a
`docker run <image> x`	replaces CMD
`docker run --entrypoint <p> <image>`	replaces ENTRYPOINT

SYNTAX AND CONTEXT

 # syntax=docker/dockerfile:1
Parser directive: current stable frontend

RUN <<EOF
Multi-line script with a heredoc

<command> ****
Continue a line

 # comment
Comment line

docker build -f *<file>* *<context>*
Use another Dockerfile

docker build -t *<name>* **.**
Build with a tag

node_modules
In .dockerignore: skip a path

Keep .git, node_modules and build output out of the context: it makes builds faster and keeps COPY . from invalidating the cache.

CACHE AND SIZE

Copy dependency manifests first, install, then copy the rest of the source. Code edits then do not re-run the install.

Use a multi-stage build: compile in a fat stage, copy only the artifact into a slim final stage.

Clean package manager caches in the same RUN that created them. A later RUN cannot shrink an earlier layer.

Never put secrets in ARG, ENV or COPY: they stay in the layers. Use RUN --mount=type=secret.