O Dockerfile

O ficheiro dockerfile contén as instrucións necesarias para xerar unha imaxe docker, normalmente partindo de outra mais simple. Os pasos a seguir son:

1. Se escribe un ficheiro normalmente chamado **Dockerfile** que contén as instruccións para crear a imaxe
2. Se crea a imaxe chamando a ***docker build*** no directorio que se utilizará como "contexto de compilación". O sistema docker terá acceso a todo o contido do directorio salvo o indicado no ficheiro ***.dockerignore***
3. Se levan a cabo as instrucións indicadas no Dockerfile en orde. Cada instrución crea unha "capa intermedia"
4. O remate se obtén unha imaxe local etiquetada que se poderá executar con ***docker run***

Estrutura do Dockerfile

- ***`FROM <imagen>[:tag|@digest] [AS name]`***
  Base da nosa imaxe. Pódese usar en multi-stage: AS builder. Recoméndase pinzar a imaxe por tag fixo ou por @sha256: para reproducibilidade.
- ***`ARG <name>[=<default>]`***
  Variable só durante o build (non persiste na imaxe a menos que a copies a `ENV`).
- ***`ENV KEY=VAL`***
  Define variables de entorno na imaxe (persisten como parte da capa).
- ***`LABEL key="value"`***
  Metadatos (p.ex. `org.opencontainers.image.*`) — útiles para auditoría e descrición.
- ***`WORKDIR /ruta`***
  Muda o directorio de traballo; crea o directorio se non existe.
- ***`COPY [--chown=user:group] <src> <dest>`***
  Copia ficheiros desde o contexto ao sistema de ficheiros da imaxe. Preferir `COPY` salvo que queiras que extraia tar ou descargue URL (mese con `ADD`).
- ***`ADD <src> <dest>`***
  Pode descomprimir tar local e descargar URL — usar só se precisas esas funcionalidades.
- ***`RUN <comando>` (shell form) ou `RUN ["executable", "param"]` (exec form)***
  Executa comandos durante o build; xera capa. Use exec form cando non necesites shell; use shell form para encadear con &&.
- ***`EXPOSE <port>`***
  Declarativo; documenta o porto que a aplicación usa.
- ***`VOLUME /path`***
  Marca punto de volume; normalmente non se usa en imaxes base de aplicación.
- ***`USER <user|uid>`***
  Cambia o usuario que executará os procesos por defecto (boa práctica non usar root).
- ***`ENTRYPOINT ["executable","param"]`***

  e/ou

  `CMD ["arg1","arg2"]`
  `ENTRYPOINT` fixa o executable principal; `CMD` dá argumentos por defectos. Se ambos existen, CMD pasa como argumentos a ENTRYPOINT. Preferir *exec form* (array) para evitar interpretación polo shell.
- ***`HEALTHCHECK --interval=... CMD curl -f http://localhost/ || exit 1`***
  Define comprobacións de saúde para o contedor.
- ***`ONBUILD`***
  Instrucións que se executan cando outra imaxe base usa a túa imaxe con `FROM` — usar con coidado.

Construindo a Imaxe

docker build -t usuario/miapp:1.0 .
docker build -f Dockerfile.prod -t miapp:prod /ruta/contexto
docker build --no-cache -t miapp:latest .
docker build --build-arg VERSION=1.2 -t miapp:1.2 .

EXEMPLO

docker build -t usuario/miapp:1.0 .
docker run -p 3000:3000 usuario/miapp:1.0

```
# 1. base
FROM node:18-alpine

# 2. metadatos
LABEL org.opencontainers.image.author="Tú"
LABEL org.opencontainers.image.title="miapp"

# 3. workdir
WORKDIR /app

# 4. copiar só package.json primeiro para aproveitar caché
COPY package*.json ./

# 5. instalar dependencias (producción)
RUN npm ci --production

# 6. copiar código
COPY . .

# 7. variables de entorno default
ENV NODE_ENV=production
ENV PORT=3000

# 8. expor porto
EXPOSE 3000

# 9. non correr como root
RUN addgroup -S app && adduser -S app -G app
USER app

# 10. comando por defecto (exec form)
CMD ["node", "server.js"]
```

DOCKER_BUILDKIT=1 docker build --secret id=mysecret,src=./secret.txt -t miapp:secure .

```
# Sintaxe para activar o BuildKit no Dockerfile
# syntax=docker/dockerfile:1.4

FROM alpine:latest
WORKDIR /app

# O segredo é montado no contedor durante este paso RUN
RUN --mount=secret,id=mysecret,dst=/run/secrets/token.txt \
    cat /run/secrets/token.txt > config_token.tmp

# O ficheiro temporal que contén o segredo é usado aquí
RUN echo "O token é usado aquí para configurar a app" && \
    ./setup_app.sh config_token.tmp && \
    # É fundamental ELIMINAR O FICHEIRO co segredo antes de rematar
    rm config_token.tmp

# O segredo xa non existe no sistema de ficheiros da capa final
```

Construción Multietapa. Exemplo

docker build --build-arg BUILD_MODE=staging -t webapp:staging .
docker run -e MODE=debug -p 8080:80 webapp:staging

```
# Etapa 1: build
FROM node:20-alpine AS builder
ARG BUILD_MODE=production
WORKDIR /app
COPY package*.json ./
RUN npm ci
COPY . .
RUN npm run build --mode=$BUILD_MODE

# Etapa 2: imaxe final
FROM nginx:alpine
ARG BUILD_MODE
ENV MODE=$BUILD_MODE
COPY --from=builder /app/dist /usr/share/nginx/html
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

Cando desexamos crear un Docker para unha aplicación que estamos desenvolvendo a construción multietapa facilita a creación de imaxes limpas e seguras. De pequeno tamaño e sen información innecesaria.

Consiste en desenvolver o docker en dúas etapas. Na primeira se crea a imaxe de desenvolvemento creando a aplicación dentro da imaxe orixinal, e na segunda etapa se crea a imaxe final copiando os arquivos necesarios da imaxe de desenvolvemento