# Docker Build

# Índice



# 1. Docker Build

# 1. Docker Build

Docker Build implementa una arquitectura cliente-servidor:

- Cliente: Buildx es el cliente y la interfaz de usuario para ejecutar y gestionar builds.
- Servidor: BuildKit es el servidor, o compilador, que gestiona la ejecución de la build.

**Docker Build Options:**

- Fichero Dockerfile ←
- Argumentos
- Opciones de exportación
- Opciones de Caché

**Docker Build Resources:**

- Filesystem del contexto
- Build Secrets ←
- SSH Sockets
- Registry Auth Tokens

# 2. Dockerfile

# 2. Dockerfile

- Un fichero Dockerfile
  - Conjunto de instrucciones
  - Ejecutadas de forma secuencial
  - Para construir una nueva imagen docker.
- Union File System:
  - Cada una de estas instrucciones crea una nueva capa en la imagen que estamos creando.
- Jerarquía y cache:
  - Docker build intentará optimizar el proceso de construcción.

# 2. Dockerfile

## Ejemplo Dockerfile

```
FROM alpine
LABEL VERSION=0.1 \
      AUTHOR=Jose \
      EMAIL=email@yo.com

RUN apk update \
 && apk add wget \
 && rm -rf /var/cache/apk/*

WORKDIR /root/

ENTRYPOINT [ "wget" ]
CMD [ "--help" ]
```

# 2. Dockerfile

**Ejemplo Dockerfile**

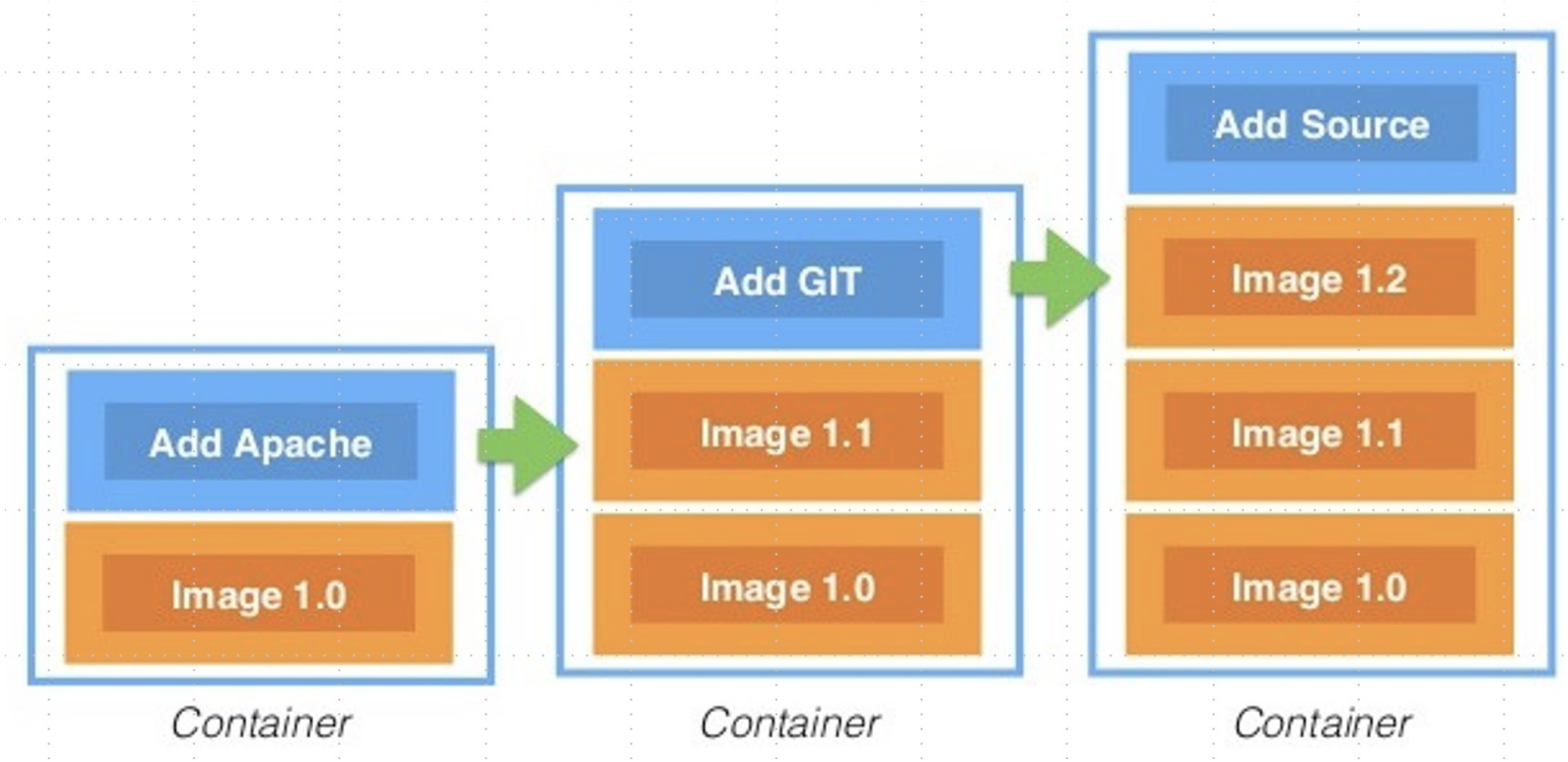
```
FROM ubuntu
EXPOSE 3306
RUN apt-get update &&\
    apt-get -y install && \
        mysql && \
    git && \
    rm -rf /var/lib/apt/lists/*
COPY my.cnf /etc/mysql/my.cnf
VOLUME ["/var/lib/mysql"]

CMD ["/start.sh"]
```

# 2. Dockerfile

**Ejemplo Dockerfile**

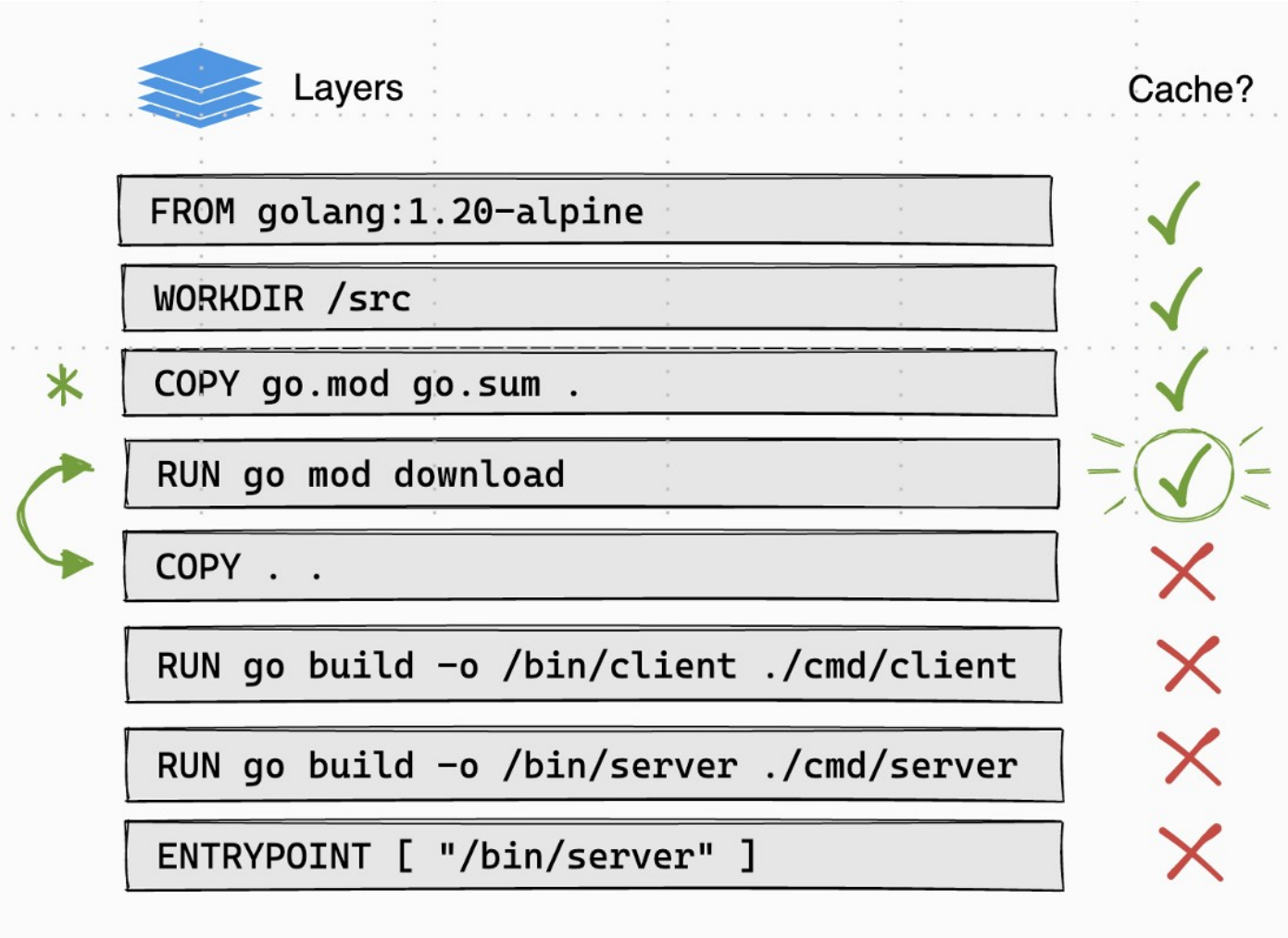
```
FROM python

WORKDIR /app
RUN pip i ansible\
 && pip i ansible-docker\
 && pip i setuptools

ONBUILD ADD . /app/src
ONBUILD RUN python-create-exe
```

# 2. Dockerfile
## Union File System




Capas inmutables: no se puede eliminar o modificar

# 2. Dockerfile
## Jerarquia y Cache




Cuando se ejecuta docker build, el constructor intenta reutilizar las capas de compilaciones anteriores. Si una capa no se modifica, esta se obtiene de cache.

Si una capa cambia desde la última construcción, esa capa y todas las siguientes deben reconstruirse.

# 3. Etiquetas

# 3. Etiquetas

- Imagen base: FROM
- Metadatos: LABEL
- Instrucciones de construcción:
  RUN, COPY, ADD, WORKDIR
- Configuración:
  USER, EXPOSE, ENV
- Instrucciones de arranque:
  CMD, ENTRYPOINT

Documentación: https://docs.docker.com/reference/dockerfile/

# 3. Etiquetas: FROM

- Es obligatorio y debe ser la primera etiqueta.

**FROM imagen[:tag] [AS alias]**

  - Tag: por defecto, latest
  - Alias: Útil cuando se trabaja con múltiples etapas de construcción. Ejemplo.

- Seleccionar una imagen base adecuada para tu aplicación puede simplificar mucho el desarrollo.
- Se recomienda utilizar imágenes base oficiales y de confianza, así como mantenerlas actualizadas para evitar vulnerabilidades de seguridad.

# 3. Etiquetas: LABEL

- Se utiliza para añadir metadatos que documenten y faciliten el mantenimiento de la imagen.

```
LABEL clave=valor clave=valor ...
```

- Aunque puedes definir cualquier etiqueta que desee, Docker recomienda: mantainer, description, versión y vendor (proveedor).
- Se pueden usar variables de entorno:

```
LABEL build_date=$BUILD_DATE
```

- Los metadatos son visibles con el comando:

```
docker inspect <image>
```

# 3. Etiquetas: COPY, ADD, RUN y WORKDIR

- **COPY**: Para copiar ficheros desde mi equipo a la imagen. Esos ficheros deben estar en el mismo contexto (carpeta o repositorio).
- **ADD**:  Es similar a COPY pero tiene funcionalidades adicionales:
  - permite especificar una URL como fuente
  - descomprimir automáticamente los archivos comprimidos

**ADD/COPY [--chown=<usuario>:<grupo>] <fuente> <destino>**

- **RUN**: Ejecuta una orden creando una nueva capa.

**RUN orden**

  - Importante: Durante el proceso de construcción no puede haber interacción con el usuario (-y).
- **WORKDIR**: Establece el directorio de trabajo dentro de la imagen.

# 3-. Instrucciones de construcción

- **COPY**: Para copiar ficheros desde mi equipo a la imagen.
  - Esos ficheros deben estar en el mismo contexto (carpeta o repositorio).
- **ADD**: Es similar a COPY pero tiene funcionalidades adicionales:
  - permite especificar una URL como fuente
  - descomprimir automáticamente los archivos comprimidos

sintaxis es :

**ADD/COPY [--chown=<usuario>:<grupo>] <fuente> <destino>**

- **RUN**: Ejecuta una orden creando una nueva capa.
  - Sintaxis es:

    **RUN orden**
  - Importante: Durante el proceso de construcción no puede haber interacción con el usuario. (-y)
- **WORKDIR**: Establece el directorio de trabajo dentro de la imagen que estoy creando para posteriormente usar las órdenes.

```
# Dockerfile
FROM Ubuntu

LABEL mantainer="Nombre del Autor" \
        vendor=”UA” \
                versión="1.0" \
        descripción="Esta es una imagen de

ejemplo.”

COPY . /app
ADD ./bin/archivo.zip /app2
RUN apt update && \
        apt install –y git
WORKDIR /app
```

# 4-. Variables de entorno

- **USER**: Para especificar (por nombre o UID/GID) el usuario de trabajo para todas las órdenes RUN,CMD Y ENTRYPOINT posteriores.
  - La instrucción USER también afecta a las capas posteriores de la imagen, lo que significa que el usuario especificado se utilizará en todas las capas subsecuentes del Dockerfile a menos que se cambie nuevamente.
- **EXPOSE**: No publica realmente los puertos. Nos da información acerca de qué puertos tendrá abiertos el contenedor cuando se cree uno en base a la imagen que estamos creando. Es meramente informativo.
- **ENV**: Para establecer variables de entorno dentro del contenedor.
  - Se pueden sobrescribir en tiempo de ejecución.

```
# Dockerfile
FROM Ubuntu

LABEL mantainer="Nombre del Autor" \
          vendor=”UA” \
                    versión="1.0" \
         descripción="Esta es una imagen de

ejemplo.”
COPY . /app
ADD ./bin/archivo.zip /app2
RUN apt update && \
       apt install –y git
WORKDIR /app

USER Jenkins
EXPOSE 80
EXPOSE 443
ENV DB_HOST=localhost \
              DB_USER=admin \
DB_PASSWORD=password
```

# 5-. Instrucciónes de arranque

- **ENTRYPOINT**: Un ENTRYPOINT permite configurar un contenedor que se ejecutará como un ejecutable.

  Sintaxis:

  - ENTRYPOINT ["executable", "param1", "param2"]
  - ENTRYPOINT command param1 param2

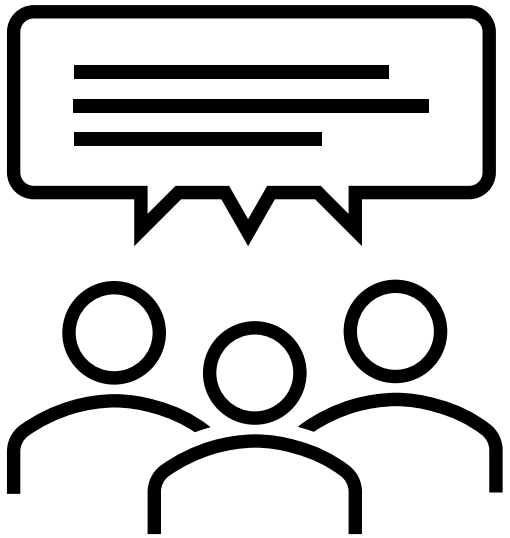
- **CMD**: La instrucción CMD establece el comando a ejecutar cuando se ejecuta un contenedor desde una imagen.

  Sintaxis:

  - CMD ["param1", "param2"]
  - CMD [”executable”, ”param1", "param2"]
  - CMD command param1 param2

- Diferencias entre los formatos exec y shell: evitar que la Shell manipule el formato.

```
# Dockerfile
FROM Ubuntu
LABEL mantainer="Nombre del Autor" \
          vendor=”UA” \
                    versión="1.0" \
          descripción="Esta es una imagen de
                                                ejemplo.”
COPY . /app
ADD ./bin/archivo.zip /app2
RUN apt update && \
       apt install –y git
WORKDIR /app
USER Jenkins
EXPOSE 80
EXPOSE 443
ENV DB_HOST=localhost \
                DB_USER=admin \
DB_PASSWORD=password

ENTRYPOINT [ "wget" ]
CMD [ "--help" ]
# otra opción:
```



¿Preguntas?

# Ejercicios

1. Revisar documentación Dockerfile Reference. https://docs.docker.com/reference/dockerfile/

2. Ejercicios guiados y proyectos propuestos. Ejercicios 2

3. Cualquier proyecto que consideres interesante.

# 2. Build Context