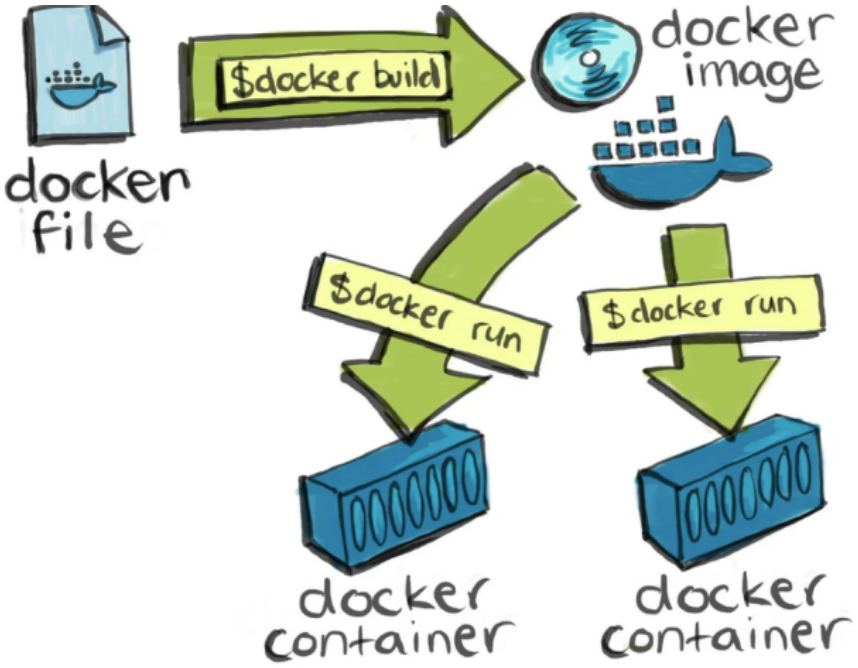
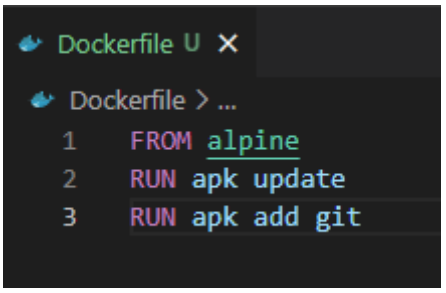
DOCKERFILE

Dockerfile es una receta para construir una imagen.



Creando nuestro primer dockerfile



Para construir una imagen a partir de un archivo Dockerfile.

```
docker build -t sercha30/mi-primera-imagen .
```

La sintaxis del anterior comando es:
docker build [opciones] RUTA | URL | -:
Las opciones más comunes son:

- **-t, nombre [:etiqueta]**. Crea una imagen con el nombre y la etiqueta especificada a partir de las instrucciones indicadas en el fichero. Es una opción muy recomendable.

- –**no-cache**. Por defecto, Docker guarda en memoria caché las acciones realizadas recientemente. Si se diese el caso de que ejecutamos un docker build varias veces, Docker comprobará si el fichero contiene las mismas instrucciones y, en caso afirmativo, no generará una nueva imagen. Para generar una nueva imagen omitiendo la memoria caché utilizaremos siempre esta opción.
- –**pull**. También por defecto. Docker solo descargará la imagen especificada en la expresión FROM si no existe. Para forzar que descargue la nueva versión de la imagen utilizaremos esta opción.
- –**quiet**. Por defecto, se muestra todo el proceso de creación, los comandos ejecutados y su salida. Utilizando esta opción sólo mostrará el identificador de la imagen creada.

Ejecutar un contenedor con una imagen propia y ejecutar un comando al iniciar.

```
docker run --rm -it sercha30/mi-primera-imagen git version
```

Ejercicio de práctica con Node

Crear antes un package*.json

```
FROM node

LABEL maintainer="luismi.lopez@salesianos.edu"

ARG PORT=3000
ENV APP_PORT=${PORT}

RUN npm update npm -g

# Establecemos el directorio de trabajo
WORKDIR /opt/app

# Copiamos el código fuente
# Se trata de uno de nuestros proyectos de ejempo

# Copiamos primero el fichero de manifiesto para instalar las librerías
COPY package*.json ./

# Instalamos todos los módulos y dependencias
RUN npm install

# Copiamos el resto del código
COPY . .

# Exponemos el puerto PORT
EXPOSE ${APP_PORT}

# Ejecutamos la aplicación
CMD [ "npm", "start" ]
```

Dockerizar una aplicación de Spring Boot

Instalar maven y luego pasar nuestra app a jar.

docker build --build-arg JAR_FILE=target/*.jar -t rutademicarpeta/nombreproyecto .

Después crearemos el fichero dockerfile

```
FROM openjdk:8-jdk-alpine
VOLUME /tmp
ARG JAR_FILE
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java","-jar","/app.jar"]
```

COPY

Donde pone app.jar iría el nombre de nuestra aplicación, el 8 del jdk lo tenemos que cambiar por el 17 si nuestra app tiene Java 17.

docker build -t myorg/myapp .

docker run -p 8080:8080 myorg/myapp

A Better Dockerfile(más mejor xd)

```
mkdir target/dependency
(cd target/dependency; jar -xf ../*.jar)
docker build -t myorg/myapp .
```

COPY

Then we can use the following `Dockerfile`

`Dockerfile`

```
FROM openjdk:8-jdk-alpine
VOLUME /tmp
ARG DEPENDENCY=target/dependency
COPY ${DEPENDENCY}/BOOT-INF/lib /app/lib
COPY ${DEPENDENCY}/META-INF /app/META-INF
COPY ${DEPENDENCY}/BOOT-INF/classes /app
ENTRYPOINT ["java","-cp","app:app/lib/*","hello.Application"]
```

COPY