Curso de Docker Ops

Unidad Didáctica 03: Imágenes

http://cursosdedesarrollo.com/

Ayuntamiento
de Vitoria-Gasteiz
Vitoria-Gasteizko
Udala

Índice de contenidos

http://cursosdedesarrollo.com/

Introducción

A través del fichero Dockerfile podemos generar nuevas imágenes quesos van a permitir gestionar nuestras instalaciones

http://cursosdedesarrollo.com/

Introducción

En esta unidad veremos cómo crear estos ficheros de construcción

http://cursosdedesarrollo.com/

Dockerfile

Es el fichero que definirá la imagen a construir

http://cursosdedesarrollo.com/

Dockerfile

Una vez creado y configurado el fichero crearemos la imagen con:

docker build .

docker build -f /path/to/a/Dockerfile .

http://cursosdedesarrollo.com/

Instrucciones

Las instrucciones van a definir las distintas acciones que se deben realizar para crear la imagen

http://cursosdedesarrollo.com/

Instrucciones

Dichas instrucciones deberán de ser lo más genéricas posibles

http://cursosdedesarrollo.com/

Instrucciones

Cada instrucción tendrá la siguiente sintaxis

INSTRUCCIÓN argumentos

http://cursosdedesarrollo.com/

Instrucciones

Ejemplo:

Comentario

RUN echo ‘estamos creado cosas guapas'

http://cursosdedesarrollo.com/

Referencias entres imágenes: FROM

Es la instrucción que permite heredar todas las características iniciales de la imagen

http://cursosdedesarrollo.com/

Referencias entres imágenes: FROM

Normalmente suele definir una base de creación, tal como centos o ubuntu, incluso su versión

FROM ubuntu

FROM ubuntu:16.04

Referencias entres imágenes: FROM

También podríamos empezar de cero

FROM scratch

http://cursosdedesarrollo.com/

Definiciones

Label permite crear clave con valores

LABEL version="1.0"

LABEL description="This text illustrates \

that label-values can span multiple lines."

http://cursosdedesarrollo.com/

Definiciones

WORKDIR permite definir un directorio de actuación para el resto de comandos: RUN, CMD, ENTRYPOINT, COPY y ADD

WORKDIR /path/to/workdir

WORKDIR /var/www

http://cursosdedesarrollo.com/

Definiciones

ENV permite definir variables de entorno, muy utilizado para gestionar versiones o valores a utilizar por instrucciones más adelante

ENV <key> <value>

ENV <key>=<value>

Definiciones

ENV myName=“John Doe”

ENV myDog Rex The Dog

http://cursosdedesarrollo.com/

Comandos

RUN permite la ejecución de comandos en el contenedor, crea una nueva imagen

CMD permite ejecutar comandos que son tuneables, cuando se lanza el contenedor, si no se especifica uno en el lanzamiento

ENTRYPOINT configura que se ejecutará cuando se arranque el contenedor, y lo hará siempre

http://cursosdedesarrollo.com/

Comandos

USER permite definir con qué usuario se ejecutarán los comandos:

USER daemon

Comandos

Diferencias entre ellos:

http://goinbigdata.com/docker-run-vs-cmd-vs-entrypoint/

http://cursosdedesarrollo.com/

Comandos

RUN apt-get install python3

CMD configura “cosas”

ENTRYPOINT ejecuta mongo

http://cursosdedesarrollo.com/

Comandos

RUN ["apt-get", "install", "python3"]

CMD ["/bin/echo", "Hello world"]

ENTRYPOINT ["/bin/echo", "Hello world"]

http://cursosdedesarrollo.com/

Comandos

RUN permite la ejecución de comandos en el contenedor

RUN <command>

por defecto es /bin/sh -c en Linux

RUN ["executable", "param1", "param2"]

http://cursosdedesarrollo.com/

Comandos

```
RUN /bin/bash -c 'source $HOME/.bashrc; \

echo $HOME'
```

http://cursosdedesarrollo.com/

Comandos

CMD echo "Hello world"

http://cursosdedesarrollo.com/

Comandos

ENV name John Dow

ENTRYPOINT echo "Hello, $name"

http://cursosdedesarrollo.com/

Comandos

ENV name John Dow

ENTRYPOINT ["/bin/echo", "Hello, $name"]

http://cursosdedesarrollo.com/

Comandos

ENV name John Dow

ENTRYPOINT ["/bin/bash", "-c", "echo Hello, $name"]

http://cursosdedesarrollo.com/

Comandos

COPY permite copiar ficheros de un sitio a otro

COPY <src>... <dest>

COPY ["<src>",... "<dest>"]

http://cursosdedesarrollo.com/

Comandos

COPY test relativeDir/   # adds "test" to `WORKDIR`/relativeDir/

COPY test /absoluteDir/ # adds "test" to /absoluteDir/

http://cursosdedesarrollo.com/

Comandos

ONBUILD permite ejecutar un comando justo después de que se quiera usar esta imagen por parte de otra, justo después del FROM del nuevo Dockerfile

http://cursosdedesarrollo.com/

Exponiendo Recursos

VOLUME permite crear un punto de montaje dentro de la imagen:

VOLUME ["/data"]

http://cursosdedesarrollo.com/

Exponiendo Recursos

Ese directorio interno puede ser asociado a un directorio externo desde la máquina que ejecuta el contenedor

http://cursosdedesarrollo.com/

Exponiendo Recursos

```
FROM ubuntu

RUN mkdir /myvol

RUN echo "hello world" > /myvol/greeting

VOLUME /myvol
```

http://cursosdedesarrollo.com/

Exponiendo Recursos

docker run -v /Users/<path>:/<container path>:[ro|rw]

ejemplo

docker run -v /Users/pepesan/midir:/myvol:rw

http://cursosdedesarrollo.com/

Exponiendo Recursos

También se pueden montar ficheros en concreto en vez de directorios enteros

https://docs.docker.com/engine/tutorials/dockervolumes/#creating-and-mounting-a-data-volume-container

Exponiendo Recursos

También se pueden exponer puertos con EXPOSE

EXPOSE <port> [<port>...]

http://cursosdedesarrollo.com/

Exponiendo Recursos

De esta manera se exponen puertos del contenedor desde la máquina que ejecuta el contenedor

EXPOSE 80

EXPOSE 80 443

http://cursosdedesarrollo.com/

Exponiendo Recursos

Luego cuando se va a arrancar

docker run --name some-nginx -d -p 8080:80

el primer número marca el puerto en el host

el segundo el puerto expuesto desde el contenedor

http://cursosdedesarrollo.com/

Práctica: Creando una imagen

http://www.projectatomic.io/docs/docker-building-images/

http://cursosdedesarrollo.com/

Conclusiones

http://cursosdedesarrollo.com/

Datos de Contacto

http://www.cursosdedesarrollo.com
david@cursosdedesarrollo.com

http://cursosdedesarrollo.com/

Licencia

David Vaquero Santiago

http://cursosdedesarrollo.com/