
MICROSERVEIS AMB DOCKER

DIA 2





ÍNDEX

- Creació d'imatges
- Ús d'imatges
- Volumes i persistència de dades
- Enllaç entre contenidors: Composició de contenidors i xarxes.

CREACIÓ D'IMATGES: BONES PRÀCTIQUES

- Un contenidor hauria de executar únicament un procés
 - Moodle requereix com a mínim una base de dades i un servidor web. Per tant, hauria d'estar en dos contenidors.
- Sempre utilitzar "receptes" (Dockerfile)
- Quan s'utilitza la instrucció "# docker build" es passa tot el contingut del directori actual com a context (build context).
- Aquest build context es pot utilitzar per a copiar fitxers a la imatge.
- IMPORTANT! No es poden copiar fitxers fora del build context al contenidor. **Podeu esbrinar el motiu?**

CREACIÓ D'IMATGES: INSTRUCCIONS BÀSIQUES

<https://docs.docker.com/engine/reference/builder/>

- FROM
- ADD
- COPY
- RUN
- WORKDIR
- ENV
- VOLUME
- EXPOSE
- ENTRYPOINT
- CMD

FROM

- Especifica la imatge base.
- Totes les instruccions s'executaran sobre aquesta imatge base.
- La imatge s'especifica com a IMATGE:TAG (debian:wheezy).
- Si no s'indica TAG, s'utilitza per defecte "latest"
- HA DE SER LA PRIMERA INSTRUCCIÓ D'UN Dockerfile

ADD

- Permet copiar fitxers:
 - Del build context a la imatge.
 - D'una URL a la imatge.
 - D'un arxiu comprimit, que extrau automàticament.

COPY

- Molt similar a ADD, però no permet la descompressió de fitxers.
- Utilitzat més habitualment.
- Exemple:
 - `COPY --chown=55:mygroup files* /somedir/`



RUN

- Executa una instrucció dins del contenidor i fa un "commit" del resultat.



WORKDIR

- Fixa el directori base sobre el que s'executarà qualsevol comanda RUN, CMD, ENTRYPOINT, COPY o ADD posterior.
- Si aquest directori no existeix, es crea.

ENV

- Permet declarar variables.
- Les variables persisteixen a qualsevol contenidor creat a partir de la imatge resultant.
- Exemples:
 - `ENV myName="John Doe"`
 - `ENV myName John Doe`

VOLUME

- Crea un punt de muntatge al contenidor amb el nom especificat i el marca com a gestionat pel sistema operatiu hoste.
- El nom i ubicació d'aquest directori al sistema operatiu hoste es declara en temps d'execució de cada contenidor basat en aquesta imatge.
- Exemples:
 - `VOLUME /myvol`

EXPOSE

- Aquesta comanda informa al servidor Docker de que els contenidors creats a partir d'aquesta imatge escoltaran en el port indicat en temps d'execució.
- Aquest port unicament serà publicat en temps d'execució del contenidor amb "# docker run", on s'haurà d'especificar un map"real" del sistema operatiu hoste.
- Exemples:
 - EXPOSE 80/tcp
 - EXPOSE 80/udp
 - En temps d'execució farem "# docker run -p 8080:80"

ENTRYPOINT

- Permet especificar quina serà la comanda per defecte que executarà qualsevol contenidor basat en aquesta imatge, i que processarà qualsevol argument passat en temps d'execució al contenidor.
- Habitualment l'entrypoint per defecte és `"/bin/sh -c"`.
- Qualsevol argument de `"# docker run"` posterior al nom de la imatge es passarà com a argument a aquesta comanda.

- Exemple:

```
FROM debian:stable
```

```
RUN apt-get update && apt-get install -y --force-yes apache2
```

```
EXPOSE 80 443
```

```
ENTRYPOINT ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]
```

CMD

- Permet definir quina serà la comanda per defecte que es passarà a l'ENTRYPOINT si no s'especifica cap.
- Únicament pot haver una instrucció CMD a un Dockerfile. Si n'hi ha més d'una, la darrera serà la vàlida.
- Si l'usuari especifica un argument a "#docker run", aquest sobre escriu la instrucció CMD.
- Exemples:
 - Execució a shell: CMD echo "This is a test."
 - Execució sense shell: CMD ["/usr/bin/wc", "--help"]

CREACIÓ D'IMATGES: IMATGES INTERMITGES I CONTENIDORS TEMPORALS

- Observar com funciona el procés de creació d'imatges:

```
FROM busybox:latest
```

```
RUN echo "This should work"
```

```
RUN /bin/bash -c echo "This won't"
```

- `# docker build -t echotest .`
- Veure com es creen imatges intermitges i contenidors temporals.
- Veure com es poden executar contenidors de les imatges intermitges.

CREACIÓ D'UN SERVIDOR WEB AMB CONTINGUT ESTÀTIC

FROM nginx

COPY index.html /usr/share/nginx/html/

- # docker build .
- Què cal afegir?
- Com executaríeu el contenidor?
- Com podem veure el resultat?
- Com mapegem ports?



CREACIÓ D'UNA APLICACIÓ PYTHON SENZILLA

<https://github.com/docker/labs/blob/master/beginner/chapters/webapps.md#23-create-your-first-image>



OPTIMIZANT IMAGES

- Multi stage builds:
 - <https://docs.docker.com/develop/develop-images/multistage-build/>



CREACIÓ D'UNA APLICACIÓ PYTHON SENZILLA II

- Consultar la documentació a <https://docs.docker.com/get-started/part2/>

CREACIÓ D'UNA IMATGE AMB JAVA

- <https://github.com/docker/labs/blob/master/developer-tools/java/chapters/ch03-build-image-java-9.adoc#build-a-docker-image-with-jdk-9>



VOLUMS I PERSISTÈNCIA

- <https://docs.docker.com/storage/>
- <https://docs.docker.com/storage/volumes/>
- https://docs.docker.com/engine/reference/commandline/volume_create/

COMPOSICIÓ DE CONTENIDORS: DOCKER COMPOSE

- Utilitzarem Docker Compose
- Instal·lació:
 - <https://docs.docker.com/compose/install/>



COMPOSICIÓ DE CONTENIDORS: PRIMER EXEMPLE

- <https://docs.docker.com/compose/gettingstarted/>



COMPOSICIÓ DE CONTENIDORS: WORDPRESS

- <https://docs.docker.com/samples/library/wordpress/>



COMPOSICIÓ DE CONTENIDORS: MOODLE

- <https://github.com/bitnami/bitnami-docker-moodle>



COMPOSICIÓ DE CONTENIDORS: XARXES DE COMUNICACIÓ

- <https://docs.docker.com/network/network-tutorial-standalone/>