

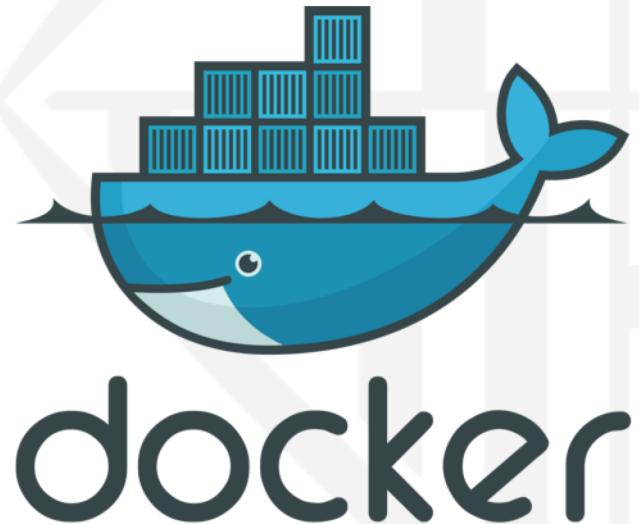


# Introducció a Docker

Carlos Alonso Martínez  
[carlos.alonso@slashmobility.com](mailto:carlos.alonso@slashmobility.com)

Slashmobility

Setembre 2022



# Agenda

Què és Docker?

Instal·lació Docker

Executant contenidors

Exemples

Eines i utilitats

Persistència de dades

Exemple entorns

L'arxiu Dockerfile

Docker Hub

Xarxes a Docker

Docker-Compose

Orquestració i cloud

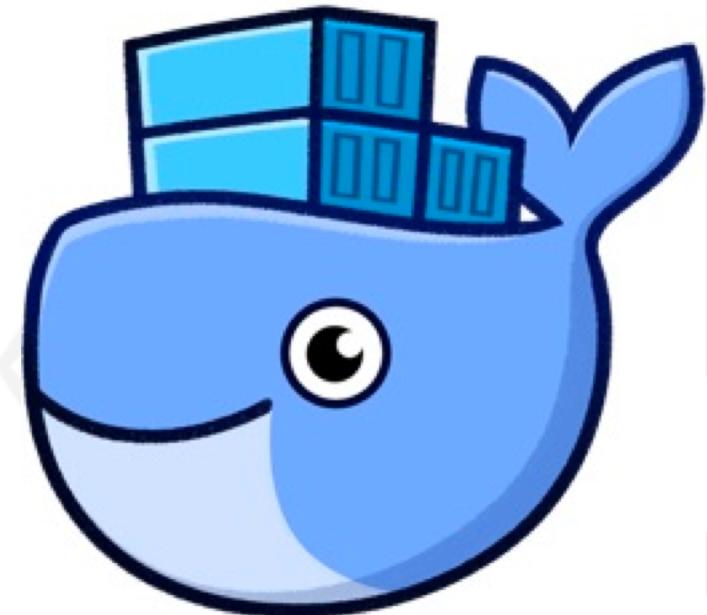
# Què és Docker

Plataforma de software per gestionar contenidors.

Simplifica la creació, distribució i execució d'aplicacions.

Execució nativa en Windows Server i Linux.

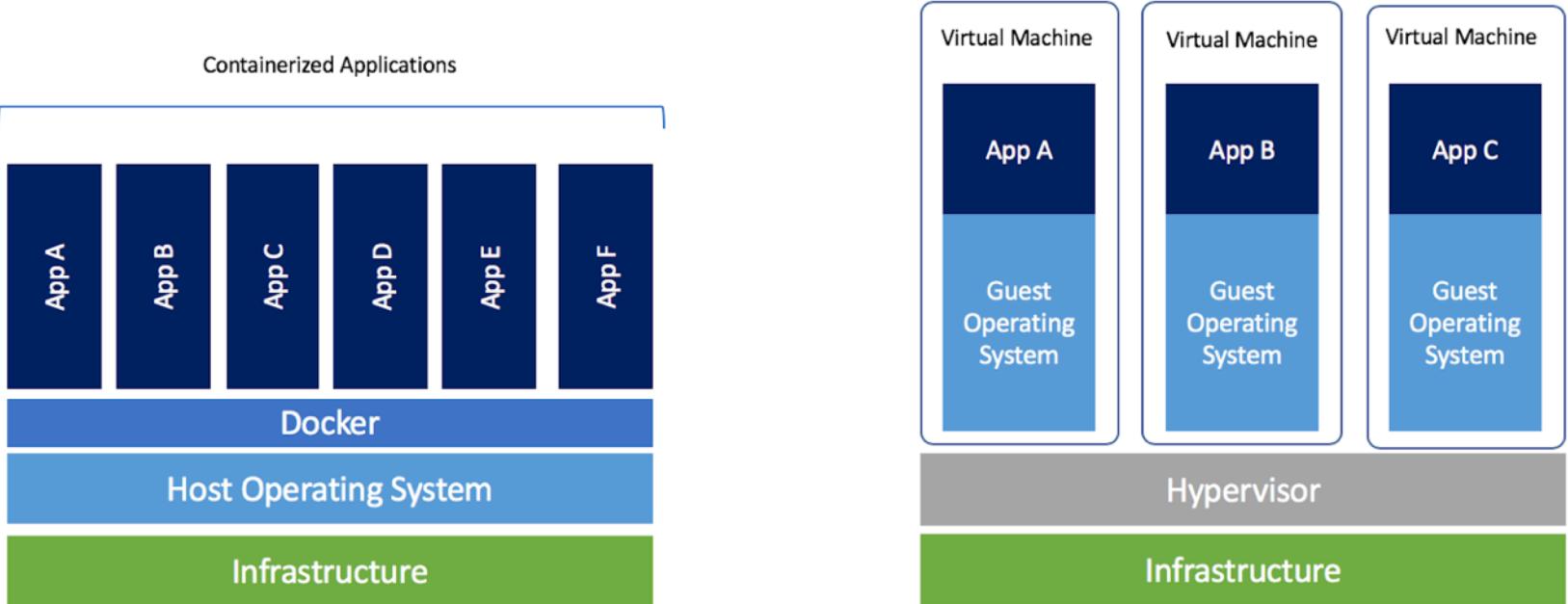
Entorns desenvolupament per Windows i Mac.



# Què és un contenidor?

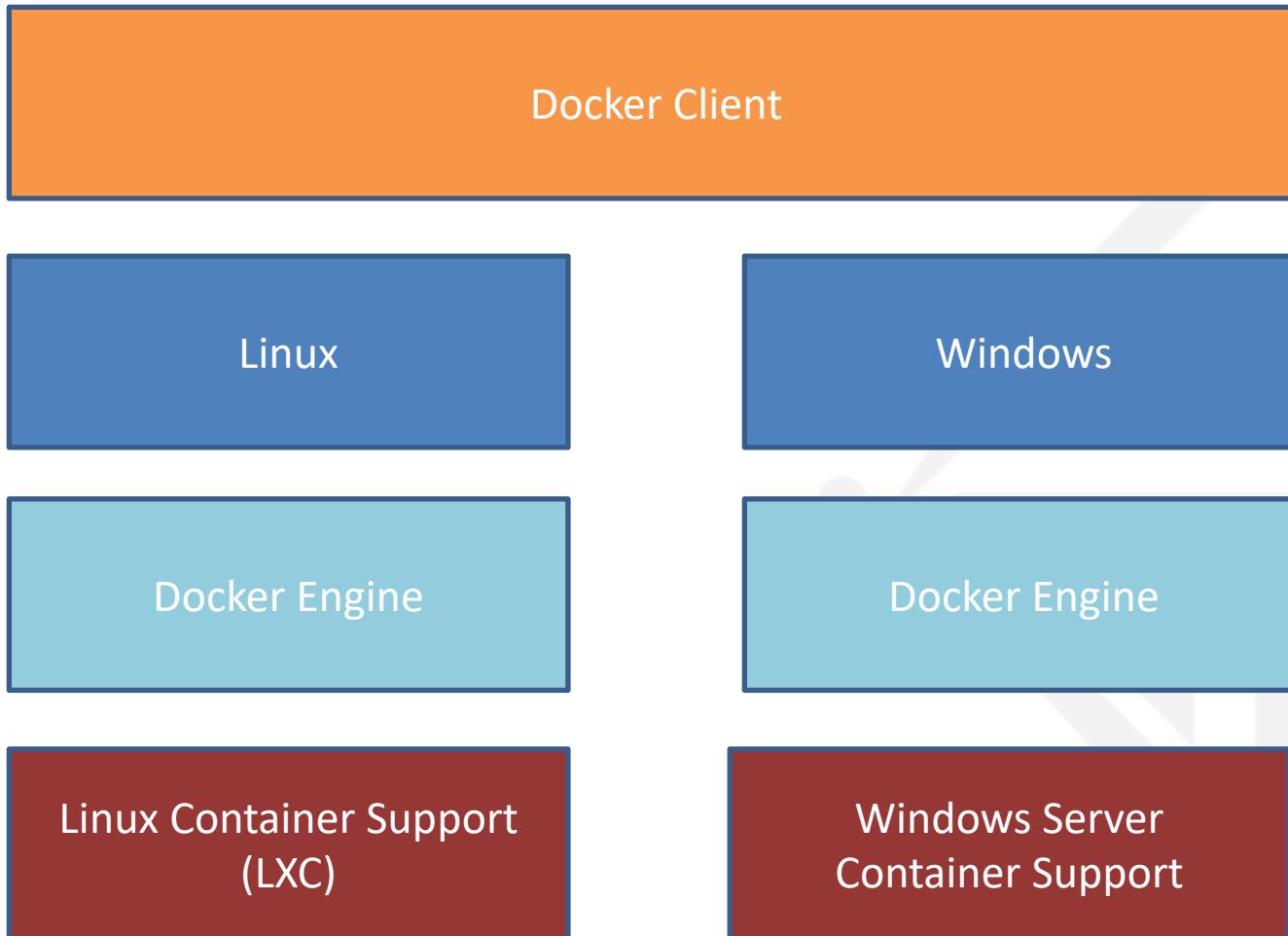
- Un contenidor és un procés aïllat del sistema.
- Contindrà una aplicació i totes les dependències associades.
- Té el seu propi sistema de fitxers.
- Té la seva pròpia xarxa definida.
- Virtualització a nivell de SO.

# VM vs contenidors



font: [blog.docker.com](http://blog.docker.com)

# Docker: execució contenidors



# Avantatges Docker

- Evita el conflicte entre diferents aplicacions.
- Disposar de múltiples entorns de desenvolupament de forma senzilla.
- Desplegar exactament el mateix que s'ha desenvolupat → desplegant contenidors enllot de codi.

# Instal·lant Docker al nostre equip

- Diferents opcions en funció del nostres SO:
  - Docker engine: Linux
  - Docker Desktop: Windows, Mac i Linux



# Docker Desktop

- Windows 10-11 Pro o Enterprise:
  - Contenidors Windows usant WCE
  - Contenidors Linux usant Hyper-V o WSL2
- Mac fa servir Hyperkit com virtualitzador (versions Yosemite o superiors).
- Linux: Ubuntu i altres derivats de Debian.

# Requeriments Windows

Activar o desactivar las características de Windows

Para activar una característica, active la casilla correspondiente. Para desactivarla, desactive la casilla. Una casilla rellena indica que solo está activada una parte de la característica.

The screenshot shows a list of Windows features under the heading 'Activate or deactivate Windows features'. The 'Plataforma de máquina virtual' and 'Subsistema de Windows para Linux' options are highlighted with red boxes. Both have checkmarks in their checkboxes. Other features listed include MultiPoint Connector, Núcleo de web hospedable de Internet Information Services, Protección de aplicaciones de Microsoft Defender, Protocolo de puente del centro de datos, Servicio WAS (Windows Process Activation Service), Servicios de impresión y documentos, Servicios de TCP/IP simple (por ej; echo, daytime etc), Servicios para NFS, Servidor de cola de mensajes Microsoft Message Queue (MSMQ), SMB directo, Windows Identity Foundation 3.5, Windows PowerShell 2.0, Windows Projected File System, and Windows TIFF IFilter.

- MultiPoint Connector
- Núcleo de web hospedable de Internet Information Services
- Plataforma de máquina virtual
- Plataforma del hipervisor de Windows
- Protección de aplicaciones de Microsoft Defender
- Protocolo de puente del centro de datos
- Servicio WAS (Windows Process Activation Service)
- Servicios de impresión y documentos
- Servicios de TCP/IP simple (por ej; echo, daytime etc)
- Servicios para NFS
- Servidor de cola de mensajes Microsoft Message Queue (MSMQ)
- SMB directo
- Subsistema de Windows para Linux
- Windows Identity Foundation 3.5
- Windows PowerShell 2.0
- Windows Projected File System
- Windows TIFF IFilter

# Docker engine

- En els sistemes Linux només cal instal·lar Docker Engine, perquè s'executa directament sobre el kernel Linux.
- Docker Engine CE.

# Docker Hub

## Personal

Ideal for individual developers, education, open source communities, and small businesses.

**\$0**

- Docker Desktop ⓘ
- Unlimited public repositories
- Docker Engine + Kubernetes ⓘ
- 200 image pulls per 6 hours
- Unlimited scoped tokens ⓘ

[Start Now](#)

## Pro

Includes pro tools for individual developers who want to accelerate their productivity.

**\$5** /month

- ← Everything in Personal plus:
- Docker Desktop ⓘ
  - Unlimited private repositories
  - 5,000 image pulls per day
  - 5 concurrent builds ⓘ
  - 300 Hub vulnerability scans

[Buy Now](#)

Billed annually for \$60.

## Team

Ideal for teams and includes capabilities for collaboration, productivity and security.

**\$7** /user/month  
Start with minimum 5 users for \$25.

- ← Everything in Pro, plus:
- Docker Desktop ⓘ
  - Unlimited teams
  - 15 concurrent builds ⓘ
  - Unlimited image scans
  - Role-based access control
  - Audit logs ⓘ

[Buy Now](#)

Billed annually starting at \$300.

## Business

Ideal for medium and large businesses who need centralized management and advanced security capabilities.

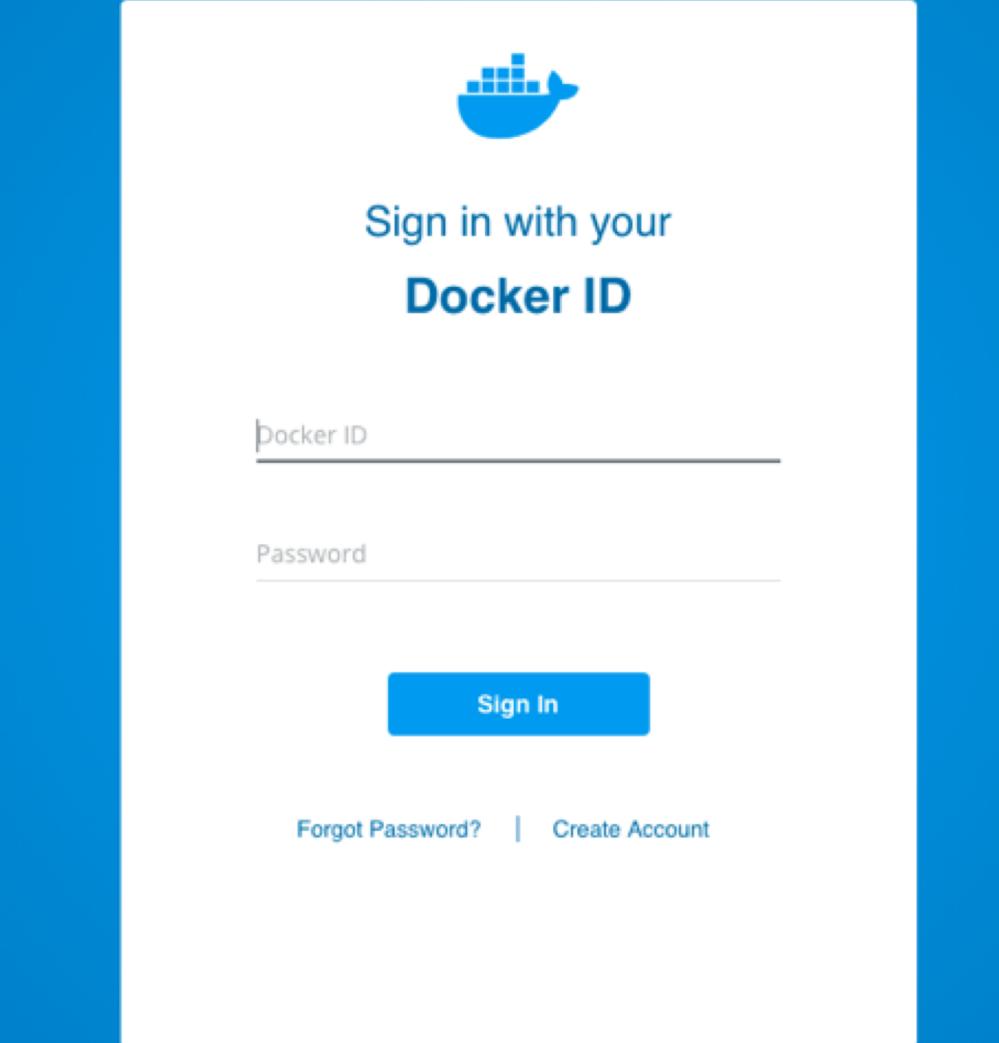
**\$21** /user/month

- ← Everything in Team, plus:
- Docker Desktop ⓘ
  - Centralized management
  - Image Access Management
  - Single Sign-On (SSO)
  - Purchase via invoice
  - Volume Pricing Available

[Contact Sales](#)

[Buy Now](#)

# Docker Hub



The image shows the Docker Hub login page. It features a blue header bar at the top and bottom. The central white area contains the Docker logo (a blue ship icon) and the text "Sign in with your Docker ID". Below this are two input fields: "Docker ID" and "Password", each with a corresponding placeholder text. A blue "Sign In" button is centered below the password field. At the bottom, there are links for "Forgot Password?" and "Create Account".

Sign in with your  
**Docker ID**

Docker ID

Password

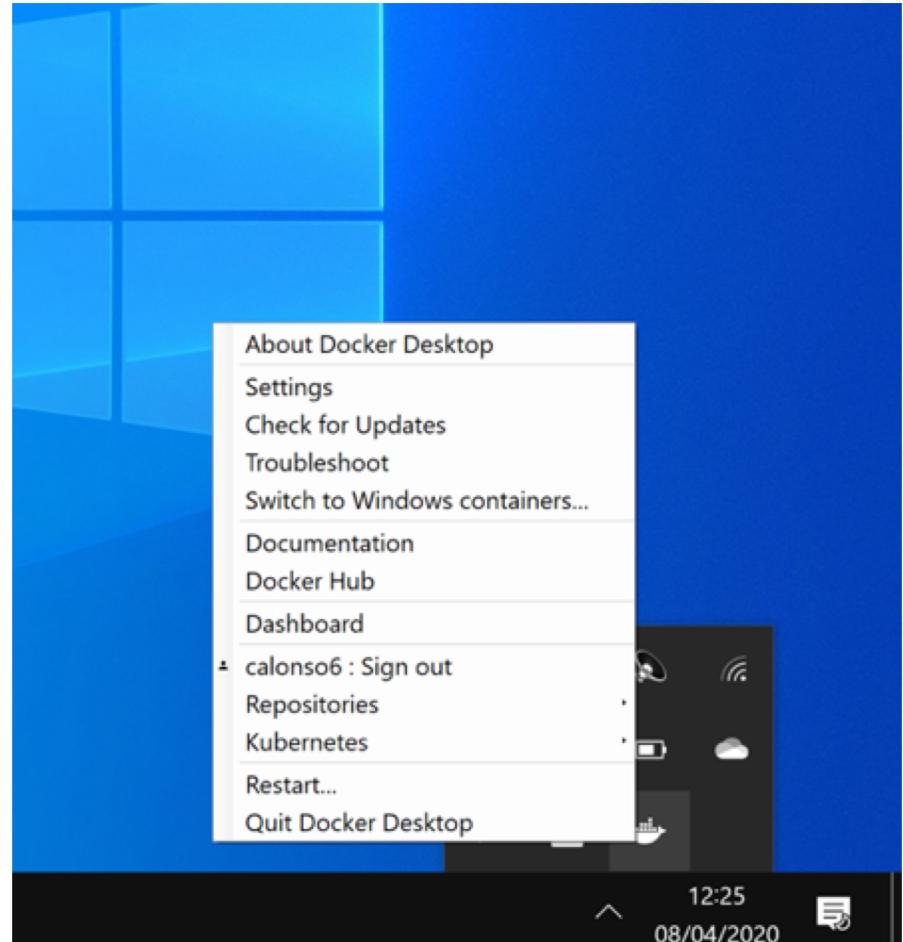
Sign In

[Forgot Password?](#) | [Create Account](#)

# Docker Desktop

Un cop instal·lat en apareix la icona a la barra de tasques.

En validarem amb el compte que hem creat.



# Docker Desktop: configuraciones

Settings X

General Resources

Docker Engine Command Line

Kubernetes

## General

Adjust how Docker Desktop behaves according to your preferences.

Start Docker Desktop when you log in

Automatically check for updates

Expose daemon on tcp://localhost:2375 without TLS

Exposing daemon on TCP without TLS helps legacy clients connect to the daemon. It also makes yourself vulnerable to remote code execution attacks. Use with caution.

Use the WSL 2 based engine

WSL 2 provides better performance than the legacy Hyper-V backend. [Learn more](#).

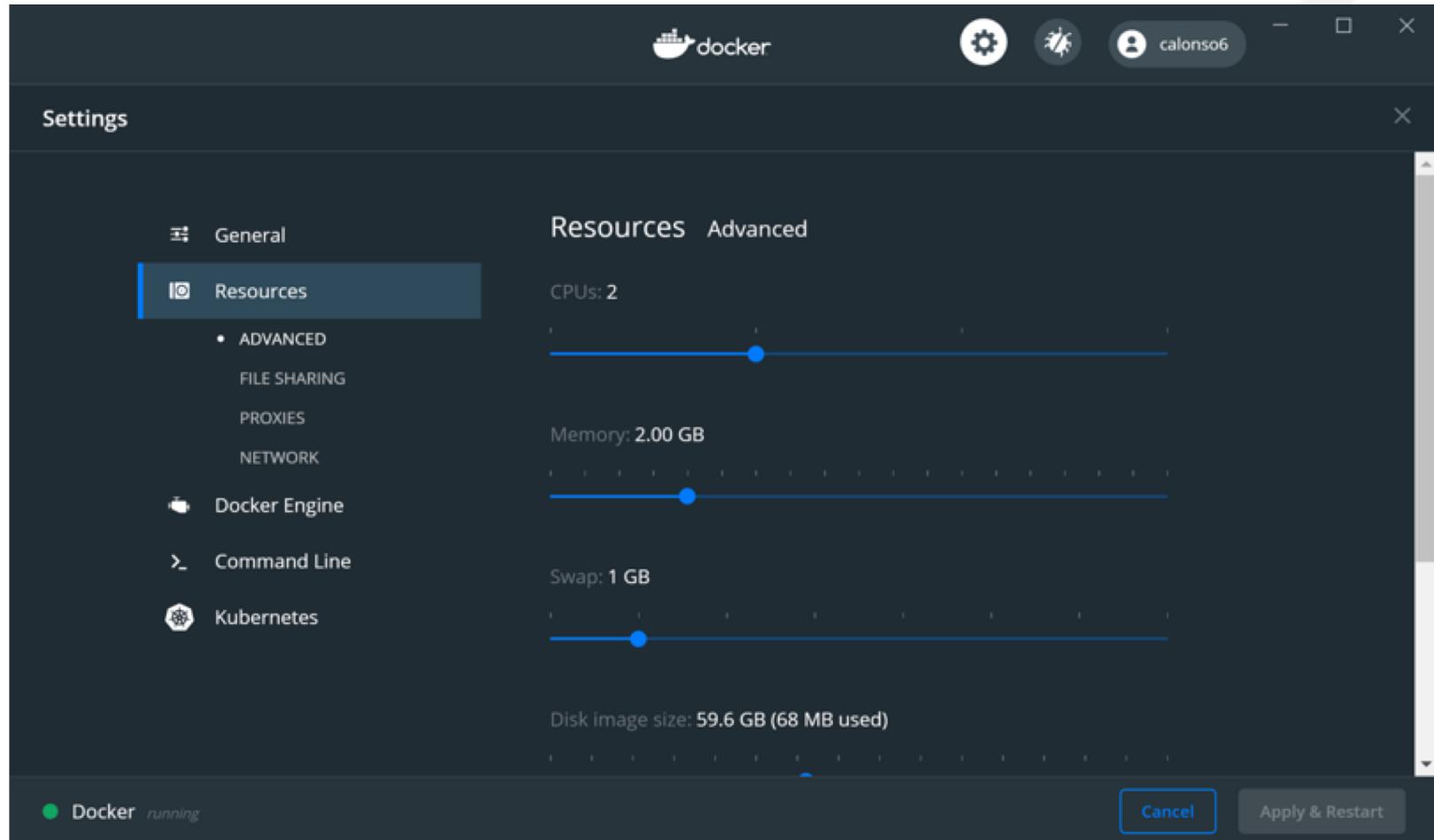
Send usage statistics

Send error reports, system version and language as well as Docker Desktop lifecycle information (e.g., starts, stops, resets).

Discover experimental features. [Switch to the Edge version](#).

● Docker *running* [Cancel](#) [Apply & Restart](#)

# Docker Desktop: configuraciones



# Docker Desktop: comprovació

```
Windows PowerShell
PS C:\Users\carlos> docker info
Client:
  Debug Mode: false

Server:
  Containers: 0
    Running: 0
    Paused: 0
    Stopped: 0
  Images: 0
  Server Version: 19.03.8
  Storage Driver: overlay2
    Backing Filesystem: <unknown>
    Supports d_type: true
    Native Overlay Diff: true
  Logging Driver: json-file
  Cgroup Driver: cgroupfs
  Plugins:
```

```
Last login: Tue Apr 21 12:02:45 on ttys001
[iMac-de-Carlos:~ cam$ docker info
Client:
  Debug Mode: false

Server:
  Containers: 0
    Running: 0
    Paused: 0
    Stopped: 0
  Images: 3
  Server Version: 19.03.8
  Storage Driver: overlay2
    Backing Filesystem: <unknown>
    Supports d_type: true
    Native Overlay Diff: true
  Logging Driver: json-file
  Cgroup Driver: cgroupfs
  Plugins:
```

# Contenidors i imatges

## Imatge

- Analogia amb una classe (POO).
- Serveix per instanciar contenidors.
- Es poden crear múltiples instàncies.
- Immutable.

## Contenidor

- Equivalent a un objecte de POO.
- S'executa en un entorn aïllat.
- Lloc on s'executarà l'aplicació.

# Executant contenidors

- Les imatges de Docker són aplicacions empaquetades.
- Les imatges es descarreguen des del registre (Docker Hub).
- Per executar l'aplicació caldrà crear un contenidor a partir de la imatge.

# Executant contenidors: hello world

```
iMac-de-Carlos:Docker cam$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest: sha256:5f179596a7335398b805f036f7e8561b6f0e32cd30a32f5e19d17a3cda6cc33d
Status: Downloaded newer image for hello-world:latest
```

```
Hello from Docker!
This message shows that your installation appears to be working correctly.
```

To generate this message, Docker took the following steps:

1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.  
(amd64)
3. The Docker daemon created a new container from that image which runs the executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it to your terminal.

To try something more ambitious, you can run an Ubuntu container with:  
\$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:  
<https://hub.docker.com/>

For more examples and ideas, visit:  
<https://docs.docker.com/get-started/>

```
iMac-de-Carlos:Docker cam$
```

- Es busca la imatge en local, com no es troba es descarrega.
- Es crea el contenidor i s'executa.
- Quan l'aplicació acaba (escriu per pantalla), el contenidor finalitza.

# Executant contenidors: servidor web

```
[Mac-de-Carlos:Docker cam$  
[Mac-de-Carlos:Docker cam$ docker run nginx:alpine  
Inable to find image 'nginx:alpine' locally  
alpine: Pulling from library/nginx  
7c96db7181b: Pull complete  
64026bbe255: Pull complete  
71634c55d29: Pull complete  
595887beb81: Pull complete  
Digest: sha256:57a226fb6ab6823027c0704a9346a890ffb0cacde06bc19bbc234c8720673555  
Status: Downloaded newer image for nginx:alpine  
]
```

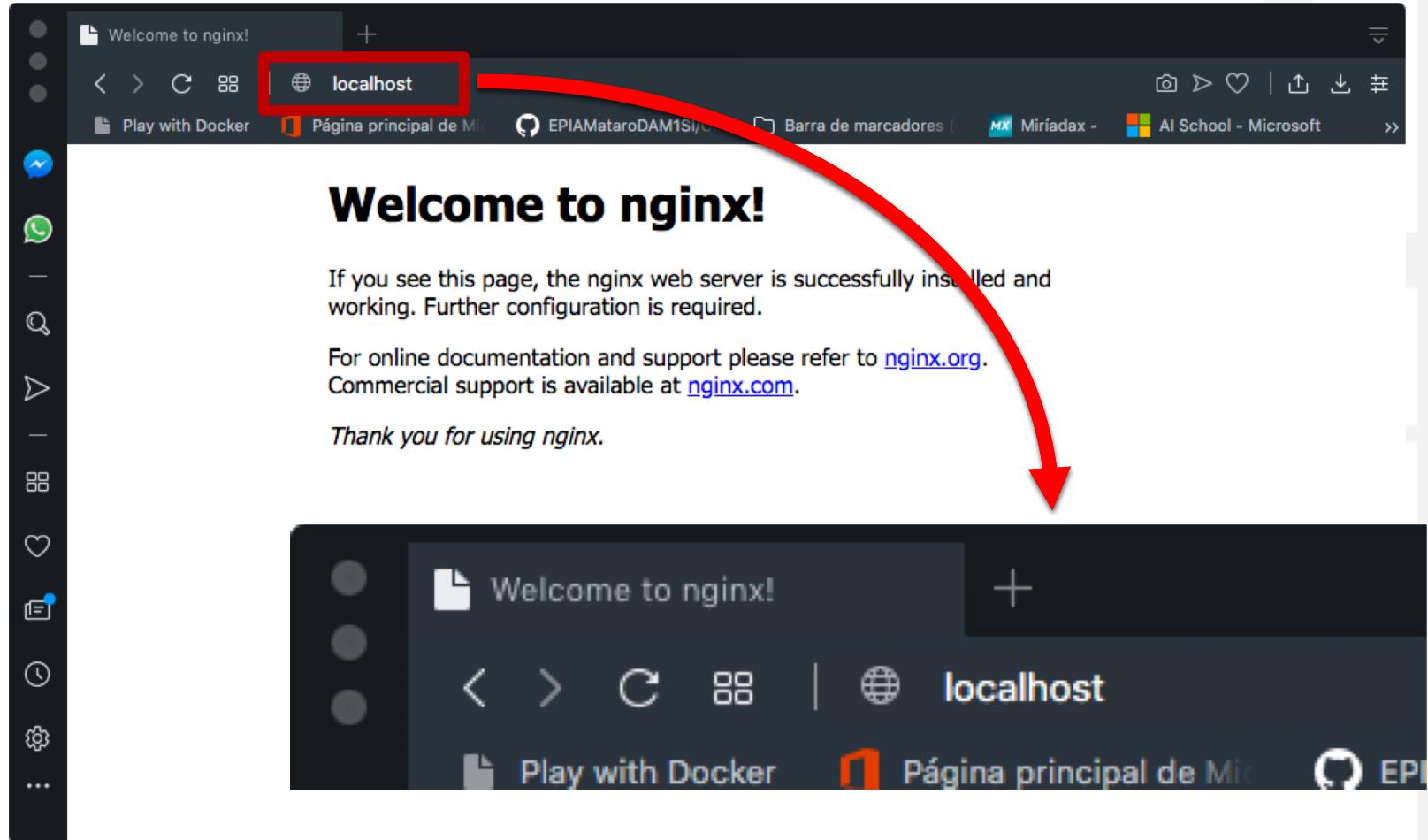
Com és un procés que no finalitza, es queda esperant connexions i no serveix per gaire

Un contenidor amb una tasca que no finalitza s'executa en segon pla (**detached mode**) a més publiquem al port 80 del host el port 80 del contenidor.



```
[iMac-de-Carlos:Docker cam$  
[iMac-de-Carlos:Docker cam$ docker container run --detach --publish 80:80 nginx:alpine  
6f9b6c254912a1127c1b9e7dff25b7dc000956e9b3f88ca84e535b5935f07e13  
iMac-de-Carlos:Docker cam$ ]
```

# Executant contenidors: servidor web



# Processos llarga durada

- Docker és ideal per executar processos típics de servidor (long-term processes).
- El servidor nginx gairebé no consumeix recursos fins que rep peticions, però quan hi rep pot anar augmentant ús CPU i RAM.
- Un servidor relativament modest pot executar un gran número de contenidors.

# Contenidor interactiu: Ubuntu

```
iMac-de-Carlos:Docker cam$  
iMac-de-Carlos:Docker cam$ docker run -i -t ubuntu:18.04 /bin/bash  
Unable to find image 'ubuntu:18.04' locally  
18.04: Pulling from library/ubuntu  
f476d66f5408: Pull complete  
8882c27f669e: Pull complete  
d9af21273955: Pull complete  
f5029279ec12: Pull complete  
Digest: sha256:70fc21e832af32eeec9b0161a805c08f6dddf64d341748379de9a527c01b6ca1  
Status: Downloaded newer image for ubuntu:18.04  
root@059d4212801c:/# hostname  
059d4212801c  
root@059d4212801c:/# █
```

# Comandes bàsiques

- docker run *imatge* (crea un contenidor)
  - -it en mode interactiu i indicant terminal
  - -d per segon pla
  - p port host: port contenidor (publicar port)
- docker stop *ID* (para contenidor)
- docker ps (mostra contenidors actius)
- docker ps -a (mostra tots els contenidors)

# Comandes bàsiques

- docker image ls (mostra imatges)
- docker rm *ID* esborra contenidor
- docker image rm *imatge* (esborra imatge)
- docker logs per tenir els logs d'un contenidor.
- docker exec per executar una comanda a dins d'un contenidor.

# Temps de practicar

- I si no tinc Docker instal·lat?



# Exercicis

- Crear contenidor bàsic
- Contenidor interactiu
- Contenidor mode background
- Mapeig de ports
- Veure contenidors i imatges
- Eliminar contenidors
- Logs
- Llençar comandes dins el contenidor.

# Eines i utilitats

- Per la segona sessió necessitarem dues eines.
- Visual Studio Code un editor vitaminat.
- Azure Data Studio

# Visual Studio Code

The screenshot shows the official Visual Studio Code website. At the top, there's a navigation bar with links for Visual Studio Code, Docs, Updates, Blog, API, Extensions, and FAQ. To the right of the navigation is a search bar labeled "Search Docs" and a blue "Download" button. Below the navigation, a banner announces "Version 1.46 is now available! Read about the new features and fixes from May." On the left side, there's a large, bold headline "Code editing. Redefined." followed by the text "Free. Built on open source. Runs everywhere." Below this, there are download buttons for "Download for Mac Stable Build" and "Other platforms and Insiders Edition". A note below the download buttons states: "By using VS Code, you agree to its license and privacy statement." On the right side of the page, a large screenshot of the Visual Studio Code interface is displayed. It shows the code editor with several files open, including "blog-post.js", "index.js", and "utils.js". The sidebar shows the "EXTENSIONS: MARKETPLACE" section with various extensions listed, such as Python, GitLens, C/C++, ESLint, Debugger for C#, Language Support, vscode-icons, and Vetur. The bottom of the screenshot shows the terminal output with messages like "Compiling...", "Compiled successfully in 26ms", and "Compiled successfully." The status bar at the bottom of the interface shows "Ln 6, Col 21 Spaces: 2 UTF-8 LF JavaScript".

<https://code.visualstudio.com>

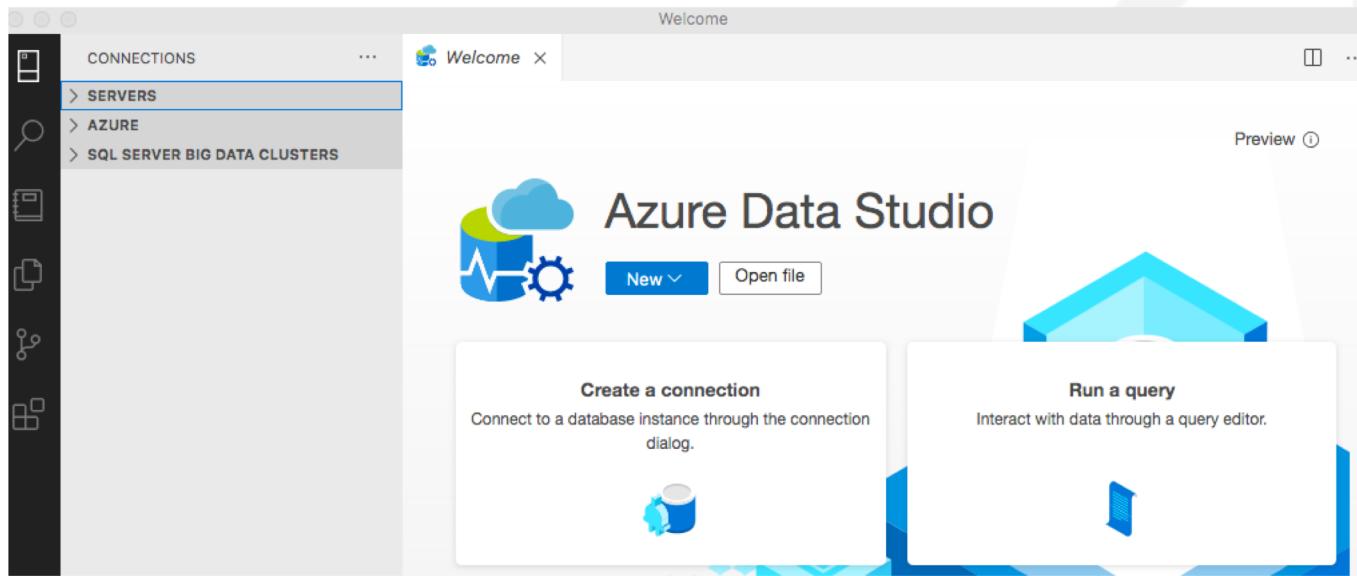
# Visual Studio Code

- Un cop instal·lat explorarem les extensions i instal·larem l'extensió de Docker.
- A més instal·larem el complement Visual Studio Remote Development.

[VS Code Remote Extension Pack](#)

# Azure Data Studio

- És una eina per gestionar de forma remota bases de dades SQL Server.



Azure Data Studio

# Contenidors i dades

I si al contenidor se li fa alguna modificació?

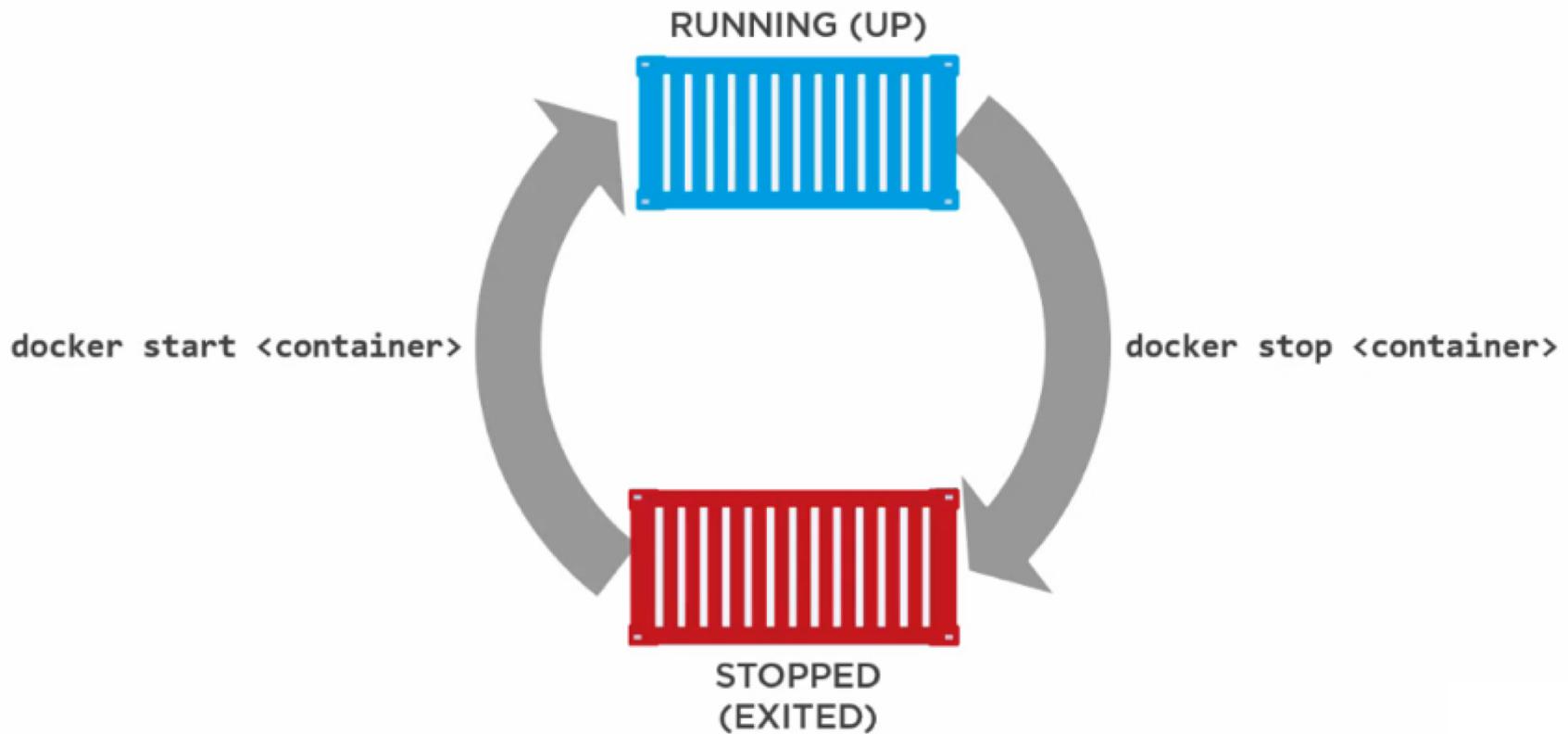
Dades

Programes

Si torno a fer la mateix instrucció **docker run** no es veuen el canvis.

Això vol dir que els contenidors no són persistents?

# Cicle d'un contenidor



# Contenidors i dades

Quan un contenidor finalitza no s'esborra.

```
[iMac-de-Carlos:~ cam$ docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS
dde26360fede        ubuntu              "/bin/bash"        3 hours ago       Exited (127) 3 hours ago
f3617c5a4996        ubuntu              "/bin/bash"        3 hours ago       Exited (0) 3 hours ago
```

Si tornem arrancar un contenidor, els canvis  
sí que hi són

```
[iMac-de-Carlos:~ cam$  
[iMac-de-Carlos:~ cam$ docker start -i f3617c5a4996  
[root@f3617c5a4996:/# ls  
bin boot dev etc hola home lib lib64 media mnt opt proc root run sbin srv sys tmp usr var  
[root@f3617c5a4996:/# ]
```

# Contenidors i dades

- Normalment no usarem aquesta persistència perquè depèn del contingidor i no de la imatge.
- Moltes vegades els contingidors no els aturem si no els eliminem directament.
- Solució a persistència?

## VOLUMS

# Volums

- Carpetes en el sistema d'arxius que tenen persistència.
- Utilitat:
  - Persistència dades
  - Compartir dades entre contenidors
  - Fer accessible carpetes de l'ordinador a dins el contingidor

# Tipus volums

- Volums de dades (data volumes)
  - Anònims
  - Amb nom

Aquests volums només són accessibles pels contenidors

- Volums muntats (mounted volumes)

Compartir una carpeta de l'ordinador per fer-la accessible a un contenidor.

- TMPFS (temporal file system)

Carpetes temporals muntades en RAM.

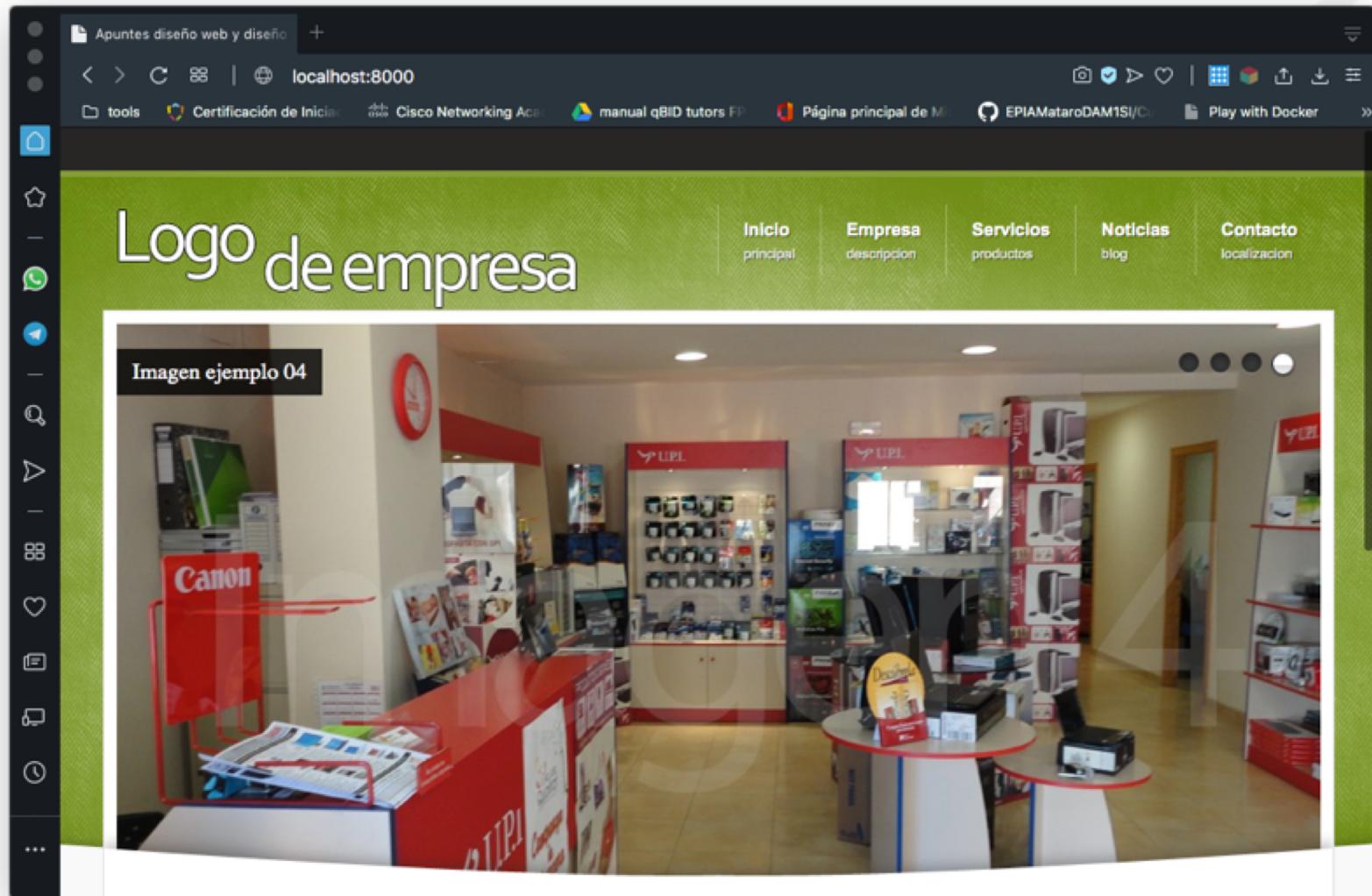
# Exemple volum muntat

- El codi es té a una carpeta de l'ordinador i es fa accessible al contenidor.

```
docker run -d -p 80:80 -v $(pwd): /usr/share/nginx/html  
nginx
```

- Això fa que la carpeta actual on estem a l'ordinador es munti al contenidor a la ruta /usr/share/nginx/html

# Exemple volum muntat



# Exemple volum dades

- Els volums de dades són objectes que es guarden com fitxers al nostre sistema.
- Només són accessibles des dels contenidors.
- Es poden connectar volums entre diferents contenidors.
- Docker no els elimina automàticament.

# Crear volum de dades

- Es pot crear directament:  
`docker volume create nom`
- Veure contenidors:  
`docker volume ls`
- Eliminar contenidors:  
`docker volume rm nom o ID`

Només es poden eliminar volums que no estiguin vinculats a cap contenidor existent.

# Usar volum de dades

- En el moment de crear el contenidor:

```
docker run -v nom:ruta_dins_contenidor imatge
```

- Exemple:
- `docker run --rm -v src:/var/www/html -d -p 8000:80 php:7.3-apache`

Si el volum no existeix, es crea en el moment d'arrancar el contenidor.

# Compartint contenidors

- Dos contenidors poden compartir dades a través de volums.
- Dos mètodes:
  - Usant un contenidor amb nom definit
  - Exposant tots els volums d'un contenidor a l'altre usant **--volumes-from**

# Exemple cas 1

Des del segon contenidor podem accedir al volum i modificar el codi que mostra el contingut web.

```
[iMac-de-Carlos:Docker cam$  
[iMac-de-Carlos:Docker cam$ docker run --rm -d -p 8000:80 -v src:/var/www/html php:7.3-apache  
746694ad5c2a237173565f827d38f467cd7c4580b02c35a872b0ffe13e3f9a8f  
[iMac-de-Carlos:Docker cam$ docker run --rm -it -v src:/var/src ubuntu  
[root@0ae2b4921a41:/# ls /var/src  
index.html  
root@0ae2b4921a41:/# ]
```

## Exemple cas 2

- El segon contenidor mapeja el volum automàticament les mateixes rutes que el primer.

```
[iMac-de-Carlos:Docker cam$  
[iMac-de-Carlos:Docker cam$ docker run --name webserver --rm -d -p 8000:80 -v src:/var/www/html php:7.3-apache  
9dfd425e7814f090975b018eb6f19124ac863bbd70157b188bfd34e7016edf9a  
[iMac-de-Carlos:Docker cam$ docker run --rm -it --volumes-from webserver ubuntu  
[root@7da51238f999:/# ls /var/www/html  
index.html  
[root@7da51238f999:/# ]
```

# Entorns de desenvolupament

- Docker és una eina ideal per desplegar entorns de desenvolupament al nostre equip, minimitzant interferències.
- Veurem dos exemples:
  - Usar SQL Server
  - Programar amb Java amb contenidors

# Exemple 1: SQL Server

- El popular gestor de BD SQL Server de Microsoft actualment té versions Windows i Linux.
- Usarem un contenidor i un volum per tenir persistència.
- Gestió SQL Server amb l'eina multiplataforma Azure Data Studio.

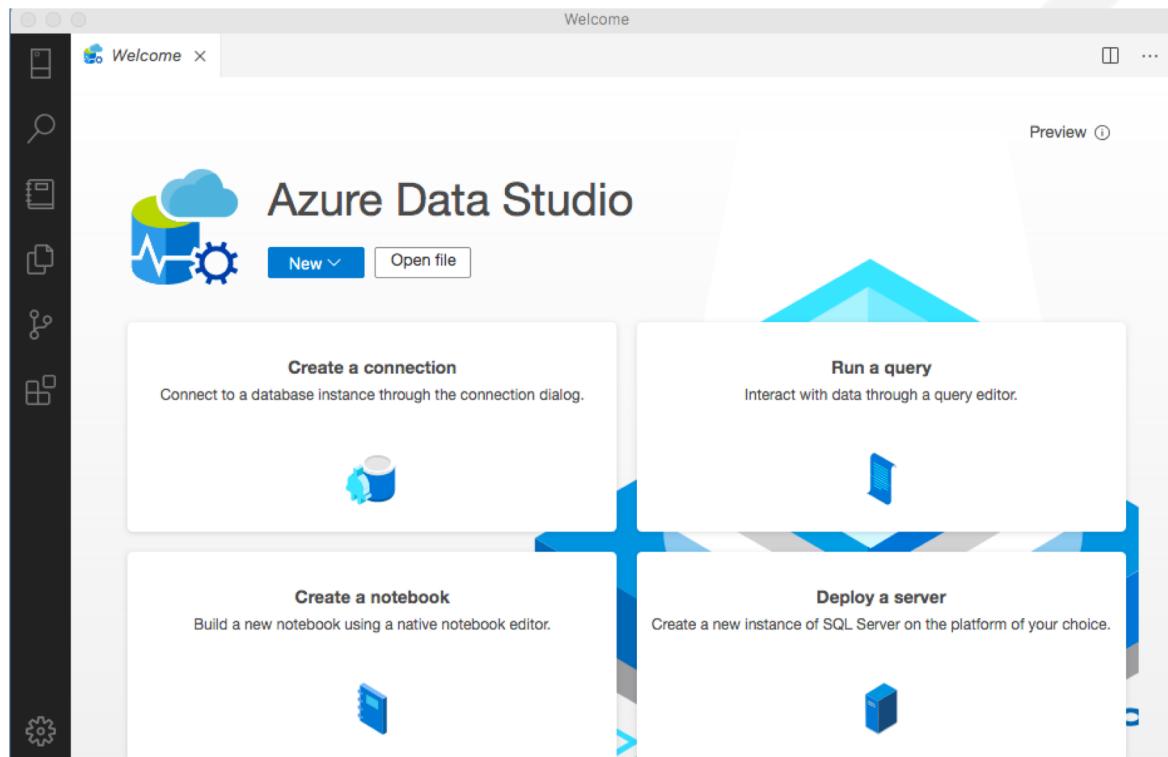
# Exemple 1: volum i contenidor

```
iMac-de-Carlos:sqlserver cam$  
iMac-de-Carlos:sqlserver cam$ docker run --rm -d --name sql1 -e 'ACCEPT_EULA=Y' -e 'SA_PASSWORD=P@ssw0rd2020' -p 1433:1433  
-v database:/var/opt/mssql mcr.microsoft.com/mssql/server  
1350f5bd3da56c3bf375c4149f86e4a84d51262bd7ceae839c26dc6e935b8f67  
iMac-de-Carlos:sqlserver cam$ docker ps  
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS  
NAMES  
1350f5bd3da5      mcr.microsoft.com/mssql/server   "/opt/mssql/bin/perm..."   4 seconds ago    Up 3 seconds     0.0.0  
.0:1433->1433/tcp  sql1  
iMac-de-Carlos:sqlserver cam$
```

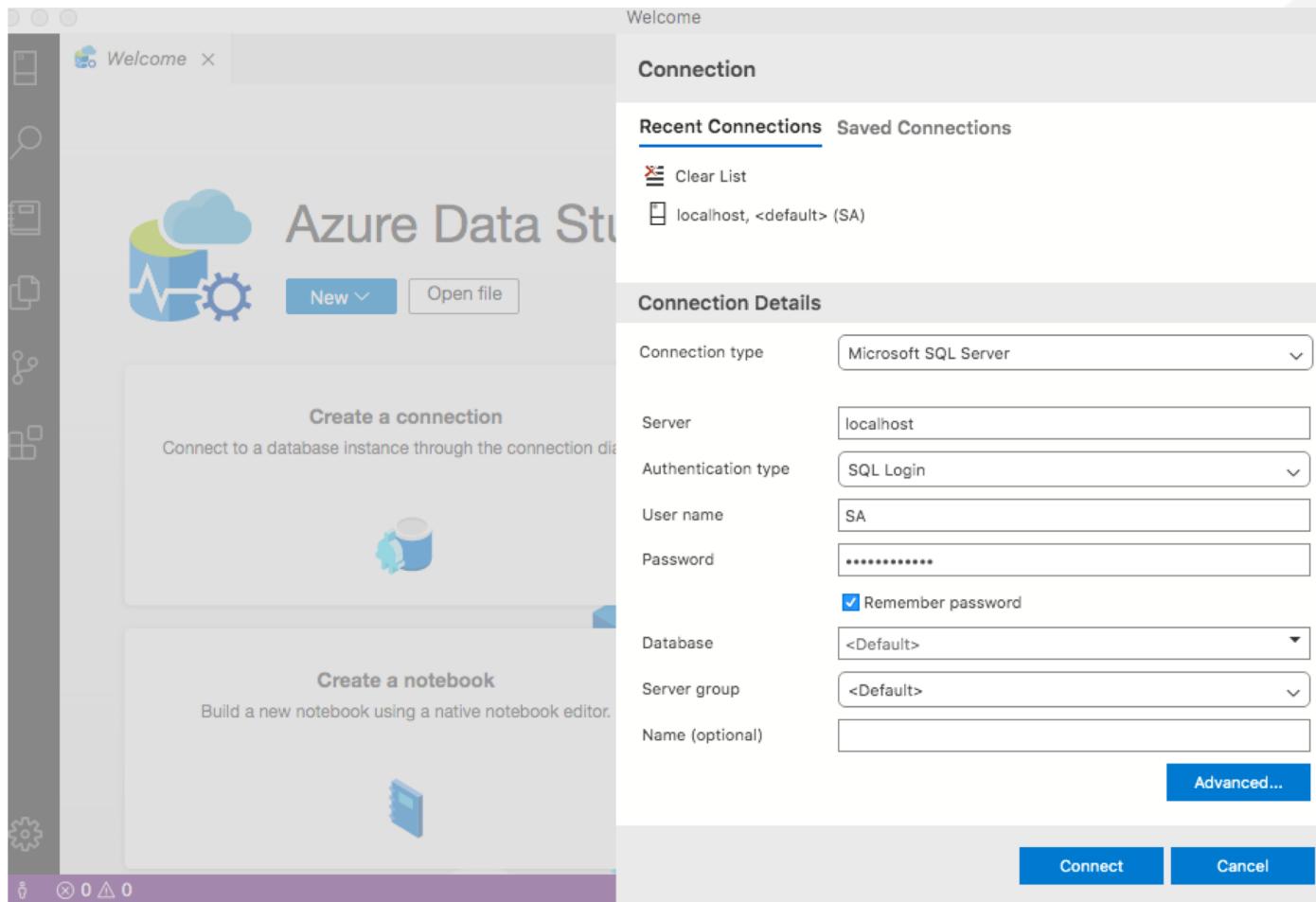
El primer cop que executem el contenidor es crearà el volum **database** que hem mapejat a la carpeta on SQL Server Linux crea les bases de dades.

# Exemple 1: administració

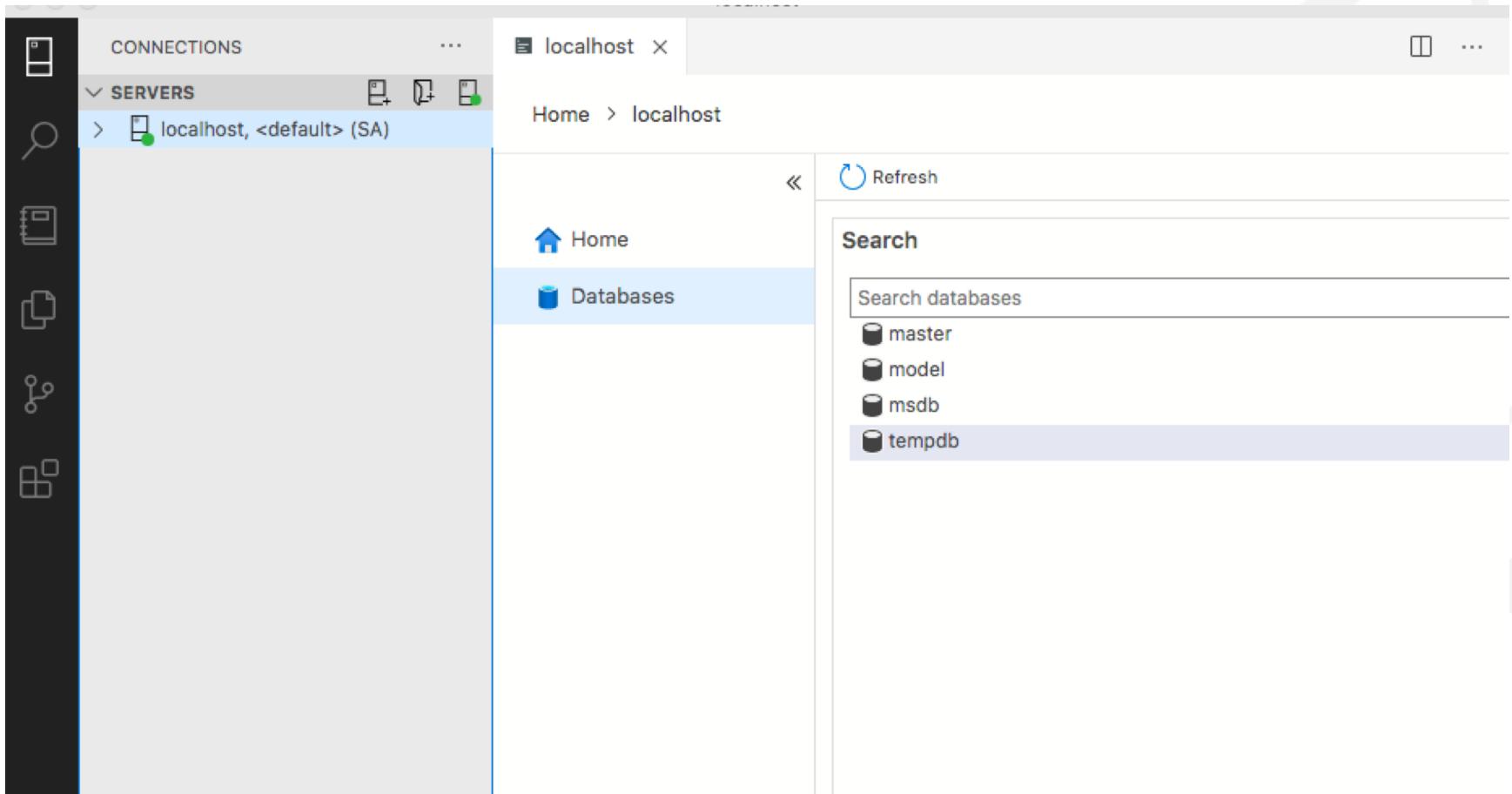
- Usarem l'eina Azure Data Studio ([link](#))



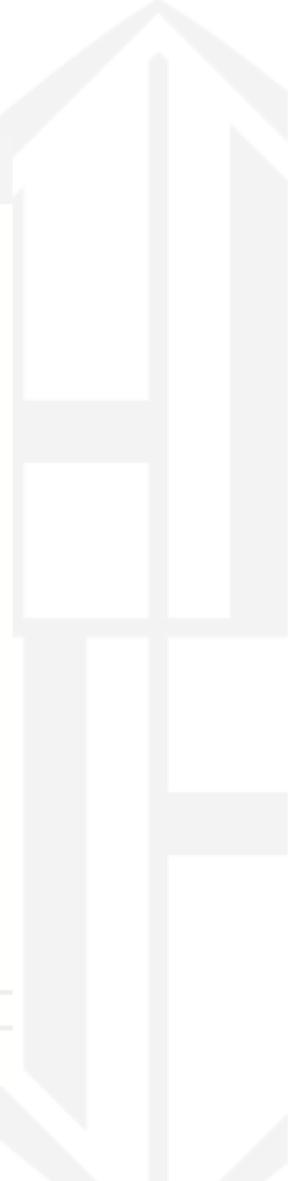
# Exemple 1: connexió



# Exemple 1: administració

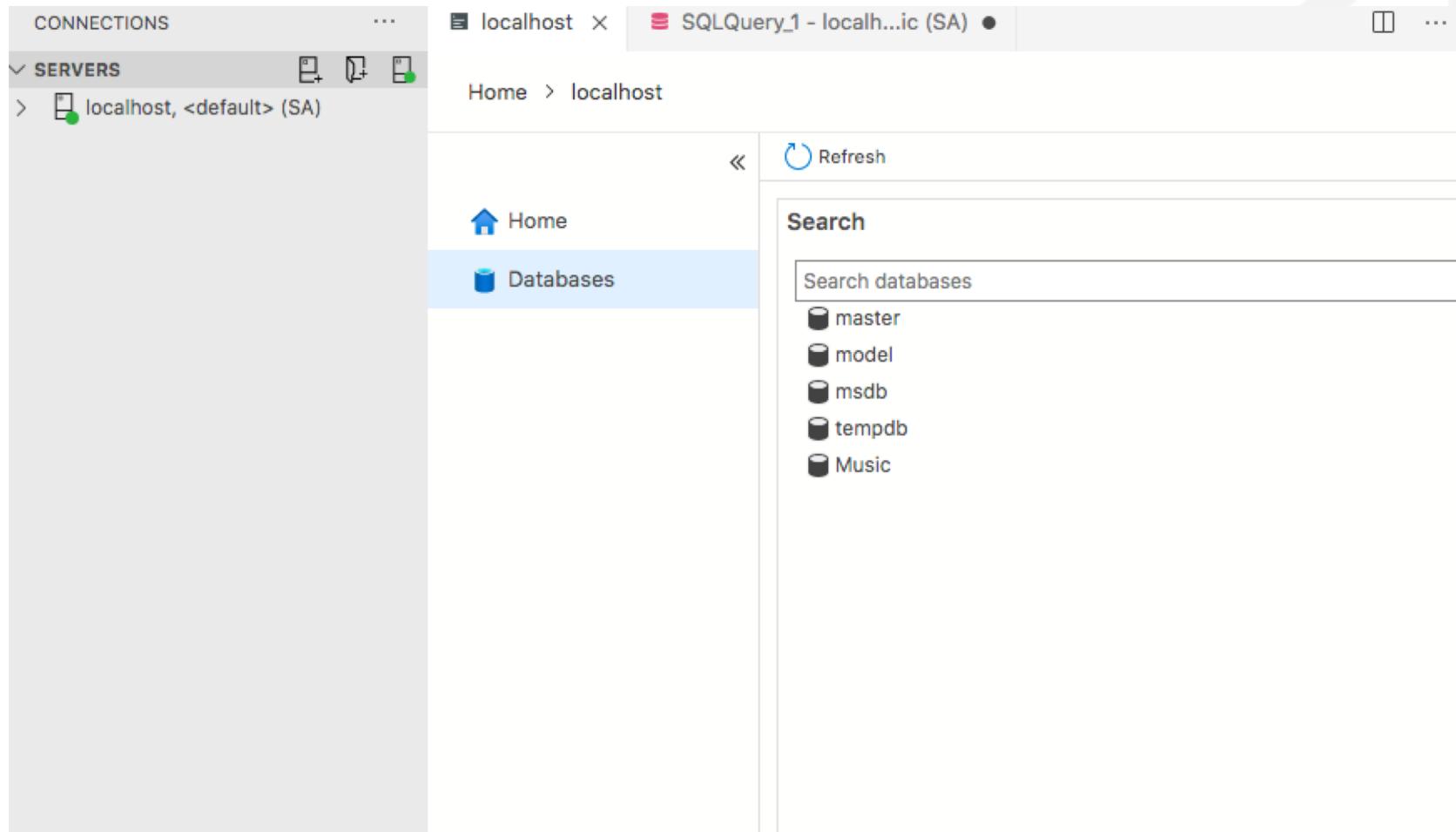


# Exemple 1: administració



```
localhost SQLQuery_1 - localh...er (SA) master Explain
Run Cancel Disconnect Change Connection
Enable SQLCMD
1  /* Create database */
2  CREATE DATABASE Music;
3  GO
4
5  /* Change to the Music database */
6  USE Music;
7  GO
8
9  /* Create tables */
10 CREATE TABLE Artists (
11     ArtistId int IDENTITY(1,1) NOT NULL PRIMARY KEY,
12     ArtistName nvarchar(255) NOT NULL,
13     ActiveFrom DATE NULL
14 );
15
16
17
18 GO
```

# Exemple 1: administració

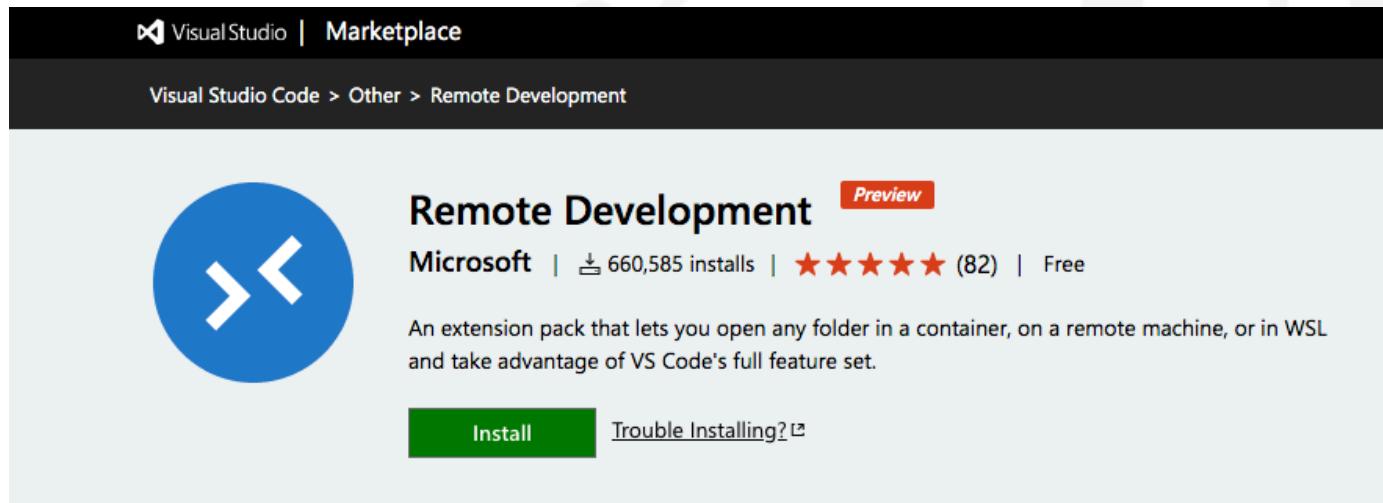


# Exemple 1: persistència

- Si ara aturem el contenidor (s'elimina automàticament) i el tornem a crear veurem com veiem la base de dades **Music** creada anteriorment.

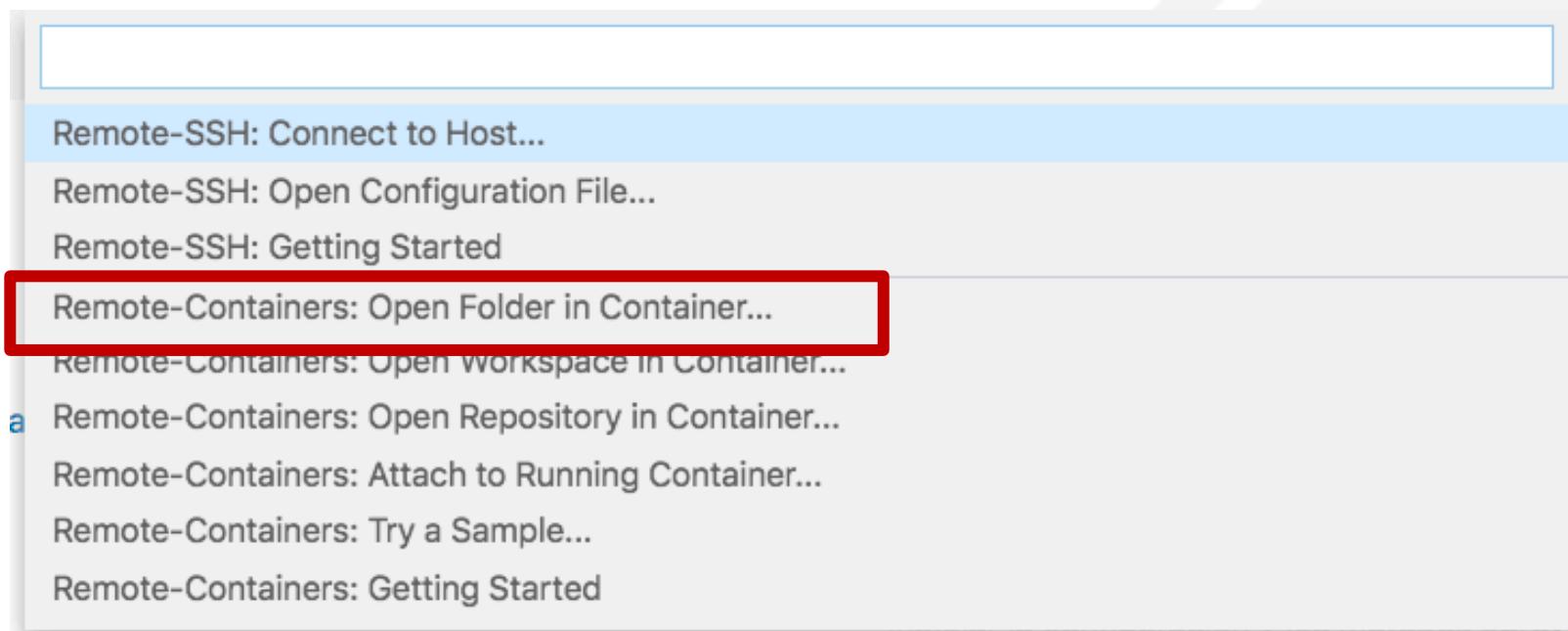
# Exemple 2: Programar en Java

- Volem desenvolupar codi en Java sense haver d'instal·lar openJDK a l'equip.
- Com editor usarem Visual Code amb el plugin Remote Development



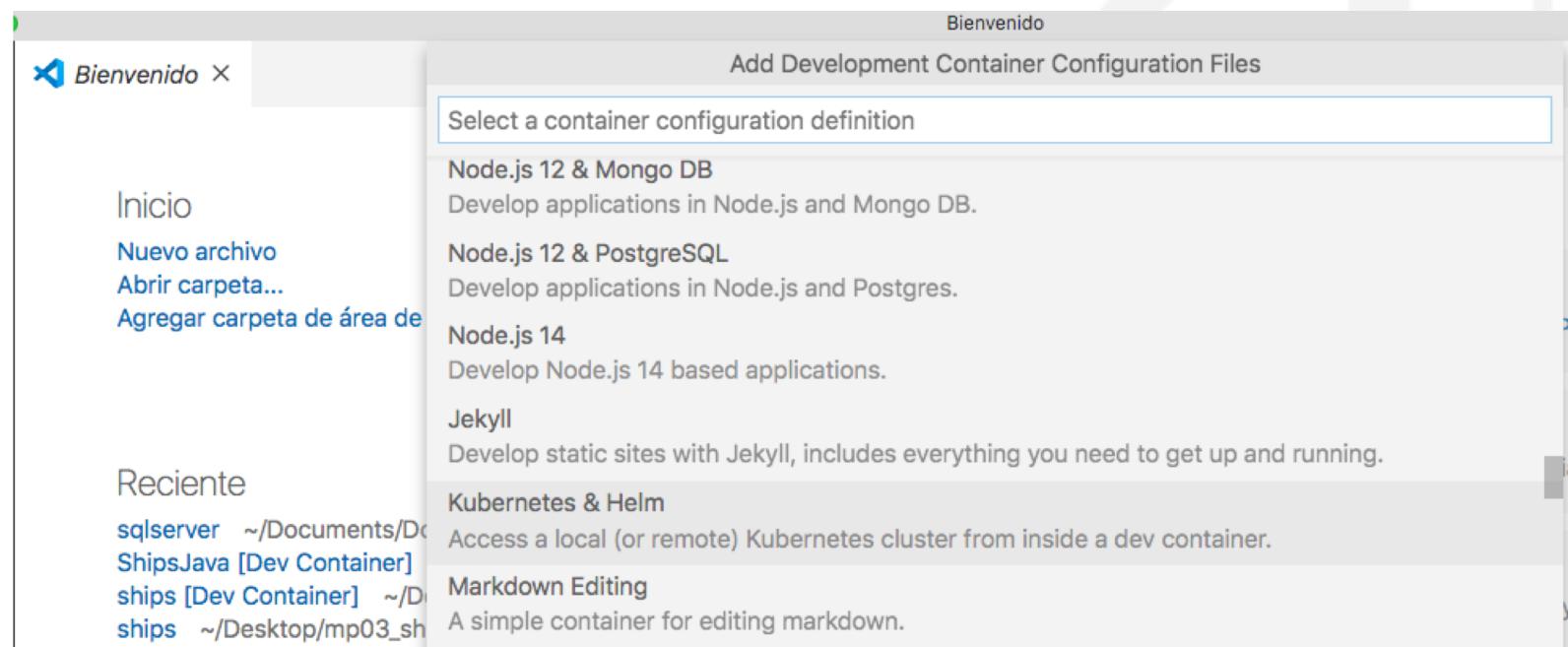
## Exemple 2: Connexió

- Utilitzarem l'opció d'utilitzar una carpeta del nostre equip on estarà el codi.



# Exemple 2: selecció contenidor

- Ens demana quin contenidor es vol utilitzar per aquesta carpeta.



# Exemple 2: Execució

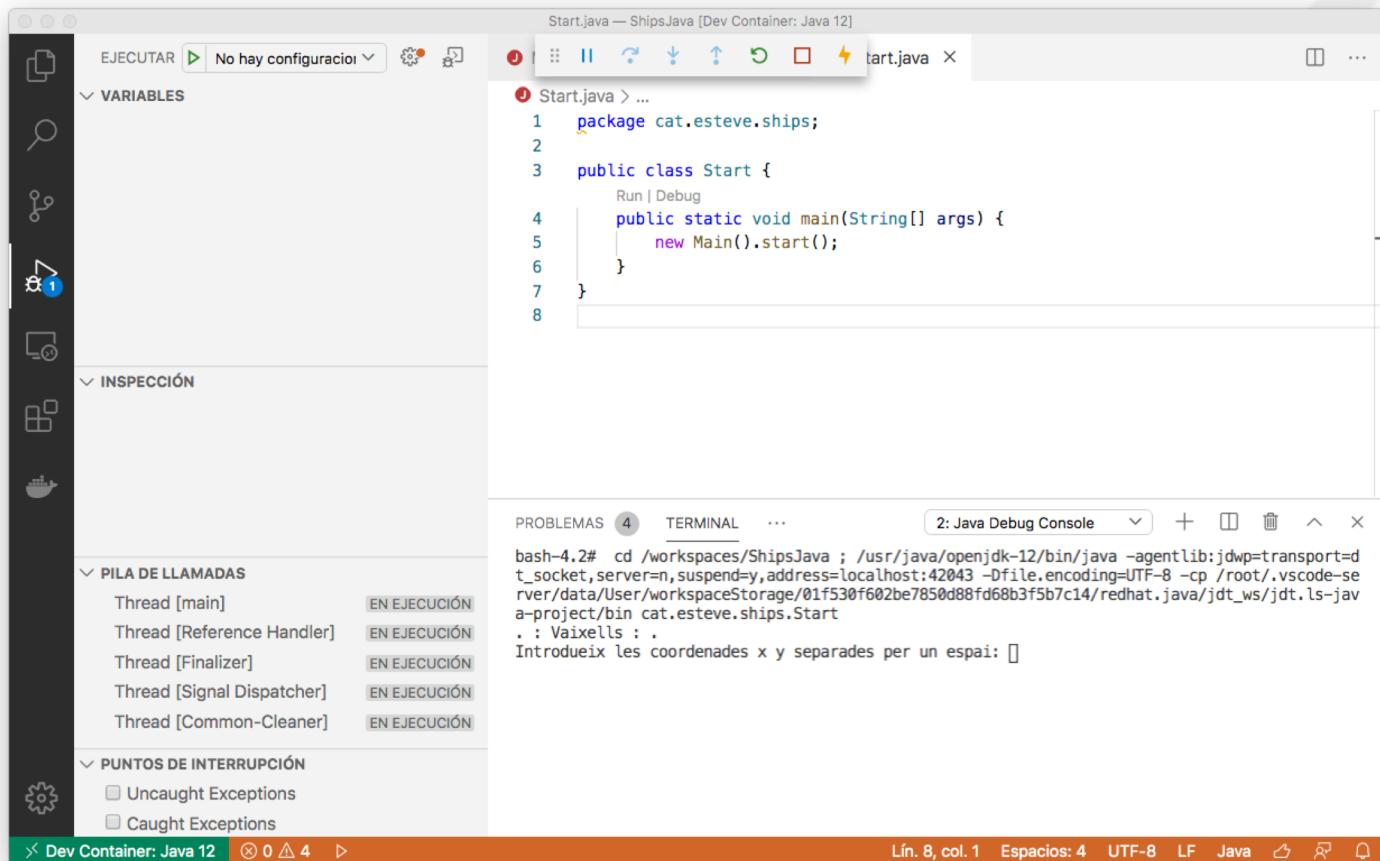
The screenshot shows a Java development environment interface. On the left is a sidebar with various icons: a clipboard, a magnifying glass, a gear, a play button, a refresh, and a terminal. The main area has a title bar "Start.java — ShipsJava [Dev Container: Java 12]". The left pane, titled "EXPLORADOR", shows a tree structure with "EDITORES ABIERTOS" containing "Main.java" and "Start.java", and a "SHIPSJAVA [DEV CONTAINER: JAVA 12]" section containing ".devcontainer", "Main.java", "Ship.java", and "Start.java". The right pane contains two tabs: "Main.java" and "Start.java". The "Start.java" tab is active and displays the following code:

```
1 package cat.esteve.ships;
2
3 public class Start {
4     public static void main(String[] args) {
5         new Main().start();
6     }
7 }
8
```

At the bottom, there is a "TERMINAL" section with a "bash-4.2#" prompt.

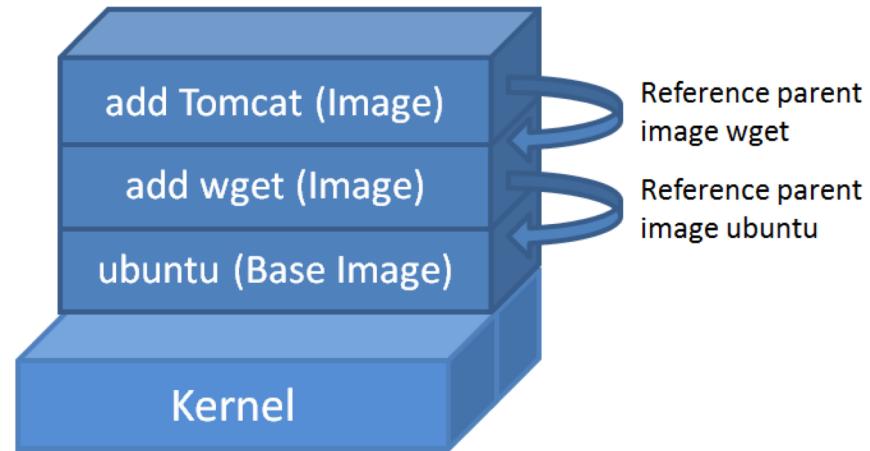
El primer cop es crearà una carpeta oculta **.devcontainer** que defineix l'entorn d'execució.

# Exemple 2: Execució



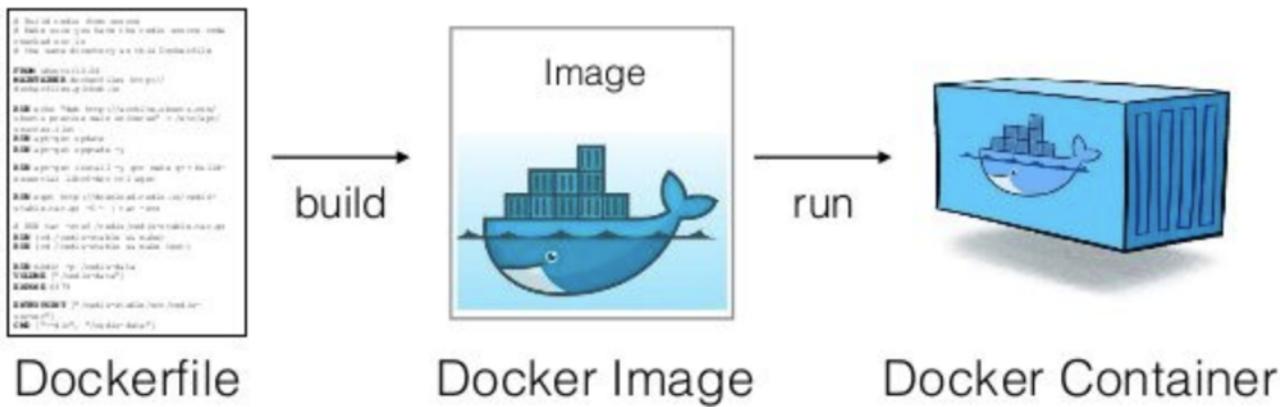
# Creant imatges

- Contenidor és l'execució de la imatge, per tant, per desplegar les nostres aplicacions caldrà crear imatges.
- Les imatges es defineixen per capes (Layered File System)



# Creant imatges

- Per empaquetar una aplicació caldrà crear una imatge específica.



# Instruccions Dockerfile

- FROM: imatge base
- LABEL: per indicar autor, etc.
- RUN: executar comandes durant el build.
- COPY: copiar arxius i carpetes a dins de la imatge. Es copien des de la carpeta del projecte.
- WORKDIR: directori treball dins la imatge.

# Instruccions Dockerfile

- CMD: comandes o paràmetres per defecte a l'arrancar el contenidor (es poden sobreescriure).
- ENTRYPOINT: configura el contenidor com un executable.
- EXPOSE: per redirigir ports.
- ENV: configuració variables.
- VOLUME: munta un volum per exposar una carpeta del filesystem de la imatge.

# .dockerignore

- Arxiu amb un objectiu similar al .gitignore
- Podem definir arxius i carpetes que no volem s'enviïn al fer el build (recordeu que al fer el build s'envia tot el context).
- Exemple:

.git

\*.tmp

# Exemple Dockerfile

```
#FROM is the base image for which we will run our
application
FROM nginx:latest
LABEL mantainer = "Carlos Alonso"
LABEL description = "2048 game"
# Copy files and directories from the application
COPY index.html /usr/share/nginx/html
COPY favicon.ico /usr/share/nginx/html
COPY Rakefile /usr/share/nginx/html
COPY style/ /usr/share/nginx/html/style/
COPY meta/ /usr/share/nginx/html/meta/
COPY js/ /usr/share/nginx/html/js/

# Tell Docker we are going to use this port
EXPOSE 80
```

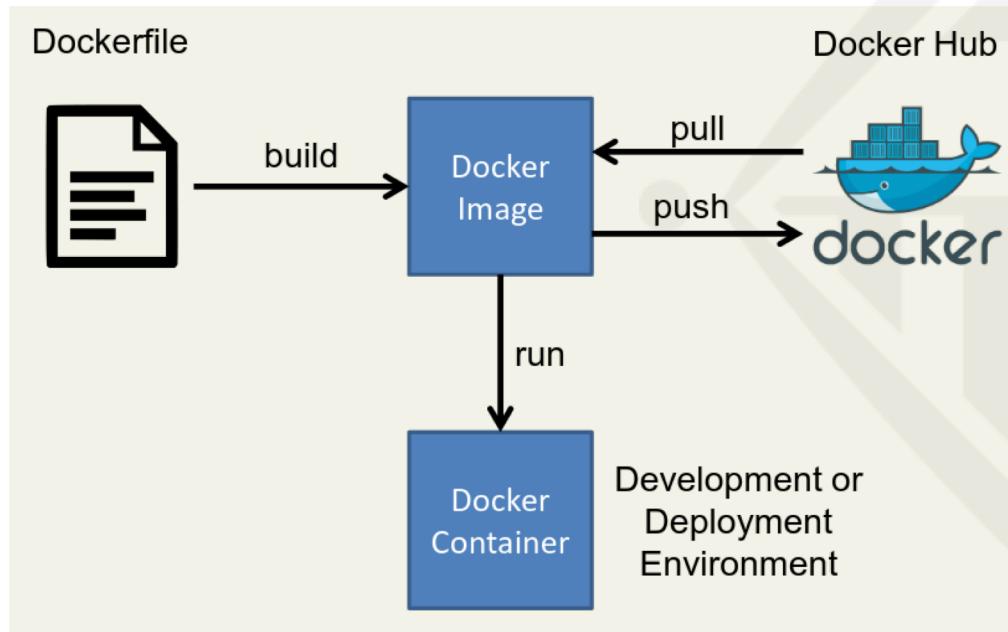
# Creació imatge

- Per construir la imatge situats a la carpeta arrel del projecte:

```
docker build -t "calonso6/2048" .
```

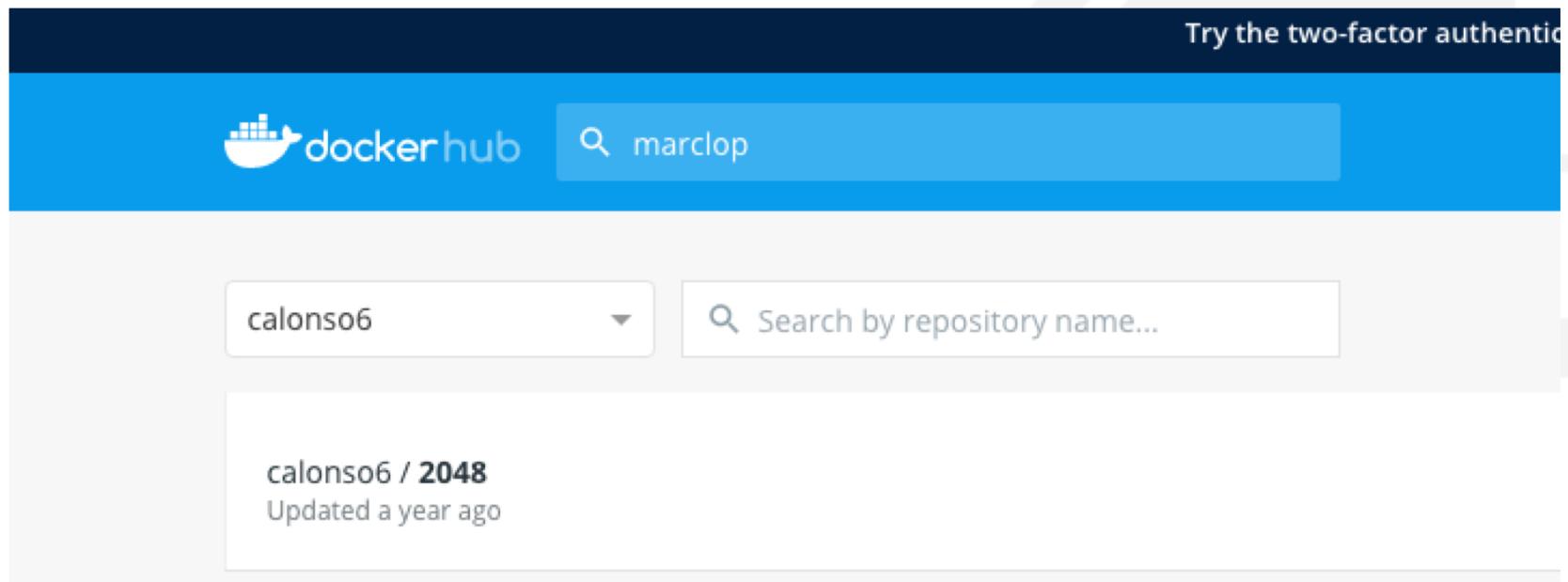
# Publicar imatges

- Aquestes noves imatges es poden publicar en Docker Hub, Azure, etc.



# Publicar imatges a Docker Hub

```
docker push calonso6/2048:1.0
```



# Neteja i manteniment

- Amb l'ús continuat acabarem tenint al nostre sistema elements de Docker que no utilitzem.
- Neteja: docker system prune

```
[iMac-de-Carlos:~ cam$ docker system prune
WARNING! This will remove:
 - all stopped containers
 - all networks not used by at least one container
 - all dangling images
 - all dangling build cache

Are you sure you want to continue? [y/N] ]
```

# Xarxes a Docker

- Fins ara, cada cop que instanciem un contenidor Docker automàticament li configura una xarxa per tal que tingui connectivitat.
- Aquesta configuració a part de donar-li una IP interna, li permet sortir a Internet i admet les configuracions de redirecció de ports (DNAT).

# Tipus de xarxes

- Per defecte les xarxes disponibles a Docker són:

```
Last login: Thu Aug 11 18:09:51 on ttys000
[iMac-de-Carlos:~ cam$ docker network ls
NETWORK ID      NAME      DRIVER      SCOPE
711f4ea34fed   bridge    bridge    local
e3ea791bf119   host      host      local
c6adced3bf8c   none      null      local
iMac-de-Carlos:~ cam$ ]
```

- A més d'aquest tres tipus, existeixen les xarxes MacVLAN i Overlay.

# Xarxa bridge

- És la xarxa per defecte i equival a una xarxa NAT.
- Contenidors aïllats dins el hosts però es poden comunicar entre sí.

```
[iMac-de-Carlos:~ cam$ docker network inspect bridge
[
  {
    "Name": "bridge",
    "Id": "711f4ea34fed78af8347b0fc96a718bc3886896b1ec58d4d2383133a2ce0bac8",
    "Created": "2022-08-11T10:42:04.9233115Z",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": null,
      "Config": [
        {
          "Subnet": "172.17.0.0/16",
          "Gateway": "172.17.0.1"
        }
      ]
    },
    "Internal": false,
```



# Xarxa host

- Aquest mode elimina l'aïllament entre la màquina física.
- Per tant, el contenidor tindrà la mateixa IP que el host.
- Només funciona amb Linux.

# Xarxa none

- Básicament estem indicant que no volem que el contenidor tingui una xarxa assignada.
- Equival al localhost, de forma que el contenidor només es pot comunicar amb sí mateix.

# Altres tipus de xarxa

- A més dels tipus per defecte, existeixen dos tipus més:
  - Macvlan: entorn híbrid, el contenidor tindrà una IP de la xarxa local on es trobi el hosts.
  - Overlay: s'utilitza en sistemes clusteritzats, és a dir, on tinc més d'un host i necessito comunicar contenidors entre màquines però aïllats de la xarxa local.

# Creació xarxes

- Podem xarxes noves tipus bridge, que s'assignen a una subxarxa diferent i per tant, permet tenir contenidors aïllats.

```
[iMac-de-Carlos:~ cam$  
[iMac-de-Carlos:~ cam$ docker network create demo-net  
94b63d3909f9208a08ceb96c91c8933d45752987e0e07ef7e0d4126413d71e3f  
[iMac-de-Carlos:~ cam$ docker network ls  
NETWORK ID      NAME      DRIVER      SCOPE  
711f4ea34fed    bridge    bridge    local  
94b63d3909f9    demo-net  bridge    local  
e3ea791bf119    host     host     local  
c6adced3bf8c    none     null     local  
[iMac-de-Carlos:~ cam$ ]
```

# Assignació de xarxes

- Si volem assignar un contenidor a una xarxa específica, es tant senzill com indicar-ho durant la creació.

```
[iMac-de-Carlos:~ cam$  
[iMac-de-Carlos:~ cam$ docker run -dit --name prova1 --network demo-net ubuntu bash  
85859d54a830ff8ee1952d03a8d525af65e8a449c3d32fc482095b7df39c84df  
[iMac-de-Carlos:~ cam$ docker run -dit --name prova2 ubuntu bash  
7e140bb6eb272809107514a0a4629c170189954ee2086f996a1f6f87df76df97
```

# Assignació de xarxes

- Amb docker inspect podem veure la IP de cada contenidor i comprovar com es troben a subxarxes diferents.

```
[iMac-de-Carlos:~ cam$  
[iMac-de-Carlos:~ cam$ docker inspect prova1 | grep "IPAddress"  
    "SecondaryIPAddresses": null,  
    "IPAddress": "",  
        "IPAddress": "172.18.0.2",  
[iMac-de-Carlos:~ cam$ docker inspect prova2 | grep "IPAddress"  
    "SecondaryIPAddresses": null,  
    "IPAddress": "172.17.0.2",  
        "IPAddress": "172.17.0.2",  
iMac-de-Carlos:~ cam$ █
```

A PowerShell l'alternativa a grep, és Select-String

# Docker compose

- És una eina pensada per simplificar el desplegament d'entorns basats en Docker.
- Permet gestionar solucions amb múltiples contenidors.
- També és una solució molt útil per situacions on només tenim un sol contenidor, però amb un variables d'entorn, volums, etc.

# Docker compose

- A docker compose les definicions es realitzen mitjançant un arxiu en format [YAML](#).
- Aquest arxiu **docker-compose.yml** conté els contenidors a crear a partir d'una imatge o d'un Dockerfile, els volums, els ports, variables d'entorn, etc.

# Docker compose

- L'arxiu té el següent format:

```
version: '3'
services:
  web:
    build: .
    ports:
      - "8088:8080"
    volumes:
      - .:/code
      - logvolume01:/var/log
    links:
      - tomcat
  tomcat:
    image: tomcat
volumes:
  logvolume01: {}
```

Services serveix per indicar els diferents sistemes que s'aixecaran.

El primer (web) es crea a partir d'un Dockerfile situat a la ruta, mentre que el tomcat utilitza una imatge existent.

Amb ports indiquem els ports que s'exposaran.

Volumes serveix per indicar els volums que s'utilitzaran, en aquest cas, un mapeig de la carpeta actual a /code del primer servei i es defineix un volum i s'assigna a una ruta dins el contenidor.

# Docker compose

Users > cam > Documents > Docker > wordpress > 📜 docker-compose.yml

```
1  version: '3.3'
2
3  services:
4      db:
5          image: mysql:5.7
6          volumes:
7              - db_data:/var/lib/mysql
8          restart: always
9          environment:
10             MYSQL_ROOT_PASSWORD: somewordpress
11             MYSQL_DATABASE: wordpress
12             MYSQL_USER: wordpress
13             MYSQL_PASSWORD: wordpress
14
15     wordpress:
16         depends_on:
17             - db
18         image: wordpress:latest
19         ports:
20             - "8000:80"
21         restart: always
22         environment:
23             WORDPRESS_DB_HOST: db:3306
24             WORDPRESS_DB_USER: wordpress
25             WORDPRESS_DB_PASSWORD: wordpress
26             WORDPRESS_DB_NAME: wordpress
27     volumes:
28         db_data: {}
```

# Orquestració

- El següent pas és aplicar els contenidors amb la filosofia dels microserveis, això implica:
  - Escalar els diferents serveis de forma independent.
  - Gestionar el conjunt de contenidors de manera que funcioni de la forma més autònoma possible.

# Orquestració

- **Docker Swarm** era l'eina proposada per Docker per crear i gestionar clústers, però actualment està en desús front Kubernetes.
- **Kubernetes** és una eina creada per Google i que serveix per gestionar clústers de contenidors i que actualment és extremadament popular.

# Desplegament

- Podem desplegar les nostres imatges directament a serveis com:
  - Heroku
  - AWS
  - Google Cloud
  - Azure
- Aquests desplegamets tant es poden fer per entorns senzills com per clústers de Kubernetes.

# Desplegament: exemple

Todos los servicios > App Services >

## Crear aplicación web

carpetas para organizar y administrar todos los recursos.

Suscripción \*

Azure para estudiantes

Grupo de recursos \*

(Nuevo) 2048

Crear nuevo

### Detalles de instancia

¿Necesita una base de datos? Pruebe la nueva experiencia de web y base de datos.

Nombre \*

2048bcnactiva

.azurewebsites.net

Publicar \*

Código  Contenedor Docker  Aplicación web estática

Sistema operativo \*

Linux  Windows

Región \*

North Europe

💡 ¿No encuentra su plan de App Service? Pruebe otra región o seleccione su App Service Environment.

### Plan de App Service

El plan de tarifa de App Service determina la ubicación, las características, los costos y los recursos del proceso asociados a la aplicación. [Más información](#)

Plan de Linux (North Europe) \*

(Nuevo) ASP-2048-bcb9

Crear nuevo

SKU y tamaño \*

Básico B1

Total de ACU: 100, 1.75 GB de memoria

Cambiar el tamaño

Microsoft Azure

Todos los servicios > App Services >

## Crear aplicación web

Datos básicos

Docker

Redes

Supervisión

Etiquetas

Revisar y crear

Extrae imágenes de contenedor de Azure Container Registry, Docker Hub o un repositorio de Docker privado. App Service implementará la aplicación en contenedores con sus dependencias preferidas en producción en cuestión de segundos.

Opciones

Contenedor único

Origen de imagen

Docker Hub

### Opciones de Docker Hub

Tipo de acceso \*

Público

Imagen y etiqueta \*

calonso6/2048:latest

Comando de inicio

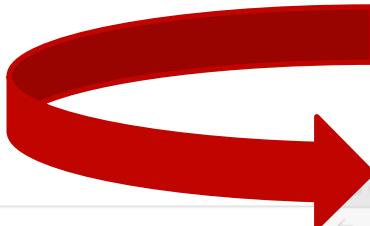
# Desplegament: exemple

Examinar Detener Intercambiar Reiniciar Eliminar Actualizar Obtener perfil de publicación Restablecer perfil de publicación Compartir con dispositivo móvil Envíenos sus co

^ Información esencial

Grupo de recursos (mover) : 2048  
Estado : Running  
Ubicación : North Europe  
Suscripción (mover) : Azure para estudiantes  
Id. de suscripción : 915cb12e-d13e-4dd0-b279-e232184d3353  
Etiquetas (editar) : Haga clic aquí para agregar etiquetas.

URL : <https://2048bcnactiva.azurewebsites.net>  
Plan del servicio de aplic... : ASP-2048-bcb9 (81:1)  
FTP/Nombre de usuario ... : 2048bcnactiva\carlos\_BCN\_Activa  
Nombre de host de FTP : ftp://waws-prod-db3-173.ftp.azurewebsites.windows.net/site/wwwroot  
Nombre de host de FTPS : https://waws-prod-db3-173.ftp.azurewebsites.windows.net/site/wwwroot



A Microsoft Azure Student Hub Overview

Join the numbers and get to the 2048 tile! New Game

SCORE 0 BEST 0

2048

4

2

Barcelona Activa

# Per seguir...

- Docker Documentation:  
<https://docs.docker.com>
- Play with Docker (laboratoris)  
<https://www.docker.com/play-with-docker>





[barcelona.cat/barcelonactiva](http://barcelona.cat/barcelonactiva)