

Docker per a desenvolupadors

U2. Connectors

Accés a Dades



Continguts

1. Què és docker?	3
Contenidor i imatges	4
1.1. Preparació de l'entorn	4
1.2. Instal·lació de Docker-CE	6
Docker i serveis	7
1.3. Docker i Busybox	7
1.4. L'ordre «docker ps»	9
1.5. Execució interactiva	10
1.6. Netejant espai	10
2. Servidor MariaDB amb Docker	11
2.1. Descàrrega de la imatge	12
2.2. Creació d'un contenidor	12
Engegant un contenidor ja creat	13
2.3. Connexió des d'un client	14

1. Què és docker?



Figura 1: Logo de Docker

Segons la Wikipèdia:

Docker és un projecte de codi obert que automatitza el desplegament d'aplicacions dins de contenidors de programari, i proporciona una capa addicional d'abstracció i automatització de virtualització d'aplicacions en múltiples sistemes operatius. Docker fa ús de les característiques d'aïllament de recursos del nucli de Linux, tals com els cgroups i els namespaces que permeten que contenidors independents s'executen dins una mateixa instància de Linux, evitant la sobrecàrrega d'iniciar i mantenir màquines virtuals.

Una definició més pràctica ens la dona la firma analista *451 Research*:

Docker és una eina que pot empaquetar una aplicació i les seues dependències en un contenidor virtual que es pot executar en qualsevol servidor Linux. Açò aporta flexibilitat i portabilitat a les aplicacions, ja que aquestes es poden executar en instal·lacions físiques, el núvol públic, núvols privats, etc.

Bàsicament, el que aconseguix Docker és oferir contenidors d'aplicacions que aprofiten les capacitats de virtualització del kernel de Linux per poder executar processos i serveis de forma aïllada. Es tracta d'un concepte paregut al de màquina virtual, però que no requereix d'un sistema operatiu, sinò que aprofita el kernel del propi Linux i les capacitats que té d'aïllar recursos, tals com la CPU, la memòria, la xarxa o l'entrada/eixida. Així doncs, podem tindre diversos contenidors compartint el mateix kernel de Linux, però cadascun amb restriccions d'accés a determinats recursos.

En els últims anys, ha estat una tecnologia cada vegada més utilitzada per la versatilitat que dona. Vegem un gràfic de les cerques de «Docker» des del 2004 (Google Trends).

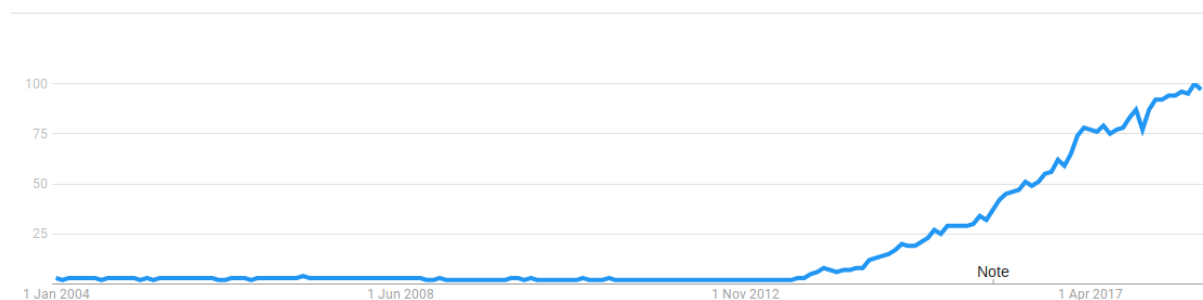


Figura 2: Logo de Docker

Els principals avantatges de l'ús de contenidors són:

- La flexibilitat, ja que fins i tot les aplicacions més complexes poden incloure's als contenidors,
- La poca càrrega que suposen per al sistema, al compartir el mateix kernel que l'amfitrió,
- La possibilitat de desplegar actualitzacions *en calent*,
- La portabilitat, ja que es poden desenvolupar localment, desplegar al núvol i llançar-los en qualsevol lloc,
- L'escalabilitat, ja que permet incrementar automàticament rèpliques dels contenidors,
- Els serveis en contenidors poden apilar-se *on the fly*

Contenedor i imatges

Al llarg del document parlarem sobre imatges i contenidors, pel que convé aclarir aquests conceptes.

Una **imatge** és un paquet executable que inclou tot allò necessari per executar una aplicació: el codi, l'entorn d'execució, llibreríes, variables d'entorn i fitxers de configuració.

Un **contenedor**, per la seua banda és una instància d'una imatge en execució: allò que es crea quan posem en marxa una imatge. Podríem dir que un contenedor és a una imatge el que un procés a un programa (procés=programa en execució -> contenedor=imatge en execució).

1.1. Preparació de l'entorn

Font: <https://docs.docker.com/install/linux/docker-ce/ubuntu/>

El projecte Docker disposa dels seus propis repositoris de programari. En aquest apartat anem a veure com descarregar-nos la versió Community de Docker, orientada a desenvolupadors i equips menuts que comencen amb Docker. L'alternativa empresarial seria Docker Enterprise Edition (EE).

En aquest apartat anem a instal·lar les eines necessàries per poder descarregar i instal·lar Docker CE al nostre equip (o màquina virtual).

En primer lloc, actualitzem la caché de paquets de l'ordinador:

```
1 $ sudo apt-get update
```

I instal·lem els paquets següents:

```
1 $ sudo apt-get install \
2   apt-transport-https \
3   ca-certificates \
4   curl \
5   software-properties-common
```

Amb açò, descarreguem la clau GPG del lloc de Docker (amb l'ordre `curl`) i la incorporem al sistema (amb `apt-key add -`), per tal que el nostre sistema confien en el lloc per a la descàrrega de programari (fixeu-se que hi ha una canonada `|`):

```
1 $ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-
   key add -
2 OK
```

Per tal de comprovar que la clau s'ha instal·lat correctament:

```
1 $ sudo apt-key fingerprint 0EBFCD88
2 pub   rsa4096 2017-02-22 [SCEA]
3       9DC8 5822 9FC7 DD38 854A  E2D8 8D81 803C 0EBF CD88
4 uid           [ unknown] Docker Release (CE deb) <docker@docker.com>
5 sub   rsa4096 2017-02-22 [S]
```

Si tot és correcte i han aparegut bé les claus de dalt, podem continuar afegint el repositori als orígens de programari del nostre sistema (el que tenim en `/etc/apt/sources.list*`):

```
1 $ sudo add-apt-repository \
2   "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
3   $(lsb_release -cs) \
4   stable"
```

Fet açò, en versions anteriors a Bionic (Ubuntu 18.04) caldria fer un `apt-get update`, per refrescar la llista de paquets de programari disponibles al nostre sistema. Com que Ubuntu 18.04 ja refresca la llista de paquets cada vegada que afegim un dipòsit nou, no és necessari fer-ho en aquesta distribució.

I ara, ja instal·lem el paquet `docker-ce`.

1.2. Instal·lació de Docker-CE

Amb els repositoris ja configurats, només hem de realitzar la instal·lació amb `apt-get`:

```
1 $ sudo apt-get install docker-ce
```

Per tal de comprovar que tot ha funcionat bé, anem a llançar un contenidor amb la imatge «*hello-world*»:

```
1 $ sudo docker run hello-world
2 Unable to find image 'hello-world:latest' locally
3 latest: Pulling from library/hello-world
4 d1725b59e92d: Pull complete
5 Digest: sha256:0
   add3ace90ecb4adbf7777e9aacf18357296e799f81cab9fde470971e499788
6 Status: Downloaded newer image for hello-world:latest
7
8 Hello from Docker!
9 This message shows that your installation appears to be working
   correctly.
10
11 To generate this message, Docker took the following steps:
12 1. The Docker client contacted the Docker daemon.
13 2. The Docker daemon pulled the "hello-world" image from the Docker
   Hub.
14    (amd64)
15 3. The Docker daemon created a new container from that image which
   runs the
16    executable that produces the output you are currently reading.
17 4. The Docker daemon streamed that output to the Docker client, which
   sent it
18    to your terminal.
19
20 To try something more ambitious, you can run an Ubuntu container with:
21 $ docker run -it ubuntu bash
22
23 Share images, automate workflows, and more with a free Docker ID:
24 https://hub.docker.com/
25
26 For more examples and ideas, visit:
27 https://docs.docker.com/get-started/
```

Analitzem alguns detalls:

```
1 $ sudo docker run hello-world
```

Com veiem, l'execució de docker s'ha de realitzar com a sudo. Si volem utilitzar docker per a altres usuaris, només haurem de crear un grup docker i afegir ahi els usuaris que el puguin utilitzar.

Per altra banda, veiem com utilitzar docker per tal de llançar una imatge: `docker run`. En aquest cas,

una imatge de prova anomenat *hello-world*.

```
1 Unable to find image 'hello-world:latest' locally
2 latest: Pulling from library/hello-world
3 d1725b59e92d: Pull complete
4 Digest: sha256:0
   add3ace90ecb4adbf7777e9aacf18357296e799f81cab9fde470971e499788
5 Status: Downloaded newer image for hello-world:latest
6
7 Hello from Docker!
8 This message shows that your installation appears to be working
   correctly.
```

Com veiem, ens indica que no troba la imatge «hello-world:latest», pel que la descarrega de la llibreria (pull). Una vegada descarregada ja ens mostra el missatge que comença amb «*Hello from Docker*», corresponent a aquesta imatge.

Si tornem a llançar l'ordre, comprovarem com ja no realitza la descàrrega, sinò que executa directament la imatge Hello World.

Docker i serveis

El dimoni (daemon) de Docker és el servei que gestiona la creació, execució i distribució de contenidors. L'ordre `docker` que hem vist anteriorment, és el client de docker, que permet a l'usuari interactuar amb el sistema, encara que també hi ha altres aplicacions clints.

Si volem que el servei de Docker s'iniciï en arrancar el sistema (systemd), farem:

```
1 sudo systemctl enable docker
```

I si volem llevar-lo de l'inici:

```
1 sudo systemctl disable docker
```

Podem trobar més informació sobre tot el que podem fer després d'haver instal·lat Docker en: <https://docs.docker.com/install/linux/linux-postinstall/#configure-docker-to-start-on-boot>, així com documentació per a la configuració del servei de Docker per a que escolte determinats ports mitjançant el fitxer `/etc/docker/daemon.json`: <https://docs.docker.com/install/linux/linux-postinstall/#configure-where-the-docker-daemon-listens-for-connections>

1.3. Docker i Busybox

A mode d'exemple, anem a veure com descarregar i llançar una imatge per a Docker de Busybox. Busybox no és més que una utilitat que combina moltes eines estàndards d'Unix i ordres de Linux en

un sol fitxer.

Per descarregar el busybox fem ús de l'ordre de Docker `docker pull` (fixeu-se que som l'usuari administrador):

```
1 # docker pull busybox
2 Using default tag: latest
3 latest: Pulling from library/busybox
4 8c5a7da1afbc: Pull complete
5 Digest: sha256:
   cb63aa0641a885f54de20f61d152187419e8f6b159ed11a251a09d115fdff9bd
6 Status: Downloaded newer image for busybox:latest
```

Amb açò hem descarregar la imatge de busybox pe a Docker des del Docker Hub (<https://hub.docker.com/explore/>).

Per tal de vore les imatges de Docker que tenim instal·lades, podem fer:

1	# docker images			
2	REPOSITORY	TAG	IMAGE ID	CREATED
		SIZE		
3	hello-world	latest	4ab4c602aa5e	3 weeks ago
		1.84kB		
4	busybox	latest	e1ddd7948a1c	2 months ago
		1.16MB		

Com podem comprovar, tenim la imatge de busybox i hello-world.

Anem ara a llançar el Busybox. Per a això farem:

```
1 # docker run busybox
```

Amb açò, si no tenim descarregada la imatge, el primer que farà és descarregar-la. Si ja la tenim descarregada, ometrà aquest pas. Amb la imatge de busybox al sistema, Docker la busca, la carrega en un contenidor i executa al busybox les ordres que li passem. Com que en aquest cas no li hem passat cap ordre, aparentment, no farà res.

Així doncs, per executar alguna cosa dins el docker li haurem de passar com a paràmetre:

```
1 # docker run busybox echo "hola"
2 hola
3
4 # docker run busybox ls
5 bin
6 dev
7 etc
8 home
9 proc
10 root
11 sys
12 tmp
```



```

13  usr
14  var
15
16  # docker run busybox cat /etc/passwd
17  root:x:0:0:root:/root:/bin/sh
18  daemon:x:1:1:daemon:/usr/sbin:/bin/false
19  bin:x:2:2:bin:/bin:/bin/false
20  sys:x:3:3:sys:/dev:/bin/false
21  sync:x:4:100:sync:/bin:/bin/sync
22  mail:x:8:8:mail:/var/spool/mail:/bin/false
23  www-data:x:33:33:www-data:/var/www:/bin/false
24  operator:x:37:37:Operator:/var:/bin/false
25  nobody:x:65534:65534:nobody:/home:/bin/false

```

Com podem veure, hem llançat tres ordres diferents sobre el Busybox: hem escrit hola, hem llistat el sistema de fitxers, i hem consultat el fitxer `/etc/passwd`. Fixeu-se que ni el sistema de fitxers que hem mostrat ni el fitxer `passwd` es corresponen amb l'estructura de fitxers del sistema o el fitxer `passwd` del nostre sistema. Estem accedint al sistema de fitxers i el fitxer `passwd` del propi Busybox. De fet, si fem un `ps aux`, veurem que no hi ha cap altre procés en el sistema:

```

1  # docker run busybox ps aux
2  PID    USER      TIME  COMMAND
3      1   root         0:00  ps aux

```

1.4. L'ordre «docker ps»

L'ordre `docker ps` serveix per veure els contenidors que s'estan executant en un moment donat. Si fem en un terminal:

```
1 $ docker run busybox sleep 10
```

I des d'altre:

```

1 $ sudo docker ps
2  CONTAINER ID        IMAGE               COMMAND             CREATED
3  d4018e11cc64        busybox            "sleep 10"         5 seconds
4  ago                Up 4 seconds       elegant_keldysh
joamuran@toki:~$

```

Veiem que tenim l'ordre `sleep 10` funcionant sobre la imatge de busybox en el contenidor `d4018e11cc64`.

1.5. Execució interactiva

Si volem llançar més d'una ordre per contenidor, podem fer ús del paràmetre -it (flag interactive):

```

1 root@toki:/home/joamuran# docker run -it busybox sh
2 / # ls
3 bin    dev    etc    home  proc  root  sys    tmp    usr    var
4 / # users
5
6 / # cat /etc/passwd
7 root:x:0:0:root:/root:/bin/sh
8 daemon:x:1:1:daemon:/usr/sbin:/bin/false
9 bin:x:2:2:bin:/bin:/bin/false
10 sys:x:3:3:sys:/dev:/bin/false
11 sync:x:4:100:sync:/bin:/bin/sync
12 mail:x:8:8:mail:/var/spool/mail:/bin/false
13 www-data:x:33:33:www-data:/var/www:/bin/false
14 operator:x:37:37:Operator:/var:/bin/false
15 nobody:x:65534:65534:nobody:/home:/bin/false
16 / # exit

```

1.6. Netejant espai

Amb l'opció -a de docker ps podem obtindre tots els contenidors que s'han creat en la sessió actual:

```

1 # docker ps -a
2 CONTAINER ID        IMAGE               COMMAND             CREATED             NAMES
3 5dde99182cec        busybox            "sh"               2 minutes ago      festive_ride
4 d4018e11cc64        busybox            "sleep 10"         16 minutes ago     elegant_keldysh
5 2a2b08d458c5        busybox            "ps aux"           20 minutes ago     zen_bassi
6 b17136bc9389        busybox            "cat /etc/passwd"  22 minutes ago     gifted_archimedes
7 a475a7587143        busybox            "ls"               23 minutes ago     jovial_leavitt
8 1febe609a9a9        busybox            "echo hola"        23 minutes ago     compassionate_spence
9 dc64b0307514        busybox            "sh"               25 minutes ago     clever_lalande

```

10	2a4808f37627	hello-world	"/hello"	About an hour ago	Exited (0) About an hour ago
11	31545996c7f7	hello-world	"/hello"	About an hour ago	Exited (0) About an hour ago

Com veiem, la columna *status* indica que els contenidors han acabat. So volem eliminar-los, podríem fer:

```
1 # docker rm 5dde99182cec d4018e11cc64...
```

Aquest mecanisme és un poc tediós, pel que anem a fer-ho més senzill.

Amb la següent ordre, podem obtenir els ids dels contenidors que ja han acabat:

```
1 toki:/home/joamuran# sudo docker ps -a -q -f status=exited
2 5dde99182cec
3 d4018e11cc64
4 2a2b08d458c5
5 b17136bc9389
6 a475a7587143
7 1febe609a9a9
8 dc64b0307514
9 2a4808f37627
10 31545996c7f7
```

Pel que si combinem aquesta ordre amb docker rm:

```
1 sudo docker rm $(docker ps -a -q -f status=exited)
```

2. Servidor MariaDB amb Docker

Com hem comentat anteriorment, el Docker Hub (<https://hub.docker.com/explore/>) oferix una gran quantitat d'imatges. Aquest lloc pot entendre's com una espècie de Github per a imatges de Docker.

Les imatges es poden classificar de diverses formes atenent a diferents criteris.

Atenent al procés de creació de la imatge, podem distingir:

- **Imatges de base:** Aquelles que han estat creades de zero, generalment a partir de sistemes operatius com Ubuntu, Busybox o Debian, o bé
- **Imatges filles:** Aquelles que estan construïdes sobre una imatge base, amb funcionalitat addicional.

Per altra banda, atenent a qui ha creat les imatges, distingim:

- **Imatges oficials:** Són aquelles mantinguts per Docker, i que generalment són imatges de base. Generalment el nom és una cadena, com `busybox` i `hello-world`.
- **Imatges d'usuari:** Són aquelles creades i compartides pels usuaris. Generalment, són imatges basades en les imatges de base amb funcionalitat addicional. Normalment, s'anomenen amb `usuari/imatge`.

Per tal de disposar d'un servidor de Subversion, farem ús de la imatge `mamohr/subversion-edge` (<https://hub.docker.com/r/mamohr/subversion-edge>), basat en el servidor de subversion Subversion Edge de CollabNet, que disposa d'interfície web per a la seua gestió.

2.1. Descàrrega de la imatge

Anem a treballar amb la imatge oficial del SGBD MySQL (https://hub.docker.com/_/mysql). Per talde descarregar-nos la imatge més recent del SGBD, farem:

```
1 $ docker pull mysql
```

2.2. Creació d'un contenidor

Una vegada descarregada la imatge, podem llançar-la amb un simple `docker run mysql`, però no tindríem el servei disponible des del nostre equip, i les seues dades desapareixerien quan esborrarem el contenidor.

Per tal d'ferir persistència, haurem d'enllaçar la carpeta on guarda MySQL tota la informació amb una carpeta del nostre equip, coneguda en terminologia de Docker com a *volum*.

Per a això, crearem una carpeta, per exemple a `/srv/mysql-data`:

```
1 $ sudo mkdir /srv/mariadb-data
```

I ara llançarem un contenidor amb la següent ordre (les `\` són per indicar canvi de línia en *bash*, però no s'han de posar si ho escrivim tot a una línia):

```
1 $ docker run --name mysql-srv \  
2     -p 3308:3306 \  
3     -v /srv/mysql:/var/lib/mysql \  
4     -e MYSQL_ROOT_PASSWORD="root" \  
5     -d mysql
```

Les opcions que hem utilitzat han estat:

- `--name mysql-srv`: Li donem un nom (*mysql-srv*) al contenidor, per a quan hajam d'aturar-lo o eliminar-lo referir-nos a ell de forma més senzilla.

- `-p 3308:3306`: Aci fem el que es coneix com *exposició de ports*, és a dir, *exposem* els ports pels que treballa el contenidor per defecte a través del ports del nostre equip. En aquest cas, el port per defecte de MySQL (3306) del contenidor, estarà visible en la nostra màquina a través del port 3308.
- `-v /srv/mysql:/var/lib/mysql`: Enllacem el volum que acabem de crear (la carpeta `/srv/mysql`), amb la carpeta `/var/lib/mysql`, que és on el servidor de MySQL emmagatzema les dades.
- `-e MYSQL_ROOT_PASSWORD="root"`: Amb `-e` establim variables d'entorn. En aquest cas, estem establint el valor de la variable `MYSQL_ROOT_PASSWORD` (és a dir, la contrassenya de `root`) com a `root`.
- `-d`: Indica que anem a llançar el contenidor en background, sense que ens mostre tots els missatges de log per pantalla.

I finalment, hem indicat el nom de la imatge `mysql` a llançar.

Amb açò, si fem un `docker ps`, obtindrem:

1						
2	CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	
		PORTS		NAMES		
3	2a89233a9d13	mysql	"docker-entrypoint....s"	19 minutes ago	Up 19	
	minutes	33060/tcp	, 0.0.0.0:3308->3306/tcp	mysql-srv		

Amb açò ja tindrem un contenidor amb un servidor de MySQL funcionant en estos moments, i que podrem aturar i engegar quan el necessitem.

Enggant un contenidor ja creat

Una vegada hem llançat l'ordre `docker run` anterior, creem el contenidor a partir de la imatge, i podem aturar-lo amb `docker stop`. De tota manera, com hem vist, amb açò aturem el contenidor però no l'eliminem. D'aquesta manera, si intentem tornar a crear de nou el contenidor amb `docker run`, obtindrem el següent error:

```
1 $ docker run --name mysql-srv \  
2 > -p 3308:3306 \  
3 > -v /srv/mysql:/var/lib/mysql \  
4 > -e MYSQL_ROOT_PASSWORD="root" \  
5 > -d mysql  
6 docker: Error response from daemon: Conflict. The container name "/  
mysql-srv" is already in use by container "2  
a89233a9d13fc9db96565f5858fdaa7a58b68b95e262467391f3eb68c6d5997".  
You have to remove (or rename) that container to be able to reuse  
that name.  
7 See 'docker run --help'.
```

Açò ens està indicant que el nom de contenidor `mysql-srv` ja està en ús per altre contenidor. Si fem un `docker ps -a`, veurem que tenim aquest contenidor creat:

```
1 docker ps -a
```

2	CONTAINER ID	IMAGE	COMMAND	
	CREATED	STATUS		PORTS
	NAMES			
3	2a89233a9d13	mysql	"docker-entrypoint....s"	3 days ago
	ago	Exited (0) 3 days ago		mysql-srv

Ara tenim dues possibilitats, eliminar aquest contenidor (amb `docker rm`) o bé seguir executant-lo. Aquesta última opció té l'avantatge que, com que no hem eliminat el contenidor, la informació que haverem guardat, encara que no haverem utilitzat cap volum, seguiria estant disponible. Així, per tal de posar en marxa de nou aquest contenidor ja creat, farem:

```
1 $ docker start mysql-srv
2 mysql-srv
```

Així doncs, i a mode de conclusió, podem establir les següents diferències entre `docker run` i `docker start`:

- `docker run`: Crea un nou contenidor a partir d'una imatge i executa les ordres que indiquem.
- `docker start`: Inicia un contenidor aturat, mantenint aquest tal i com estava en el moment d'aturar-lo, pel que manté la informació que aquest estiguera gestionant sense necessitat d'utilitzar volums.

2.3. Connexió des d'un client

Per tal de connectar-nos des de la terminal, haurem d'indicar-li la IP local 127.0.0.1 (no ens serveix el `localhost` per defecte) i el port pel que ens anem a connectar (3308):

```
1 ~$ mysql -u root --host=127.0.0.1 --port=3308 -p
2 Enter password:
3 Welcome to the MySQL monitor.  Commands end with ; or \g.
4 Your MySQL connection id is 12
5 Server version: 8.0.18 MySQL Community Server - GPL
6
7 Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights
  reserved.
8
9 Oracle is a registered trademark of Oracle Corporation and/or its
10 affiliates. Other names may be trademarks of their respective
11 owners.
12
13 Type 'help;' or '\h' for help. Type '\c' to clear the current input
  statement.
```

Com veiem, ens demana el password de root (ja que hem indicat l'opció `-p`), i una vegada introduït, ens mostra el prompt de MySQL.

També podem connectar-nos des d'altre client que ens dóna més joc, com el MySQL Workbench. Si no el tenim instal·lat podem fer-ho amb `sudo apt-get install mysql-workbench`. Amb aquesta eina, només haurem de crear una nova connexió amb els paràmetres amb què hem configurat el servidor. Per a això, des de la finestra principal, farem clic al símbol **+** que tenim ubicat al costat de *MySQL Connections* i configurarem la nova connexió amb els següents paràmetres:

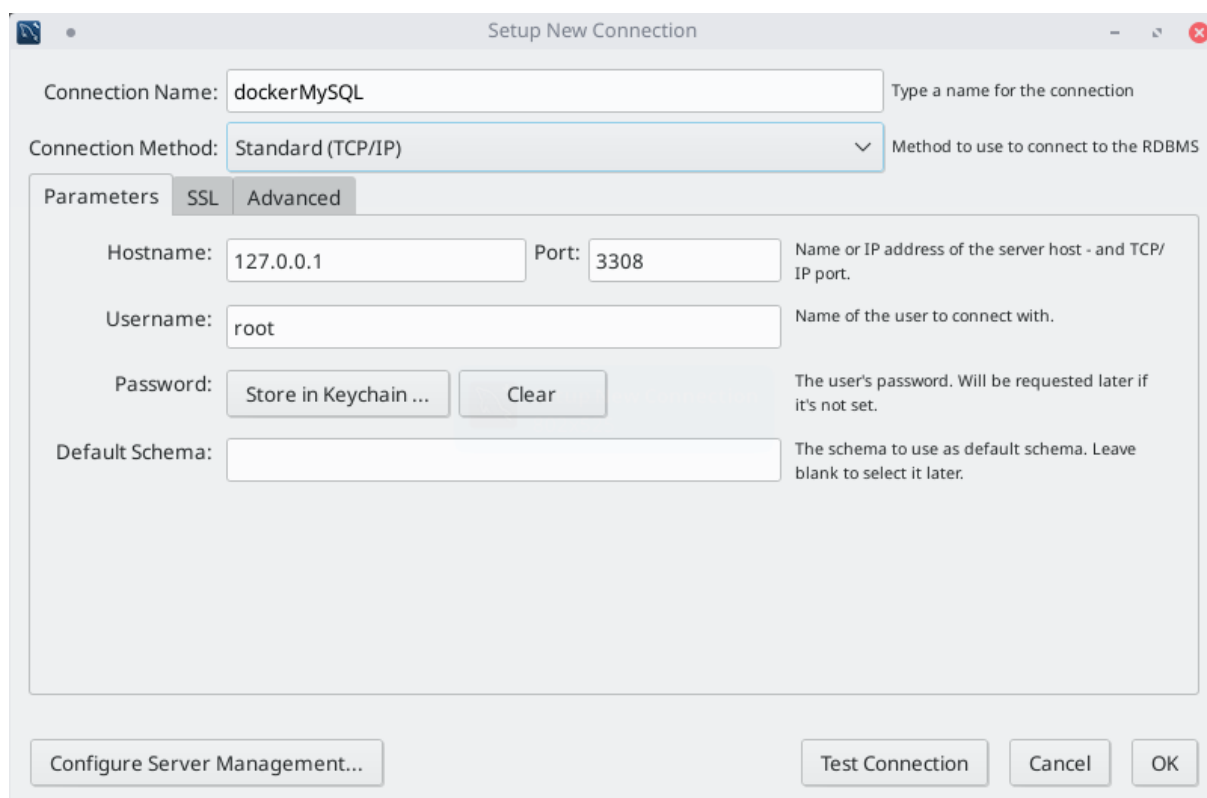


Figura 3: Configuració de la connexió

Fet açò, podem testejar la connexió per veure si està tot correcte i connectar-nos al servidor.