

Contenedores para utilizar en clase

Los contenedores pueden ejecutarlos los usuarios sin ser root pero no pueden usar puertos menores a 1024.

Se va a implementar los contenedores para que se inicien cuando inicie sesión el usuario y se pararan cuando se apague la máquina (ojo, cuando se cierra sesión no se paran)

Configuración contenedor para clase con el usuario alumno

- Crear directorio para que el usuario pueda crear servicios

```
mkdir -p ~/.config/systemd/user/
```

- Descargar y ejecutar el contenedor:

```
podman run --name mati-postgresql -e POSTGRES_USER=mati -e POSTGRES_PASSWORD=mati -p 5432:5432 -v vol_postgres_mati:/var/lib/postgresql/data -d docker.io/library/postgres:latest
```

- Para probar si funciona el contenedor conectamos a la base de datos:

```
psql "user=mati hostaddr=127.0.0.1"
```

- Cambiar al directorio de los servicios:

```
cd ~/.config/systemd/user/
```

- Crear fichero para el servicio:

```
podman generate systemd --name mati-postgresql --files --new
```

- Recargamos el servicio:

```
systemctl --user daemon-reload
```

- Activamos e inicializamos el servicio:

```
systemctl --user enable container-mati-postgresql.service
```

```
systemctl --user start container-mati-postgresql.service
```

- Paramos el contenedor:

```
systemctl --user stop container-mati-postgresql.service
```

- Iniciamos el contenedor:

```
systemctl --user start container-mati-postgresql.service
```

Exportar bbdd

Una vez el contenedor este en ejecución, verificamos su id mediante el comando:

```
podman ps
```

El resultado sería similar a este:

```
ce40ec8343fd docker.io/library/postgres:latest postgres About a minute ago Up About a minute ago 0.0.0.0:5432->5432/tcp mati-postgresql
```

Cogemos el primer string "ce40ec8343fd" y ejecutamos el siguiente comando:

```
podman exec -it ce40ec8343fd /bin/bash
```

Y obtendremos un terminal dentro del contenedor:

```
pg_dump --file "nombrebackup.sql" -U nombreusuario -W -h localhost --verbose nombrebdd
```

Ejecutamos `ls -ltr` y vemos nuestro fichero "nombrebackup.sql"

Salimos con `exit`

Copiamos a nuestro home el fichero sql:

```
podman cp ce40ec8343fd:/nombrebackup.sql ~/
```

Una vez parado el contenedor se destruirá el fichero "nombrebackup.sql"

Eliminar volumen del contenedor vinculado a un servicio

Eliminamos el volumen ya que el usuario ha hecho algo que no debía y no va el contenedor o se quieren borrar sus datos.

- Paramos el contenedor:

```
systemctl --user stop container-mati-postgresql.service
```

- Eliminamos el volumen:

```
podman volume rm vol_postgres_mati
```

- Volvemos a iniciar el contenedor:

```
systemctl --user start container-mati-postgresql.service
```

Ubicación de los contenidos de los volúmenes

Para el usuario alumnom y el contenedor vol_mysql_mati los archivos estarían en:

```
/home/alumnom/.local/share/containers/storage/volumes/vol_postgres_mati/_data
```

Sólo si es muy necesario:

- Podemos usar un directorio en vez de un volumen:
 - Creamos el directorio:

```
mkdir -p /home/alumnom/contenedor/postgres
```

- En el apartado anterior *“Descargar y ejecutar el contenedor”* cambiamos vol_postgres_mati por /home/alumnom/contenedor/postgres

Comandos útiles

- Parar contenedor en ejecución que no esté vinculado a un servicio:

```
podman stop mati-mysql
```

- Borrar contenedor:

```
podman rm mati-mysql
```

- Ver contenedores en ejecución

```
podman ps
```

- Ver todos los contenedores (parados y en ejecución)

podman ps -a

- ver volúmenes

podman volume ls

- Eliminar un volumen.

podman volume rm vol_mysql_mati

- Podman run:

podman run --name nombreunico -v dirlocal_o_vol:dircontenedor -e VARIABLECONTENDOR=valorvariable -p puertolocal:puertocontenedor -d docker.io/library/nombrecontendor:version

- nombreunico: Nombre único que queramos poner
- dirlocal_o_vol: directorio local que se montará en el equipo o volumen para que cree el directorio automáticamente podman y lo gestione
- dircontenedor: Directorio del contenedor que se vinculará al volumen o al directorio local
- VARIABLECONTENDOR=valorvariable: Variable que se le pasa al contenedor para la use al arrancarlo (como una password o similar)
- puertolocal:puertocontenedor: puerto que se vinculará de nuestra máquina al contenedor (puertolocal debe ser mayor a 1024)
- docker.io/library/nombrecontendor:version: Imagen de contenedor que se usará
- Por ejemplo:

podman run --name mati-mysql -v vol_mysql_mati:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=mati -p 3306:3306 -d docker.io/library/mysql:latest

mysql -u root -p --port=3306 --host=127.0.0.1

Listar contenedores

Parar el conetnedor

```
podman stop mati-mysql
```

8.7 Elimine todos los contenedores.

```
[student@servera ~]$ podman rm -a
```

8.8 Verifique que se hayan eliminado todos los contenedores.

```
[student@servera ~]$ podman ps -a
```

Cree un volumen mediante podman volume.

```
podman volume create my_vol
```

Mostrar volúmenes.

```
podman volume ls
```

Eliminar un volumen.

```
podman volume rm my_vol
```

Inicie un contenedor y cree un volumen asociado a él.

```
podman run -it -v my_data:/data --name box1 oraclelinux:8
```

```
podman rm -v box2
```

```
podman ps -a
```

podman volume ls

podman volume prune

podman volume ls