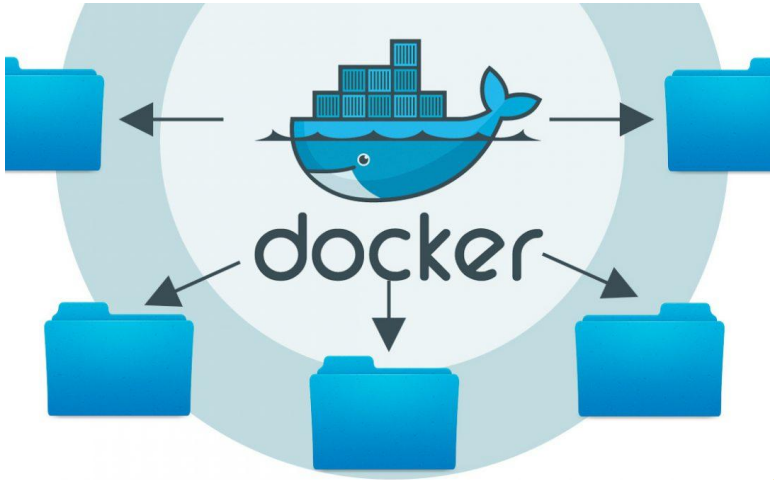
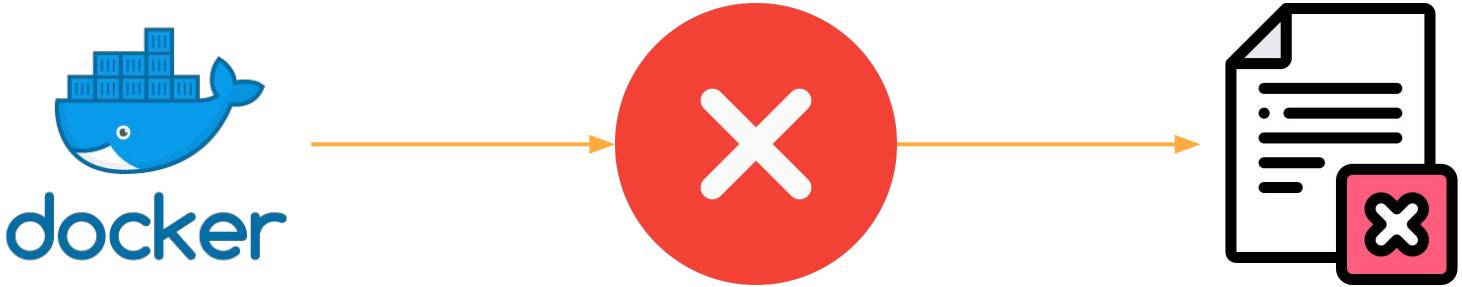
Docker Volume

Docker Volume bisa dianggap sebagai storage atau tempat penyimpanan data di container. Tentunya saat kita membuat container kita tidak ingin ketika container kita mati data yang ada pada container ikut terhapus juga. Untuk itu kita dapat memanfaatkan Volume pada docker.



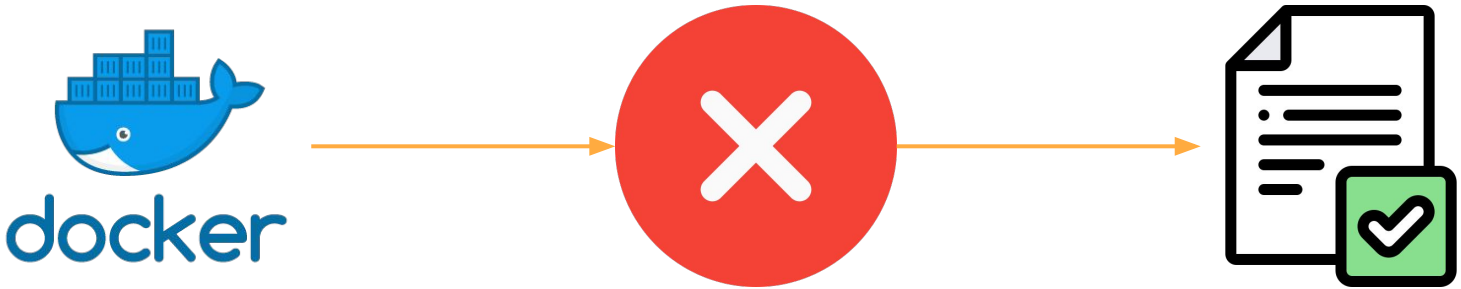
Running Container Without Volume

Ketika container terhapus, data yang ada pada container juga ikut terhapus



Running Container With Volume

Ketika container terhapus, data yang ada pada container tetap tersimpan dan tidak akan ikut terhapus.



Basic Command

Create Volume

```
$ docker create volume <name_volume>
```

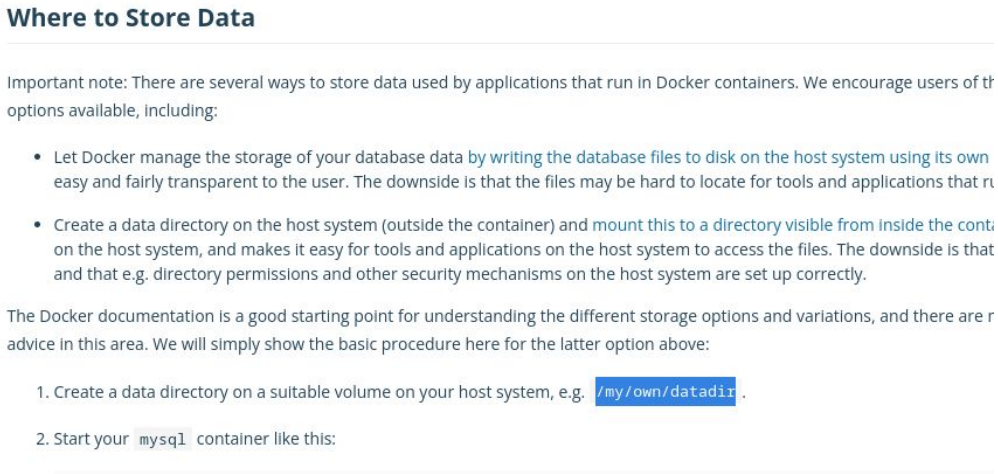
Remove Volume

```
$ docker volume rm <name_volume>
```

Add container to volume

```
$ docker container create --name <name_container> \
--mount "type=volume, source=<name_volume>, dst=<folder_destionation>"
```

Example



Where to Store Data

Important note: There are several ways to store data used by applications that run in Docker containers. We encourage users of th
options available, including:

- Let Docker manage the storage of your database data by writing the database files to disk on the host system using its own
  easy and fairly transparent to the user. The downside is that the files may be hard to locate for tools and applications that ru
- Create a data directory on the host system (outside the container) and mount this to a directory visible from inside the conta
  on the host system, and makes it easy for tools and applications on the host system to access the files. The downside is that
  and that e.g. directory permissions and other security mechanisms on the host system are set up correctly.

The Docker documentation is a good starting point for understanding the different storage options and variations, and there are r
advice in this area. We will simply show the basic procedure here for the latter option above:

1. Create a data directory on a suitable volume on your host system, e.g. `/my/own/datadir` .
2. Start your `mysql` container like this:

Sebelum kita membuat volume kita perlu tau dimana container menyimpan data, pada contoh kali ini kita akan memakai image **mysql**.

Kita bisa mengetahuinya melalui docker hub, disana kita tau bahwa mysql menyimpan data pada **/my/own/datadir**.

Example

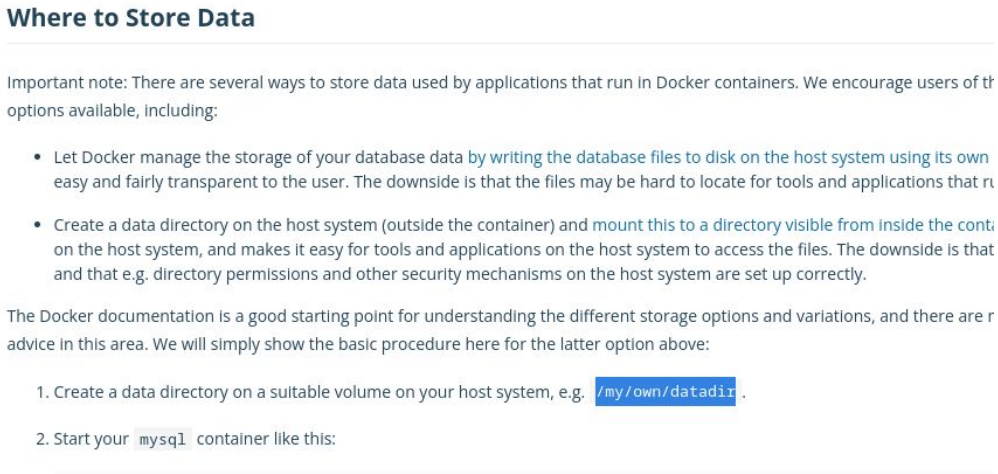
Create Volume

```
$ docker create volume data-mysql
```

Add container to volume

```
$ docker container create --name own-mysql \
--mount "type=volume, source=data-mysql, dst=/my/own/datadir" \
mysql:latest
```

Example



Where to Store Data

Important note: There are several ways to store data used by applications that run in Docker containers. We encourage users of th
options available, including:

- Let Docker manage the storage of your database data by writing the database files to disk on the host system using its own
  easy and fairly transparent to the user. The downside is that the files may be hard to locate for tools and applications that ru
- Create a data directory on the host system (outside the container) and mount this to a directory visible from inside the conta
  on the host system, and makes it easy for tools and applications on the host system to access the files. The downside is that
  and that e.g. directory permissions and other security mechanisms on the host system are set up correctly.

The Docker documentation is a good starting point for understanding the different storage options and variations, and there are r
advice in this area. We will simply show the basic procedure here for the latter option above:

1. Create a data directory on a suitable volume on your host system, e.g. `/my/own/datadir` .
2. Start your `mysql` container like this:

Sebelum kita membuat volume kita perlu tau dimana container menyimpan data, pada contoh kali ini kita akan memakai image **mysql**.

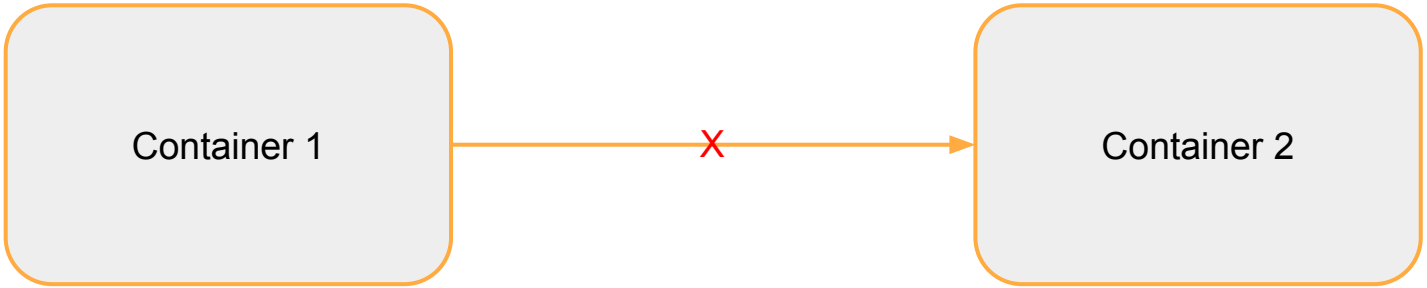
Kita bisa mengetahuinya melalui docker hub, disana kita tau bahwa mysql menyimpan data pada **/my/own/datadir**.

Docker Network

Defaultnya container pada docker akan saling terisolasi satu sama lain. Kita tidak dapat melakukan request api (misal) dari container satu ke container lain. Untuk itu kita harus membuat dan mendaftarkan container pada network yang sama.

Docker Network

Secara default container tidak bisa saling berkomunikasi satu sama lain



Docker Network

Untuk itu kita perlu **docker network** agar container bisa berkomunikasi satu sama lain

Basic command

alterra academy

Create Network

```
# list network
$ docker network ls
# create network
$ docker network create <name_network>
# delete network
$ docker network rm <name_network>
```

Regist container to network

```
# regist container to network
$ docker container create --name <container_name> \
--network <name_network>
# delete container from network
$ docker network disconect <name_network> <container_name>
```

Example without network

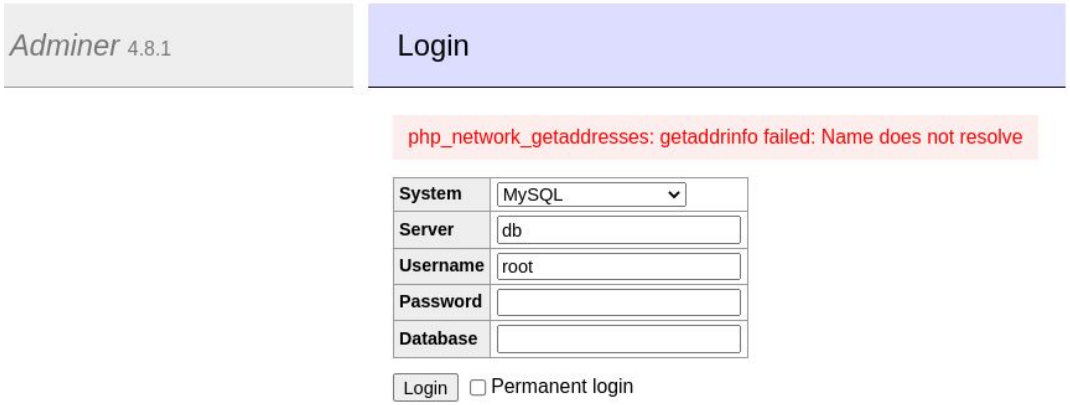
Kita akan menggunakan mysql dan adminer sebagai contoh, kita akan mengetes tanpa docker network.

Create Container

```
# mysql
$ docker container create --name my-sql -p 3306:3306 mysql:latest & \
docker container start my-sql
# adminer
$ docker container create --name my-adminer -p 8080:8080 adminer:latest & \
docker container start my-adminer
```

Example Without Network

Kalau kita coba akses db melalui adminer maka akan mendapatkan error karena adminer tidak bisa melakukan komunikasi dengan mysql



Example with network

Create Network

```
$ docker network create database-network
```

Create Container

```
# mysql
$ docker container create --name my-sql -p 3306:3306 mysql:latest \
--network database-network & \
docker container start my-sql
# adminer
$ docker container create --name my-adminer -p 8080:8080 adminer:latest \
--network database-network & \
docker container start my-adminer
```