

# Docker Management

Adfinis**sy**Group

Be smart. Think open source.

# Docker Management

## **Disclaimer**

This presentation is geared towards management of development environments. Not all of the recommendations apply for production environments!

# Docker volume Management

## Automatically created volumes

Docker Engine automatically creates a volume for every `VOLUME` command in the Dockerfile

```
docker run -d mariadb  
docker volume ls
```

# Mount a host path in the containers

Volumes can also be paths on the host

**Paths specified this way must be absolute paths**

```
docker run -d -v "${PWD}/dockerdata:/var/lib/mysql" --name mariadb mariadb
docker inspect -f '{{ json .Mounts }}' mariadb | jq
[
  {
    "Source": "/home/lukasgr/Docker_Workshop/dockerdata",
    "Destination": "/var/lib/mysql",
    "Mode": "",
    "RW": true,
    "Propagation": "rprivate"
  }
]
```

**docker-compose**



## What is docker-compose

docker-compose provides a layer of abstraction to avoid running multiple docker commands or using crazy long commands with multiple options

To configure it create a **docker-compose.yml** file in a directory, an example is provided on the next slide

# Applications with docker-compose

`docker-compose` allows to combine multiple Docker containers to an application

```
version: '2'
services:
  db:
    image: mariadb:10.1
    volumes:
      - "./.data/db:/var/lib/mysql"
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: wordpress
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: wordpress

  wordpress:
    depends_on:
      - db
    image: wordpress:latest
    volumes:
      - "./.data/wp-content:/var/www/html/wp-content"
    links:
      - db
    ports:
      - "8000:80"
    restart: always
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_PASSWORD: wordpress
```

## Start you application with docker-compose

Copy the example from the previous slide to `docker-compose.yml` and run

```
docker-compose up
```

and open <http://localhost:8000/> in your browser

**Keep your environment clean**

## Remove exited containers

This command removes all containers in the exited state

```
docker ps -q -f status=exited | xargs -r docker rm -v
```

## Remove dangling images

This command removes all Docker images which are not part of a Docker layer anymore

```
docker images -q -f dangling=true | xargs -r docker rmi
```

Since version 1.25:

```
docker system prune
```

## Remove dangling volumes

This command removes all automatically created volumes which are not attached to a container anymore

```
docker volume ls -q -f dangling=true | xargs -r docker volume rm
```

Since version 1.25:

```
docker system prune --volumes
```

## Update local Docker images to the newest version

This command will check for new versions of the each image:tag combination in `docker images`

```
docker images | awk '(NR>1) && ($2!~/none/) {print $1":"$2}' | xargs -L 1 docker pull
```



# Feel Free to Contact Us

[www.adfinis-sygroup.ch](http://www.adfinis-sygroup.ch)

[Tech Blog](#)

[GitHub](#)

[info@adfinis-sygroup.ch](mailto:info@adfinis-sygroup.ch)

[Twitter](#)

