# Docker quickguide

***Data Engineering Frühlingssemester 2020***

Marcel Huber

Version 20.4.0, 2020-04-06: updated Datastructures and Indexes

# Inhaltsverzeichnis

This document describes some of the helper tools used within this course.

# Docker

Instead of installing the tools as native packages for your operating system, we can make use of Docker images for the different services we use.

To get a short introduction into Docker including some basic examples, check out these slides of the Cloud Solutions lecture.

## Installation

### Linux

Check the Linux installation page on how to install docker community edition after having checked the prerequisites.

### Mac

Check the Mac installation page on how to install docker community edition after having checked the prerequisites.

> Docker for Mac requires OS X El Capitan 10.11 or newer macOS release running on a 2010 or newer Mac, with Intel's hardware support for MMU virtualization. The app will run on 10.10.3 Yosemite, but with limited support. Please see What to know before you install for a full explanation and list of prerequisites.

> If you want to share files outside the `/Users` directory with your container, you have to add these directories in the docker ***->Preferences->File sharing***.

### Windows

Check the Windows installation page on how to install docker community edition after having checked the prerequisites.

> Docker for Windows requires 64bit Windows 10 Pro and Microsoft Hyper-V. Please see What to know before you install for a full list of prerequisites.

> - If you want to share directories with your container, you have to add these directories in the docker ***->Settings->Shared Drives***.
> - If you get errors starting your container when using `--publish` run option, you might need to disable **Experimental Features** in the docker ***->Settings ->Daemon***. (this issue)

## Special note on how to use X client containers (linux, mac)

In order to access your X server from within a docker X client, follow these steps based on your platform.

#### Linux

1. Pass your `DISPLAY` variable when creating/running the container

   ```
   --env DISPLAY=unix$DISPLAY
   ```

2. Map your X11 socket as volume into the container

   ```
   --volume /tmp/.X11-unix:/tmp/.X11-unix
   ```

3. Allow access to your X11 server from local clients

   ```
   # allow local clients only
   xhost local:
   # or allow all
   xhost +
   ```

#### Mac

According to this issue comment, the following steps are needed.

1. Start `socat` to expose local xquartz socket on a TCP port

   ```
   socat TCP-LISTEN:6000,reuseaddr,fork UNIX-CLIENT:\"$DISPLAY\"
   ```

   This is insecure on public networks, add `bind`, `su` and `range` options to `socat` to limit access.

2. Pass the display to container (assuming virtualbox host is available on 192.168.59.3):

   ```
   --env DISPLAY=192.168.59.3:0
   ```

#### Windows (unverified!)

The following procedure, as stated here might theoretically work but isn't verified yet.

1. Install XMing
2. Start XMing like this, `-ac` disables access control checks:

```
XMing -ac -multiwindow -clipboard
```

3. Pass the display to the container:

   ```
   --env DISPLAY=192.168.59.3:0.0
   ```

   Replace the above IP address with your hosts current address.

## Basic commands and images

*show available images*

```
docker image ls
```

*show running and dead containers*

```
docker container ls -a
```

**PostgreSQL + PostGIS + cstore_fdw**

This image contains the latest postgres server including the cli tool `psql`. Additionally, postgis and `cstore_fdw` extensions are installed and ready to use.

*Build image*

```
docker-compose
  --file Helpers/docker-compose.yml
  --project-name dataeng
  --project-directory Helpers/
  build postgres
```

*Run the server in the background and make it accessible from the host at `localhost:5432`:*

```
docker-compose
  --file Helpers/docker-compose.yml
  --project-name dataeng
  --project-directory .
  up -d                    ①
  postgres
```

① `-d` runs the service in the background

*Use the `psql` cli tool to access the server*

```
docker-compose
  --file Helpers/docker-compose.yml
  --project-name dataeng
  --project-directory .                     ①
  run postgres                              ②
  psql                                      ③
    -h postgres                             ④
    -U postgres                             ⑤
```

① Bind mount the specified directory to `/src` in the container.

② Run service `postgres` interactively in the foreground.

③ Instead of the default command, execute `psql …`.

④ Hostname of the `postgres` database.

⑤ Username of the dba, usually `postgres`.

*Execute (p)sql commands from a file which resides in your local file system.*

To access files from the local file system, we need to map it using the `--volume` option. The service entry in the compose file already maps the `project-directory` as `/src` into the container.

```
docker-compose
  --file Helpers/docker-compose.yml
  --project-name dataeng
  --project-directory .
  run postgres
  psql
    -h postgres
    -U postgres
    -v ON_ERROR_STOP=1                      ①
    -f /src/Databases/bank/0_runAllScripts.sql  ②
```

① Abort the script in case of errors. Default is to silently continue running the script which might lead to a non-working database in case of errors.

② Full path name to the SQL script file to import. This path needs to be present within the container.

### Eclipse-Java-EE

To update or manually build an eclipse-jee image use the following command.

*Build an updated image*

```
docker-compose
  --file Helpers/docker-compose.yml
  --project-name dataeng
  --project-directory Helpers/
  build eclipse-jee
```

*Run eclipse in the background*

```
xhost local:
docker-compose
  --file Helpers/docker-compose.yml
  --project-name dataeng
  --project-directory .
  up -d eclipse-jee
```

If you place your workspace directory below the default folder `/nobody`, changes will automatically be persisted between sessions. The corresponding volume is probably named `dataeng_eclipse-jee-data`. Plugins and updates to them will also be stored in that volume when updated.

The following tools will help you to keep your code clean but are not pre-installed. You can easily find and install them using Eclipse-Marketplace.

*Useful tools*

- EclEmma
- Checkstyle Plug-in
- PMD Plug-in
- FindBugs Feature
- LiClipseText Feature