USING DOCKER FOR MXCUBE DEVELOPMENT AT MAX IV

Antonio Milán Otero

KITS - 2017

OVERVIEW

- What is Docker?
- How does it work?
- How we use it for MxCUBE
- How to create a container
- Services
- Other tools

WHAT IS DOCKER?

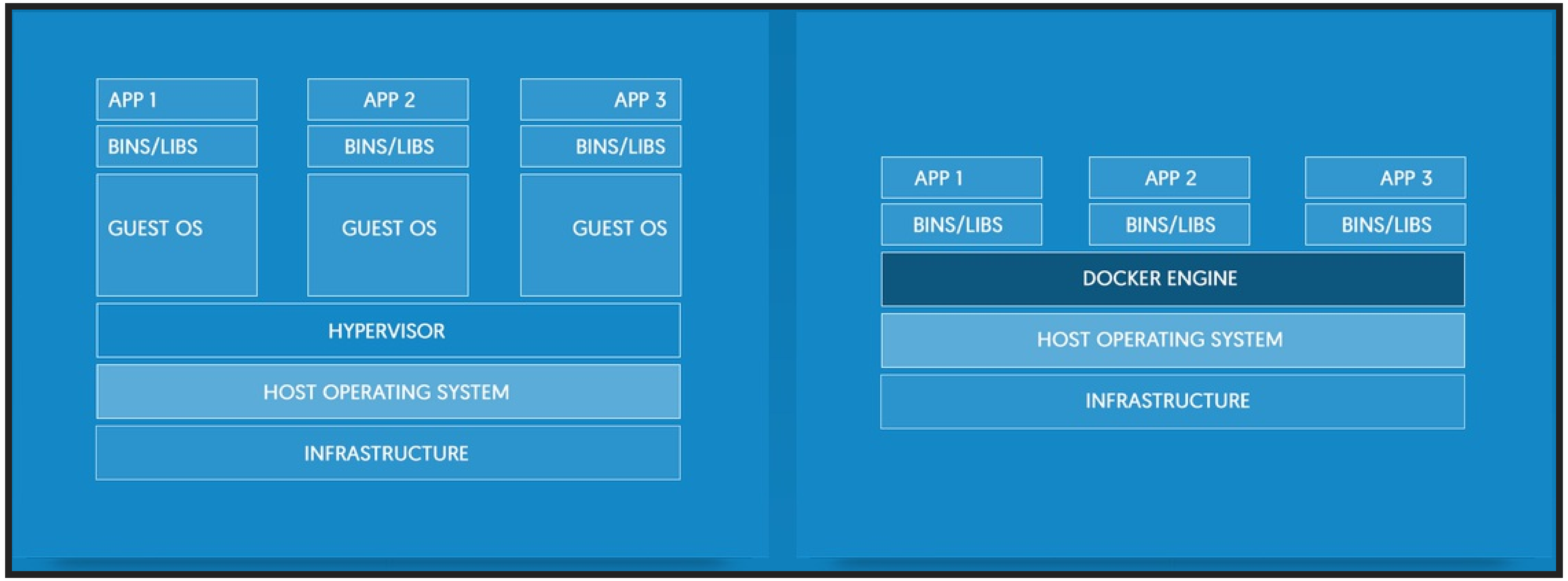
Docker containers wrap a piece of software in a complete filesystem that contains everything needed to run:

- Code
- Runtime
- System tools
- System libraries

This guarantees that **the software will always run the same, regardless of its environment.**

WHAT DOCKER IS NOT

IT'S NOT A VIRTUAL MACHINE



Virtual Machines include: Application, binaries, libraries and the entire guest OS.

Containers include the application and its dependencies.

Run in an **isolated process** in user space on the host operating system.

MAYBE NOW YOU FEEL LIKE ...

Imagine you have an app and its dependencies to deploy.

You put it all together on the container.

Ship it together.

More details on:

http://www.docker.com/what-docker

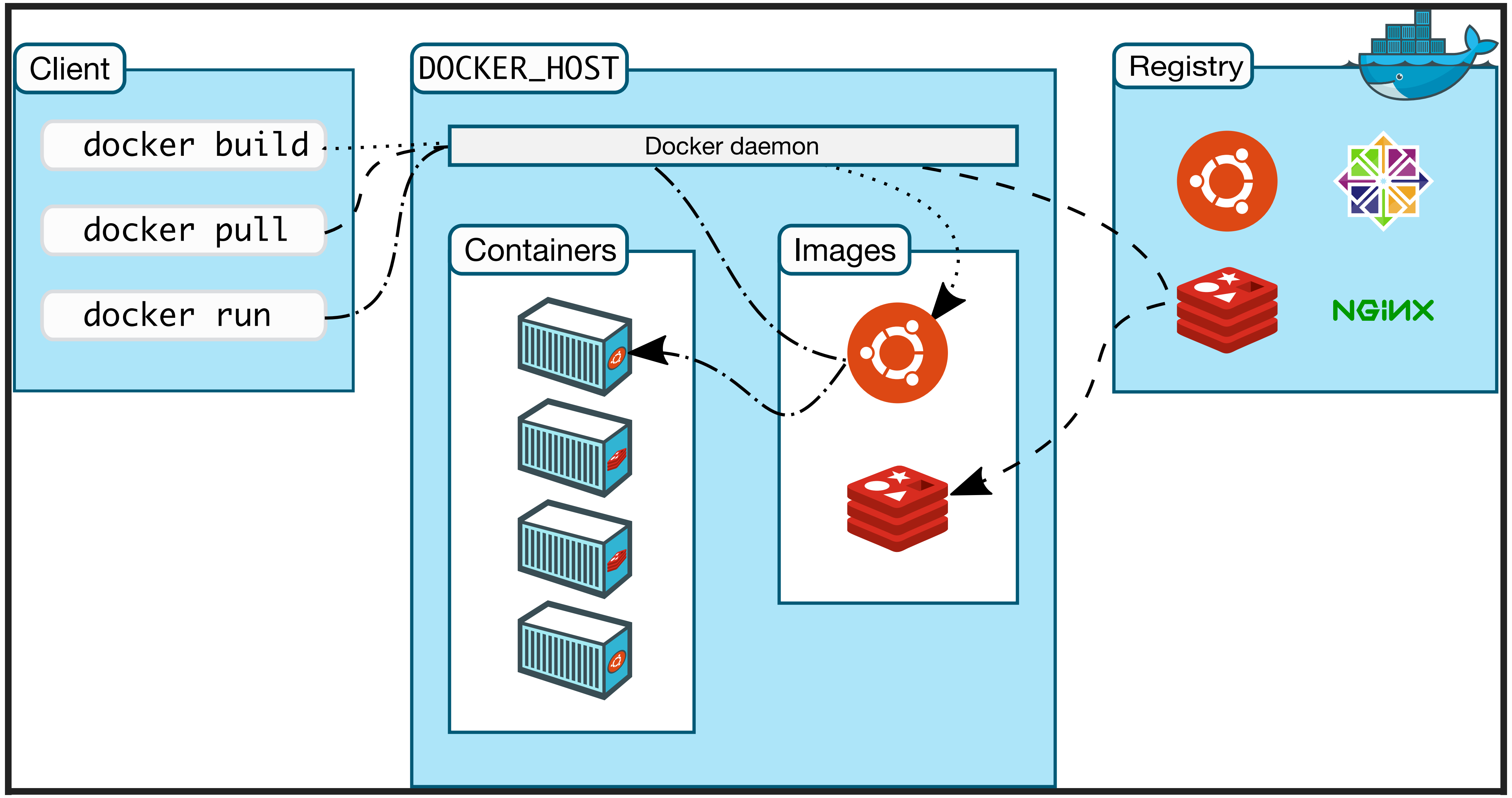
https://docs.docker.com/engine/understanding-docker/

HOW DOES IT WORKS?

3 main components:

- Images
- Containers
- Registries

ARCHITECTURE



HOW WE USED IT FOR MXCUBE

... BUT FIRST LET'S PUT YOU ON CONTEXT ...

In our previous meeting in Hamburg, we ran a workshop to show & help with the MxCUBE 3 installation process.

We wanted to let the people experiment with our application, but at the end we discovered that they were having plenty of problems to install it and run it.

USE CASE 1: DISTRIBUTING A MOCKUP VERSION FOR NEWCOMERS

So, we decided to distribute a VM to make easier for newcomers to test the app.

But why to use a VM when we can use a docker container and simplify the process and reduce the weight?

USE CASE 2: DEVELOPMENT ENVIRONMENT

Docker it's so nice that after playing with it, we started to consider its usage as a development environment.

USE CASE 3: DEPLOYMENT ENVIRONMENT

Also, it will not be so complex to use it for deployment.

But we are not there yet.

MXCUBE 3 OVERVIEW

As you know, MxCUBE 3 is composed by 3 main services:

- Redis
- NPM
- MxCUBE

Good Practice: Split your monolithic application into several microservices.

FIRST THING FIRST: HOW TO BUILD A DOCKER IMAGE. DOCKERFILE

Here we define everything needed in our container to run the app, and also the configuration that has to be done.

```
##############################################################################
# Dockerfile to run MXCuBE web server
##############################################################################

FROM centos:7
MAINTAINER Antonio Milan Otero <antonio.milan_otero.maxiv.lu.se>
```

Then you install whatever is needed in your container.

```
# Install ####
RUN yum install -y curl
RUN curl --silent --location https://rpm.nodesource.com/setup_4.x | bash -

RUN yum install -y gcc-c++ make
# or: yum groupinstall 'Development Tools'

# Install nodejs from epel repository ####
RUN yum install -y nodejs npm

# Install more dependencies ####
RUN yum install -y \
        python-devel \
        openldap-devel \
        lapack-devel \
        zlib-devel \
        libjpeg-turbo-devel \
        libxml2-devel \
        libxslt-devel \
        openssl-devel \
        libgfortran \
        cyrus-sasl-devel
```

Then we get the latest version of the app.

```
                 In the future it will be pointing official releases.
```

```
# Install git ####
RUN yum install -y git

# Get MxCUBE code ####
RUN mkdir /mxcube
WORKDIR /mxcube
RUN git clone https://github.com/mxcube/mxcube3.git --recursive
WORKDIR mxcube3
```

After that, more dependencies are installed

```
# Install EPEL repository ####
RUN yum install -y epel-release
RUN yum makecache # && yum update -y

# Install python pip ####
RUN yum install -y python-pip
RUN yum install -y redis

# Install requirements ####
RUN pip install -r requirements.txt

# Install supervisor
RUN pip install supervisor

# Install npm ####
RUN npm install
RUN npm install fabric
RUN npm install --dev
```

Some configuration now.

```
RUN cp backend_server.js.example backend_server.js

COPY supervisord.conf /etc/supervisor/supervisord.conf

COPY run_mxcube /usr/local/bin/

EXPOSE 8090

CMD ["/usr/bin/supervisord"]
```

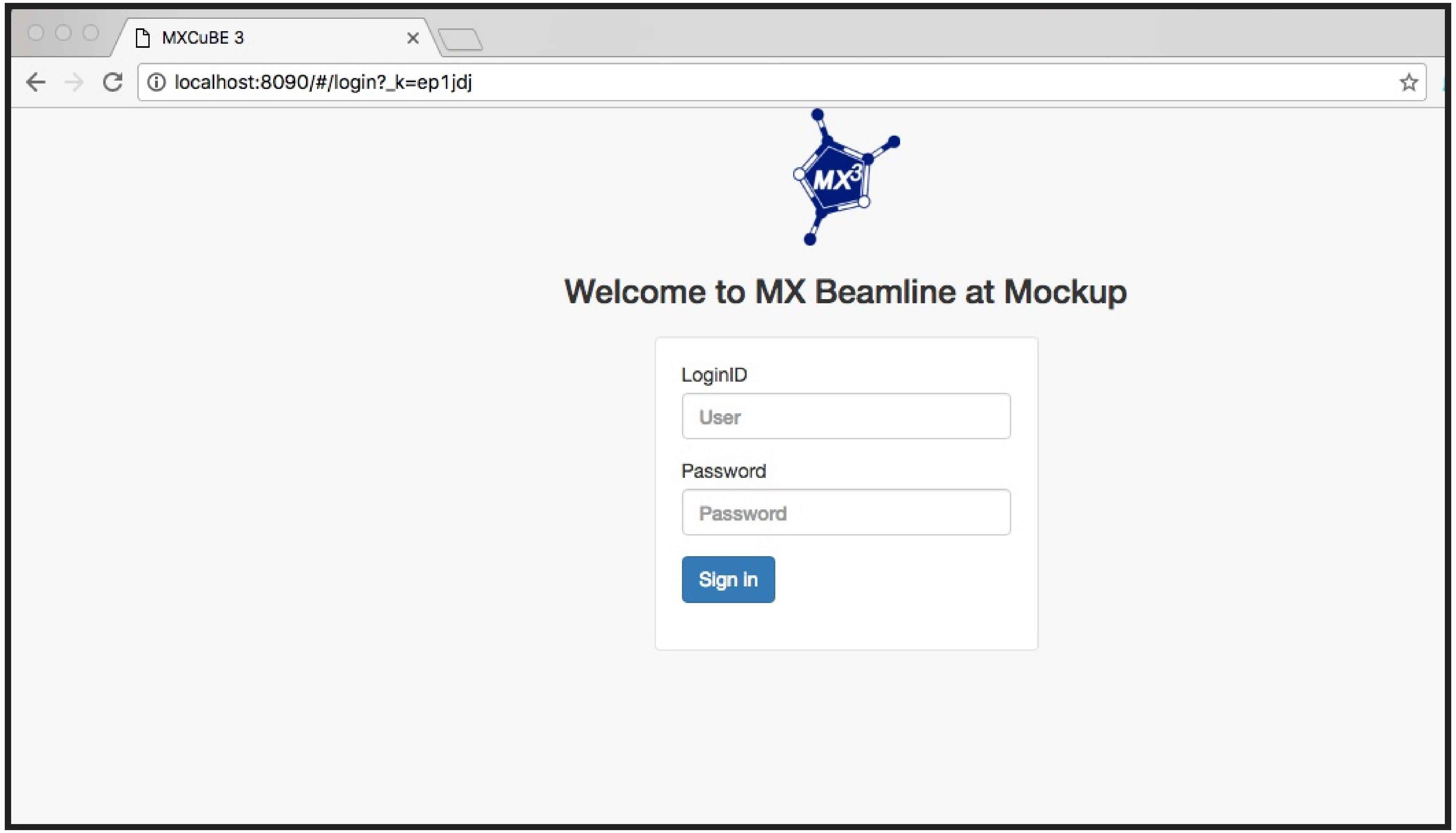
We use supervisor in order to run and monitor all the processes. More details here: http://supervisord.org/

Supervisor configuration example:

```
[supervisord]
nodaemon=true

[program:redis]
command=redis-server

[program:mxcube]
command=python mxcube3-server -r test/HardwareObjectsMockup.xml --log-file mxcube.log

[program:npm]
command=npm start
```

AND RUN IT

```
docker build -t mxcube_web .
docker run -i -p 8090:8090 -t mxcube_web
```



MXCuBE 3

localhost:8090/#/login?_k=ep1jdj

Welcome to MX Beamline at Mockup

LoginID

User

Password

Password

Sign in

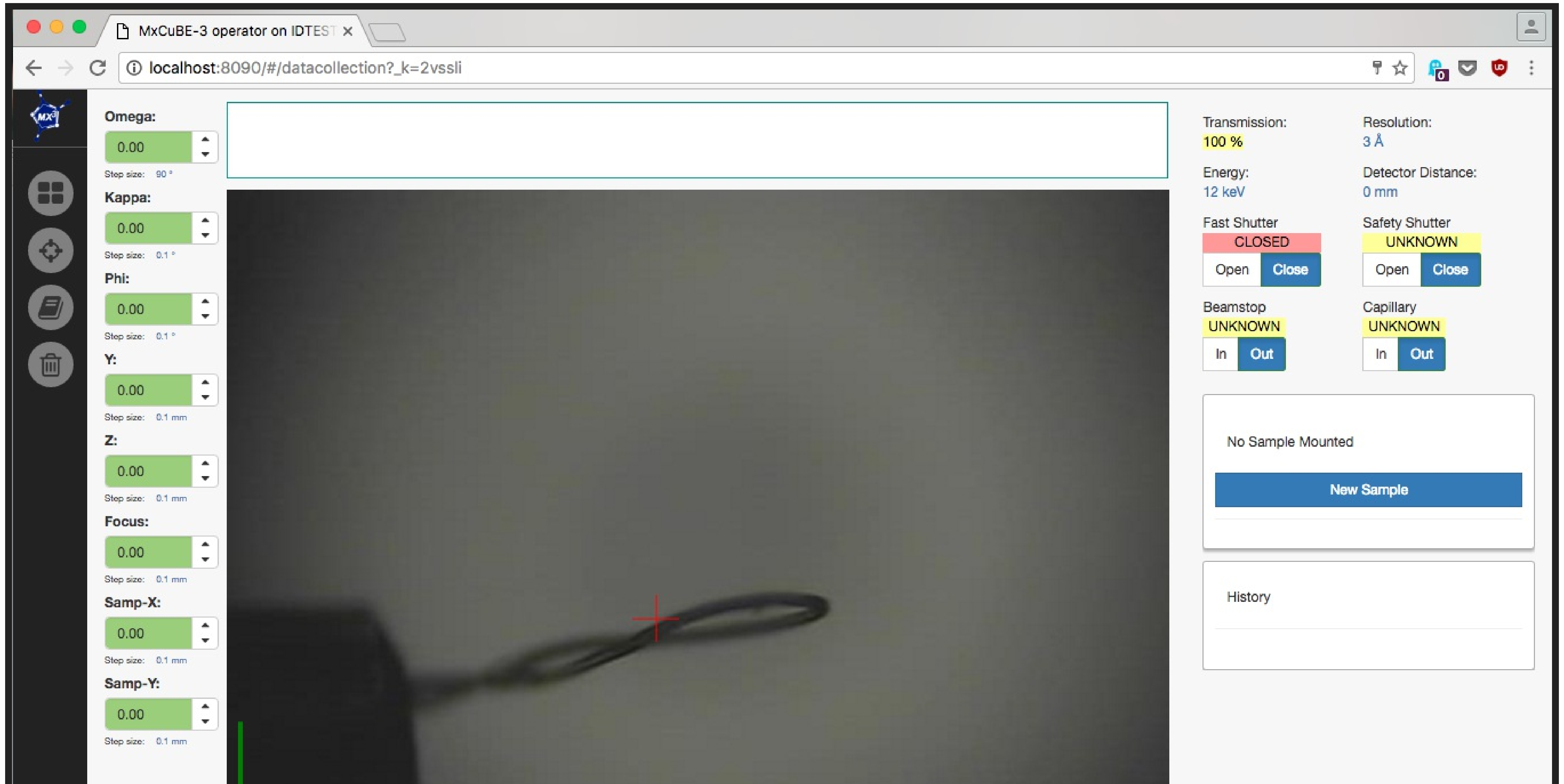
USE CASE 1: DISTRIBUTING A MOCKUP

Actually you don't need to build anything, it's already available in docker hub.

So, execute:

```
docker run -i -p 8090:8090 -t amilan/mxcube_web
```

And enjoy it.



MxCuBE-3 operator on IDTEST

localhost:8090/#/datacollection?_k=2vssli

Omega: 0.00
Step size: 90 °

Kappa: 0.00
Step size: 0.1 °

Phi: 0.00
Step size: 0.1 °

Y: 0.00
Step size: 0.1 mm

Z: 0.00
Step size: 0.1 mm

Focus: 0.00
Step size: 0.1 mm

Samp-X: 0.00
Step size: 0.1 mm

Samp-Y: 0.00
Step size: 0.1 mm

Transmission: 100 %

Resolution: 3 Å

Energy: 12 keV

Detector Distance: 0 mm

Fast Shutter: CLOSED

Safety Shutter: UNKNOWN

Beamstop: UNKNOWN

Capillary: UNKNOWN

No Sample Mounted

History

It's like magic.

USE CASE 2: DEVELOPMENT ENVIRONMENT

You can use it also as a part of your development environment as a test bench.

One way of doing that is:

1. Get the latest image.
2. Clone the mxcube3 repository in your host.
3. Run the container mounting your host folder.
4. Disable Supervisor in your container.
5. Run the services manually.

Looks more difficult than what it is.

1- GET THE LATEST IMAGE

```
docker pull amilan/mxcube_web
```

2- CLONE THE MXCUBE3 REPOSITORY

```
git clone https://github.com/mxcube/mxcube3.git
```

3- RUN THE CONTAINER MOUNTING YOUR HOST FOLDER

```
docker run -i -p 8090:8090 -v /Path/to/your/repo/mxcube3:/mxcube/mxcube3 --name mxcube3 -t amilan/mxcube_web
```

- See details about installing npm in the documentation.

4- DISABLE SUPERVISOR

Edit the file: /etc/supervisor/supervisord.conf and let only this:

```
[supervisord]
nodaemon=true

[program:redis]
command=redis-server

# [program:mxcube]
# command=python mxcube3-server -r test/HardwareObjectsMockup.xml --log-file mxcube.log

# [program:npm]
# command=npm start
```

And then, restart the container:

```
docker restart mxcube3
```

5- RUN THE SERVICES MANUALLY

```
docker exec -it mxcube3 redis-server
docker exec -it mxcube3 python mxcube3-server -r test/HardwareObjectsMockup.xml --log-file mxcube.log
docker exec -it mxcube3 npm start
```

More detailed info here:

https://github.com/amilan/mxcube_web

USE CASE 3: DEPLOYMENT ENVIRONMENT

Not used yet ... maybe one day ...

EXTRA TIME FOR SOME MORE DOCKER STUFF?

If not, jump to slide 21.

DOCKER COMPOSE

A nice tool to orchestrate several docker containers and manage your microservices.

Configurable with a .yml file.

```
version: '2'
          services:
            mxcube:
              build: .
              expose:
                - "8081"
              ports:
                - "8081:8081"
              links:
                - redis
              command: python mxcube3-server -r test/HardwareObjectsMockup.xml --log-file /tmp/mxcube.log
              volumes:
                - ./tmp:/tmp
            redis:
              image: redis
              expose:
                - "6379"
              ports:
                - "6379:6379"
              volumes:
                - ./data:/data
```

And then:

```
docker-compose build
docker-compose up
```

And the magic happens.

Example available in: https://github.com/amilan/mxcube_web

Under the branch: **use_composer**

But wait! It's still under development. Use it at your own risk!

DOCKER REGISTRY

Basically it's like a repository of images.

Docker hub is the official one, but you can have your own one.

It provides:

- Storage for your containers.
- Push and pull commands (for maintenance).
- Notification system (webhooks)
- Able to connect to a Continuous Integration / Delivery system.

ANSIBLE INTEGRATION?

Why not? It will provide:

- Flexibility:
  - Portable playbooks
  - Reproducible environments (using docker, vagrant, etc)
- Auditability: Simple to review and update content in a container.
- Ubiquity: You can manage container environments and also host environments.

DOCKER SWARM

It provides features to orchestrate a cluster of Docker Engines.

Main features:

- Cluster management
- Decentralized design
- Declarative service model
- Scaling
- Desired state reconciliation
- Multi-host networking
- Service discovery
- Load balancing
- Secure by default
- Rolling updates

ANY QUESTIONS?

THANKS FOR YOUR ATTENTION!

https://hub.docker.com/r/amilan/mxcube_web/

https://github.com/amilan/mxcube_web