AMD-01 (102386): Development Environment

Applications for mobile devices & Fall 2020-2021

Jordi Mateo Fornés jordi.mateo@udl.cat

Presentation

- Dr. Jordi Mateo Fornés
- **Office**:
  - Office A.12 (Campus Igualada)
  - Office 3.08 (EPS Lleida)
- **Email**: jordi.mateo@udl.cat
- **Twitter**: https://twitter.com/MatForJordi
- **Github**: https://github/JordiMateoUdL

Course

- **Applications for mobile devices**.
- Grau en Tècniques d'Interacció Digital i de Computació
- Campus Igualada - Escola Politècnica Superior - Universitat de Lleida
- All the code developed in this course can be found in this repository: DAM Course.

Agenda

Agenda

1. HandsOn01 feedback, comments, and discussion.
2. Prepare the development environment and tools
3. Introduction to Falcon
4. DAMCore
5. Production remote environment

HandsOn01

HandsOn01

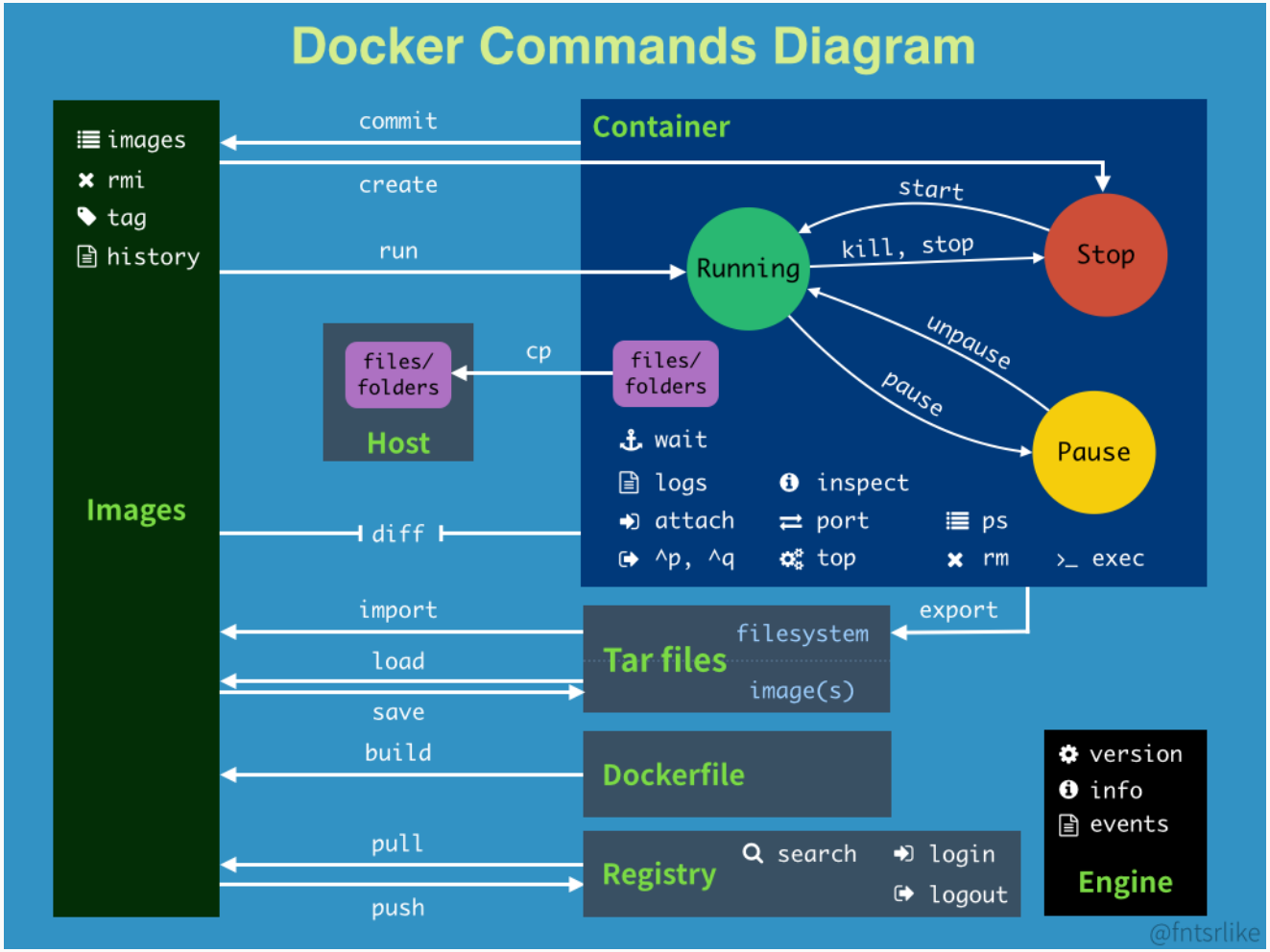
- I expect that this first hands-on was really easy for the majority?
- Do you want/need another similar but remote git tracking, or do you think it will be boring?
- What is a pull request?
- What do you things about this hands-on methodology? Do you like it? Do you want more?

Prepare the development environment and tools

Tools to install

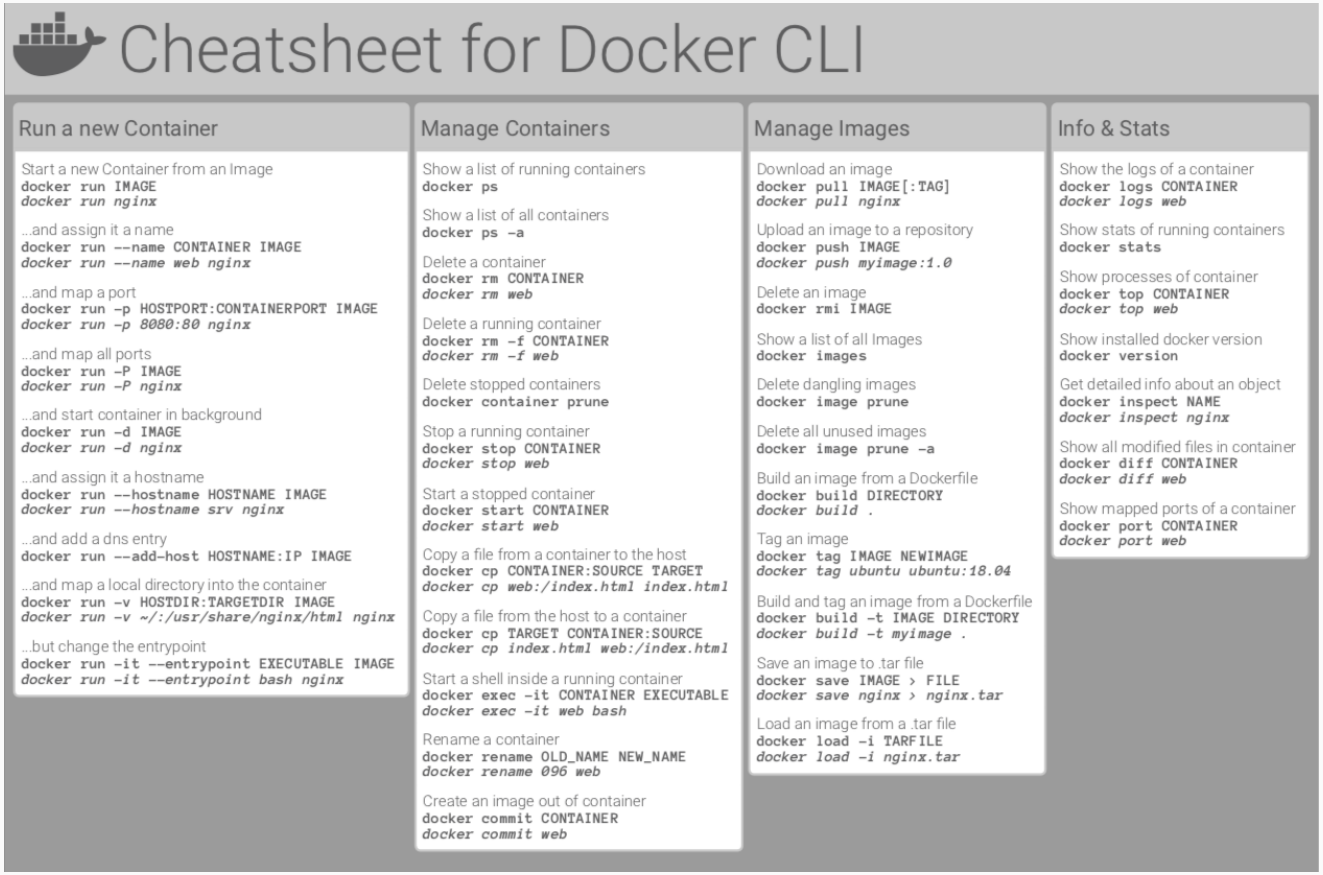
- Windows:
  - https://docs.docker.com/docker-for-windows/install/
- Linux:
  - https://computingforgeeks.com/install-docker-ce-on-linux-systems/
- Mac:
  - https://docs.docker.com/docker-for-mac/install/
- **Compose**: https://docs.docker.com/compose/install/
- **Postman:** https://www.postman.com/downloads/

Docker



Campus Universitari Igualada-UdL

Cheatsheet



Cheatsheet for Docker CLI

Run a new Container

Start a new Container from an Image
```
docker run IMAGE
docker run nginx
```

...and assign it a name
```
docker run --name CONTAINER IMAGE
docker run --name web nginx
```

...and map a port
```
docker run -p HOSTPORT:CONTAINERPORT IMAGE
docker run -p 8080:80 nginx
```

...and map all ports
```
docker run -P IMAGE
docker run -P nginx
```

...and start container in background
```
docker run -d IMAGE
docker run -d nginx
```

...and assign it a hostname
```
docker run --hostname HOSTNAME IMAGE
docker run --hostname srv nginx
```

...and add a dns entry
```
docker run --add-host HOSTNAME:IP IMAGE
```

...and map a local directory into the container
```
docker run -v HOSTDIR:TARGETDIR IMAGE
docker run -v ~/:/usr/share/nginx/html nginx
```

...but change the entrypoint
```
docker run -it --entrypoint EXECUTABLE IMAGE
docker run -it --entrypoint bash nginx
```

Manage Containers

Show a list of running containers
```
docker ps
```

Show a list of all containers
```
docker ps -a
```

Delete a container
```
docker rm CONTAINER
docker rm web
```

Delete a running container
```
docker rm -f CONTAINER
docker rm -f web
```

Delete stopped containers
```
docker container prune
```

Stop a running container
```
docker stop CONTAINER
docker stop web
```

Start a stopped container
```
docker start CONTAINER
docker start web
```

Copy a file from a container to the host
```
docker cp CONTAINER:SOURCE TARGET
docker cp web:/index.html index.html
```

Copy a file from the host to a container
```
docker cp TARGET CONTAINER:SOURCE
docker cp index.html web:/index.html
```

Start a shell inside a running container
```
docker exec -it CONTAINER EXECUTABLE
docker exec -it web bash
```

Rename a container
```
docker rename OLD_NAME NEW_NAME
docker rename 096 web
```

Create an image out of container
```
docker commit CONTAINER
docker commit web
```

Manage Images

Download an image
```
docker pull IMAGE[:TAG]
docker pull nginx
```

Upload an image to a repository
```
docker push IMAGE
docker push myimage:1.0
```

Delete an image
```
docker rmi IMAGE
```

Show a list of all Images
```
docker images
```

Delete dangling images
```
docker image prune
```

Delete all unused images
```
docker image prune -a
```

Build an image from a Dockerfile
```
docker build DIRECTORY
docker build .
```

Tag an image
```
docker tag IMAGE NEWIMAGE
docker tag ubuntu ubuntu:18.04
```

Build and tag an image from a Dockerfile
```
docker build -t IMAGE DIRECTORY
docker build -t myimage .
```

Save an image to .tar file
```
docker save IMAGE > FILE
docker save nginx > nginx.tar
```

Load an image from a .tar file
```
docker load -i TARFILE
docker load -i nginx.tar
```

Info & Stats

Show the logs of a container
```
docker logs CONTAINER
docker logs web
```

Show stats of running containers
```
docker stats
```

Show processes of container
```
docker top CONTAINER
docker top web
```

Show installed docker version
```
docker version
```

Get detailed info about an object
```
docker inspect NAME
docker inspect nginx
```

Show all modified files in container
```
docker diff CONTAINER
docker diff web
```

Show mapped ports of a container
```
docker port CONTAINER
docker port web
```

Docker-Compose (1)

```
# Starts existing containers for a service.
docker-compose start
# Stops running containers without removing them.
docker-compose stop
# Pauses running containers of a service.
docker-compose pause
# Unpauses paused containers of a service.
docker-compose unpause
# Lists containers.
docker-compose ps
```

Docker-Compose (2)

```
# Builds, (re)creates, starts, and attaches
# to containers for a service.
docker-compose up (-d) (--build)
# Stops containers and removes containers,
# networks, volumes, and images created by up.
docker-compose down
# Logs
docker-compose logs
```

Falcon

Introduction

Definition minimalist Python web API framework for building reliable app backends and microservices. (“Falcon Official Webpage,” n.d.)

The Falcon web framework encourages the **REST** architectural style. Resource classes implement HTTP method handlers that resolve requests and perform state transitions.

Features (1)

- Highly-optimized, extensible code base
- Intuitive routing via URI templates and REST-inspired resource classes
- Easy access to headers and bodies through request and response classes
- DRY request processing via middleware components and hooks
- Idiomatic HTTP error responses
- Straightforward exception handling
- Snappy unit testing through WSGI helpers and mocks
- CPython 2.6-2.7 and 3.4+, or PyPy 2.7 and 3.5+
- Cython support for an extra speed boost under CPython

People feedback

- We have been using Falcon as a replacement for [framework] and we simply love the performance (three times faster) and code base size (easily half of our original [framework] code).
- Falcon looks great so far. I hacked together a quick test for a tiny server of mine and was ~40% faster with only 20 minutes of work." "Falcon is rock solid and it's fast.
- I'm loving #falconframework! Super clean and simple, I finally have the speed and flexibility I need!
- I feel like I'm just talking HTTP at last, with nothing in the middle. Falcon seems like the requests of backend.
- The source code for Falcon is so good, I almost prefer it to documentation. It basically can't be wrong.
- What other framework has integrated support for 786 TRY IT NOW ?

REST architectural style

```
class QuoteResource:
    def on_get(self, req, resp):
        """Handles GET requests"""
        quote = {
            'quote': (
                "I've always been more interested in "
                "the future than in the past."
            ),
            'author': 'Grace Hopper'
        }
        resp.media = quote
api = falcon.API()
api.add_route('/quote', QuoteResource())
```

Middleware and Hooks

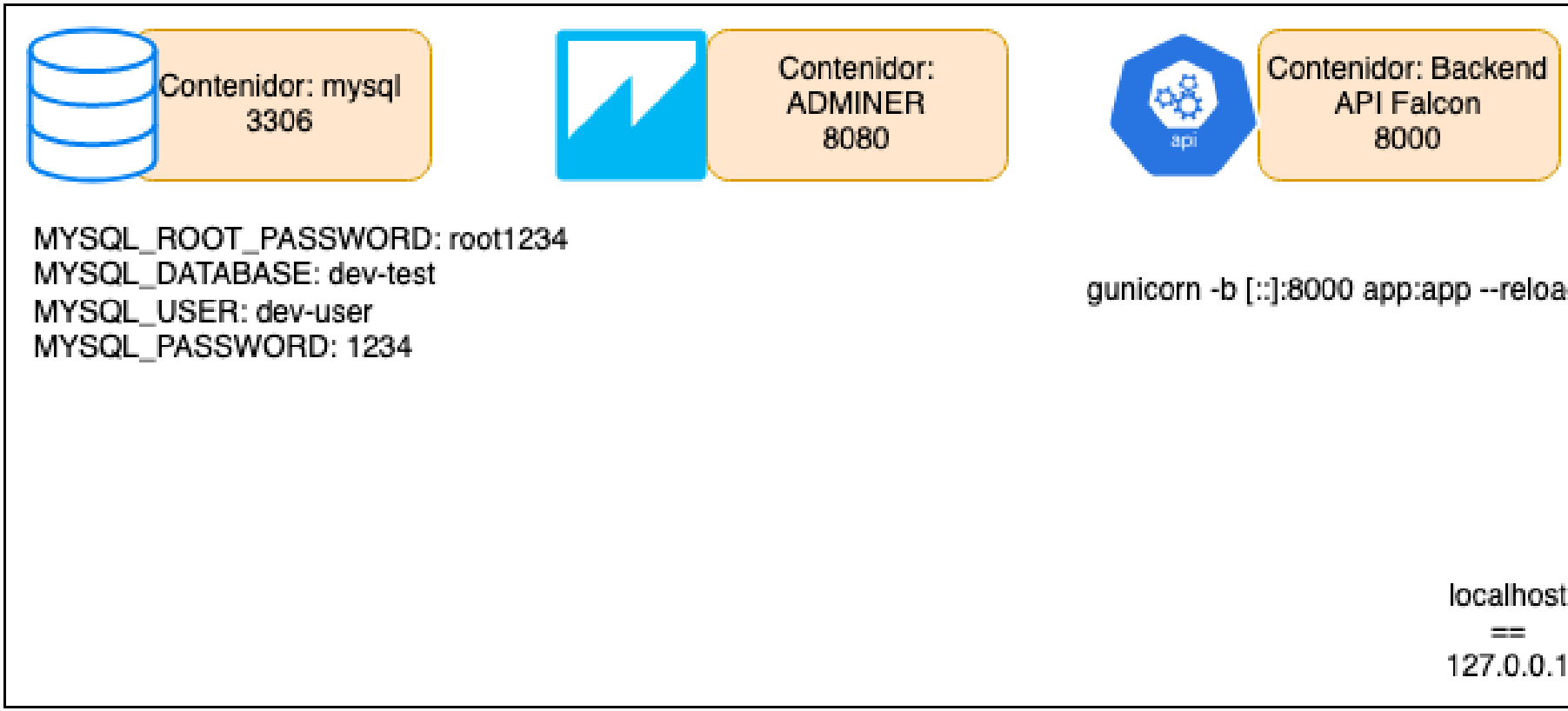
Falcon supports before and after **hooks**. You install a hook simply by applying one of the decorators below, either to an individual responder or to an entire resource.

Middleware components provide a way to execute logic before the framework routes each request, after each request is routed but before the target responder is called, or just before the response is returned for each request. Components are registered with the middleware kwarg when instantiating Falcon's API class.

Unlike hooks, middleware methods apply globally to the entire API.

DAMCore

Overview


Contenidor: mysql
3306
Contenidor:
ADMINER
8080
Contenidor: Backend
API Falcon
8000
MYSQL_ROOT_PASSWORD: root1234
MYSQL_DATABASE: dev-test
MYSQL_USER: dev-user
MYSQL_PASSWORD: 1234
gunicorn -b [::]:8000 app:app --reload
localhost
==
127.0.0.1

Workflow

- **Project**:
  - Android folder
  - DamCore folder
  - Documentation
  - static folder
  - .gitignore (static folder...)
- Integrate this backend to the project: Github.

Let us see the code and explore the repo.

Production remote environment

Production remote environment

Servers:

- Group 1: 192.168.101.84
- Group 2: 192.168.101.85
- Group 3: 192.168.101.86

Install sftp (filezila (gui) or cmd) ssh(putt (windows) or cmd)

That's all folks

www — jordimateofornes.com

github — github.com/JordiMateoUdL

twitter — @MatForJordi

gdc — Distributed computation group

References:

"Falcon Official Webpage." n.d. https://falcon.readthedocs.io/en/stable/api/middleware.html.