Westfälische
Wilhelms-Universität
Münster

Easily deploy Opencast with Docker

eLectures at the University of Münster

living.knowledge
WWU Münster

Jan Koppe,
Matthias Neugebauer

March 16, 2016

About us

jan.koppe@wwu.de
GitHub: JanKoppe

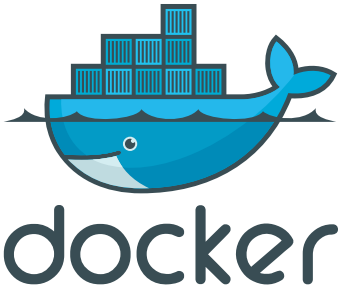
matthias.neugebauer@wwu.de
GitHub: mtneug

eLectures Project at the University of Münster

- Goal
  - 20 rooms until the end of 2017
  - 3 rooms in this summer term
- Capture Agents
  - Currently Galicaster Pro
  - Custom built
- Server
  - Opencast
  - Distributed setup
  - VM infrastructure of University
  - Docker

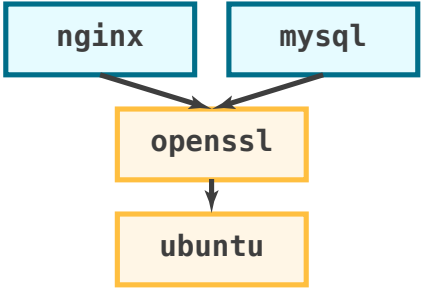
Docker

- Linux Kernel offers operating-system / container virtualization (cgroups, namespaces)
- Lightweight isolated environments using the same kernel
- Features easily accessible with Docker
- Aim: Easy and reliable packaging, distributing and running of applications (systems)
- Client-Server architecture



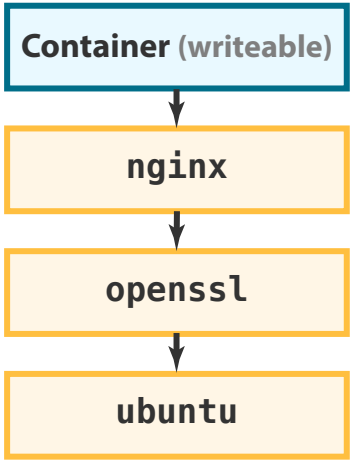
Docker Images

- Read-only layered filesystem + Metadata
- Packaging of applications
- Distributing via Docker Registries
  (e.g. Docker Hub)



Docker Container

- Isolated process(-es)
  filesystem, network, processes, user, ...
- Based on Docker Image
- New writeable filesystem layer added



Creation of Docker Images

1. Manually commit a container
2. Automated via `Dockerfile`

```
1  FROM debian:jessie
2  MAINTAINER Some One <me@example.org>
3
4  RUN apt-get update \
5   && DEBIAN_FRONTEND=noninteractive apt-get install -y --no-install-
       recommends icecast2 \
6   && rm -rf /var/lib/apt/lists/*
7  COPY icecast.xml /etc/icecast2/icecast.xml
8  RUN chown icecast2:icecast /etc/icecast2/icecast.xml
9
10 USER icecast2
11
12 EXPOSE 8000
13 CMD ["icecast2", "-c", "/etc/icecast2/icecast.xml"]
```

Features

- Container lifecycle: created, started, paused, stopped
- Mount files/folders via volumes
- Networking
  - Creation of networks
  - Port forwarding
- Plugin system for volumes and networks
- Control over resources (memory, CPU, devices, ...)
- Many accompanying applications
  - Orchestrator
  - Private/Public Cloud
  - Monitoring
  - ...

Advantages

- Fast, lightweight virtualization
- Reproducible running behavior
  Development › Testing (CI/CD) › Production
- All applications are deployed in the same way
- Focus on applications not machines → clustering
- Multiple versions running side by side
  - Dependencies
  - Updates

Opencast Image

- Opensource under ECL-2.0
- **Contribution welcome!**

- git.io/docker-opencast
- hub.docker.com/r/learnweb/opencast/

Opencast Image Dockerfiles

- Based on `maven:3-jdk8` › `debian:jessie`
- Compiled from source
- Multiple tags:
  `allinone` `admin` `worker` `presentation` + version
- Auto-configuration via environment variables
- `docker-compose` examples

Example use cases

Containerizing Opencast can be useful in various situations:

- Lower the entry barrier
- Development and Testing
- Fast and consistent deployment
- Outsource Opencast hosting
- Granular resource control

Lower the entry barrier

- How to compile?
- What distribution to choose?
- How to configure Opencast?
- What other services/dependencies do I need?
- Documentation up to date?

With Docker simply run

```
$ docker-compose -p [name] -f [example.yml] up
```

Demo Time!

Development and Testing

While developing Opencast itself

- Consistent environment
- Reproducibility (of bugs)
- Run multiple versions or distributions in parallel
- Basis for Continuous Integration with automatic testing (not yet)
- Feature Instances

Fast and consistent deployment

- Enable fast, reliable, dynamic scaling and updating
- With VMs
  - Clone VM / provision everything again
  - Change keys, hostname, ...
  - Error prone → security
- With Docker
  - Install Docker
  - Start container
  - Test new versions before deploying
- Updates
  - Pull new images
  - Create new container

Outsource Opencast hosting

- Container as a Service (CaaS)
  Google, AWS, Joyent, tutum, ...
- Small deployments
- Quick scaling possible
  Dynamically spawn workers to process recordings

Granular resource control

- `cgroups` gives control over resources
  - Memory
  - CPU
- Run Opencast worker on capture agents (leverage unused resources)
- Doesn't get in the way of Capture Agent processes
- ...maybe dockerize your CA Software, too?

About us

jan.koppe@wwu.de
GitHub: JanKoppe

matthias.neugebauer@wwu.de
GitHub: mtneug

Opencast Image

- Opensource under ECL-2.0
- **Contribution welcome!**

- git.io/docker-opencast
- hub.docker.com/r/learnweb/opencast/
- git.io/docker-opencast-presentation

Questions?