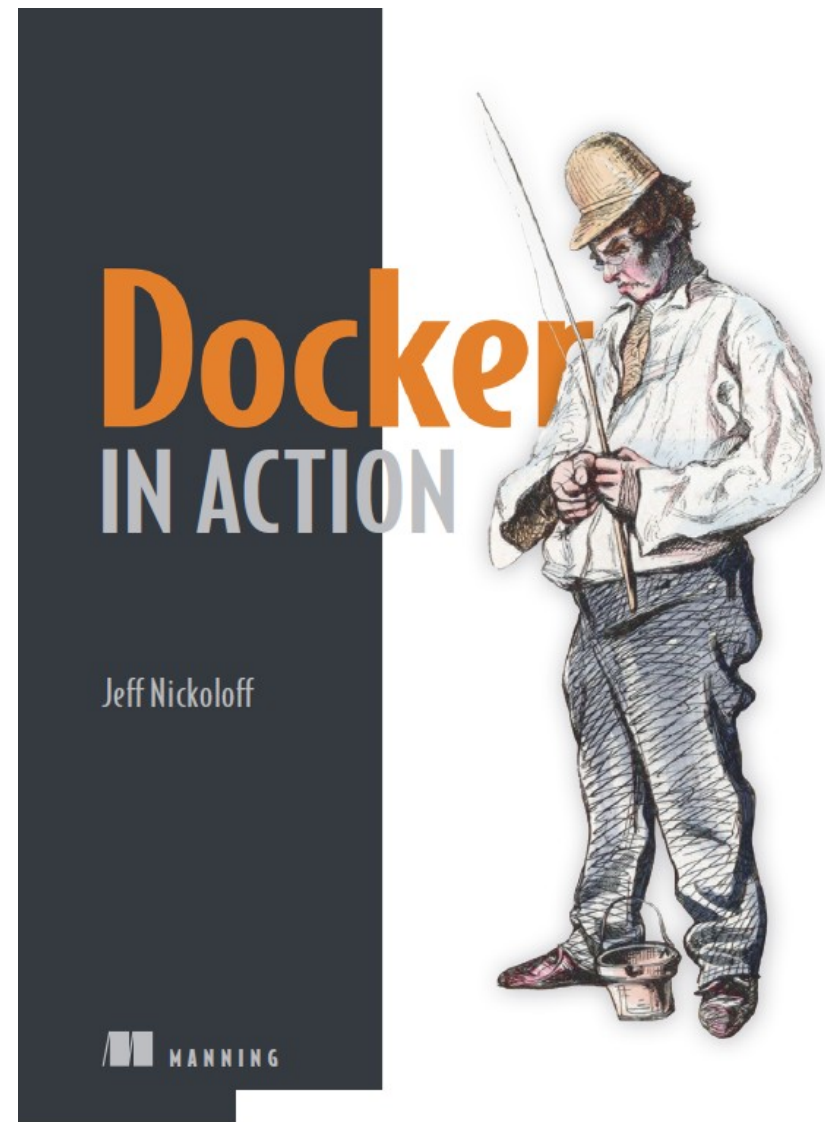
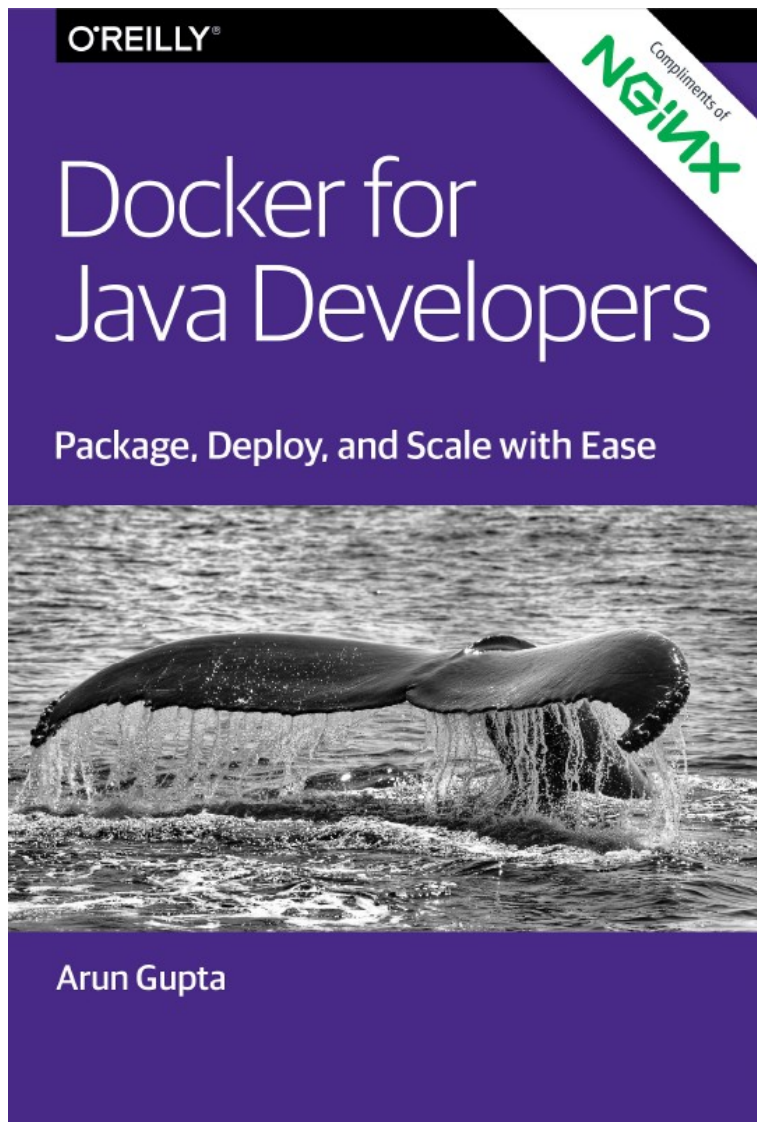


Docker und Java

Container für die Java-Entwicklung



- Die in diesem Seminar verwendete Werkzeuge und Frameworks sind Open Source
 - LPGL Lizenzmodell
- Dies ist ein technisches Seminar mit Übungsanteil
 - Damit werden die Inhalte durch Übungen vertieft und verinnerlicht
 - Musterbeispiele werden zur Verfügung gestellt
 - Diese können am Ende des Seminars als ZIP-Datei kopiert werden
 - USB-Stick oder ähnliches
- Dokumentation und Ressourcen stehen auch im Internet zur Verfügung
 - Insbesondere die API-Dokumentation

Kapitel 1: Einführung	5
Kapitel 2: Docker im Detail	35
Kapitel 3: Java und Docker	48
Kapitel 4: System-Werkzeuge und Tools	76
Kapitel 5: Java Enterprise Anwendungen	86

1

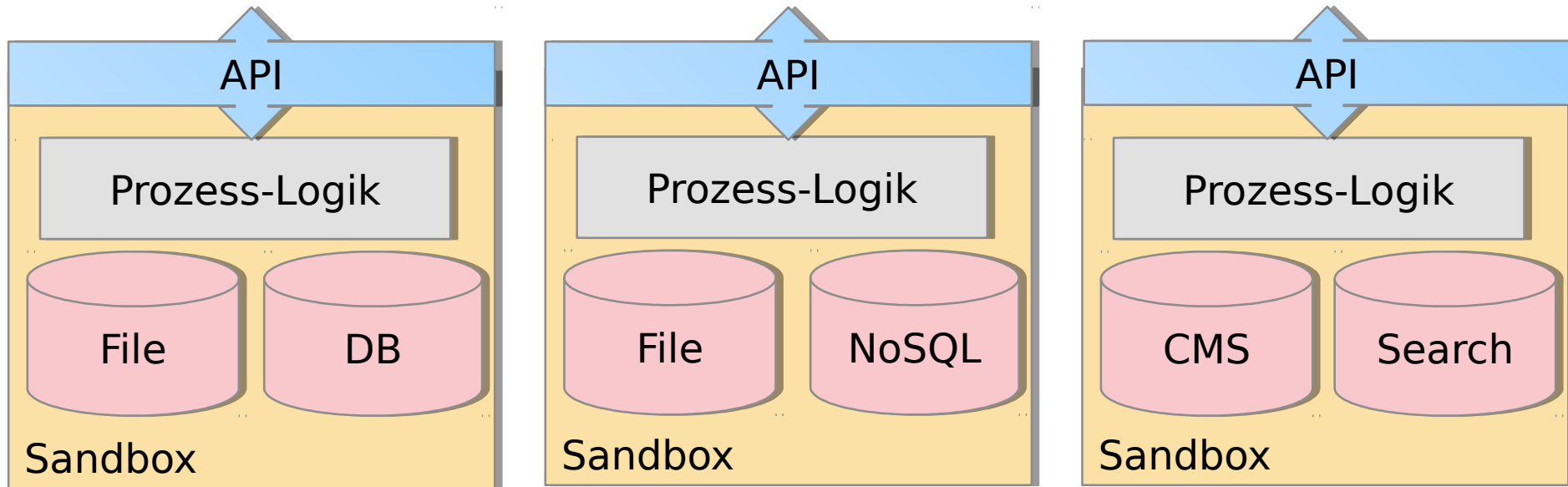
EINFÜHRUNG

1.1

AUSGANGSSITUATION

- Wir benötigen:
 - Kapselung
 - Öffentliche „API“
 - Rein interne Implementierung
 - Vermeidung von Redundanzen
 - Wartbarkeit
 - Dependency Management
 - Statisch determinierte Abhängigkeiten oder
 - lose gekoppelte Systeme
- Diese Aufgabenstellung sieht nach Architektur-Vorgaben für die Programmentwicklung aus!
 - Kurzversion einer Service-Definition

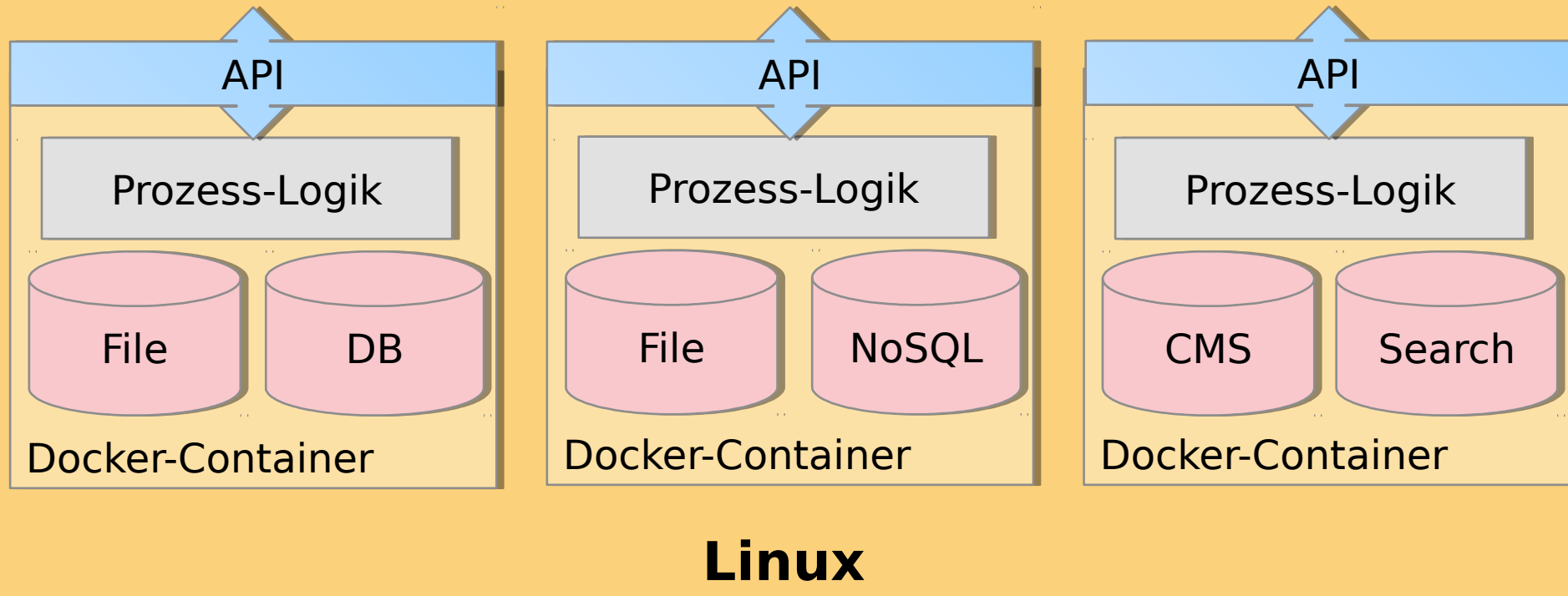
Gekapselte Services



- PID namespace
 - Process identifiers und Capabilities
- UTS namespace
 - Host und Domain Name
- MNT namespace
 - File System
- IPC namespace
 - Process Communication über Shared Memory
- NET namespace
 - Network Access
- USR namespace
 - User names und Identifiers
- chroot()
 - Lokation des File System Root
- cgroups
 - Schützen von Ressource

- Die Implementierung einer Linux-Sandbox ist aufwändig und komplex
- Docker ist ein Framework, das eine fertig konfigurierte und implementierte Sandbox-Lösung zur Verfügung stellt.
 - Fehler in der Sandbox: Bug im Docker-Produkt
- Docker ist mittlerweile auch für Windows erhältlich
 - Ab Windows 10 nativ
 - Benutzt Hyper-V

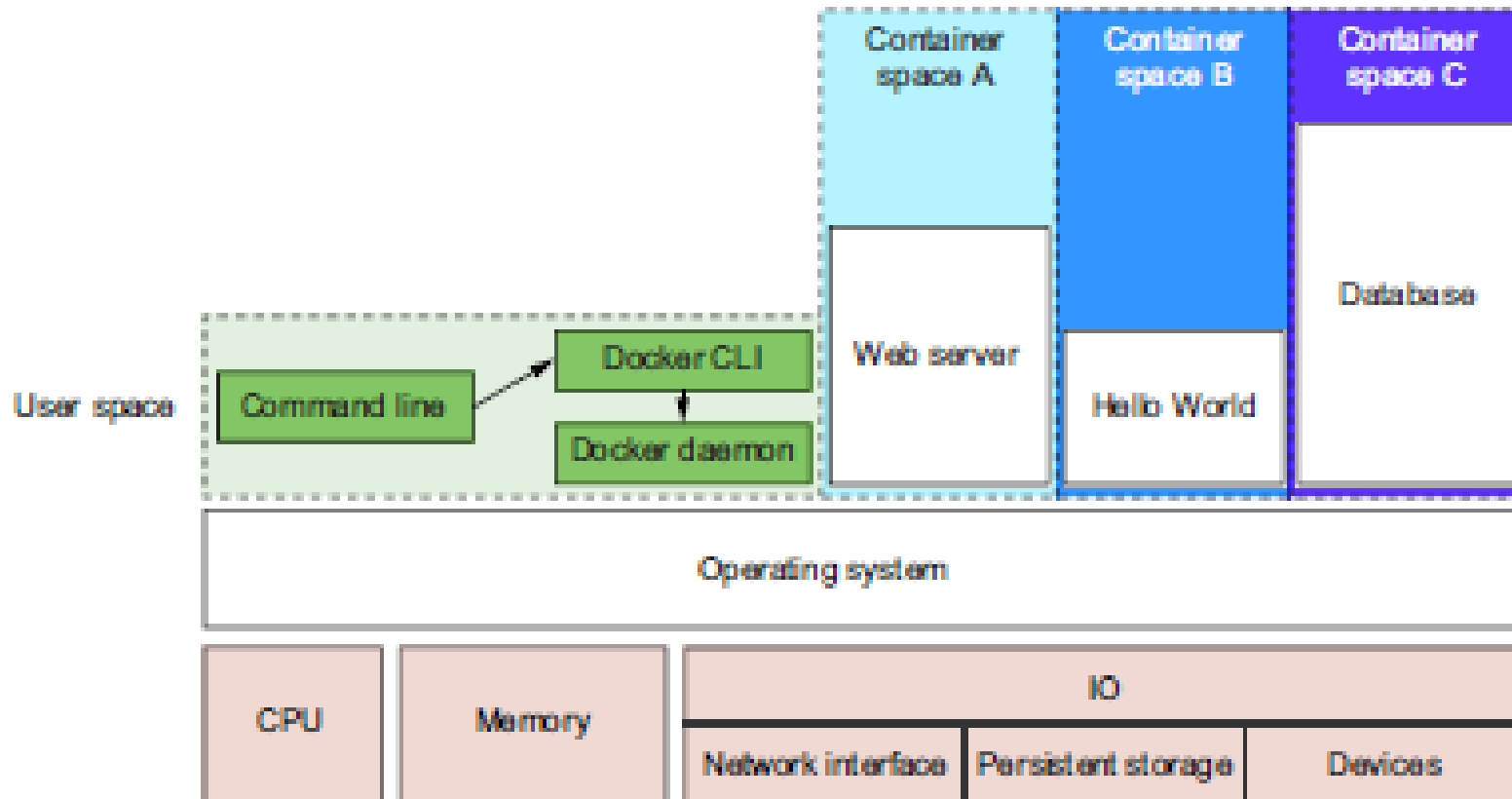
Gekapselte Services mit Docker



- Der Docker selbst stellt keine hochwertigen Protokolle zur Verfügung
 - SOAP
 - REST
 - Java RMI
- Es stehen zur Verfügung
 - Netzwerk-Sockets
 - Server-Implementierungen
 - Client-Zugriffe aus dem Container heraus
 - Dateisystem

- "Ein Container ist eine (modifizierte) Laufzeitumgebung, die der darin laufenden Anwendung den Zugriff auf geschützte Ressourcen unmöglich macht."
 - Unterschied zu Virtualisierung: Hier wird eine komplette Hardware durch eine "Virtuelle Maschine" abstrahiert
- Docker-Container laufen direkt ohne weitere Emulation auf dem Linux-Kernel
 - dabei werden etablierte Linux-Features wie namespaces und cgroups benutzt

Container-Isolation



1.2

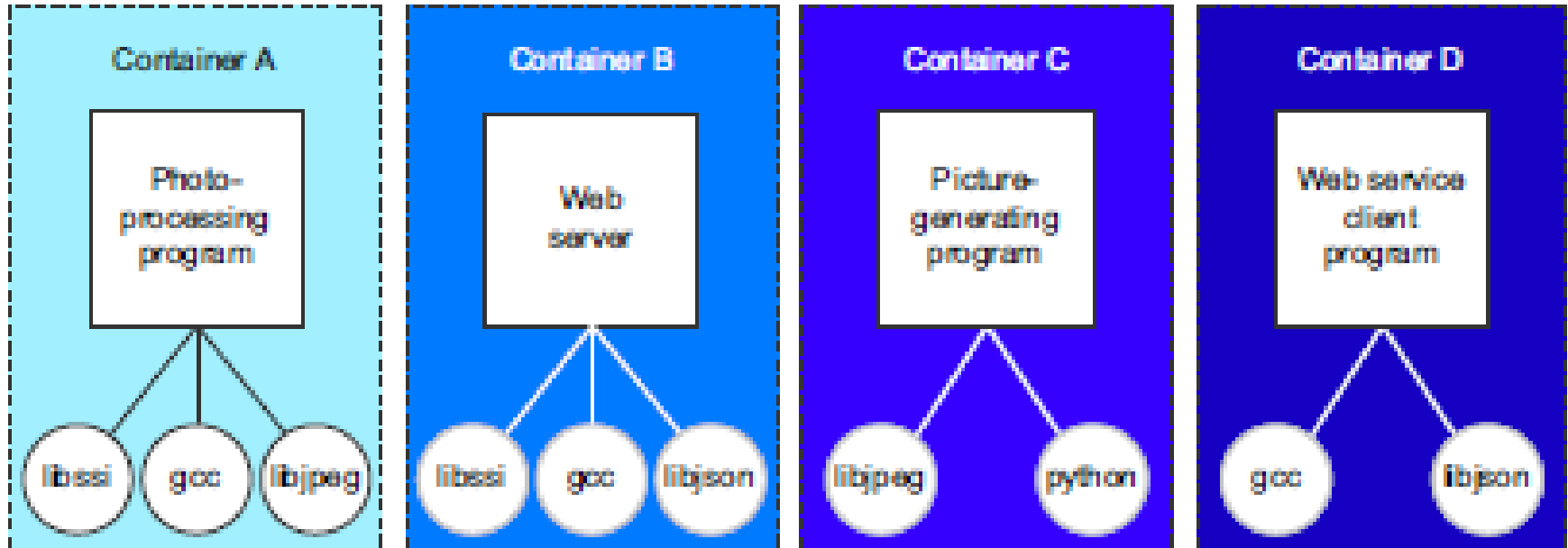
BESTANDTEILE VON DOCKER

- Docker Command Line
 - Dieser Befehl wird vom Docker-Anwender direkt benutzt
- Docker Host
 - Eine Maschine mit installierter Docker Umgebung
 - Docker Dämon/Engine
 - Lokale Registry
- Docker Dämon/Engine
 - Über den Dämon werden die Docker-Container gestartet
 - Technisch gesprochen sind die Container Child-Prozesse des Dämons
- Remote Services
 - Registries für Docker Images
 - DockerHub

- In der Objekt-orientierten Programmierung werden Datenstrukturen und Verhalten in Objekten gekapselt
 - Ein Container kapselt einen kompletten Service samt Infrastruktur
- Objekte kommunizieren miteinander über ein Interface
 - Container kommunizieren über Netzwerk und Streams
- Eine OOP-Anwendung wird durch ein Context&Dependency Injection –Framework dynamisch aufgebaut
 - Docker-Container definieren Dependencies
- Eine OOP-Anwendung wird durch einen Build-Prozess erzeugt
 - Docker-Images werden durch eine Konfigurationsdatei definiert und mit Docker-Kommandos erzeugt

- Microservices
 - Wohl-definiertes API
 - hier nicht relevant
 - Netzwerk
 - Isolation der internen Implementierung
 - Bereitstellung aller benötigten Ressourcen
 - Interne Ressourcen wie Datenbanken
 - Dependencies auf andere Services
- Docker passt!

Microservices mit Docker-Containern



1.3

INSTALLATION

- <https://docs.docker.com/engine/installation/>
- Beispiel:
 - SUSE Linux Enterprise
 - Windows 10 Enterprise mit Hyper-V

- Download der passenden Distribution von docker.com
- Anschließend ist Docker auf dem System aktiv
 - Docker CLI
 - Docker Dämon
 - Dieser muss eventuell noch manuell gestartet werden
- Vorsicht: Der Zugriff auf den Docker-Dämon benötigt Berechtigungen
 - `sudo`
 - Hinzufügen des Benutzers zur `docker`-Gruppe
 - Diese wurde während der Installation automatisch eingerichtet

- **Hinzufügen des Docker-Repositories**
 - `sudo zypper addrepo
https://yum.dockerproject.org/repo/main/opensuse/13.2/
docker-main`
- **Refresh des SUSE Package Managers**
 - `sudo zypper refresh`
- **Die eigentliche Installation**
 - `sudo zypper install docker-engine`
- **Auf Grund eines kleinen Problems mit den Berechtigungen muss der Docker-Service manuell gestartet werden**
 - `sudo service docker start`

1.4

EIN ERSTES BEISPIEL

- Nach der Installation kann der erste Container gestartet werden (!)
 - `docker run <docker_image>`
 - z.B. `docker run hello-world`
- Das wars

```
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
535020c3e8ad: Pull complete
af340544ed62: Pull complete
Digest: sha256:a68868bfe696c00866942e8f5ca39e3e31b79c1e50feaaa4ce5e28df2f051d5c
Status: Downloaded newer image for hello-world:latest
Hello from Docker.

This message shows that your installation appears to be working correctly.
To generate this message, Docker took the following steps:
1. The Docker Engine CLI client contacted the Docker Engine daemon.
2. The Docker Engine daemon pulled the "hello-world" image from the Docker Hub.
3. The Docker Engine daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker Engine daemon streamed that output to the Docker Engine CLI client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

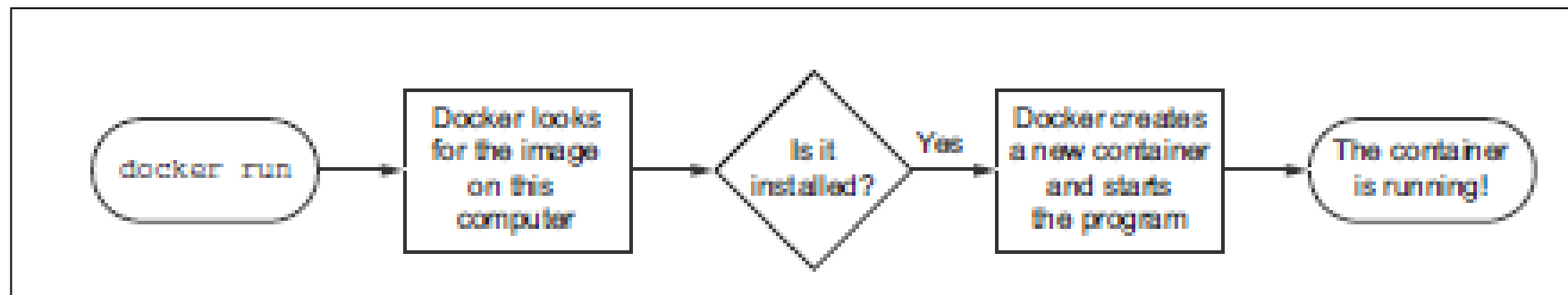
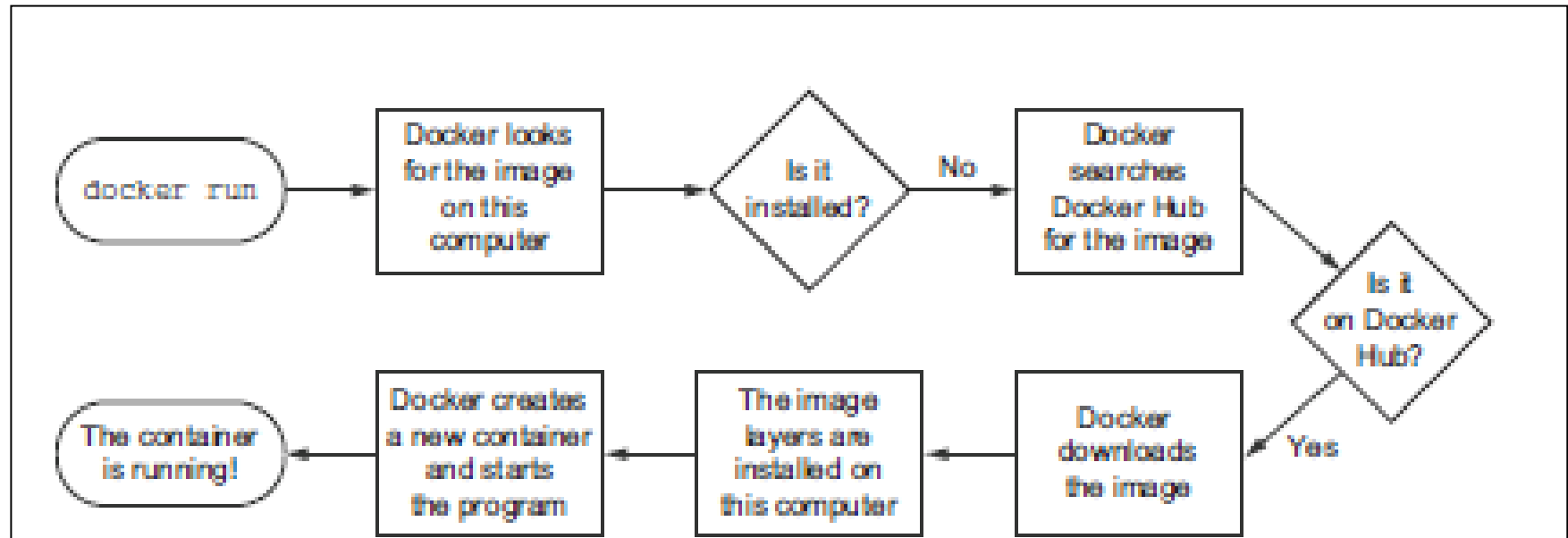
Share images, automate workflows, and more with a free Docker Hub account:
https://hub.docker.com

For more examples and ideas, visit:
https://docs.docker.com/userguide/
```

Wie funktioniert das denn?



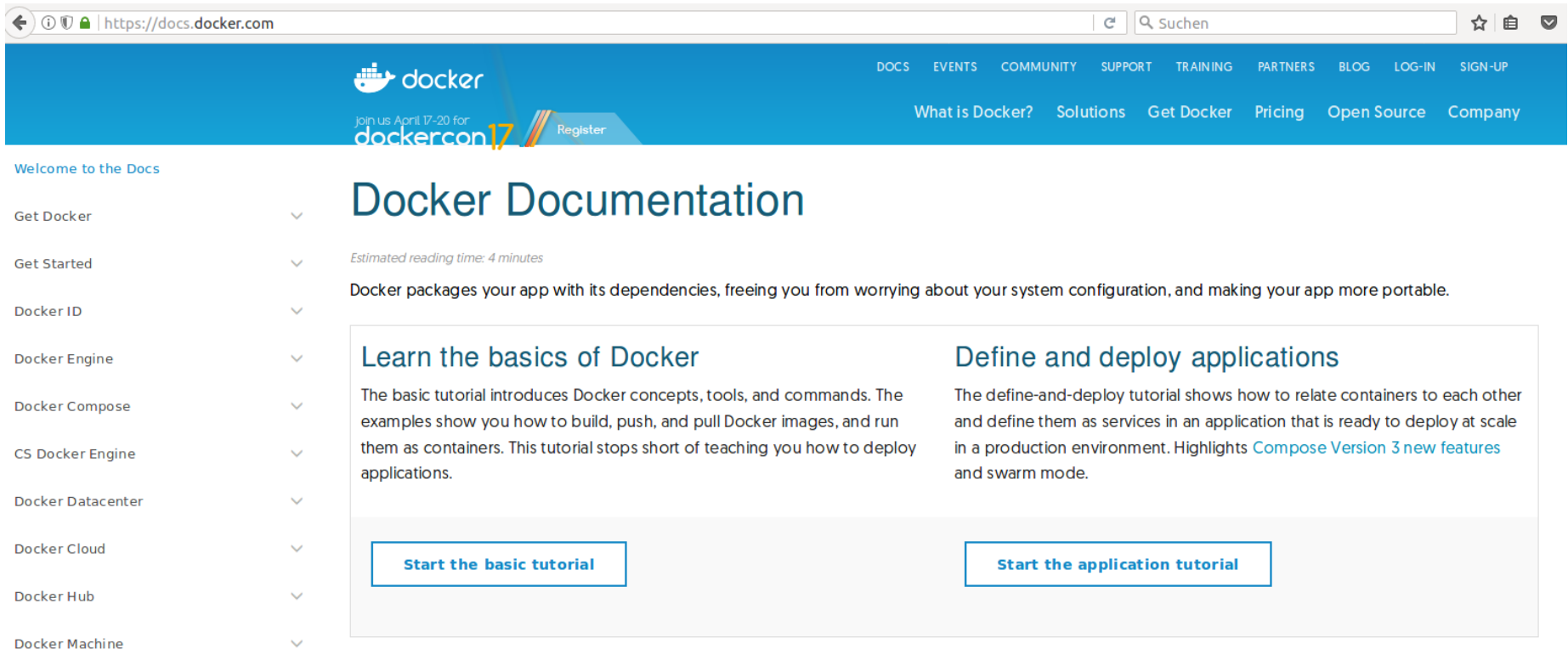
Docker-Workflow: 1. und 2. Lauf



- Der Containers wird gestartet
 - Nochmal: Keine VM, sondern ein gekapselter Prozess auf dem Host-System
- Innerhalb des Containers werden Prozesse gestartet
 - Diese werden sind der Definition des Images angegeben
- Der Container läuft, solange ein definierter Prozess läuft
 - Anschließend wird der Container selbst beendet

1.5

DOKUMENTATION, COMMUNITY UND RESSOURCEN



The screenshot shows the Docker Documentation website. The browser address bar displays `https://docs.docker.com`. The website header is blue with the Docker logo and navigation links: DOCS, EVENTS, COMMUNITY, SUPPORT, TRAINING, PARTNERS, BLOG, LOG-IN, SIGN-UP. Below the header, there's a banner for DockerCon 17 with a 'Register' button. The main content area has a left sidebar with a list of topics: Welcome to the Docs, Get Docker, Get Started, Docker ID, Docker Engine, Docker Compose, CS Docker Engine, Docker Datacenter, Docker Cloud, Docker Hub, and Docker Machine. The main content area is titled 'Docker Documentation' and includes an estimated reading time of 4 minutes. It features two columns of introductory text and two buttons: 'Start the basic tutorial' and 'Start the application tutorial'.

Welcome to the Docs

- Get Docker
- Get Started
- Docker ID
- Docker Engine
- Docker Compose
- CS Docker Engine
- Docker Datacenter
- Docker Cloud
- Docker Hub
- Docker Machine

Docker Documentation

Estimated reading time: 4 minutes

Docker packages your app with its dependencies, freeing you from worrying about your system configuration, and making your app more portable.

Learn the basics of Docker

The basic tutorial introduces Docker concepts, tools, and commands. The examples show you how to build, push, and pull Docker images, and run them as containers. This tutorial stops short of teaching you how to deploy applications.

[Start the basic tutorial](#)

Define and deploy applications

The define-and-deploy tutorial shows how to relate containers to each other and define them as services in an application that is ready to deploy at scale in a production environment. Highlights [Compose Version 3 new features](#) and swarm mode.

[Start the application tutorial](#)

Browser address bar: <https://forums.docker.com> | Suchen

Success Center
▼

Knowledge Base

Documentation

Community Forums

Technical Support

Service Status

docker

Docs | Events | Community | Support | Training | Partners | Blog

What Is Docker? | Enterprise | Get Docker | Pricing | Open Source | Company

Log In | 🔍 | ☰

Welcome to the Docker Community Forums!




This is a public forum for users to discuss questions and explore current design patterns and best practices about **Docker** and related projects in the **Docker Ecosystem**.

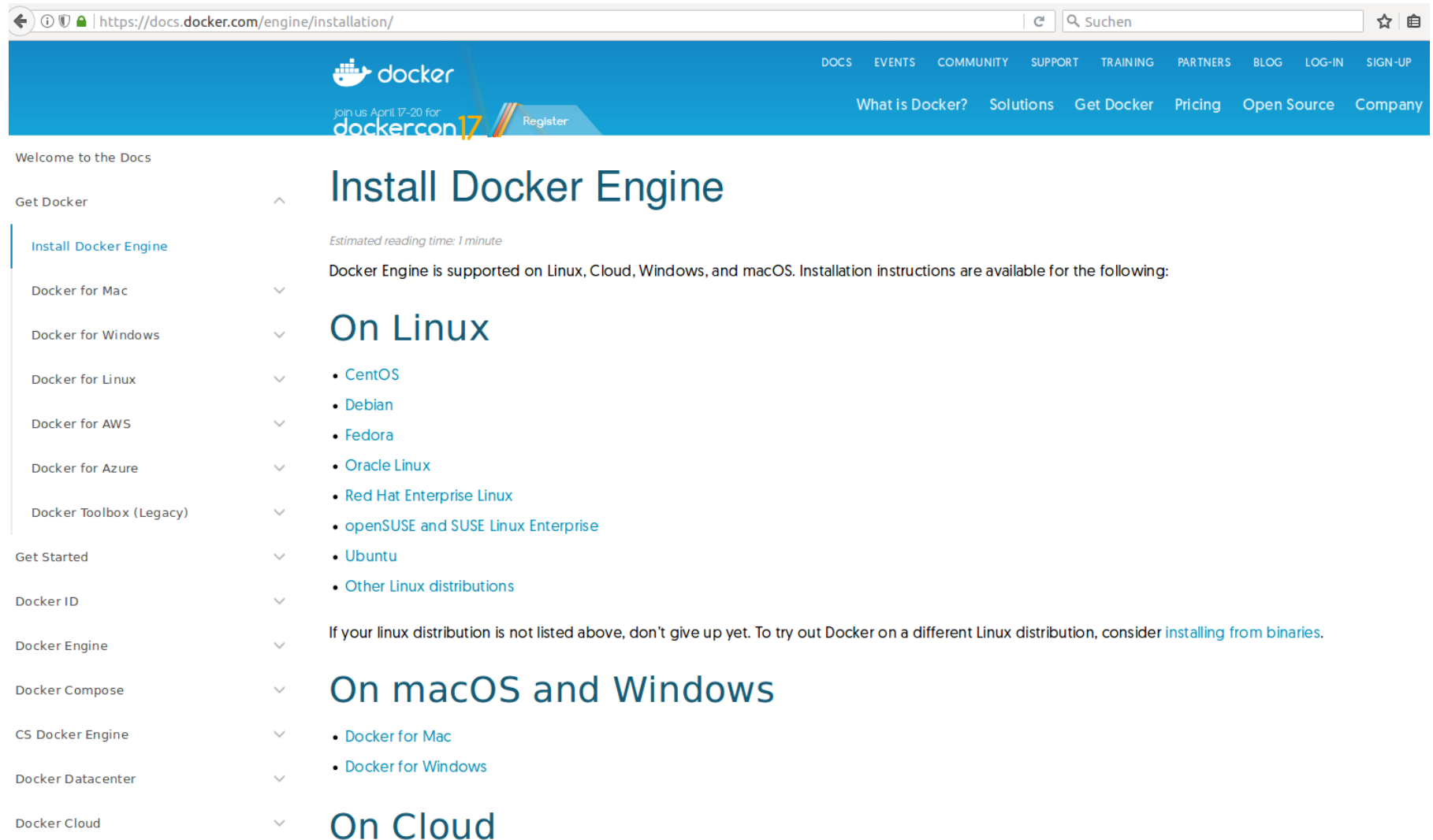
To participate, just log in with your Docker Hub account. Make sure to also review our [Community Guidelines](#), [Terms of Service](#), and [Privacy Policy](#).

When posting issues or feedback, make sure to remove any sensitive information and please provide the following:

- Issue type
- OS Version/build
- App version
- Steps to reproduce

all categories ▾ | **Categories** | Latest | Top

Category	Topics	Latest
Announcements Category for announcing new or updates to the Docker community, products, projects, training, etc, including ...	15	 Docker Containers IP externally ■ General Discussions 0 4m
General Discussions General discussions, feature requests, and training inquiries. ■ General ■ Feature Requests	29 / week	 Docker on windows fails with //./pipe/docker_engine: The system cannot find the file specified ■ Docker for Windows 1 9m
Docker Data Center	6 / week	 Issues Installing UCP 2.1.0 on Docker 1.13 ■ UCP 4 30m



The screenshot shows the Docker documentation website. The browser address bar displays <https://docs.docker.com/engine/installation/>. The page header is blue with the Docker logo and navigation links: DOCS, EVENTS, COMMUNITY, SUPPORT, TRAINING, PARTNERS, BLOG, LOG-IN, SIGN-UP. Below the header, a banner promotes DockerCon 17. The main content area has a left sidebar with a 'Get Docker' section containing links to 'Install Docker Engine', 'Docker for Mac', 'Docker for Windows', 'Docker for Linux', 'Docker for AWS', 'Docker for Azure', and 'Docker Toolbox (Legacy)'. The 'Install Docker Engine' link is active. The main heading is 'Install Docker Engine' with an estimated reading time of 1 minute. The text states that Docker Engine is supported on Linux, Cloud, Windows, and macOS. A list of Linux distributions is provided: CentOS, Debian, Fedora, Oracle Linux, Red Hat Enterprise Linux, openSUSE and SUSE Linux Enterprise, Ubuntu, and Other Linux distributions. A note suggests installing from binaries if the distribution is not listed. Below this, sections for 'On macOS and Windows' and 'On Cloud' are visible, each with a list of links to specific installation guides.

Welcome to the Docs

Get Docker

- Install Docker Engine
- Docker for Mac
- Docker for Windows
- Docker for Linux
- Docker for AWS
- Docker for Azure
- Docker Toolbox (Legacy)

Get Started

Docker ID

Docker Engine

Docker Compose

CS Docker Engine

Docker Datacenter

Docker Cloud

Install Docker Engine

Estimated reading time: 1 minute

Docker Engine is supported on Linux, Cloud, Windows, and macOS. Installation instructions are available for the following:

On Linux

- CentOS
- Debian
- Fedora
- Oracle Linux
- Red Hat Enterprise Linux
- openSUSE and SUSE Linux Enterprise
- Ubuntu
- Other Linux distributions

If your linux distribution is not listed above, don't give up yet. To try out Docker on a different Linux distribution, consider [installing from binaries](#).

On macOS and Windows

- Docker for Mac
- Docker for Windows

On Cloud

← → ↻ Sicher | <https://hub.docker.com/explore/>







Docker Store is the new place to discover public Docker content. [Check it out →](#)



Search

Dashboard Explore Organizations Create  javacream ▾

Explore Official Repositories

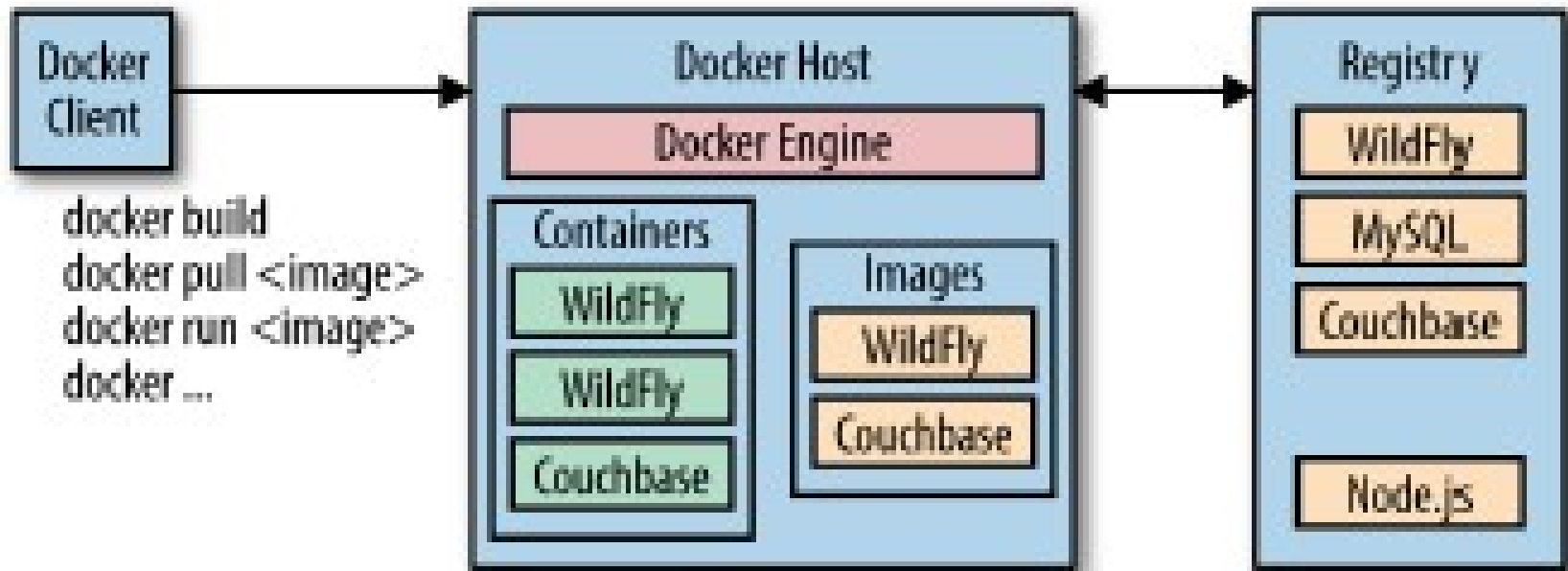
 nginx official	5.2K STARS	10M+ PULLS	➤ DETAILS
 redis official	3.3K STARS	10M+ PULLS	➤ DETAILS
 busybox official	922 STARS	10M+ PULLS	➤ DETAILS
 ubuntu official	5.5K STARS	10M+ PULLS	➤ DETAILS
 registry official	1.3K STARS	10M+ PULLS	➤ DETAILS
 alpine official	1.9K STARS	10M+ PULLS	➤ DETAILS

2

DOCKER IM DETAIL

2.1

DOCKER IMAGES UND CONTAINER



- Stellt eine Read-Only-Umgebung zur Verfügung
 - Analogie OOP: Eine Klassen-Definition
- Im Gegensatz zu VMWare-Images aus Layern aufgebaut
 - Vererbung und Assoziation
- Durch die Layerung sind einzelne Images deshalb relativ schlank
- Images und Layers werden in Repositories verwaltet
 - Inklusive Meta-Informationen
 - Versionierung!
 - Die Docker-Client-Installation verwaltet ein lokales Repository

Beispiel: Ein Java-Image (<https://microbadger.com/images/java>)

Tags	openjdk-8u111 openjdk-8u111-jdk openjdk-8 openjdk-8-jdk latest jdk 8u111 8u111-jdk 8 8-jdk
Created	January 17, 2017 at 01:52 AM
ID	d11c3799fa6a
Download Size	232.1 MB
Labels	No labels
Layers	14

107.2 MB

buildpack-deps

[scm](#) [jessie-scm](#)

What's this? -

49.0 MB

ADD file:89ecb642d662ee7edbb868340551106d51336c7e589... +

CMD ["/bin/bash"]

17.7 MB

RUN apt-get update && apt-get install -y --no-instal... +

40.5 MB

RUN apt-get update && apt-get install -y --no-instal... +

579.2 kB

RUN apt-get update && apt-get install -y --no-install-recommends b... +

214 bytes

RUN echo 'deb http://deb.debian.org/debian jessie-backports main' > ... +

ENV LANG=C.UTF-8

242 bytes

RUN { echo '#!/bin/sh'; echo 'set -e'; echo; echo 'dirname "... +

ENV JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64

ENV JAVA_VERSION=8u111

ENV JAVA_DEBIAN_VERSION=8u111-b14-2-bpo8+1

32 bytes

ENV CA_CERTIFICATES_JAVA_VERSION=20140324

124.1 MB

RUN set -x && apt-get update && apt-get install -y openjdk-8-jdk... +

282.3 kB

RUN /var/lib/dpkg/info/ca-certificates-java.postinst configure

- Eine Instanz eines Images
 - Analogie zu OOP: Ein Objekt
 - Identifikation über einen Namen bzw. über einen technischen Schlüssel
 - Ein Hash
- Ein Container hat einen Lebenszyklus
 - Gesteuert über Docker-Kommandos
 - `create`
 - `run`
 - `start`
 - `stop`
 - `delete`
- Im Gegensatz zu Images können Container einen Zustand aufweisen
 - Container-Environment
 - Interne Ressourcen
 - Dateisystem des Containers

- Container sind komplett gekapselt
 - Interaktion nur über Docker-Kommandos
 - Auslesen des Docker-Logs
 - Inspect
 - Ausführen einer Shell im Container
- Benutzung externer Volumes
 - Ein Teil des Dateisystems des Containers vom Host zur Verfügung gestellt
- Netzwerk-Kommunikation
 - Container können Netzwerk-Sockets bereit stellen
 - Diese werden auf reale Sockets des Hosts gemapped
- Container-Linking
 - Direkte Kommunikation der Container untereinander
 - Verschiedene Abstufungen
 - Gemeinsames Netzwerk
 - Gemeinsames File-Layer
 - Shared Memory

2.2

DOCKER COMMAND LINE

- <https://docs.docker.com/engine/reference/commandline/docker/#/related-commands>
- **Elementare Befehle**
 - **Container anlegen**
 - `sudo docker create <<Image-Name>>:<<Versionsnummer>>`
 - **Container löschen**
 - `sudo docker rm <<containerId>>`
 - **Docker-Container automatisch nach Ausführung löschen**
 - `sudo docker run -rm <<Image-Name>>`
 - **Alle Container anzeigen lassen**
 - `sudo docker ps -a`
 - **Container starten**
 - `sudo docker start <<containerId>>`
 - **Container anlegen, starten und assoziiert eigene Ausgabe-Konsole (nicht in der Terminal-Session, in der gearbeitet wird)**
 - `sudo docker run --detach <<Image-Name>>`
 - **Images anzeigen lassen**
 - `sudo docker images`

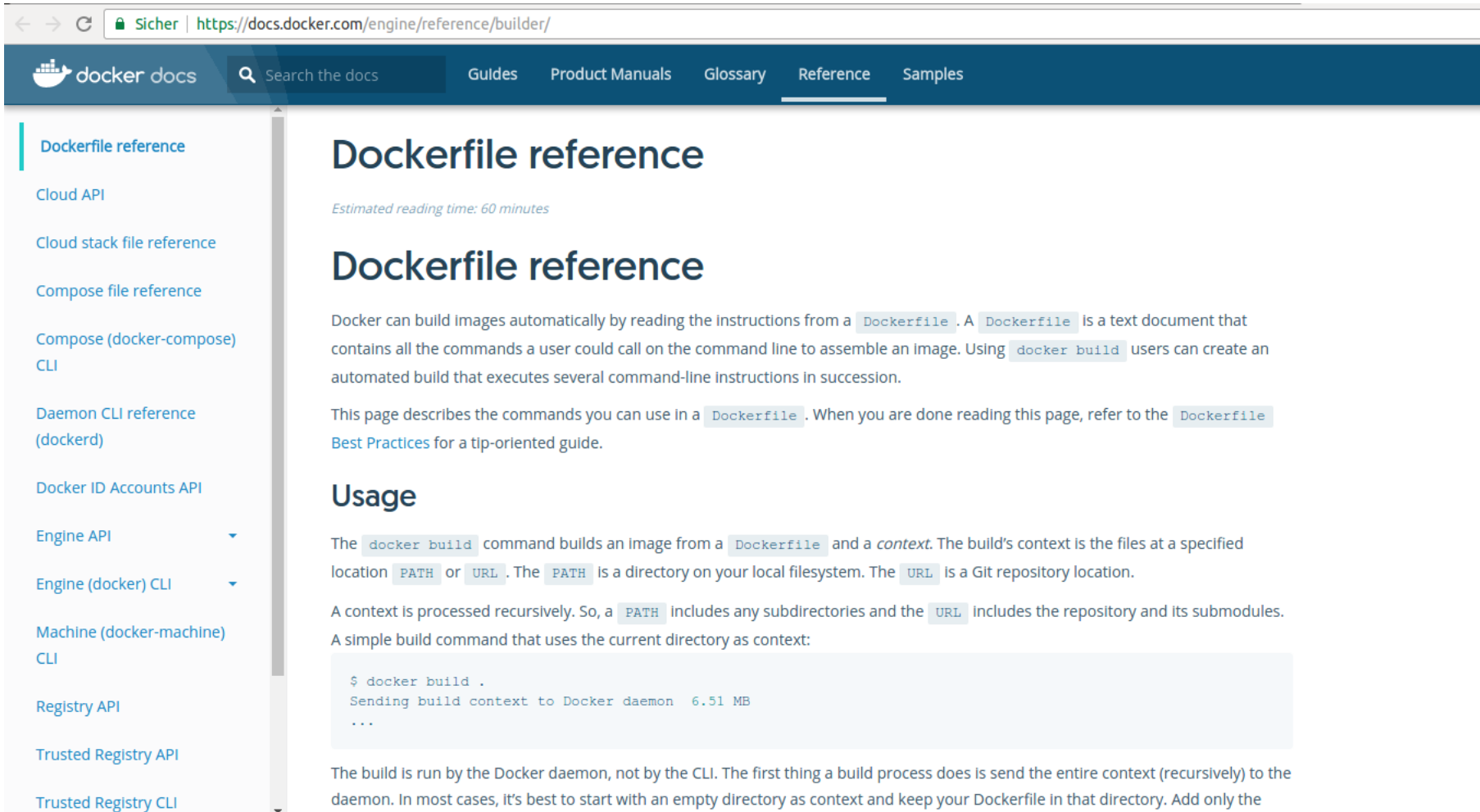
2.3

DEFINITION EINES DOCKER IMAGES

- Anlegen eines leeren Verzeichnisses
 - Alle Verzeichnisse und Unterverzeichnisse werden Bestandteil des Images
 - `excludes` in `.dockerignore`
- Erstellen eines Docker-Files
 - Bestehend aus Kommandos
 - `FROM`
 - `COPY`
 - `ENV`
 - `RUN`
 - `EXPOSE`

- Das Dockerfile selbst ist ein simples Text-Dokument
 - Referenz unter <https://docs.docker.com/engine/reference/builder/>
 - Beispiel

```
FROM openjdk
CMD ["java", "-version"]
```
- Erzeugen des Images
 - `docker build -tag <name>`



The screenshot shows a web browser displaying the Docker documentation page for Dockerfile reference. The browser's address bar shows the URL <https://docs.docker.com/engine/reference/builder/>. The page has a dark blue header with the Docker logo and navigation links: Guides, Product Manuals, Glossary, Reference (selected), and Samples. A search bar is also present. On the left, a sidebar lists various Docker documentation topics, with 'Dockerfile reference' highlighted. The main content area has the title 'Dockerfile reference' and an estimated reading time of 60 minutes. The text explains that Docker can build images automatically by reading instructions from a Dockerfile, which is a text document containing commands for assembling an image. It mentions the 'docker build' command and refers to 'Best Practices' for a tip-oriented guide. A section titled 'Usage' describes how the 'docker build' command works, mentioning 'PATH' and 'URL' for the build context. A code block shows a terminal snippet:

```
$ docker build .  
Sending build context to Docker daemon 6.51 MB  
...
```

 The text concludes by stating that the build is run by the Docker daemon, not the CLI, and that the first step is to send the entire context to the daemon.

← → ↻ Sicher | <https://docs.docker.com/engine/reference/builder/>

docker docs 🔍 Search the docs Guides Product Manuals Glossary **Reference** Samples

Dockerfile reference

Cloud API

Cloud stack file reference

Compose file reference

Compose (docker-compose) CLI

Daemon CLI reference (dockerd)

Docker ID Accounts API

Engine API ▾

Engine (docker) CLI ▾

Machine (docker-machine) CLI

Registry API

Trusted Registry API

Trusted Registry CLI

Dockerfile reference

Estimated reading time: 60 minutes

Dockerfile reference

Docker can build images automatically by reading the instructions from a `Dockerfile`. A `Dockerfile` is a text document that contains all the commands a user could call on the command line to assemble an image. Using `docker build` users can create an automated build that executes several command-line instructions in succession.

This page describes the commands you can use in a `Dockerfile`. When you are done reading this page, refer to the `Dockerfile Best Practices` for a tip-oriented guide.

Usage

The `docker build` command builds an image from a `Dockerfile` and a `context`. The build's context is the files at a specified location `PATH` or `URL`. The `PATH` is a directory on your local filesystem. The `URL` is a Git repository location.

A context is processed recursively. So, a `PATH` includes any subdirectories and the `URL` includes the repository and its submodules. A simple build command that uses the current directory as context:

```
$ docker build .  
Sending build context to Docker daemon 6.51 MB  
...
```

The build is run by the Docker daemon, not by the CLI. The first thing a build process does is send the entire context (recursively) to the daemon. In most cases, it's best to start with an empty directory as context and keep your Dockerfile in that directory. Add only the

3

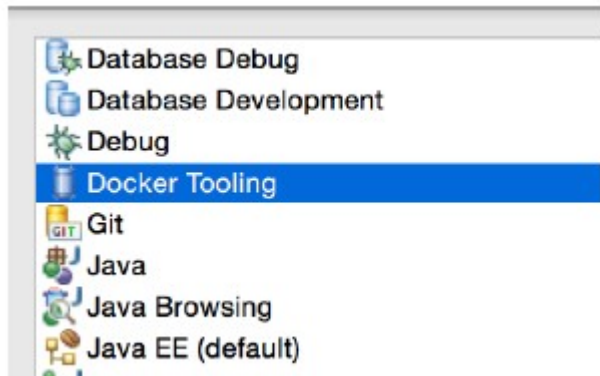
JAVA UND DOCKER

3.1

ENTWICKLUNGSUMGEBUNG EN

- Produktauswahl
 - Eclipse
 - IntelliJ
 - ...
- Docker wird durch PlugIns unterstützt!

- Perspektive

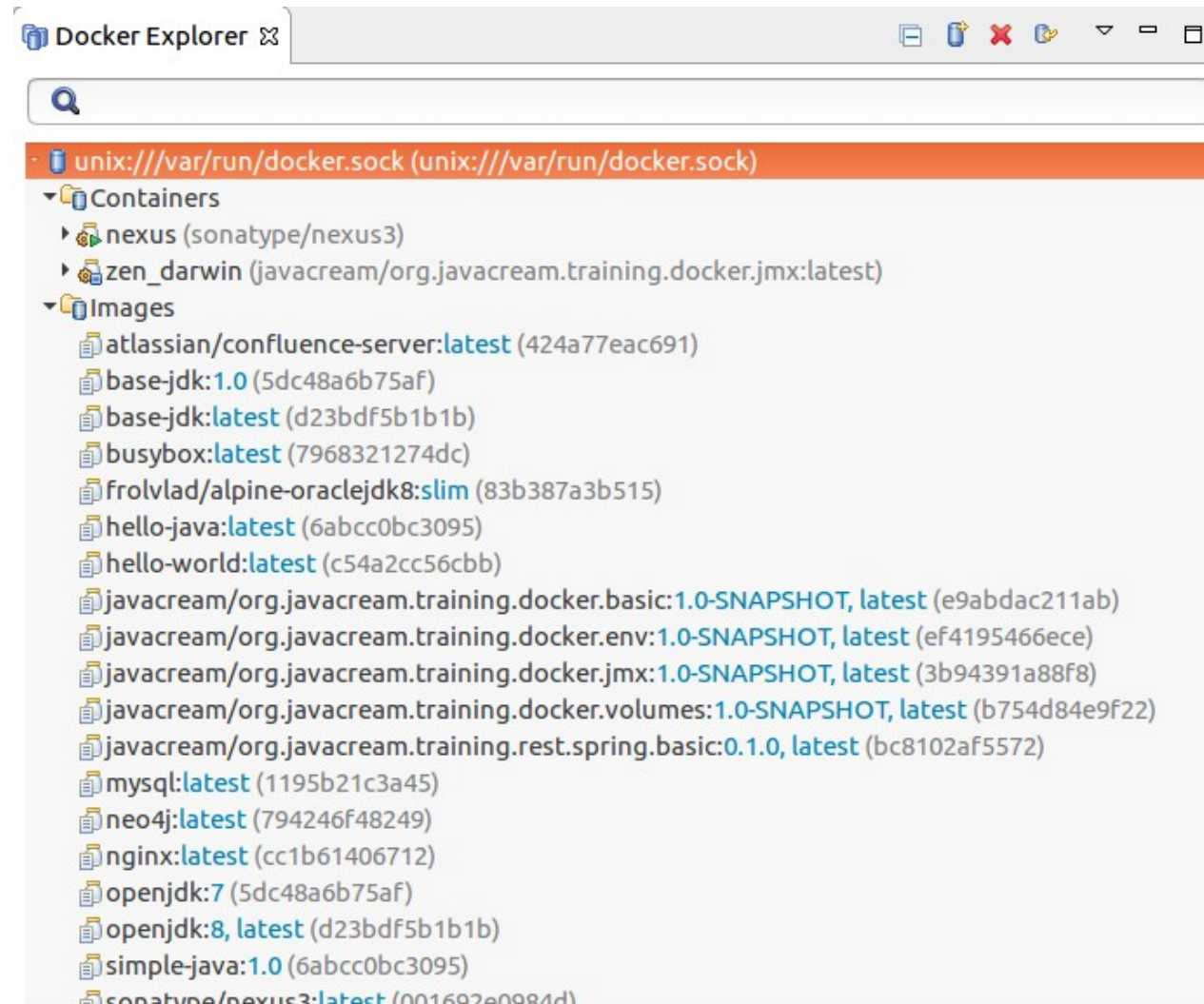


- Views

- Docker Explorer
- Docker Images
- Docker Containers
- Docker Console

- Wizards

- Build
- Run Container



Docker Containers ⓘ

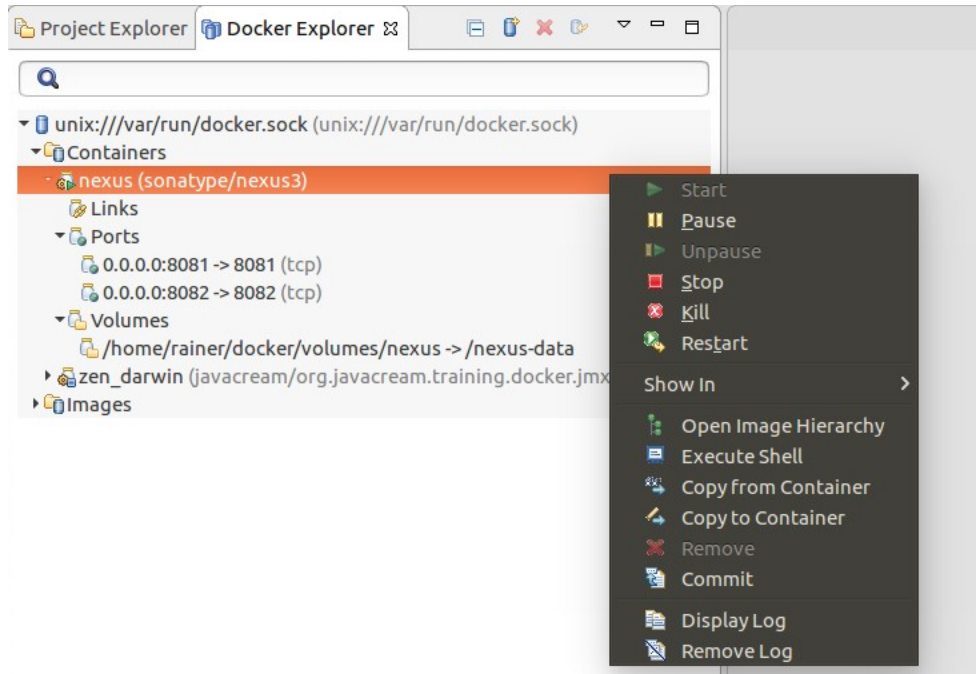
unix:///var/run/docker.sock (2 Containers)

Name	Image	Created	Command	Ports	Status
zen_darwin	javacream/org.javacream.training	2017-02-23	/bin/sh -c 'java -javaagent:jolokia-jv		Exited (137) 3 weeks ago
nexus	sonatype/nexus3	2017-02-22	bin/nexus run	0.0.0.0:8082->8082/tcp, 0.0.0.0:8081	Up About an hour

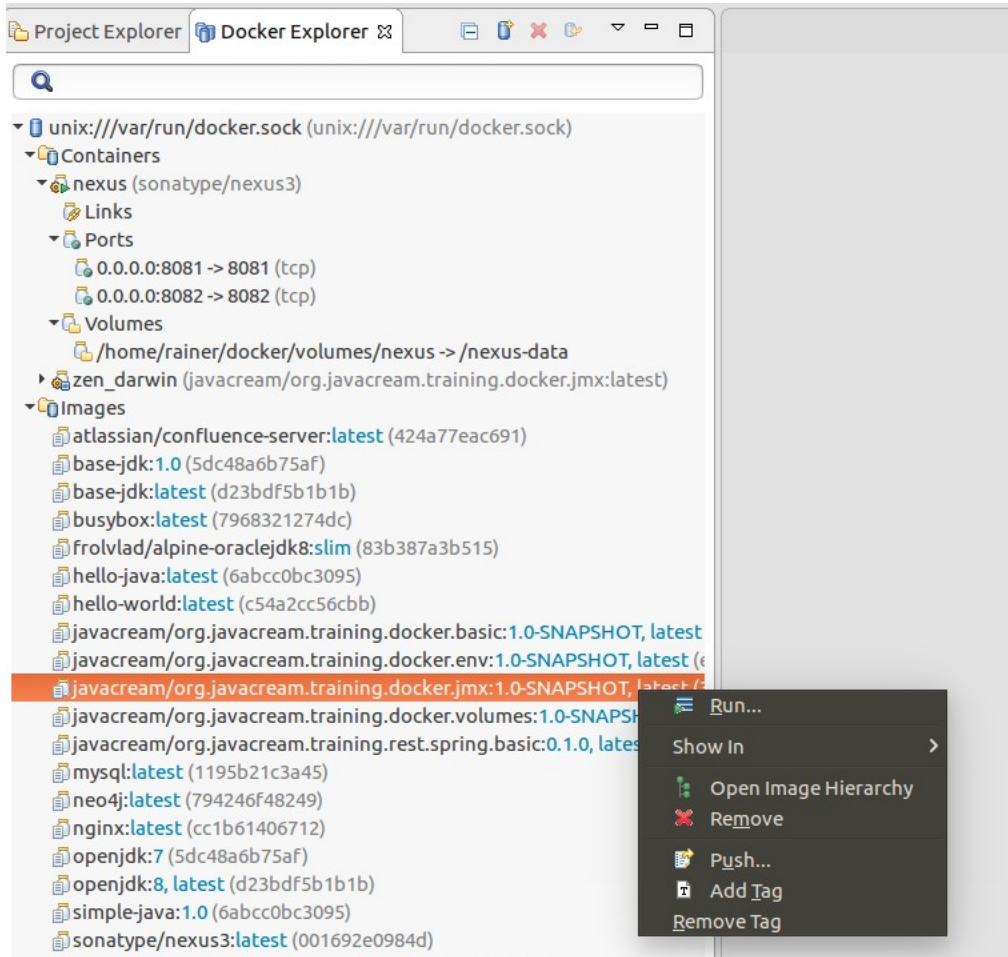
Docker Images

Docker Images			
unix:///var/run/docker.sock (18/36 Images)			
Q			
Id	Repo Tags	Created	Virtual Size
b754d84e9f22	javacream/org.javacream.training.docker.volumes:1	2017-02-25	643,2 MB
ef4195466ece	javacream/org.javacream.training.docker.volumes:l	2017-02-25	643,2 MB
e9abdac211ab	javacream/org.javacream.training.docker.env:1.0-SN	2017-02-25	643,2 MB
3b94391a88f8	javacream/org.javacream.training.docker.env:latest	2017-02-25	643,2 MB
bc8102af5572	javacream/org.javacream.training.docker.basic:1.0-S	2017-02-23	643,6 MB
001692e0984d	javacream/org.javacream.training.docker.basic:late	2017-02-22	698,8 MB
2dc5069d7582	javacream/org.javacream.training.rest.spring.basic:	2017-02-16	460,4 MB
aa26cdcc81ea	sonatype/nexus3:latest	2017-02-10	400,2 MB
6abcc0bc3095	wordpress:latest	2017-02-07	222,1 MB
cc1b61406712	springio/gs-spring-boot-docker:latest	2017-02-06	643,2 MB
83b387a3b515	hello-java:latest	2017-01-24	181,8 MB
d23bdf5b1b1b	simple-java:1.0	2017-01-22	166,5 MB
5dc48a6b75af	nginx:latest	2017-01-17	643,2 MB
7968321274dc	frolvlad/alpine-oraclejdk8:slim	2017-01-17	584,5 MB
794246f48249	base-jdk:latest	2017-01-13	1,1 MB
424a77eac691	openjdk:8	2016-09-23	377,4 MB
c54a2cc56cbb	openjdk:latest	2016-09-09	817,2 MB
1195b21c3a45	base-jdk:1.0	2016-07-01	1,8 kB
	openjdk:7	2016-06-10	380,2 MB
	busybox:latest		
	neo4j:latest		
	atlassian/confluence-server:latest		
	hello-world:latest		
	mysql:latest		

Docker Context Menü für Container



Docker Context Menü für Images

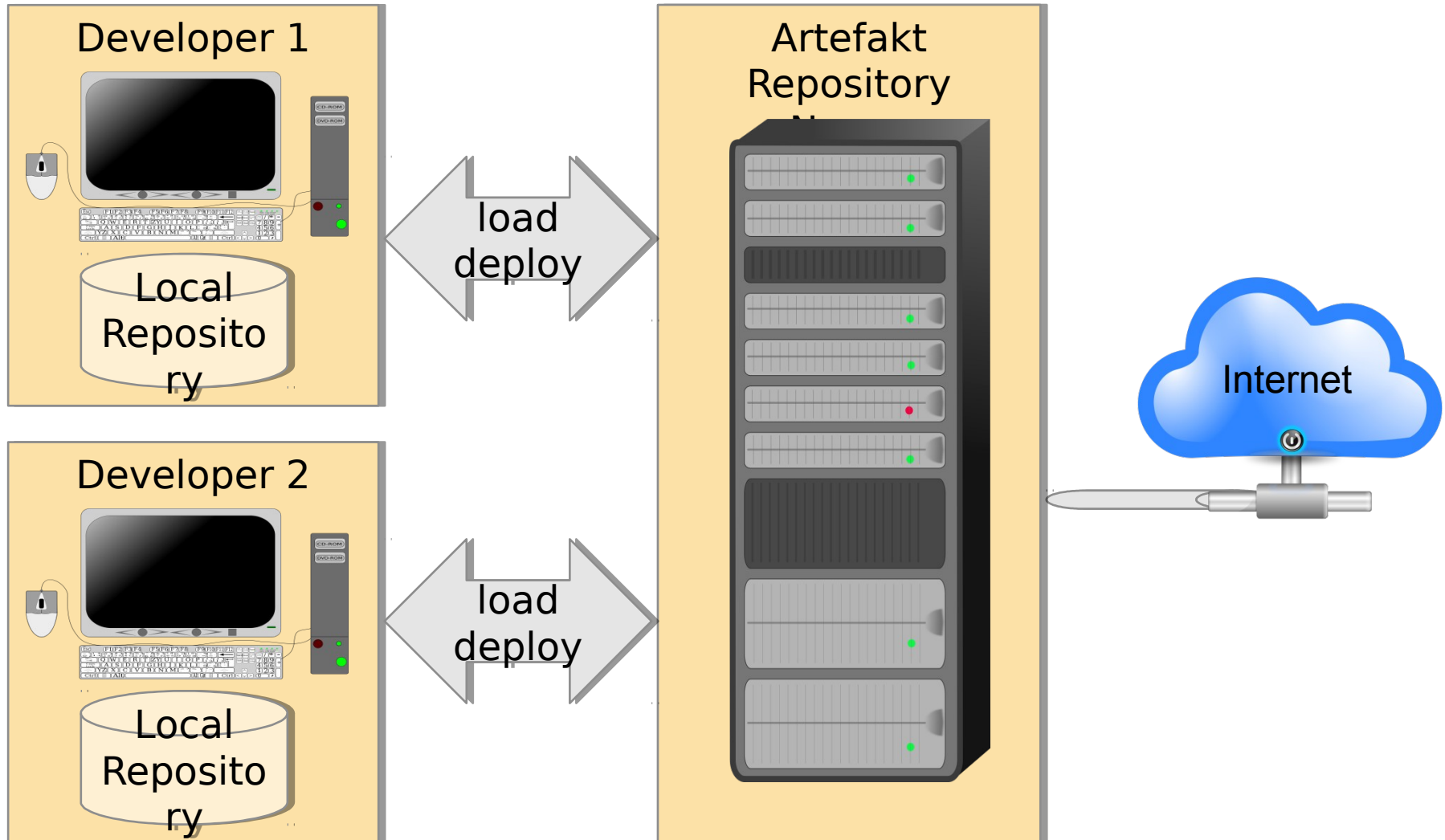


3.2

BUILD-PROZESS

- Bestandteile
 - Fester Build-Prozess
 - Aufruf mit mvn <command>
 - compile
 - package
 - install
 - deploy
- Lokaler Cache für gebildete Artefakte
- Connectivity zu einem Remote-Repository
 - Standard: Internet-Repository repo1.maven.org
 - Im Unternehmen: Eigener Repository-Server
 - Nexus
 - Artefactory
 - ...

- Unterscheiden „Werke“ (Java-Code) von „Artefakten“ (Java-Archive)
 - Der Build-Prozess wandelt Werke in Artefakte um
- Werke liegen klassischerweise in der Versionsverwaltungssysteme
 - Branches
 - Tags (Versionsnummern)
- Artefakte werden von einem Artefakt-Repository verwaltet
 - Damit können Dependencies ohne Zugriff auf das Versionsverwaltungssystem aufgelöst werden
 - Identifikation von Artefakten: "Artefakt-Koordinate"
- Build-Sequenz
 - Compiler-Aufruf
 - JAR-Archivierer
 - + x, z. B. Auschecken aus Versionsverwaltung, Testen, Ausbringen aus Servern, etc.
- Dependency Management
 - Andere interne Software-Projekte
 - Open Source-Projekte
 - Produkt-Bibliotheken



- settings.xml, allgemein gültig

```
<settings>
```

```
  <mirrors>
```

```
    <mirror>
```

Hier wird die Adresse des internen Servers eingetragen

```
    </mirror>
```

```
  </mirrors>
```

```
<servers>
```

```
  <server>
```

Authentifizierungs-Informationen für den internen
Server

```
  </server>
```

```
</servers>
```

```
</settings>
```

- Das übernimmt eine so genannte Parent-POM
 - Eine Art Superklasse für alle Maven-Buildprozesse
 - Diese wird als eigenes Artefakt im Repository abgelegt
- Beispiel:
 - Ausschnitt des Distribution-Managements

```
<distributionManagement>
  <repository>
    <uniqueVersion>false</uniqueVersion>
    <id>nexus</id>
    <name>Corporate Repository</name>
    <url>http://...</url>
  </repository>
  <snapshotRepository>
    <uniqueVersion>true</uniqueVersion>
    <id>nexus</id>
    <name>Corporate Snapshots</name>
    <url>http://...</url>
  </snapshotRepository>
</distributionManagement>
```

- Konfiguration des Java-Compilers
 - Java-Version
- Reporting
- Allgemein zu benutzende Dependencies zu anderen Artefakten
 - JUnit
 - Datenbank-Treiber
 - ...
- Konfiguration weiterer PlugIns
 - Spezielle Reports
 - DOCKER!

- fabric8io/docker-maven-plugin
- spotify/docker-maven-plugin
- wouterd/docker-maven-plugin
- alexec/docker-maven-plugin

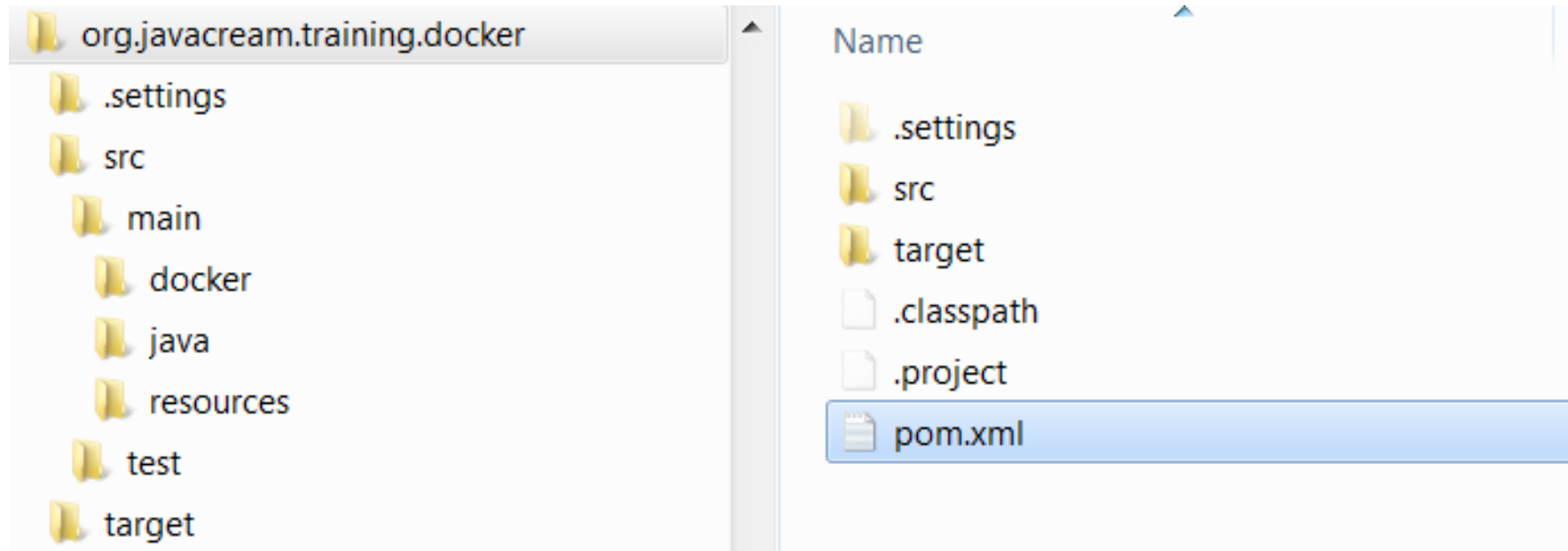
Beispiel: Das Spotify-Plugin

```
<plugin>
  <groupId>com.spotify</groupId>
  <artifactId>docker-maven-plugin</artifactId>
  <version>0.4.11</version>
  configuration
    <imageName>${docker.namespace.prefix}/${project.artifactId}</imageName>
    <dockerDirectory>src/main/docker</dockerDirectory>
    <resources>
      <resource>
        <targetPath></targetPath>
        <directory>${project.build.directory}</directory>
        <include>${project.build.finalName}.jar</include>
      </resource>
      <resource>
        <targetPath></targetPath>
        <directory>${project.build.directory}</directory>
        <include>libs/*.jar</include>
      </resource>
    </resources>
    <imageTags>
      <imageTag>${project.version}</imageTag>
      <imageTag>latest</imageTag>
      <imageTag>localhost:5000/${project.build.finalName}</imageTag>
    </imageTags>
  </configuration>
</plugin>
```

Was macht das Docker-PlugIn?

- Ab jetzt kann automatisiert ein Docker-Image erstellt werden
 - `mvn docker:build`
- Voraussetzung ist
 - ein vorhandenes Dockerfile
 - Eine zusätzliche Konfiguration des Plugins

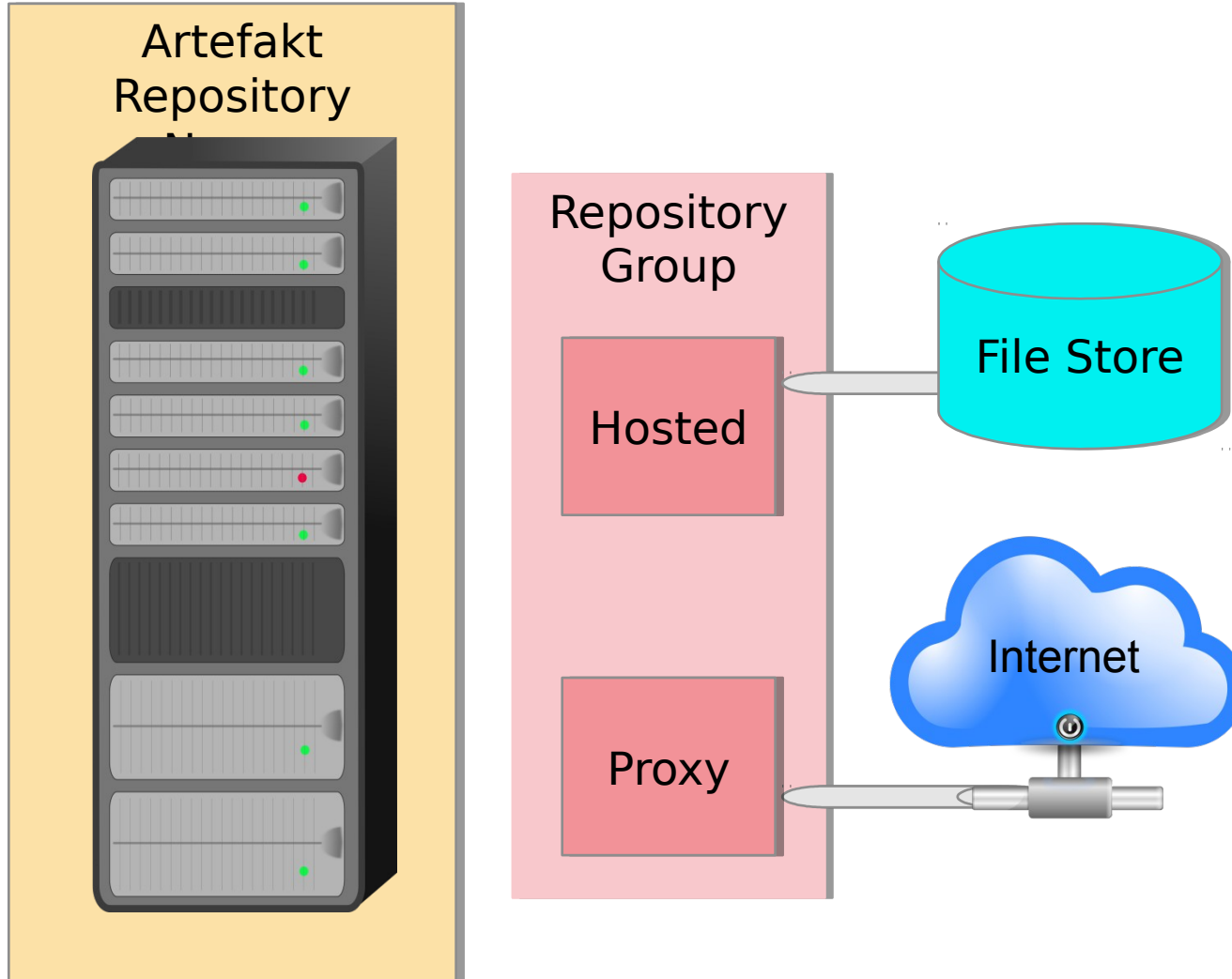
Ein fertiges Java-Projekt mit Docker-Unterstützung



3.3

DER REPOSITORY-SERVER

- Fertige Produktlösung von Sonatype
 - Community
 - Kommerzieller Support
- Repository Server für
 - Java Artefakte
 - Aber auch Docker-Images
- Alternative Produkte
 - Apache Archiva
 - JFrog Artefactory
 - Atlassian Bitbucket



- Die Repository -Group wird für Lese-Vorgänge genutzt
 - Nicht vorhandene Artefakte oder Images werden über den Internet-Proxy nachgeladen
- Das Hosted-Repository wird für die Ablage eigener Artefakte und Images genutzt

Nexus Maven-Repository

Maven - Nexus Repository Manager - Mozilla Firefox

Maven - Nexus Rep... x +

localhost:8081/#browse/search/maven

Nexus Repository Manager
OSS 3.2.1-01

Search components

admin Sign out

Browse

- Welcome
- Search
 - Custom
 - Docker
 - Maven**
 - NuGet
- Browse
 - Assets
 - Components

Maven Search for components by Maven coordinates

Group Id: Any, Artifact Id: Any, Version: Any, Base Version: Any, Classifier: Any, Extension: Any

+ More criteria

Only showing the first 1,000 of 1,145 results

Name ↑	Group	Version	Format
org.eclipse.sisu:piexus	org.eclipse.sisu	0.3.0	maven2
org.javacream.training.application.book	org.javacream.training.application.book	0.0.1-20170310....	maven2
org.javacream.training.application.book	org.javacream.training.application.book	0.0.1-20170310.1...	maven2
org.javacream.training.application.book.api	org.javacream.training.application.book	0.0.1-20170305...	maven2
org.javacream.training.application.book.api	org.javacream.training.application.book	0.0.1-20170310....	maven2
org.javacream.training.application.book.api	org.javacream.training.application.book	0.0.1-20170310.1...	maven2
org.javacream.training.application.book.cdi	org.javacream.training.application.book.cdi	0.0.1-20170310....	maven2
org.javacream.training.application.book.cdi	org.javacream.training.application.book.cdi	0.0.1-20170310.1...	maven2
org.javacream.training.application.book.cdi.jee	org.javacream.training.application.book.cdi	0.0.1-20170310....	maven2
org.javacream.training.application.book.cdi.jee	org.javacream.training.application.book.cdi	0.0.1-20170310.1...	maven2
org.javacream.training.application.book.cdi.jee.jp	org.javacream.training.application.book.cdi	0.0.1-20170310....	maven2
org.javacream.training.application.book.cdi.jee.jp	org.javacream.training.application.book.cdi	0.0.1-20170310.1...	maven2
org.javacream.training.application.book.cdi.jee.memory	org.javacream.training.application.book.cdi	0.0.1-20170310....	maven2
org.javacream.training.application.book.cdi.jee.memory	org.javacream.training.application.book.cdi	0.0.1-20170310.1...	maven2
org.javacream.training.application.book.cdi.spring	org.javacream.training.application.book.cdi	0.0.1-20170310....	maven2
org.javacream.training.application.book.cdi.spring	org.javacream.training.application.book.cdi	0.0.1-20170310.1...	maven2

Copyright © 2008-present, Sonatype Inc. All rights reserved.

Nexus Docker-Repository

Docker - Nexus Repository Manager - Mozilla Firefox

Docker - Nexus Rep... x +

localhost:8081/#browse/search/docker | Suchen

Nexus Repository Manager OSS 3.2.1-01 | Search components | admin | Sign out

Browse

- Welcome
- Search
 - Custom
 - Docker**
 - Maven
 - NuGet
- Browse
 - Assets
 - Components

Docker Search for components in Docker repositories

Image Name: Any Image Tag: Any Layer Id: Any

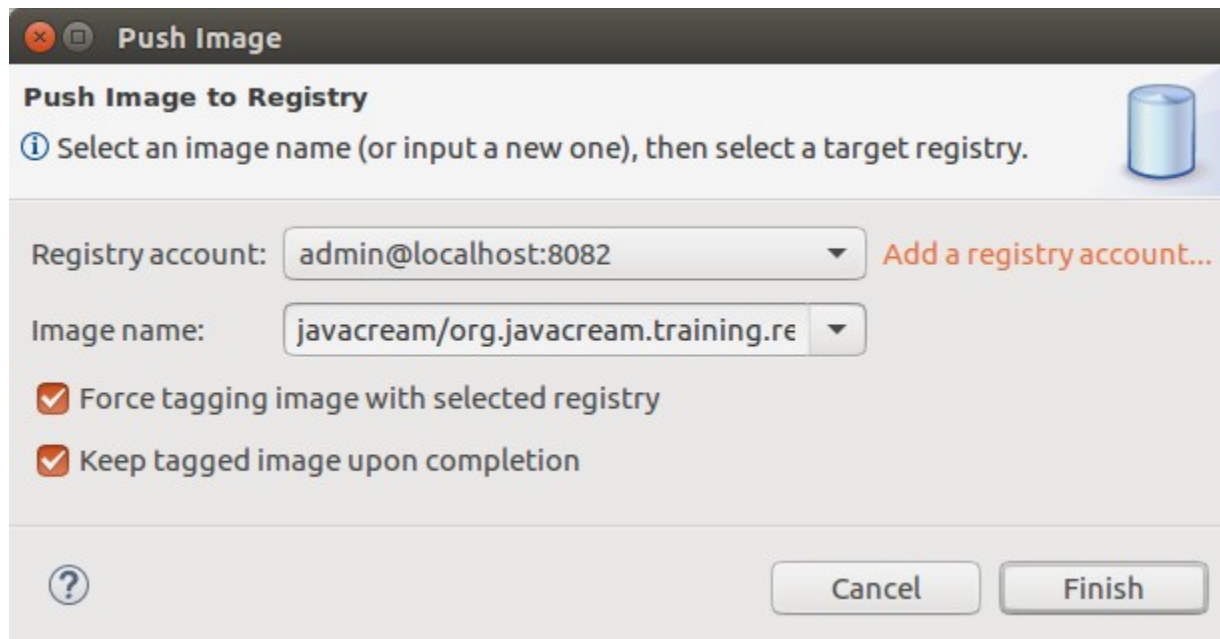
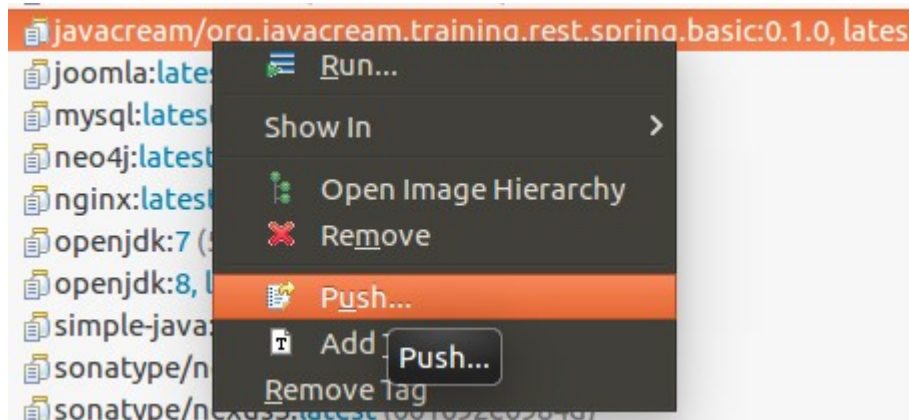
Content Digest: Any More criteria

	Name ↑	Group	Version	Format	
	javacream/org.javacream.training.docker.basic	⊗	1.0-SNAPSHOT	docker	>
	javacream/org.javacream.training.rest.spring.basic	⊗	0.1.0	docker	>

Copyright © 2008-present, Sonatype Inc. All rights reserved.

- Ablage der Credentials im Docker-System
 - `docker login -u user -p pwd`
- Images benötigen für die Ablage in eigenem Repository die Host-Information als spezielles Tag
 - `docker tag host:port/original:tag original:tag`
- Anschließend wird das Image gepushed
 - `docker push host:port/original:tag`
- Natürlich kann der Push auch über das Docker-Plugin und Maven erfolgen

Docker-Push mit Eclipse



4

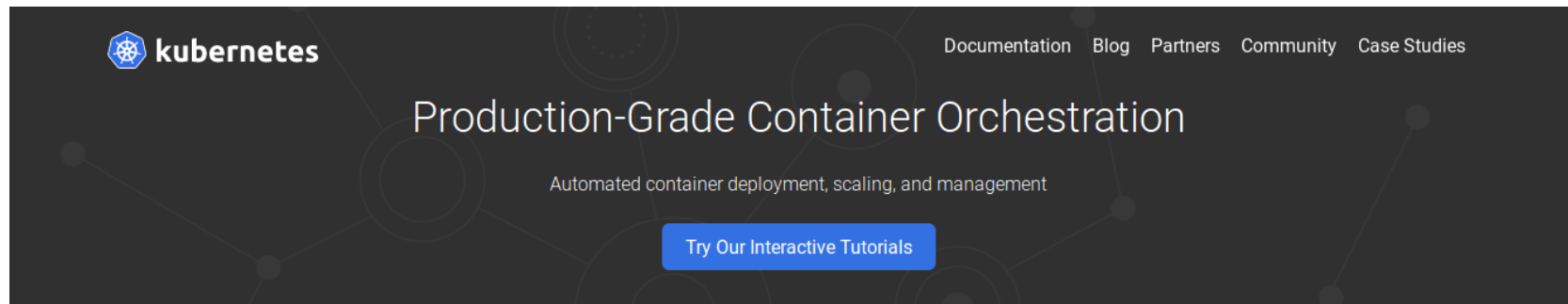
SYSTEM-WERKZEUGE UND TOOLS

4.1

DOCKER SWARM

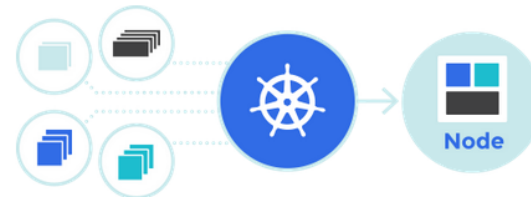
4.2

KUBERNETES



Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications.

It groups containers that make up an application into logical units for easy management and discovery. Kubernetes builds upon [15 years of experience of running production workloads at Google](#), combined with best-of-breed ideas and practices from the community.



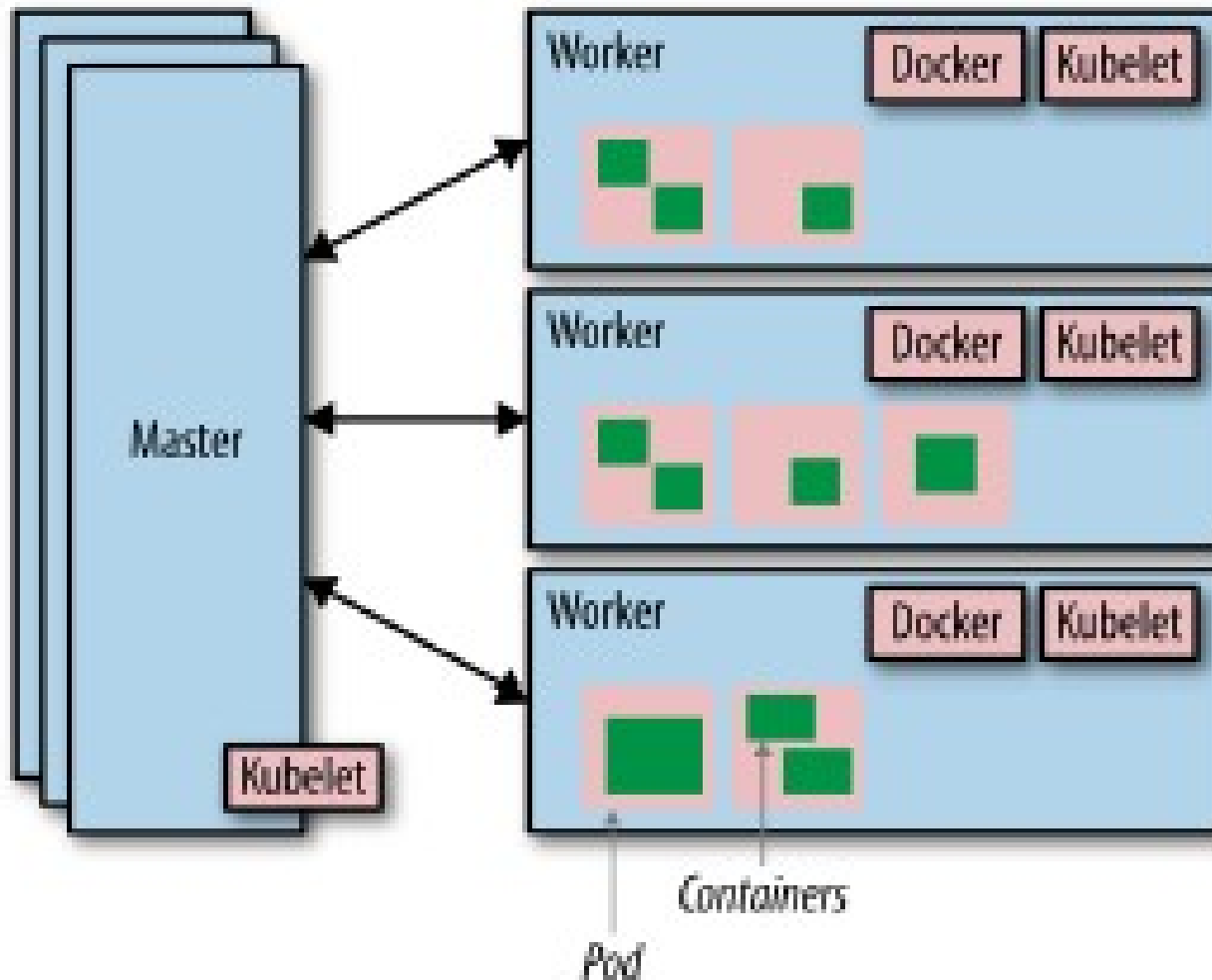
Planet Scale

Designed on the same principles that allows Google to run billions of containers a week, Kubernetes can scale without increasing your ops team.

Never Outgrow

Whether testing locally or running a global enterprise, Kubernetes flexibility grows with you to deliver your applications consistently and easily no matter how complex your need is.






4.3

OPENSIFT

Announcing Red Hat OpenShift Container Platform 3.4

MENU

 **OPENSIFT** FEATURES PRICING CONTAINER PLATFORM MORE ▾ MY ACCOUNT ▾

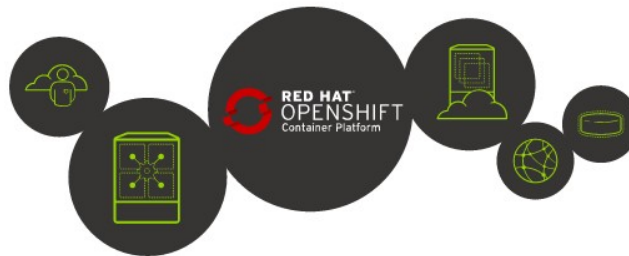
**RED HAT
OPEN INNOVATION LABS**

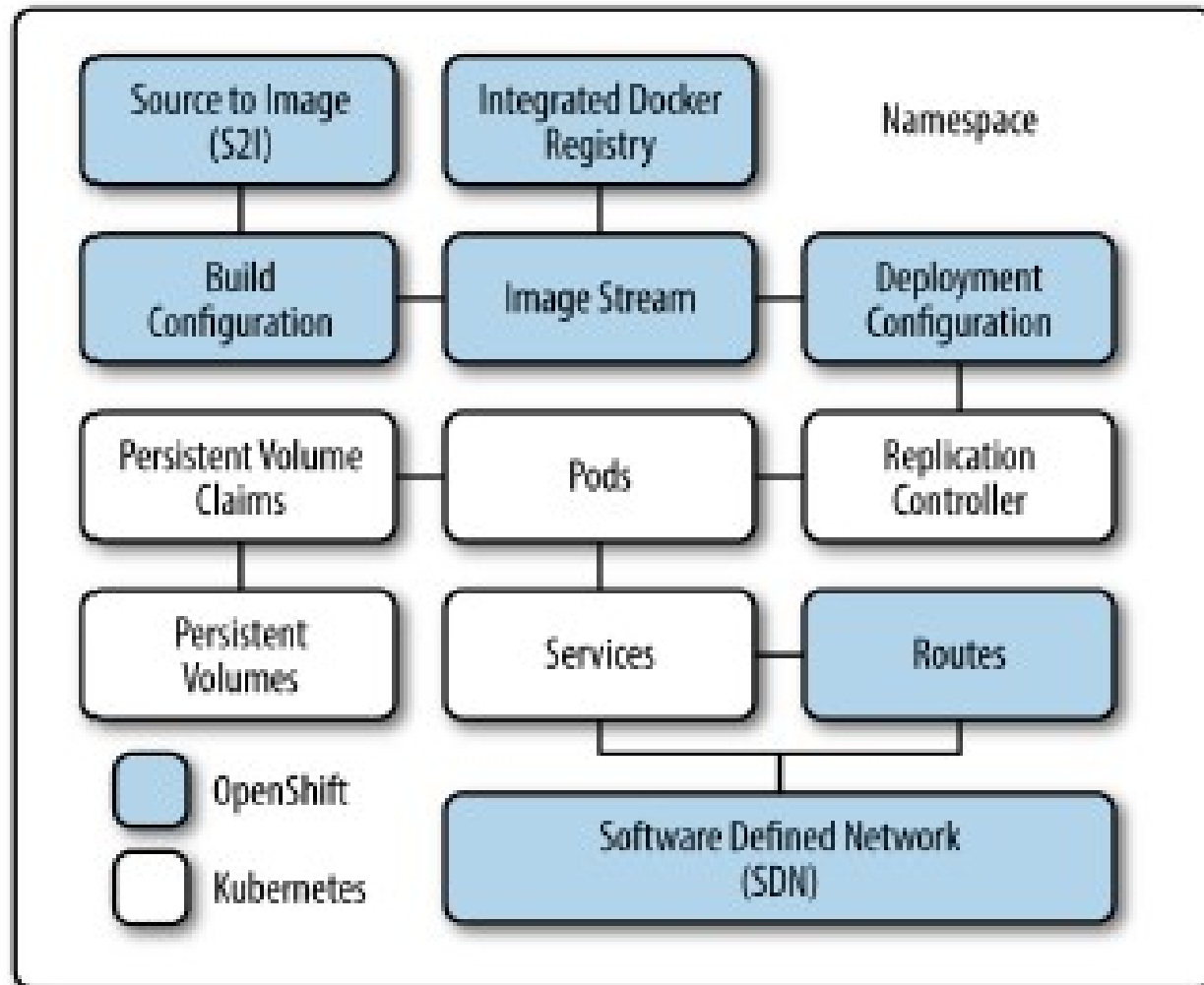
**FIND A BETTER WAY TO TACKLE A COMMON CHALLENGE
—TOGETHER.**

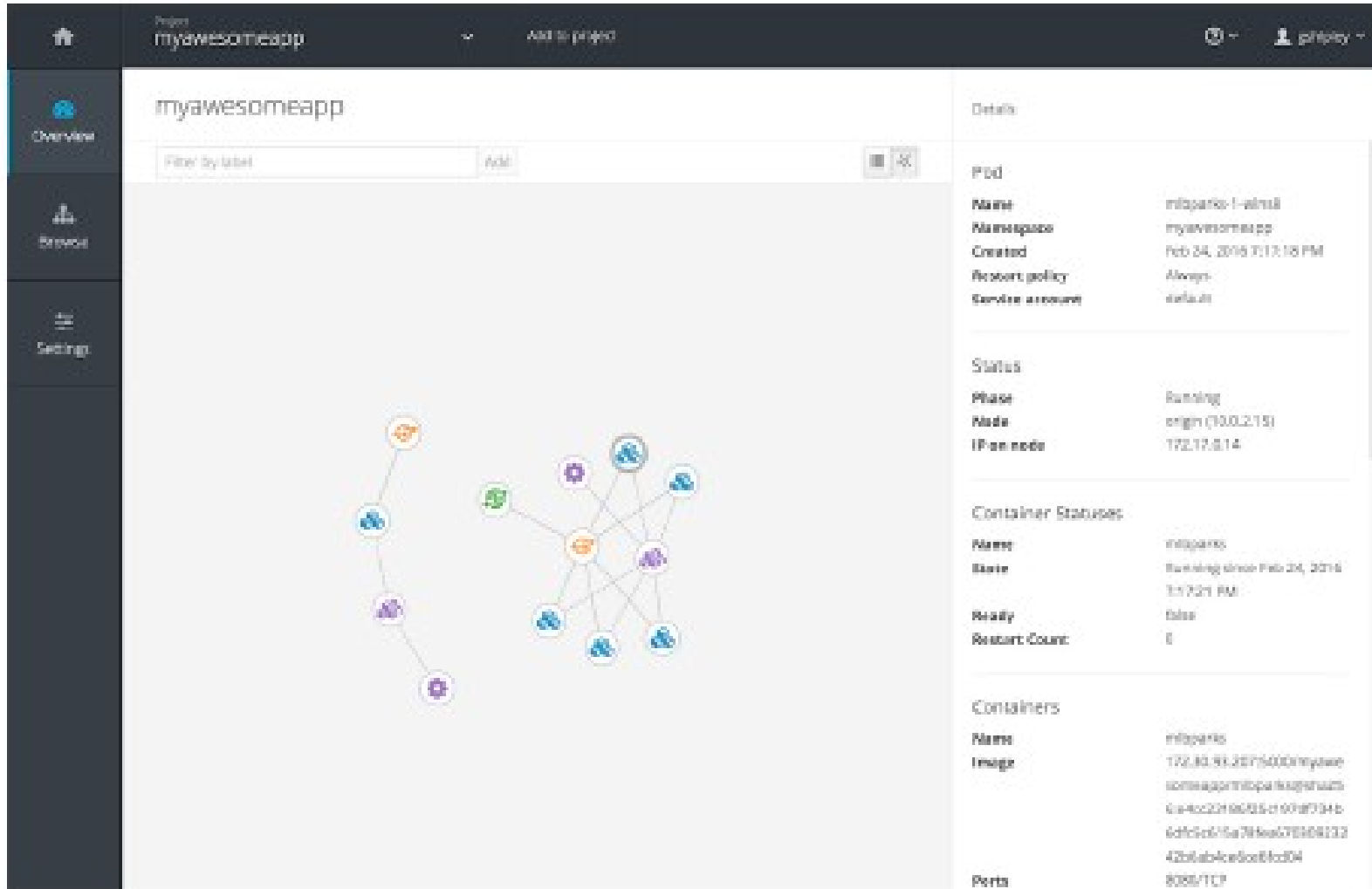
Realize the power of the open source community to jump-start modern application development.

LEARN MORE

...







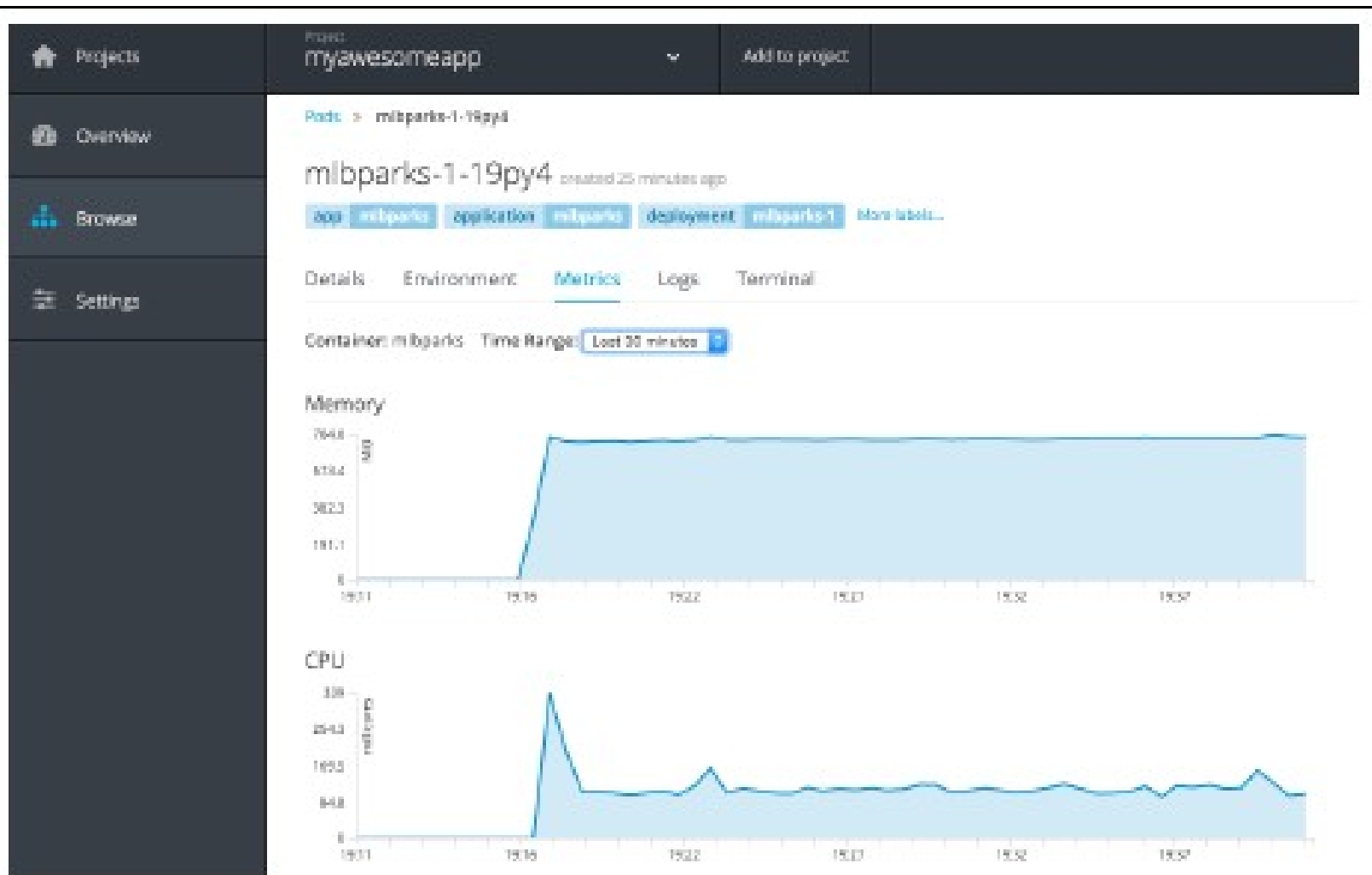
The screenshot shows the Integrata application visualization interface for a project named 'myawesomeapp'. The interface is divided into three main sections: a left sidebar, a central visualization area, and a right-hand details panel.

Left Sidebar: Contains navigation icons for 'Overview' (selected), 'Service', and 'Settings'.

Central Visualization Area: Displays a network diagram of the application. It features a central orange node connected to several blue nodes, which are further connected to other nodes, forming a complex web. A 'Filter by label' input field and an 'Add' button are located at the top of this area.

Right-hand Details Panel: Provides detailed information about the application and its components.

- Pod:**
 - Name: mtoparko-1-wlnu8
 - Namespace: myawesomeapp
 - Created: Feb 24, 2016 7:17:18 PM
 - Restart policy: Always
 - Service account: default
- Status:**
 - Phase: Running
 - Node: origin (10.0.2.15)
 - IP on node: 172.17.0.14
- Container Statuses:**
 - Name: mtoparks
 - State: Running since Feb 24, 2016 7:17:21 PM
 - Ready: false
 - Restart Count: 0
- Containers:**
 - Name: mtoparks
 - Image: 172.30.93.257:5000/myawesomesappmtoparksupstrans6a4cc23f66015e160af734b6d0c6c15a78ba67810813242b5abce5dc86cd04
- Ports:** 8085/TCP



5

JAVA ENTERPRISE ANWENDUNGEN

5.1

ÜBERWACHUNG VON JAVA- PROZESSEN MIT JMX

- Docker liefert eine Überwachungsmöglichkeit auf Ebene der Container
- Diese Überwachung ist jedoch für Java-Prozesse nicht sonderlich aufschlussreich
 - Metriken der Java-Virtual Machine
 - Heap-Speicher
 - Garbage Collections
- Hierfür wird besser JMX benutzt
 - Idee: Auf einem (im Unternehmen standardisierten) Port wird JMX beispielsweise mit Jolokia bereitgestellt
 - Hierfür existieren aber auch andere Lösungen

[Home](#) [Download](#) [Features](#) [Documentation](#) [Support](#) [Blog](#) [About](#)

Jolokia

[Download](#)
[Features](#)
[Support](#)
[Forum](#)
[IRC](#)
[License](#)

Documentation

[Overview](#)
[Tutorial](#)
[Reference Manual](#)
[Talks and Screencasts](#)

Agents

[Overview](#)
[Web Archive \(war\)](#)
[Osgi](#)
[JVM](#)
[Mule](#)

Jolokia is remote JMX with JSON over HTTP.

It is fast, simple, polyglot and has unique features. It's JMX on Capsaicin.

Jolokia is a JMX-HTTP bridge giving an alternative to JSR-160 connectors. It is an agent based approach with support for many platforms. In addition to basic JMX operations it enhances JMX remoting with unique features like bulk requests and fine grained security policies.

Starting points

- Overview of **features** which make Jolokia unique for JMX remoting.
- The **documentation** includes a **tutorial** and a **reference manual**.
- **Agents** exist for many platforms (JEE, OSGi, Mule, JVM).
- **Support** is available through various channels.
- **Contributions** are highly appreciated, too.

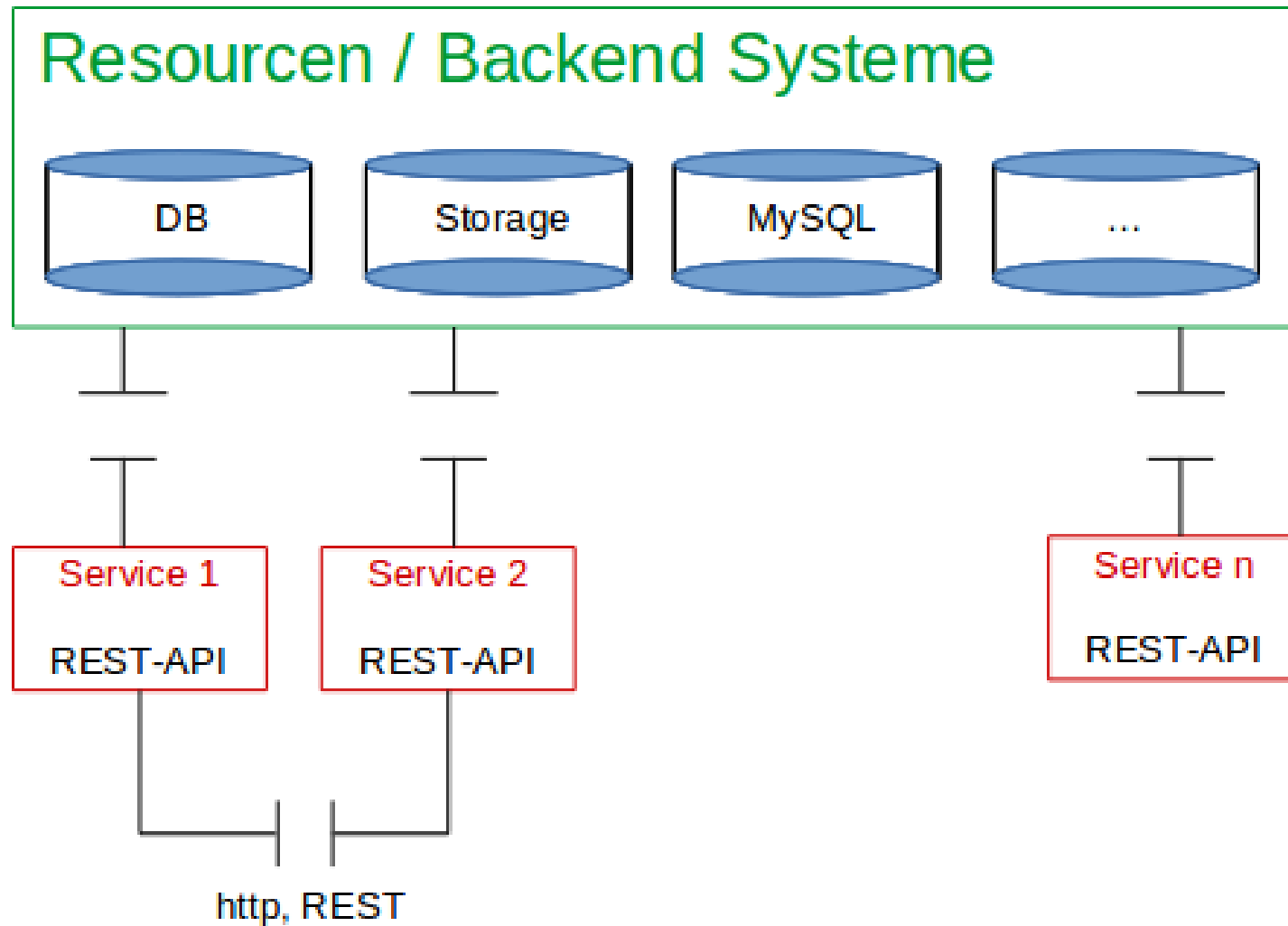
News

```
FROM openjdk:latest
RUN mkdir /in-dir
ADD org.javacream.training.docker.jmx-1.0-SNAPSHOT.jar
app.jar
ADD libs/jolokia-jvm-1.3.5-agent.jar jolokia.jar
VOLUME /in-dir
EXPOSE 7777
ENTRYPOINT java
    -javaagent:jolokia.jar=port=7777,host=0.0.0.0
    -cp app.jar
    Application
```

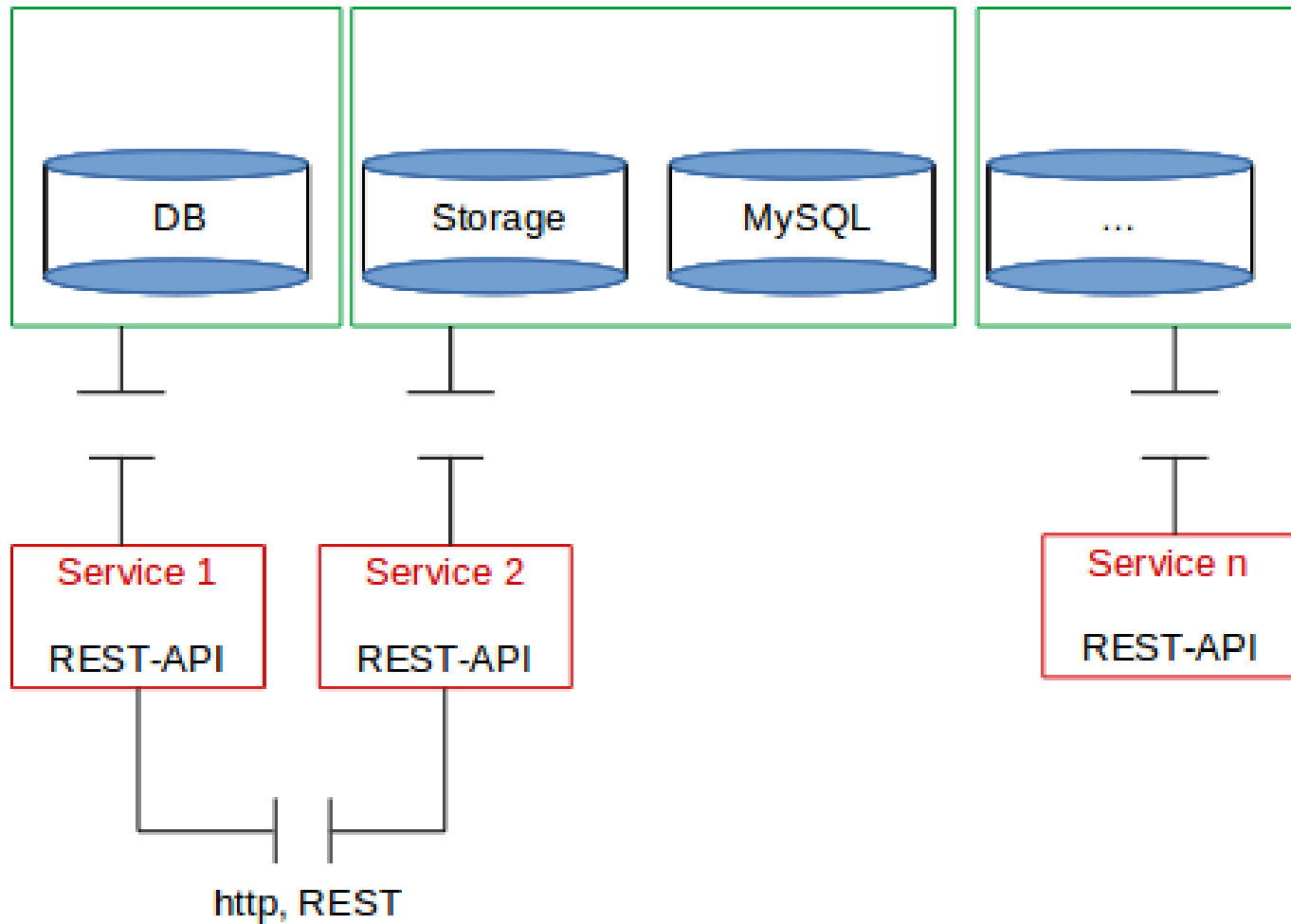
5.2

ENTERPRISE JAVA

- Enterprise-Applikationen sind meistens mehrschichtig gestaltet
 - Web Frontend
 - Service Frontend
 - z.B. RESTful Web Services
 - Transaktionelle Datenzugriffe
 - Backend-Ressourcen



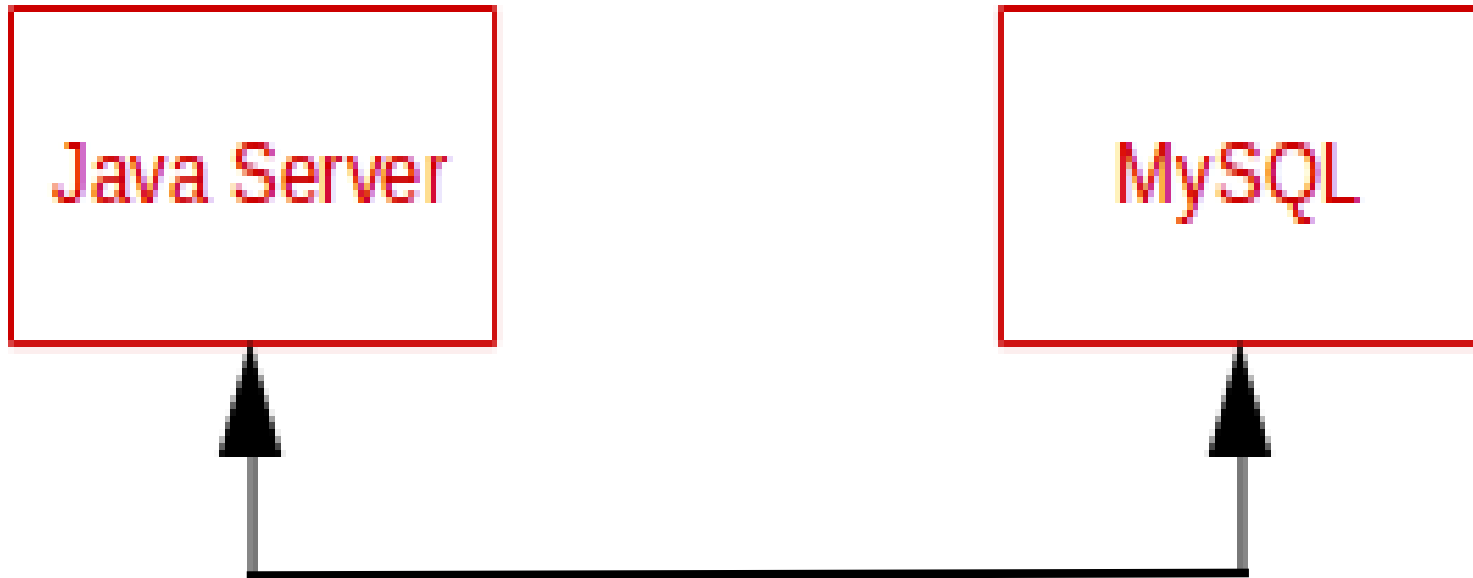
Microservices mit gekapselten Ressourcen



Realisierung mit Docker Image-Hierarchie



Realisierung mit verlinkten Containern



5.3

APPLIKATIONSSERVER

- Applikationsserver sind Bestandteil der Java Enterprise Edition
 - Eine Spezifikation
- Ausprägungen:
 - Web Profile
 - Web Anwendungen
 - Web Services
 - Full Profile
 - Transaktionelle Ressourcen-Zugriffe
 - Messaging
 - Weitere inoffizielle Profile
 - Reine Messaging Systeme
 - Web Profile mit einfachem Transaktionsmanagement

- Ausbringen von Anwendungen in den laufenden Server
 - Hot Deployment
 - File-Transfer in ein überwachtes Verzeichnis
 - Administrativer Vorgang
 - Web Console
 - Admin-Skripte
 - Konfigurativ
 - mit Neustart des Servers

- Der Server ist über JMX zugreifbar
- Web Konsolen
- Log-Dateien
- Betriebssystem-Überwachung
 - Speicher
 - CPU
 - IO
- Bei Verwendung des Docker-Containers übernimmt dieser die Rolle des Betriebssystems!

- Es existieren fertige Images für gängige Open Source Applikationsserver
 - Apache Tomcat
 - JBoss/Wildfly
 - Glassfish
 - Active MQ
 - ...
- Deployment durch
 - Erzeugen eines neuen Containers
 - Verwendung von Docker-Volumes
- Überwachung
 - Mappen der Ports und somit Netzwerk-Zugriff
 - Verzeichnis der Log-Dateien wird extern gemapped

5.4

SPRING

- Ein weit verbreitetes Open Source-Framework als Alternative zum JEE-Applikationsserver
- Spring-Core-Komponente und viele assoziierte Projekte
 - Spring Data
 - Spring MVC
 - Spring Integration
 - ...

- Eine Spring-Boot-Applikation wird als ein einziges Deployment-Archiv ausgeliefert
 - und über einen simplen Java-Aufruf gestartet
- Damit passt Spring Boot hervorragend zu Docker
 - Als Grundlage dient ein simples Java Image
 - Die zu installierende Applikation wird als eine einzige Datei hinzugefügt
 - und über einen Standard-Java-Aufruf gestartet