Docker – Containervirtualisierung leicht gemacht

CeBIT 2015

18. März 2015

Michel Rode
Linux/Unix Consultant & Trainer
B1 Systems GmbH
rode@b1-systems.de

Vorstellung B1 Systems

- gegründet 2004
- primär Linux/Open Source-Themen
- national & international tätig
- über 60 Mitarbeiter
- unabhängig von Soft- und Hardware-Herstellern
- Leistungsangebot:
  - Beratung & Consulting
  - Support
  - Entwicklung
  - Training
  - Betrieb
  - Lösungen
- dezentrale Strukturen

Schwerpunkte

- Virtualisierung (XEN, KVM & RHEV)
- Systemmanagement (Spacewalk, Red Hat Satellite, SUSE Manager)
- Konfigurationsmanagement (Puppet & Chef)
- Monitoring (Nagios & Icinga)
- IaaS Cloud (OpenStack & SUSE Cloud & RDO)
- Hochverfügbarkeit (Pacemaker)
- Shared Storage (GPFS, OCFS2, DRBD & CEPH)
- Dateiaustausch (ownCloud)
- Paketierung (Open Build Service)
- Administratoren oder Entwickler zur Unterstützung des Teams vor Ort

Docker – Build, Ship and Run Applications

Was ist Docker?

- Open Source Engine zum Standardisieren von Prozessumgebungen unter Linux
- „*build, ship and run (distributed) applications*“
- offene Plattform für Entwickler und Systemadministratoren
- nutzt Container-Technologien (LXC/libcontainer)
- einfaches Erstellen und Teilen von Container Images
- Docker ist *kein* virtueller Server! ($\Rightarrow$ wenig Overhead)
- „Docker“ bezeichnet sowohl den Server-Daemon als auch den Client samt CLI
- Open-Source-Projekt, veröffentlicht unter der Apache-2.0-Lizenz

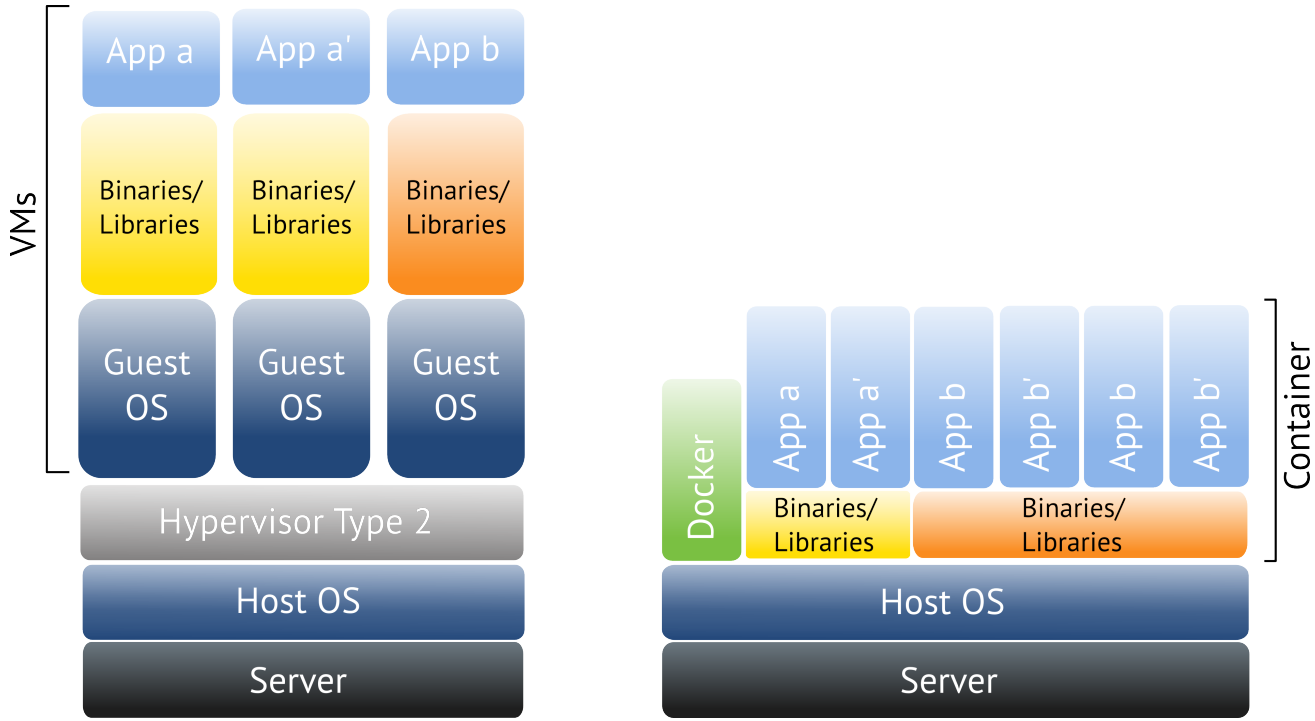
Eigenschaften

- Docker ermöglicht ein automatisches Deployment einer standardisierten Prozessumgebung für Linux-Anwendungen.
- Alle Linux-Anwendungen laufen in Docker.
- Docker läuft auf allen gängigen Linux-Distributionen.
- Alle benötigten Funktionen befinden sich innerhalb des Containers:
  - Bibliotheken
  - Abhängigkeiten
  - Binärdateien
  - ...
- Container sind auf die Architektur der Host-Plattform beschränkt.

Vorteile

- klein
- schnell
- sicher
- flexibel
- reproduzierbar
- portabel

VMs vs. Container



VMs komplette Maschine einschl. Kernel wird mit Hilfe eines Hypervisors virtualisiert.

Container nur Prozesse werden virtualisiert, das OS und Binaries/Libraries gemeinsam genutzt.

Einsatzbereiche

Docker ermöglicht ein schnelles Bereitstellen reproduzierbarer (Laufzeit-)Umgebungen für Entwickler und Systemadministratoren:

- Evaluations-Systeme
- POC-Systeme
- Test-Systeme
- QA-Systeme
- Live-Umgebungen
- Konfigurationstests
- ...

Container Execution Environments

- Docker kann derzeit zwei *Execution Driver* für die Erstellung von Containern nutzen:

  LXC
  : bewährte leichtgewichtige Prozessvirtualisierung im Linux-Kernel

  Libcontainer
  : neue Eigenentwicklung von Docker zur Nutzung der Kernel-Features

- Libcontainer ist Default seit Version 1.0
- Ein einzelner Kernel sorgt für die Limitierung und Isolation der Ressourcen.
  - ⇒ Emulation von Hardware entfällt
  - ⇒ Performance-Overhead wird minimiert

Linux Containers (LXC)

- leichtgewichtiger Virtualisierungsmechanismus (Prozessvirtualisierung) mit modularem Design
- verwaltet Ressourcen und Prozessgruppen voneinander isoliert in so genannten *Containern*
- nutzt vorhandene Kernel-Features wie Capabilities, Control Groups und Namespaces
- „*chroot on steroids*“ (`http://lxc.sourceforge.net`)
- seit 2.6.29 Teil des Upstream Kernels (und damit aller gängigen Distributionen)
- freie Software (größtenteils GNU-LGPL-Lizenz)

libcontainer I

- Go-Implementation, um Linux-Kernel-Namespaces ohne weitere Abhängigkeiten für Container zu nutzen
- vollständig unabhängig von LXC
- verwaltet als Teil von Docker die nötigen Kernel-Features:
  - Capabilities
  - Control Groups
  - Namespaces
  - Apparmor-/SELinux-Profile
  - Netzwerkschnittstellen
  - Firewall-Regeln

libcontainer II

- Grundlage für die Windows-Implementierung
- Grundlage für weitere Entwicklungen
- Red Hat, Google, Canonical & Parallels

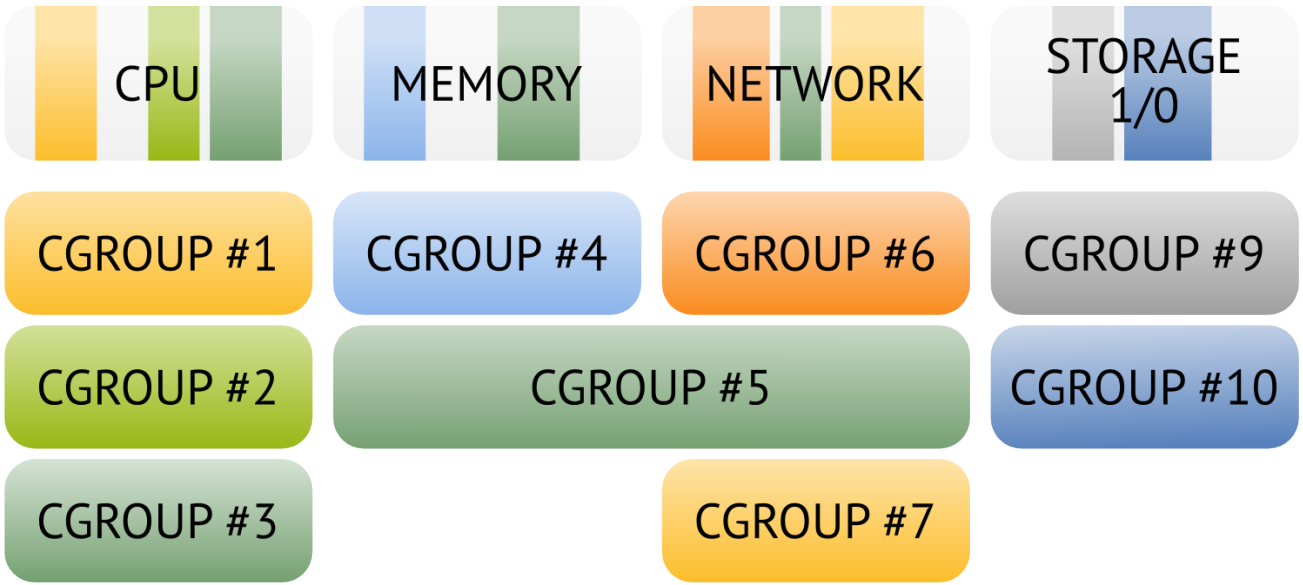
Capabilities

- teilen die Rechte eines privilegierten Benutzers (`root`) in bestimmte Teilbereiche auf
- können für unterschiedliche Prozesse individuell gesetzt werden
  $\Rightarrow$ geringere Gefahr eines Missbrauchs von Rechten
- da Container = Prozesse besteht die Möglichkeit, bestimmte Rechte innerhalb eines Containers einzuräumen
- minimieren Wechselwirkungen (z. B. beim Mounten des root-Dateisystems mit `read only` beim Herunterfahren eines Containers)

CGroups 1/2

- fassen Prozesse in einer hierarchischen Struktur zusammen.
- ermöglichen es, den Zugriff dieser Prozesse auf Ressourcen zu beschränken oder zu unterbinden und so die Nutzung dieser Ressourcen zu limitieren.
- Typische Ressourcen sind Prozessoren, Arbeitsspeicher, Netzwerkressourcen oder `Kernel Namespaces`.
- Die hierarchische Struktur ermöglicht es, Eigenschaften von Subsystemen auf mehreren Ebenen zu unterteilen.
- Zu jeder dieser Ebenen lässt sich die Nutzung der möglichen Ressourcen weiter einschränken.
- Prozesse (Container) lassen sich dann gezielt in die Tasklist der betreffenden CGroup aufnehmen.

CGroups 2/2


CPU
MEMORY
NETWORK
STORAGE 1/0
CGROUP #1
CGROUP #4
CGROUP #6
CGROUP #9
CGROUP #2
CGROUP #5
CGROUP #10
CGROUP #3
CGROUP #7

Chroot

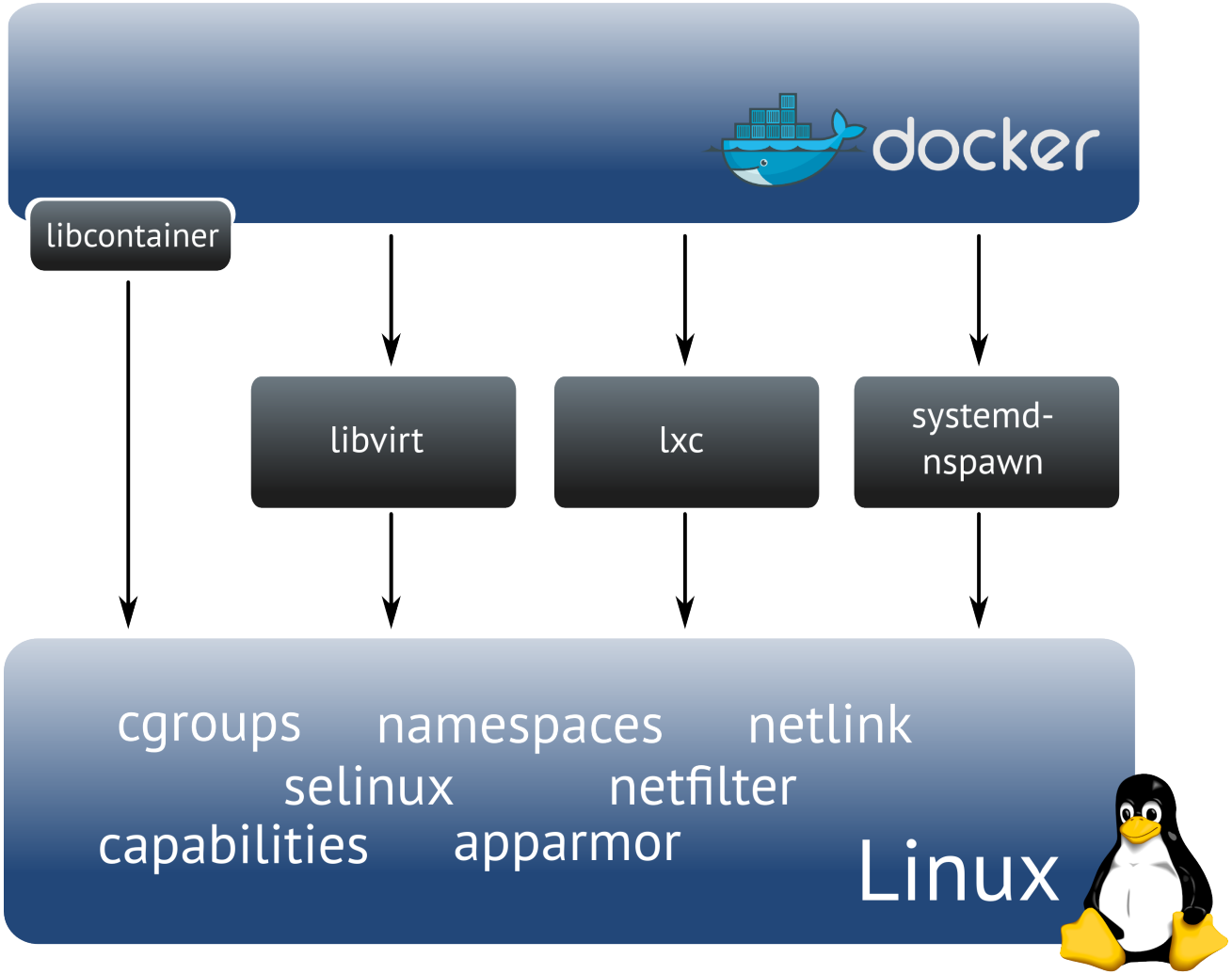
- Funktion unter Unix-Systemen, um im laufenden System ein neues Rootverzeichnis zuzuweisen
- primär für das Aufsetzen virtueller Umgebungen entwickelt
- einfache Möglichkeit, nicht vertrauenswürdige Programme zu isolieren (aka. „Sandbox“)
- einfacher Jail-Mechanismus – aus dem aber durchaus ausgebrochen werden kann
- Prozess im Chroot-Verzeichnis kann nicht mehr auf Dateien außerhalb dieses Verzeichnisses zugreifen
- Programm muss im Chroot-Verzeichnis eine komplette Umgebung vorfinden (Platz für temporäre Dateien, Konfigurationsdateien, Programmbibliotheken, . . .)
- keine Ressourcen-Limitierung

Kernel Namespaces

- Integration von Namespaces im Kernel mit Version 2.6.19
- Namespaces helfen, Prozesse voneinander zu isolieren
- mögliche Eigenschaften zum Definieren eines Namensraumes:

`ipc`	System-V-Interprozess-Kommunikation (IPC)
`uts`	Hostname
`mnt`	Mountpoints und Dateisysteme
`pid`	Prozesse
`net`	Netzwerk-Stack
`user`	Benutzer (UIDs)

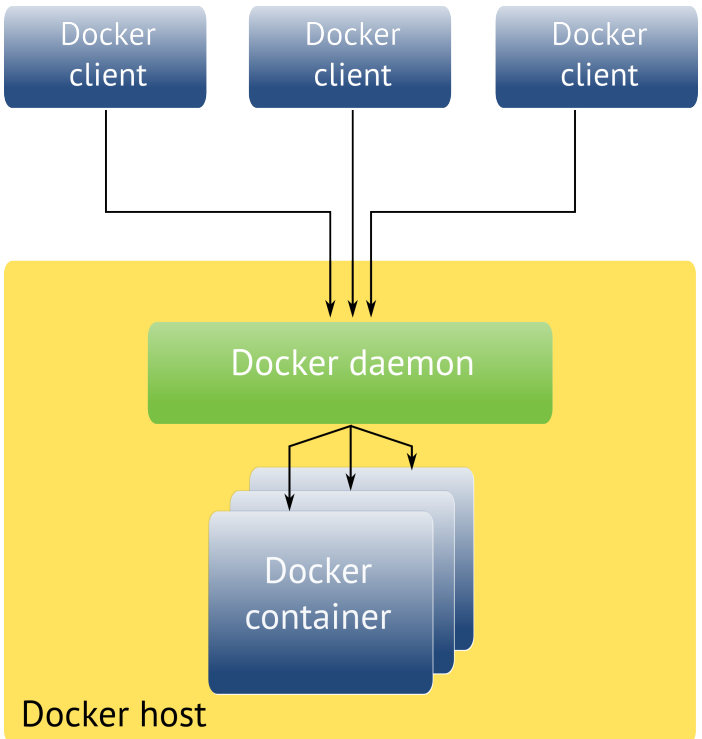
Übersicht


docker
libcontainer
libvirt
lxc
systemd-
nspawn
cgroups
namespaces
netlink
selinux
netfilter
capabilities
apparmor
Linux

Images, Container und Registries

- *Docker Images*
  - read only Template
  - Basis für Container
- *Docker Container*
  - wird auf Basis eines Image und Dockerfile erzeugt
  - hält zur Laufzeit die Änderungen vor (Read-/Write-Layer)
  - Änderungen können in einem neuen Image übernommen („committed“) werden
- *Docker Registries*
  - Image Repository
  - privat und öffentlich
  - zentrale Ressource: *Docker Hub*

Bestandteile – Die Docker-Engine



Die Docker Engine besteht aus zwei Teilen:

Server Daemon für den Serverprozess zur Verwaltung der Container

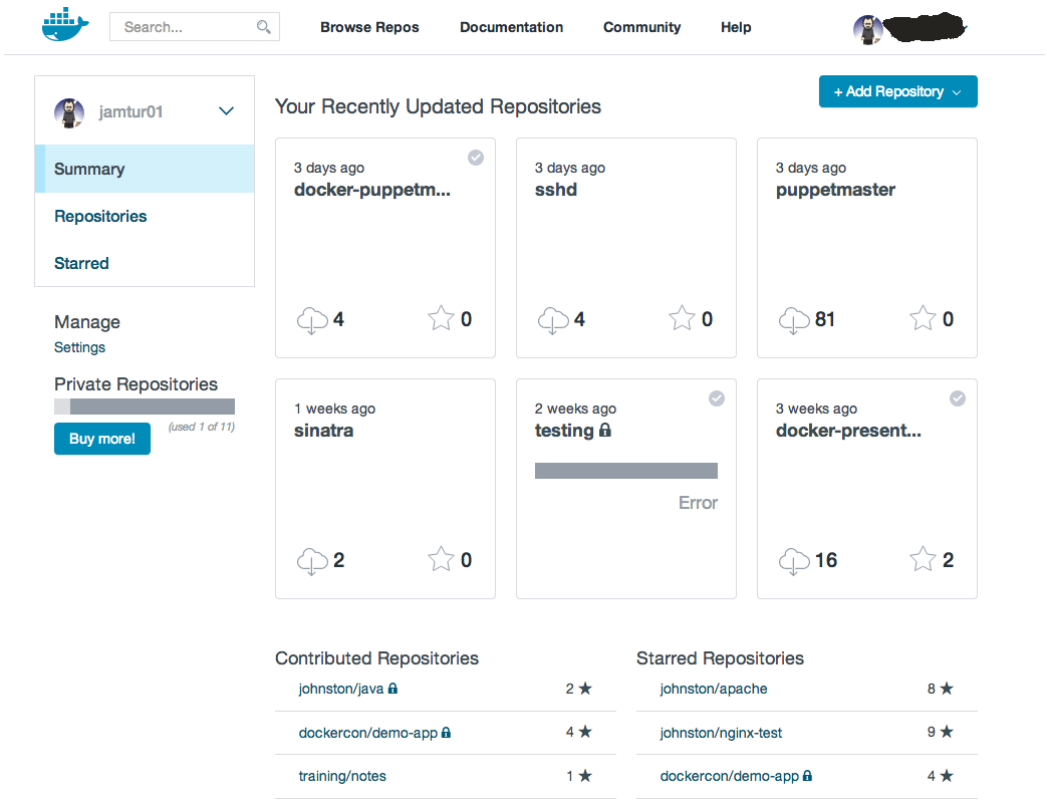
Client Client zur (Fern-)Steuerung des Daemons

Docker und die weitere Welt

Docker ist kollaborativ, modular und erweiterbar:

Atomic	atomar aufgebautes Hostsystem für Docker-Container
CoreOS	minimales Linux als Container-Betriebssystem
Gitlab	Collaboration on Code
Jenkins	Continuous Integration System für Servlet Container (z. B. Apache Tomcat)
Puppet	Konfigurationsmanagement
Fig	Orchestration
Packer	Erstellung fertiger Images mittels Templates
	...

Docker Hub


Search...
Browse Repos
Documentation
Community
Help
jamtur01
Summary
Repositories
Starred
Manage
Settings
Private Repositories
Buy more!
(used 1 of 11)
Your Recently Updated Repositories
+ Add Repository
3 days ago
docker-puppetm...
4
0
3 days ago
sshd
4
0
3 days ago
puppetmaster
81
0
1 weeks ago
sinatra
2
0
2 weeks ago
testing
Error
3 weeks ago
docker-present...
16
2
Contributed Repositories
johnston/java 2 ★
dockercon/demo-app 4 ★
training/notes 1 ★
Starred Repositories
johnston/apache 8 ★
johnston/nginx-test 9 ★
dockercon/demo-app 4 ★

Konfiguration

Container – Ressourcen (Default)

CPU	1024 CPU-Shares für neue Container durch CGroups (Default)
Speicher	gesamter Arbeitsspeicher des Hosts nutzbar (Default)
Block Devices	keine Bandbreiteneinschränkung beim Lese- und Schreibzugriff auf Blockgeräte (Default)
Netzwerk	automatisch über Linux-Bridge `docker0` mit dem Netzwerk des Hosts verbunden (Default)
Storage	10 GB Festplattenplatz pro Container (Default)

Treiber für Storage

Der Docker-Daemon unterstützt drei verschiedene Treiber für Storage:

aufs	alt, nicht im Kernel (nur via Patchset), schnell (!!!)
devicemapper	Thin Provisioning und Copy on Write (CoW), Snapshots, Direct- &Loop-LVM
btrfs	sehr schnell, Cow, Snapshots, default teilweise
OverlayFS	sehr schnell, in der Entwicklung

Treiber für Storage

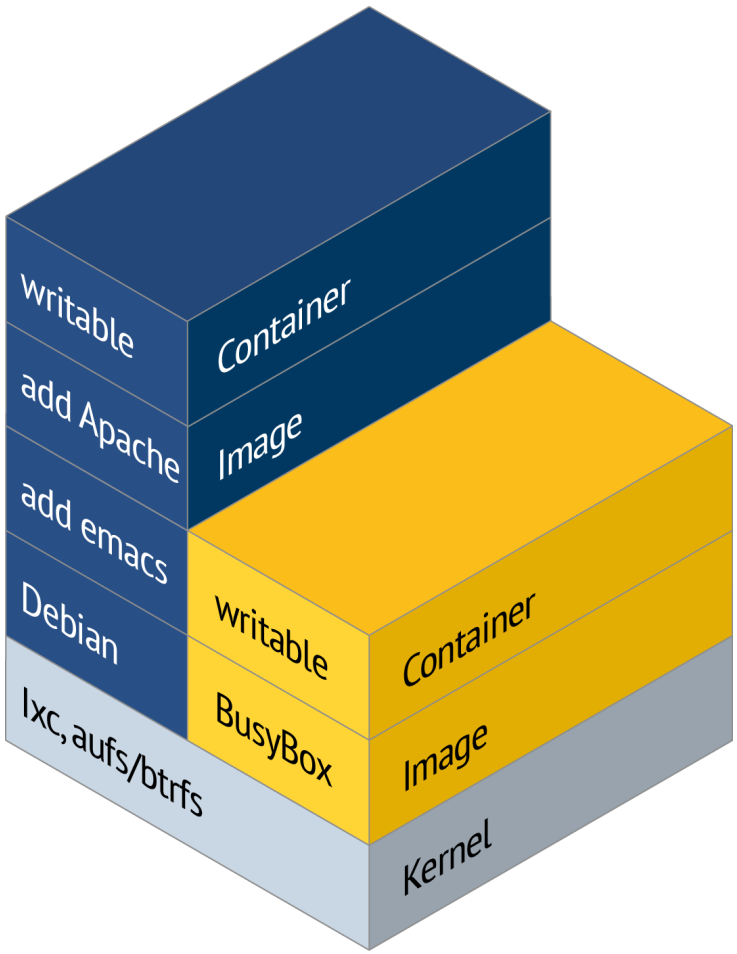
Storage-Treiber setzen:

```
# docker -s <devicemapper|btrfs|overlayfs...>
```

Storage-Informationen abfragen:

```
# docker info
[...]
Storage Driver: devicemapper
 Pool Name: docker-253:2-1443148-pool
 Pool Blocksize: 65.54 kB
 Data file: /var/lib/docker/devicemapper/devicemapper/data
 Metadata file: /var/lib/docker/devicemapper/devicemapper/metadata
 Data Space Used: 5.739 GB
 Data Space Total: 107.4 GB
 Metadata Space Used: 5.976 MB
 Metadata Space Total: 2.147 GB
```

Storage, Images und Container



- layerbasiert
- bis auf den obersten alle read-only
- beinhalten Änderungen/Diffs
- normalerweise als Base-Image-Scratch

Dockerfiles

Dockerfiles – Grundlagen

- Bauanleitung für ein Image
- enthält die notwendigen Anweisungen zum Erstellen eines Image
- stellt so die Reproduzierbarkeit eines Image bei jeder neuen Erstellung sicher
- Anweisungen sind einzeilig und beginnen mit einem Schlüsselwort
- Anweisungen sind nicht case-sensitive (Schlüsselwörter gewöhnlich groß)
- Kommentare werden durch ein #-Zeichen eingeleitet

Dockerfiles verwenden

Minimalistisches Dockerfile:

```
# johndoe/nodejs
# VERSION 0.0.1

FROM <repo>
MAINTAINER John Doe <johndoe@b1-systems.de>
```

Dockerfile verwenden (hier: aktuelles Verzeichnis):

```
$ docker build .
```

Dockerfiles – Anweisungen

`ADD/COPY`	Kopieren lokaler Dateien vom Host in den Container
`CMD`	Aufruf für Programme innerhalb eines Containers
`EXPOSE`	Portadresse für das Netzwerk
`FROM`	Repository als Basis für das Image
`RUN`	(Kommandozeilen-)Befehl ausführen
`MAINTAINER`	Ersteller des Dockerfiles (optional)
`VERSION`	Versionsnummer (optional)

Dockerfiles – Beispiel

Dockerfile für MySQL-Server (Beispiel):

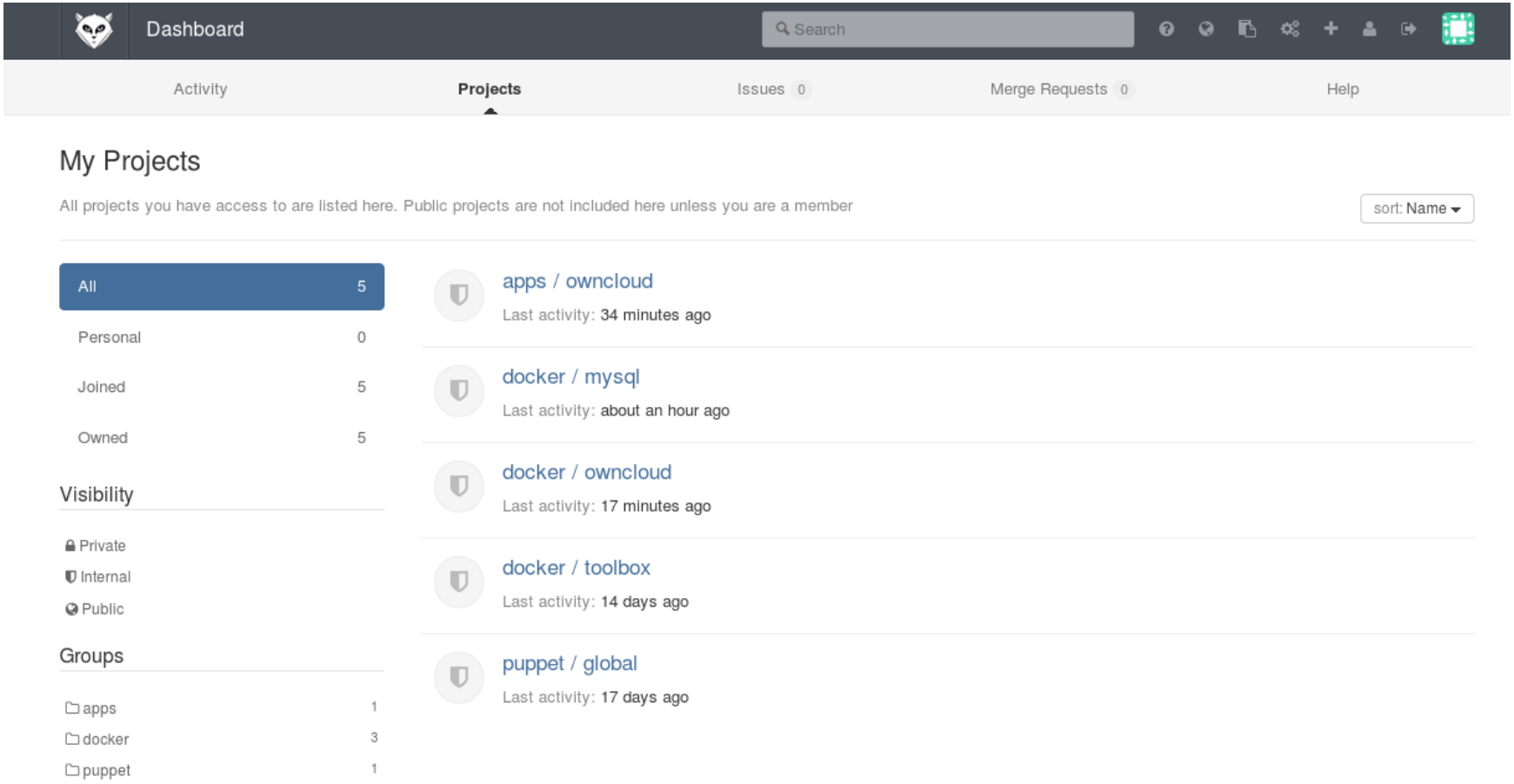
```
FROM rhel
MAINTAINER John Doe <info@b1-systems.de>
RUN yum install -y mysql mysql-server
ADD start.sh /start
RUN chmod +x /start
EXPOSE 3306
CMD ["/start"]
```

GitLab – kollaborativ entwickeln

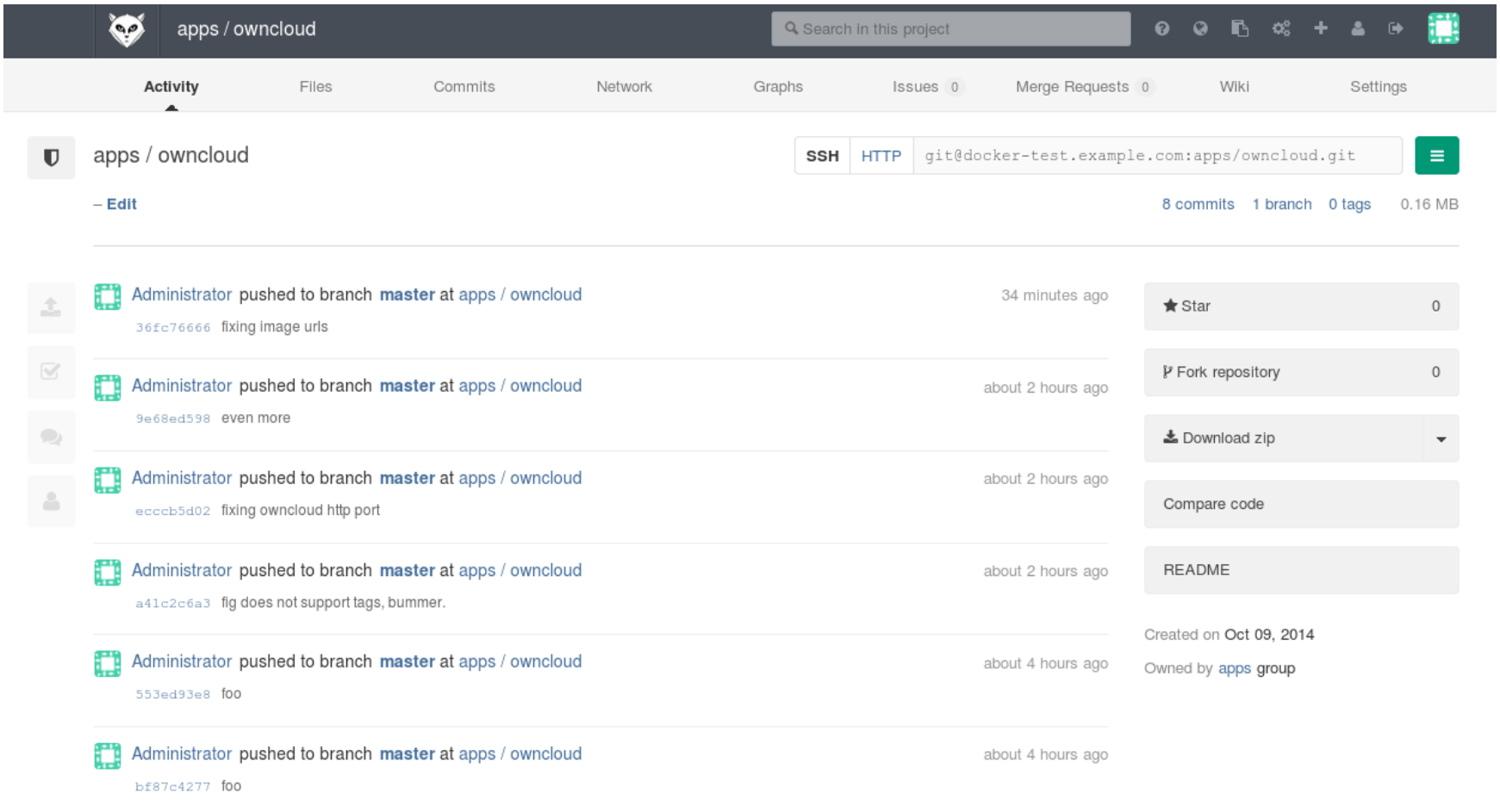
Features

- komplett frei und Open Source
- Verwalten und Durchsuchen von Git Repositories
- eigener Code auf eigenem Server
- Verwaltung von Zugriffsrechten
- Ausführen von Code Reviews und Merge Requests
- Hooks
- ...

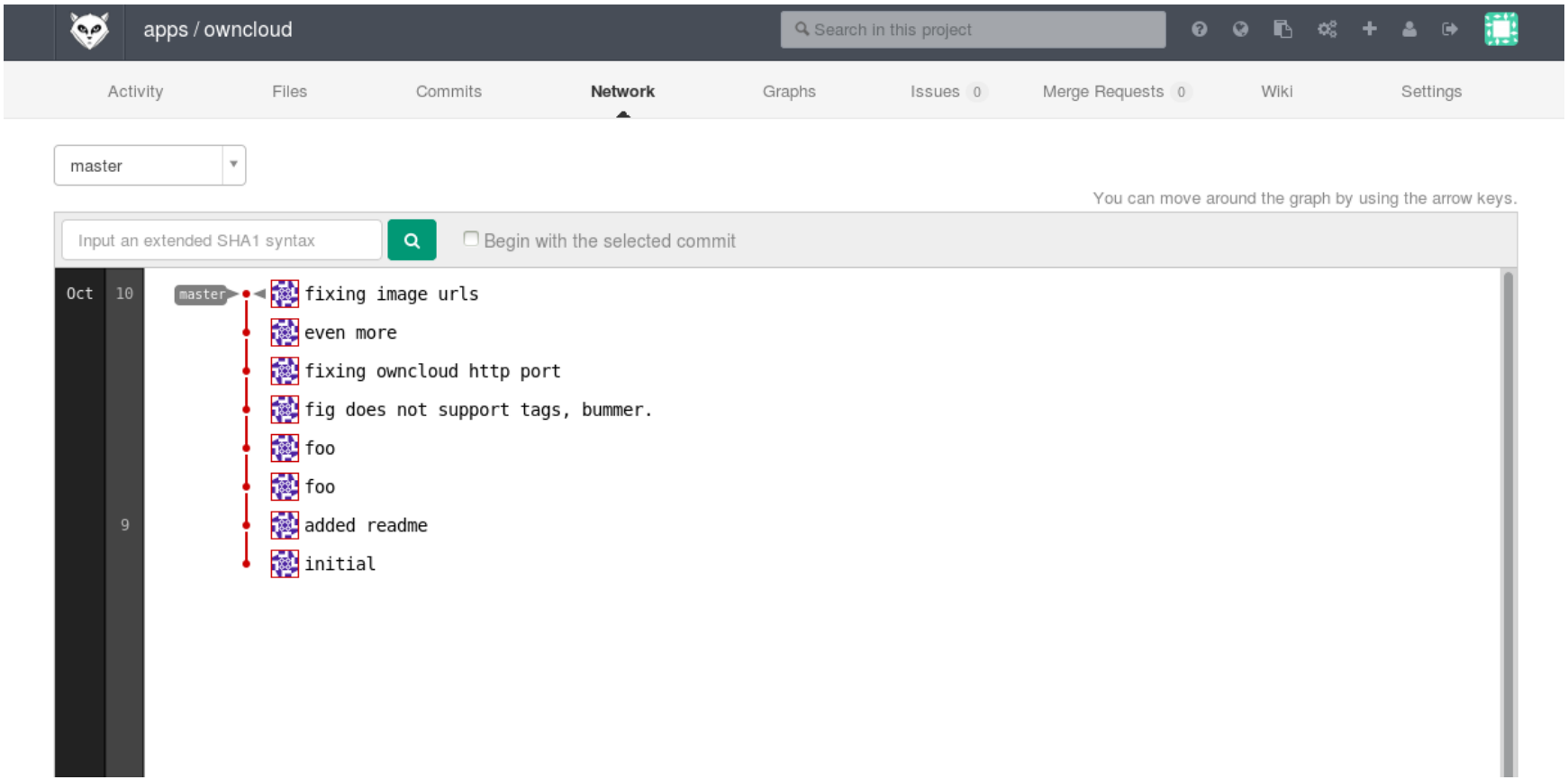
GitLab 1/3


Dashboard
Search
Activity
Projects
Issues 0
Merge Requests 0
Help
My Projects
All projects you have access to are listed here. Public projects are not included here unless you are a member
sort: Name
All 5
Personal 0
Joined 5
Owned 5
Visibility
Private
Internal
Public
Groups
apps 1
docker 3
puppet 1
apps / owncloud
Last activity: 34 minutes ago
docker / mysql
Last activity: about an hour ago
docker / owncloud
Last activity: 17 minutes ago
docker / toolbox
Last activity: 14 days ago
puppet / global
Last activity: 17 days ago

GitLab 2/3


apps / owncloud
Search in this project
Activity
Files
Commits
Network
Graphs
Issues 0
Merge Requests 0
Wiki
Settings
apps / owncloud
SSH
HTTP
git@docker-test.example.com:apps/owncloud.git
– Edit
8 commits
1 branch
0 tags
0.16 MB
Administrator pushed to branch master at apps / owncloud
34 minutes ago
36fc76666 fixing image urls
Administrator pushed to branch master at apps / owncloud
about 2 hours ago
9e68ed598 even more
Administrator pushed to branch master at apps / owncloud
about 2 hours ago
ecccb5d02 fixing owncloud http port
Administrator pushed to branch master at apps / owncloud
about 2 hours ago
a41c2c6a3 fig does not support tags, bummer.
Administrator pushed to branch master at apps / owncloud
about 4 hours ago
553ed93e8 foo
Administrator pushed to branch master at apps / owncloud
about 4 hours ago
bf87c4277 foo
Star 0
Fork repository 0
Download zip
Compare code
README
Created on Oct 09, 2014
Owned by apps group

GitLab 3/3

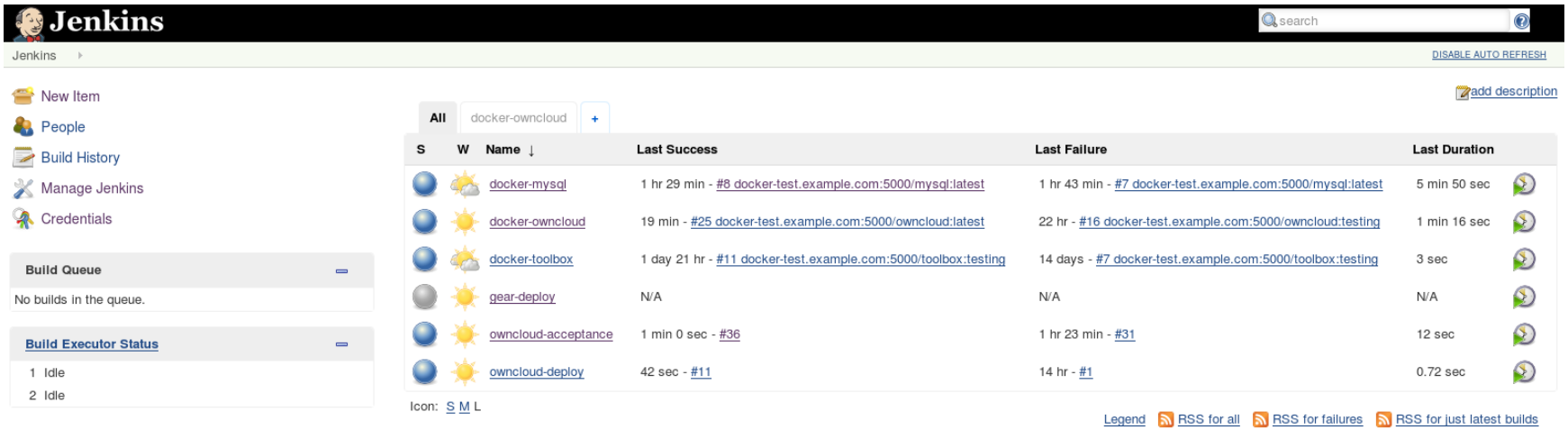

apps / owncloud
Search in this project
Activity
Files
Commits
Network
Graphs
Issues 0
Merge Requests 0
Wiki
Settings
master
You can move around the graph by using the arrow keys.
Input an extended SHA1 syntax
Begin with the selected commit
Oct 10
master
fixing image urls
even more
fixing owncloud http port
fig does not support tags, bummer.
foo
foo
9
added readme
initial

Jenkins – Continuous Integration

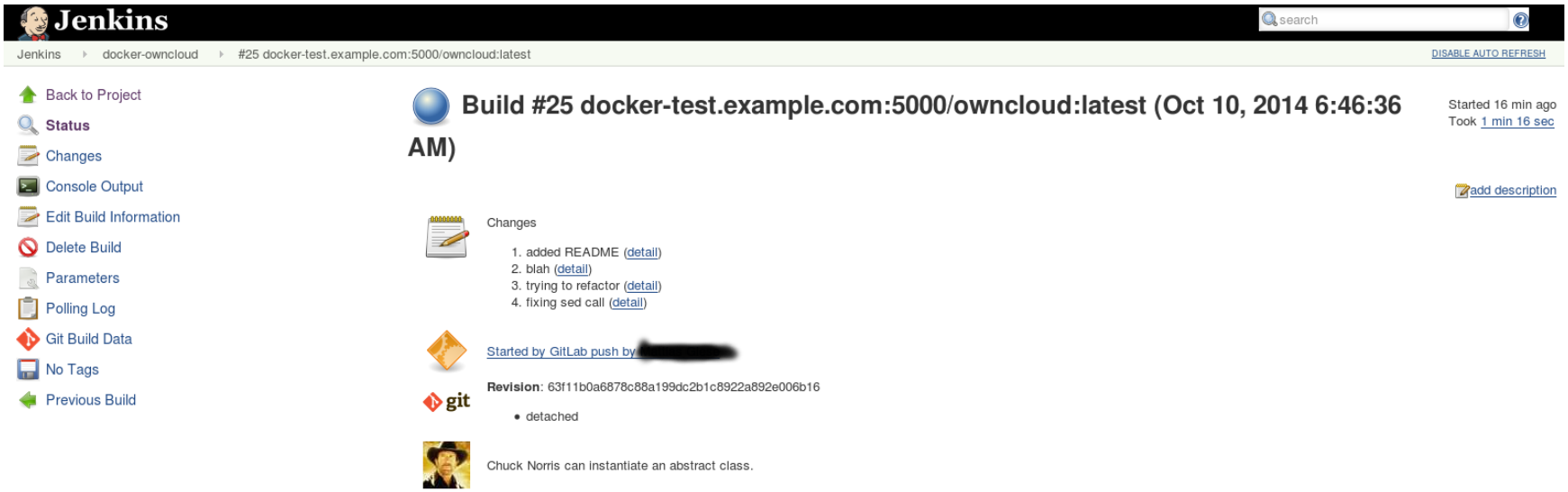
Was ist Jenkins?

- Jenkins ist ein serverbasiertes System für Continuous Integration, das in einem Servlet Container (wie Apache Tomcat) betrieben wird.
- Jenkins unterstützt SCM Tools wie AccuRev, CVS, Subversion, Git, Mercurial, Perforce, Clearcase und RCT und kann Apache Ant und Maven ausführen.
- Jenkins ist freie Software; lizenziert unter der MIT-Licence.
- Builds können auf unterschiedliche Weise gestartet werden, auch per Commit-Trigger eines Versionskontrollsystems.
- Jenkins überwacht die Ausführung wiederholter Jobs wie z. B. das Bauen eines Softwareprojekts oder Cronjobs.
- Jenkins ist in Java geschrieben.

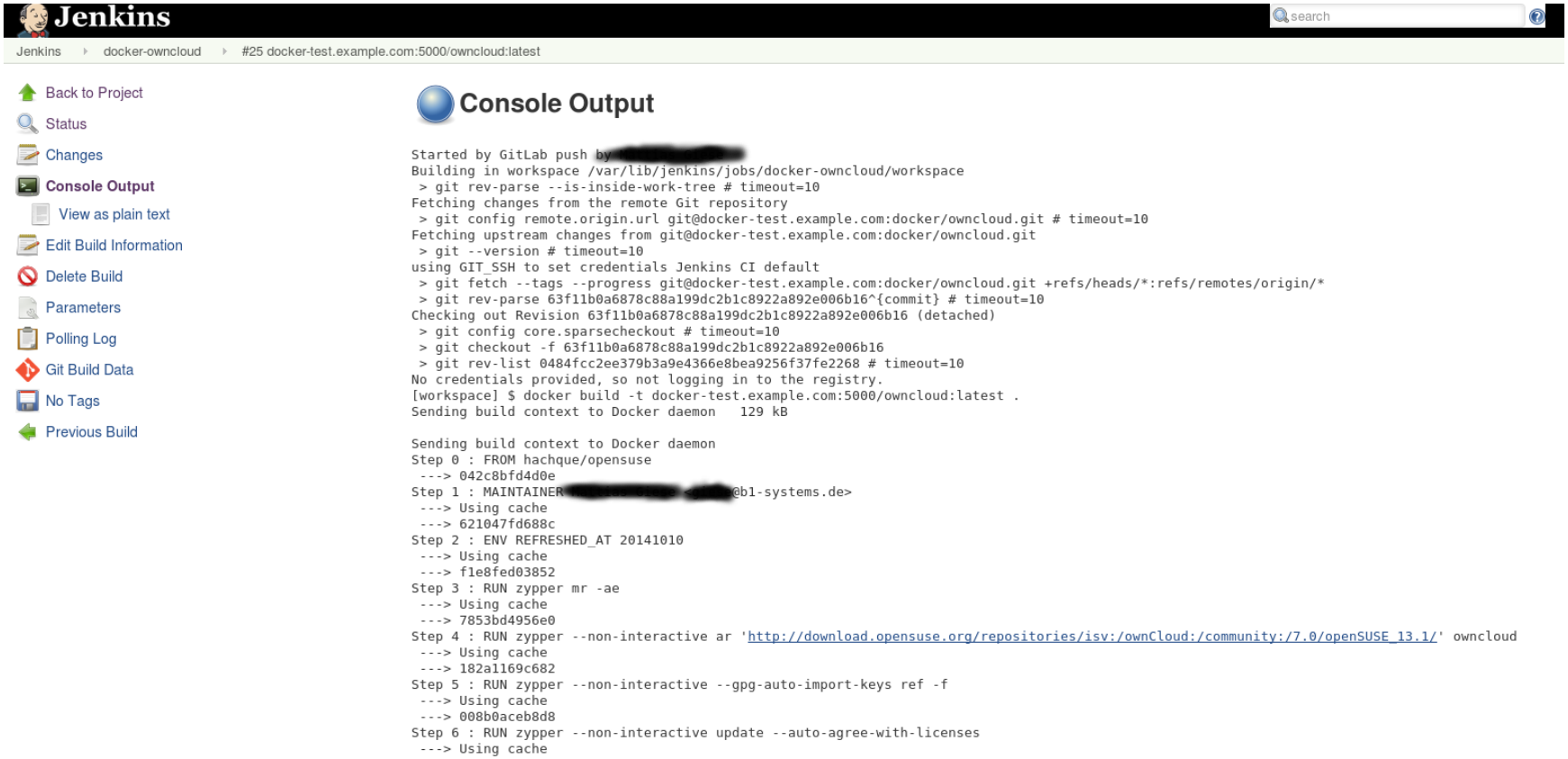
Jenkins 1/3


Jenkins
search
Jenkins
DISABLE AUTO REFRESH
New Item
People
Build History
Manage Jenkins
Credentials
add description
All
docker-owncloud
+
S W Name ↓ Last Success Last Failure Last Duration
docker-mysql 1 hr 29 min - #8 docker-test.example.com:5000/mysql:latest 1 hr 43 min - #7 docker-test.example.com:5000/mysql:latest 5 min 50 sec
docker-owncloud 19 min - #25 docker-test.example.com:5000/owncloud:latest 22 hr - #16 docker-test.example.com:5000/owncloud:testing 1 min 16 sec
docker-toolbox 1 day 21 hr - #11 docker-test.example.com:5000/toolbox:testing 14 days - #7 docker-test.example.com:5000/toolbox:testing 3 sec
gear-deploy N/A N/A N/A
owncloud-acceptance 1 min 0 sec - #36 1 hr 23 min - #31 12 sec
owncloud-deploy 42 sec - #11 14 hr - #1 0.72 sec
Build Queue
No builds in the queue.
Build Executor Status
1 Idle
2 Idle
Icon: S M L
Legend
RSS for all
RSS for failures
RSS for just latest builds

Jenkins 2/3


Jenkins
search
Jenkins ▸ docker-owncloud ▸ #25 docker-test.example.com:5000/owncloud:latest
DISABLE AUTO REFRESH
Back to Project
Status
Changes
Console Output
Edit Build Information
Delete Build
Parameters
Polling Log
Git Build Data
No Tags
Previous Build
Build #25 docker-test.example.com:5000/owncloud:latest (Oct 10, 2014 6:46:36 AM)
Started 16 min ago
Took 1 min 16 sec
add description
Changes
1. added README (detail)
2. blah (detail)
3. trying to refactor (detail)
4. fixing sed call (detail)
Started by GitLab push by [redacted]
git
Revision: 63f11b0a6878c88a199dc2b1c8922a892e006b16
• detached
Chuck Norris can instantiate an abstract class.

Jenkins 3/3


Jenkins
search
Jenkins ▸ docker-owncloud ▸ #25 docker-test.example.com:5000/owncloud:latest
Back to Project
Status
Changes
Console Output
View as plain text
Edit Build Information
Delete Build
Parameters
Polling Log
Git Build Data
No Tags
Previous Build
Console Output
Started by GitLab push by [redacted]
Building in workspace /var/lib/jenkins/jobs/docker-owncloud/workspace
> git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url git@docker-test.example.com:docker/owncloud.git # timeout=10
Fetching upstream changes from git@docker-test.example.com:docker/owncloud.git
> git --version # timeout=10
using GIT_SSH to set credentials Jenkins CI default
> git fetch --tags --progress git@docker-test.example.com:docker/owncloud.git +refs/heads/*:refs/remotes/origin/*
> git rev-parse 63f11b0a6878c88a199dc2b1c8922a892e006b16^{commit} # timeout=10
Checking out Revision 63f11b0a6878c88a199dc2b1c8922a892e006b16 (detached)
> git config core.sparsecheckout # timeout=10
> git checkout -f 63f11b0a6878c88a199dc2b1c8922a892e006b16
> git rev-list 0484fcc2ee379b3a9e4366e8bea9256f37fe2268 # timeout=10
No credentials provided, so not logging in to the registry.
[workspace] $ docker build -t docker-test.example.com:5000/owncloud:latest .
Sending build context to Docker daemon 129 kB
Sending build context to Docker daemon
Step 0 : FROM hachque/opensuse
---> 042c8bfd4d0e
Step 1 : MAINTAINER [redacted]@b1-systems.de>
---> Using cache
---> 621047fd688c
Step 2 : ENV REFRESHED_AT 20141010
---> Using cache
---> f1e8fed03852
Step 3 : RUN zypper mr -ae
---> Using cache
---> 7853bd4956e0
Step 4 : RUN zypper --non-interactive ar 'http://download.opensuse.org/repositories/isv:/ownCloud:/community:/7.0/openSUSE_13.1/' owncloud
---> Using cache
---> 182a1169c682
Step 5 : RUN zypper --non-interactive --gpg-auto-import-keys ref -f
---> Using cache
---> 008b0aceb8d8
Step 6 : RUN zypper --non-interactive update --auto-agree-with-licenses
---> Using cache

Jenkins Plugins

Docker Buildstep verschiedene Docker-Kommandos zu einem Job als Build Step hinzufügen

Docker publish Projekte mittels Dockerfile bauen und sie in die Docker Registry laden

Git Verwendung von Git als Build-SCM

Gitlab Build Trigger, der GitLab vorgaukelt, dass Jenkins ein GitLab CI ist

Build Pipeline Plugin bietet Ansicht der Build Pipeline von Upstream- und Downstream-verbundenen Jobs, die eine Build Pipeline bilden

Downstream-Ext Plugin unterstützt erweiterte Konfiguration zum Triggern von Downstream Builds.

Publish Over SSH Plugin Daten- und Dateitransfer (SSH)

Jenkins – Build Pipeline Plugin

- bildet die Build Pipeline ab
- erlaubt Festlegung manueller Trigger für Jobs, die vor der Ausführung ein Eingreifen erfordern, z. B. Abnahmeprozess außerhalb von Jenkins

Jenkins – Downstream-Ext Plugin

Dieses Plugin bietet weitreichende Konfigurationsmöglichkeiten zum Triggern von Downstream Builds:

- triggert Builds nur, wenn ein Downstream-Job Änderungen im SCM aufweist
- triggert Builds, wenn ein Upstream Build besser/gleich/schlechter als ein definiertes Ergebnis abschneidet (SUCCESS, UNSTABLE, FAILURE, ABORTED)
- für Matrix (aka Multi-Configuration) Jobs kann festgelegt werden, welcher Teil des Jobs den Downstream Job triggern soll: *parent only*, *configuration only* oder beide

Vielen Dank für Ihre Aufmerksamkeit!

Bei weiteren Fragen wenden Sie sich bitte an info@b1-systems.de
oder +49 (0)8457 - 931096.

Besuchen Sie uns auch hier auf der CeBIT,
Halle 6, H16/312.