Ausrollen von Multi-Tier-Applikationen mit Docker

CommitterConf 2015, Essen

10. November 2015

Mattias Giese
System Management & Monitoring Architect
B1 Systems GmbH
giese@b1-systems.de

Vorstellung B1 Systems

- gegründet 2004
- primär Linux/Open Source-Themen
- national & international tätig
- über 70 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

Agenda

- Docker – Kurzer Überblick
- Maschinen erstellen mit `docker-machine`
- mehrere Maschinen ansprechen mit `docker-swarm`
- Applikationsszenarien abbilden und implementieren mit `docker-compose`

Docker – Kurzer Überblick

„Old“ technology wrapped in some Hipster . . .

docker-machine

- baut Maschinen für den Einsatz mit Docker
- viele Treiber:
  - VirtualBox
  - VMware
  - DigitalOcean
  - Azure
  - AWS
  - ...
- kann auch bestehende Maschine konfigurieren (generic)
- automatische „Absicherung“ der Hosts (Authentifizierung/Autorisierung via TLS Zertifikate)

docker-machine

Maschine auf VirtualBox mit dem Namen „foobar“ erzeugen

```
$ docker-machine create -d virtualbox foobar1
```

docker-machine

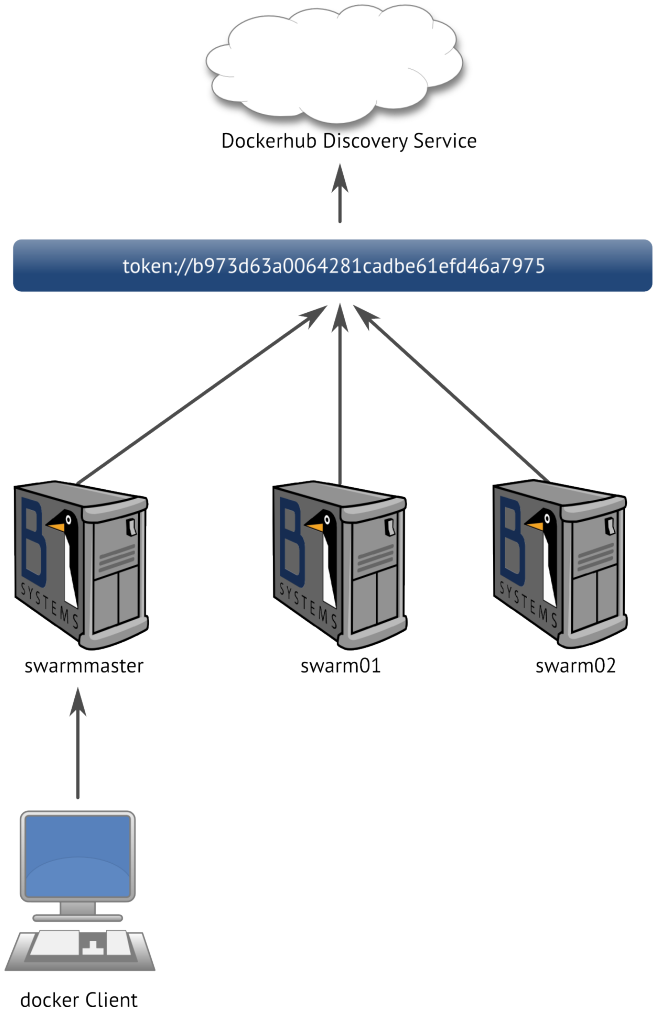
Verbindung zur Maschine

```
# docker-machine env foobar1
export DOCKER_TLS_VERIFY="1"
export DOCKER_HOST="tcp://192.168.99.100:2376"
export DOCKER_CERT_PATH="/home/mattias/.docker/
  machine/machines/foobar1"
export DOCKER_MACHINE_NAME="foobar1"

# Run this command to configure your shell:
# eval "$(docker-machine env foobar1)"
```

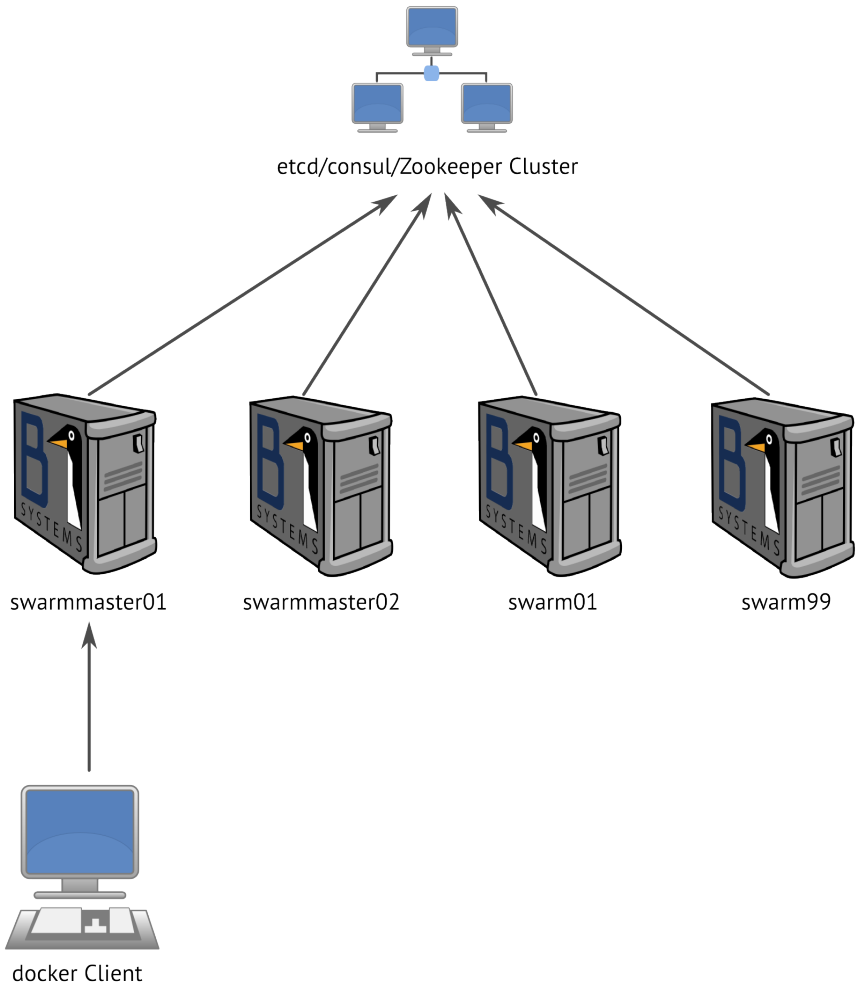
`docker-swarm`

- „Wrapper“-API für Docker
- Discovery via Docker Hub integriert
- eigene Discovery möglich:
  - statische Datei mit `Host:Port` Zeilen
  - Angabe auf Kommandozeile
  - etcd, Consul, ZooKeeper, ...

docker-swarm Docker Hub



docker-swarm etcd



`docker-swarm` – Discovery via Docker Hub

Discovery Token erstellen:

```
$ docker run swarm create
4754c10117f48dc759ec2f0989d1bdad
```

`docker-swarm` – Swarm Node

Swarm Node

```
$ docker run -d swarm join --addr=docker_ip:2375 \
  token://4754c10117f48dc759ec2f0989d1bdad
```

Docker muss via TCP lauschen!

`docker-swarm` – Swarm Manager

Swarm Manager

```
$ docker run -d swarm manager \
  token://4754c10117f48dc759ec2f0989d1bdad
```

docker-swarm via docker-machine

Master/Node

```
$ docker-machine create -d virtualbox \
  --swarm --swarm-master \
  --swarm-discovery \
  token://4754c10117f48dc759ec2f0989d1bdad \
  swarm-master
```

docker-swarm via docker-machine

Weiterer Node

```
$ docker-machine create -d virtualbox \
  --swarm --swarm-master \
  --swarm-discovery \
  token://4754c10117f48dc759ec2f0989d1bdad \
  swarm-node1
```

`docker-swarm` – Scheduler Strategies

Strategies:

- spread (Default)
- binpack
- random

In Abhängigkeit von RAM, CPU und laufenden Containern

`docker-swarm` – Constraints/Affinity Filter

Labels:	Beispiele: ssd, rz15, production
	(`docker -d --label foo=bar`)
Host Facts:	operatingsystem, provider, storagedriver, ...
weitere Filter:	image, container, node
implizit:	net namespace, volumes, ports

docker-swarm – Beispiele

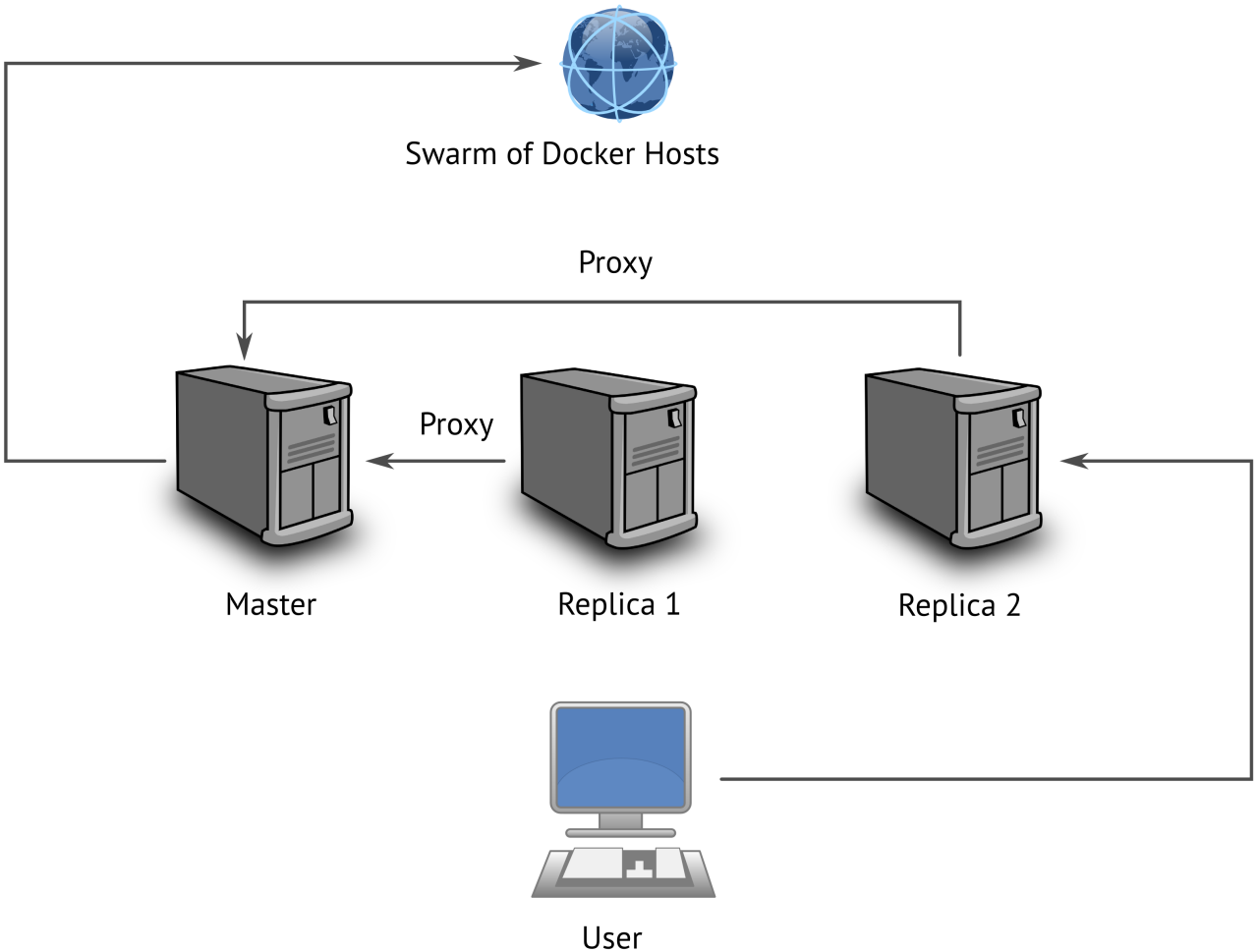
Beispiele:

```
$ docker run -d -p 80:80 -e affinity:image==~nginx \
  nginx www1

$ docker run -d -e constraint:stage!=production ...

$ docker run -d -e affinity:container==drunk_edison ...
```

`docker-swarm` – HA

- mehrere `docker-swarm` Master möglich
- Master wird dynamisch gewählt (Raft)
- Kommandos werden automatisch repliziert

docker-swarm – HA



docker-compose

- *„The artist formerly known as fig“*
- Simple YAML „Sprache“ zur Beschreibung des Anwendungsfalls
- baut Images, startet Container nach Beschreibung

Beispiel: docker-compose – YAML

```
database:
  build: mysql/
  volumes:
    - /var/lib/mysql
  expose:
    - "3306"
  environment:
    - DB_ADMIN_USER=admin
    - DB_ADMIN_PASS=admin
owncloud:
  build: httpd/
  links:
    - database:db
  volumes:
    - /srv/www/htdocs/owncloud/data
    - /srv/www/htdocs/owncloud/config
  ports:
    - "80:80"
```

docker-compose – Demonstration

https://github.com/mattiasgiese/committerconf2015-docker

Vielen Dank für Ihre Aufmerksamkeit!

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