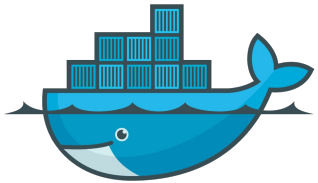


docker Workshop

Docker Swarm Mode

Agenda

neofonie*
Data. Technology. Services.

10:00 - 10:20	Begrüßung & Vorstellung (Neofonie Weg zu Docker, Vorstellung aller)
10:20 -	Einführung Docker Swarm Mode
11:00 -	Installation
11:30	Images & Containers
12:45 - 14:00	- *Mittagspause* -
13:30	Loadbalancing, DNS, Network
14:30	Swarm Recovery
15:30	docker-machine und Docker Ökosystem
16:00 - 18:00	FAQ und praxisnahe Aufgabenstellungen

Docker Swarm Mode

neofonie*
Data. Technology. Services.

Was ist Docker Swarm / Docker Swarm Mode ?

- über mehrere Hosts verteilte Docker Infrastruktur
- Container Orchestration / Deployment Tool
- Tool zur Resourcen Verwaltung (RAM, CPU)
- virtuelles Netzwerk und Routing Engine

Docker Swarm Mode

Mit Docker Swarm kann man seine Container in einem Cluster aus mehreren Hosts betreiben. Es gibt eine “ältere” Variante der Docker Swarm Implementierung, die im Gegensatz zum Docker Swarm Mode, Docker Swarm Classic genannt wird.

Beide Varianten unterscheiden sich stark. Der neuere Swarm Mode verfolgt einen Service orientierten Ansatz und unterstützt docker-compose Konfigurations-Dateien.

Docker Swarm Features

- einfache Verwaltung des Swarm über den Docker Client
- Multi-host Networking mit Docker Overlay Network
- Service Discovery / interner DNS Service für Container- und Service-Namen
- automatisches Loadbalancing
- Service Deklaration, desired state vs. current state
- Service Management, Rolling Updates, Scalling

Docker Swarm Mode Steuerung über Docker Client

Docker Swarm CLI Komponenten:

- docker swarm
- docker service
- docker network
- docker node
- docker stack
- Unterstützung für docker-compose ab Version 1.13.1 vorhanden (Feb. 2017)

Docker Swarm Mode Architektur

neofonie*
Data. Technology. Services.

Docker Swarm Rollen und deren Verwaltung

- es gibt zwei Swarm Node Rollen
  - Swarm Manager - Verwaltung des Swarm
  - Swarm Worker - Container Runtime Environment

- Rollen werden über den Docker Client verwaltet
  - docker swarm init
  - docker swarm join
  - docker swarm promote / demote

Docker Swarm Mode Architektur

neofonie*
Data. Technology. Services.

Docker Swarm Rollen Setup

- **Manager Nodes**
  - müssen redundant sein
  - mindestens 3, besser 5 Nodes
  - sind alle Manager “zerstört” muß der Swarm neu aufgesetzt werden
  - können auch als Container Wirt dienen
  - sollten vor unberechtigtem Zugriff geschützt sein

Docker Swarm Mode Architektur

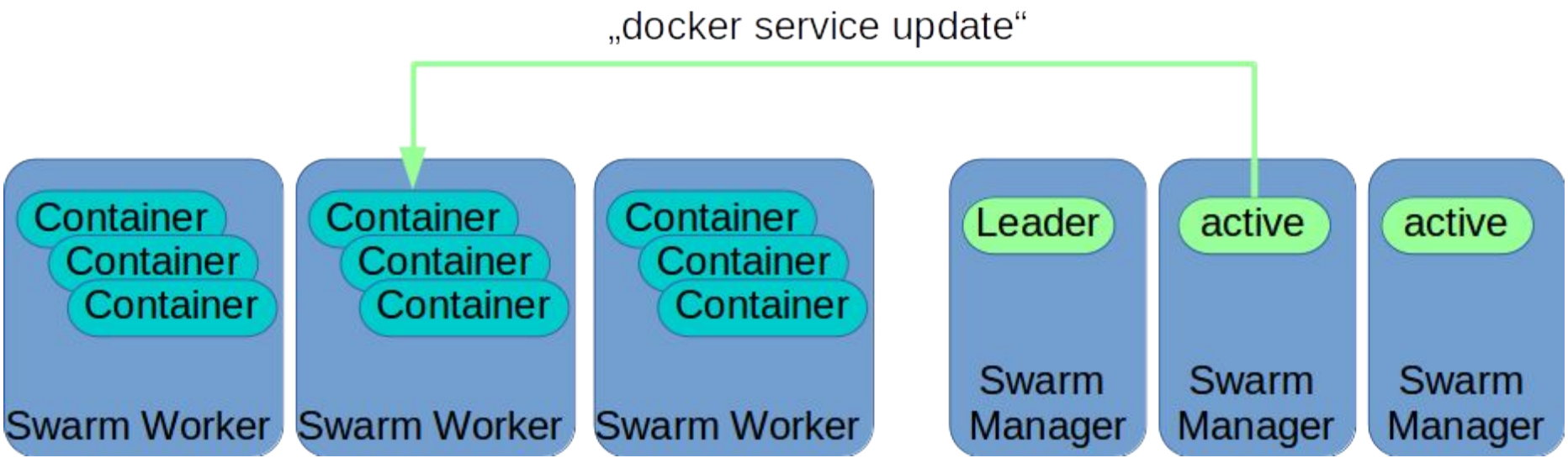
neofonie*
Data. Technology. Services.

Docker Swarm Rollen Setup

- **Worker Nodes**
  - werden durch Manager Nodes gesteuert
  - stellen Resourcen zum Start von Containern bereit
  - können zu Manager Nodes gemacht werden
  - können keine Docker-Swarm Kommandos ausführen

neofonie*
Data. Technology. Services.

Docker Swarm Mode Architektur



Docker Swarm Mode Architektur

neofonie*
Data. Technology. Services.

Docker Swarm Architektur Konzept

- Manager Nodes dienen dem Swarm Management, hier laufen keine Cotntainer
- Die Manager Nodes sollte geringe Latenzen haben
  - stehts freie CPU, also keine anderen Prozesse auf der gleichen VM
  - eigene Harddisk bzw. schnelle SSD
  - LAN Anbindung bzw. hohe Bandbreite im Netzwerk zwischen den Manager Nodes
- Worker Nodes sollten vom Sizing gleichförmig sein (kein Muß)
- Für die Verteilung der Container werden Node Label verwendet

Docker Swarm Mode Architektur

Ein Manager Node kann so konfiguriert werden, dass keine Container Tasks auf dem Node gestartet werden:

```
$ docker node update --availability drain node01
```

Nodes können mit einem Label versehen werden. Die zum Verteilen der Tasks verwendet werden können.

```
$ docker node update --label-add ssd --label-add powerline=2 node01
```

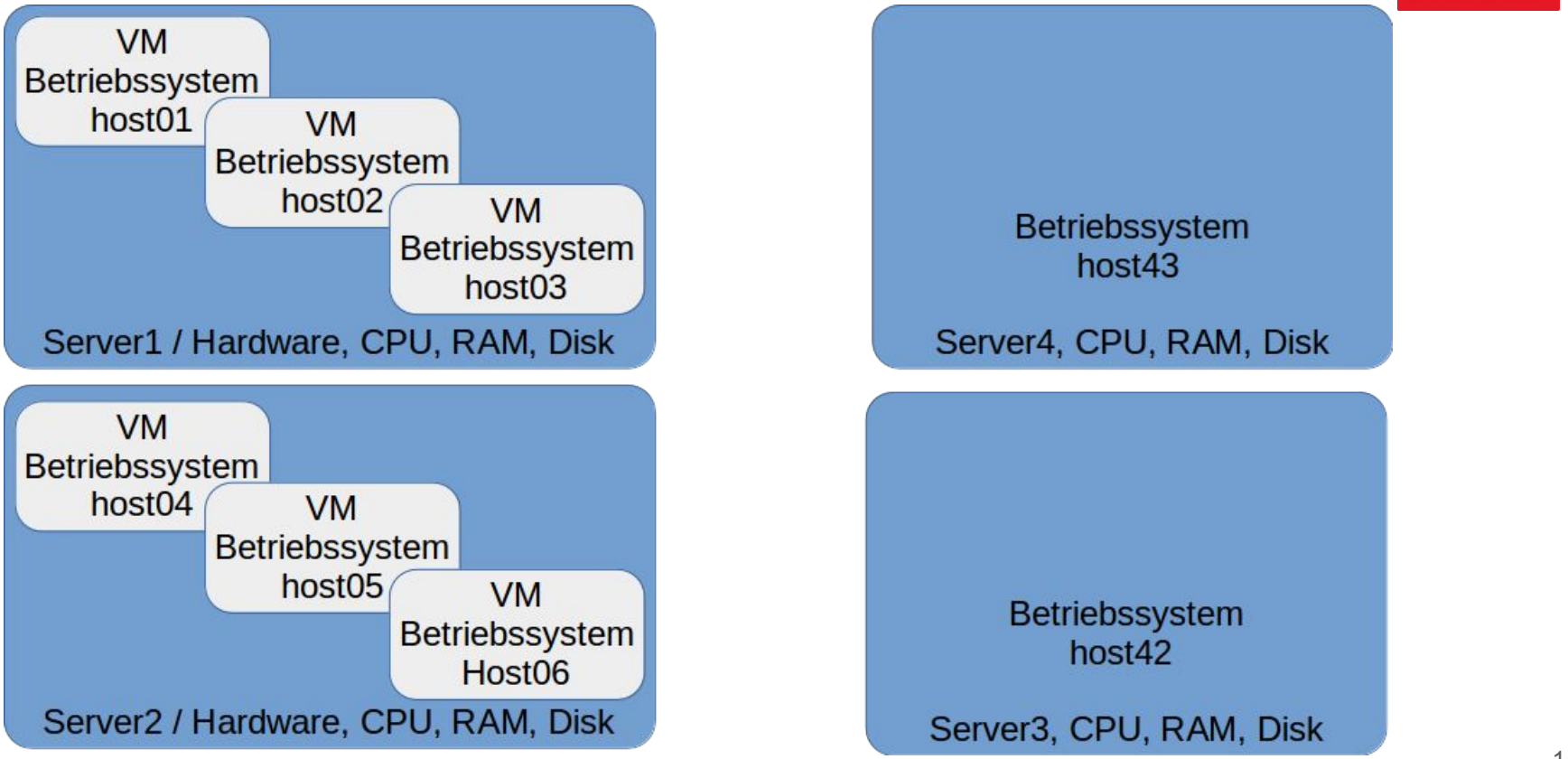
Docker Swarm Mode Integration

Um einen Docker Swarm Node aufzusetzen, benötigt man nur eine Installation der aktuellen Docker Engine. Alle Swarm-Funktionen sind dort integriert. Zusätzlich erforderlich ist ein aktueller Linux Kernel und eine Netzwerkanbindung.

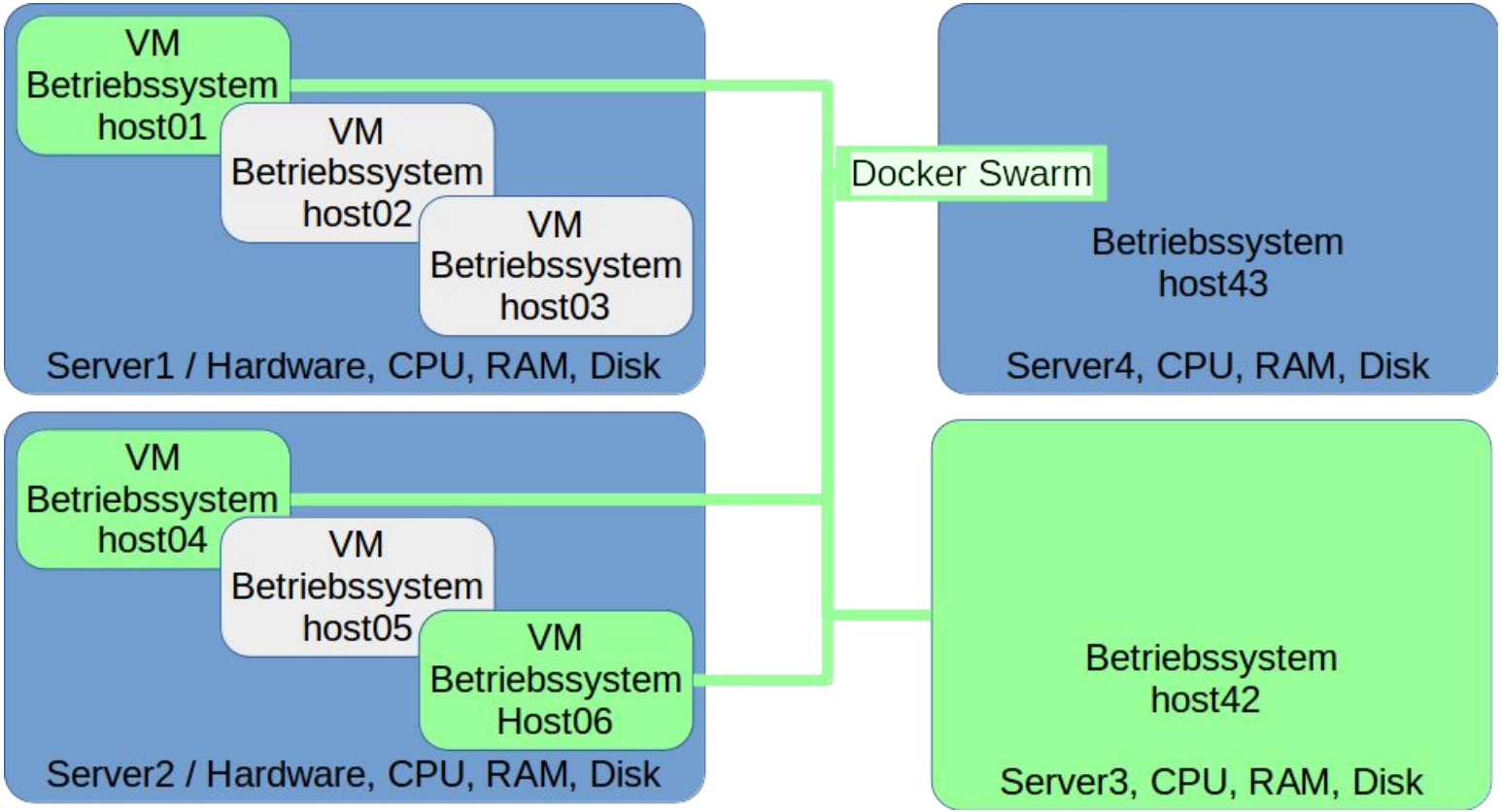
Mögliche Plattformen sind:

- virtuelle Maschine (VMWare, KVM, vSphere)
- echte Hardware
- eine Mischung aus beidem

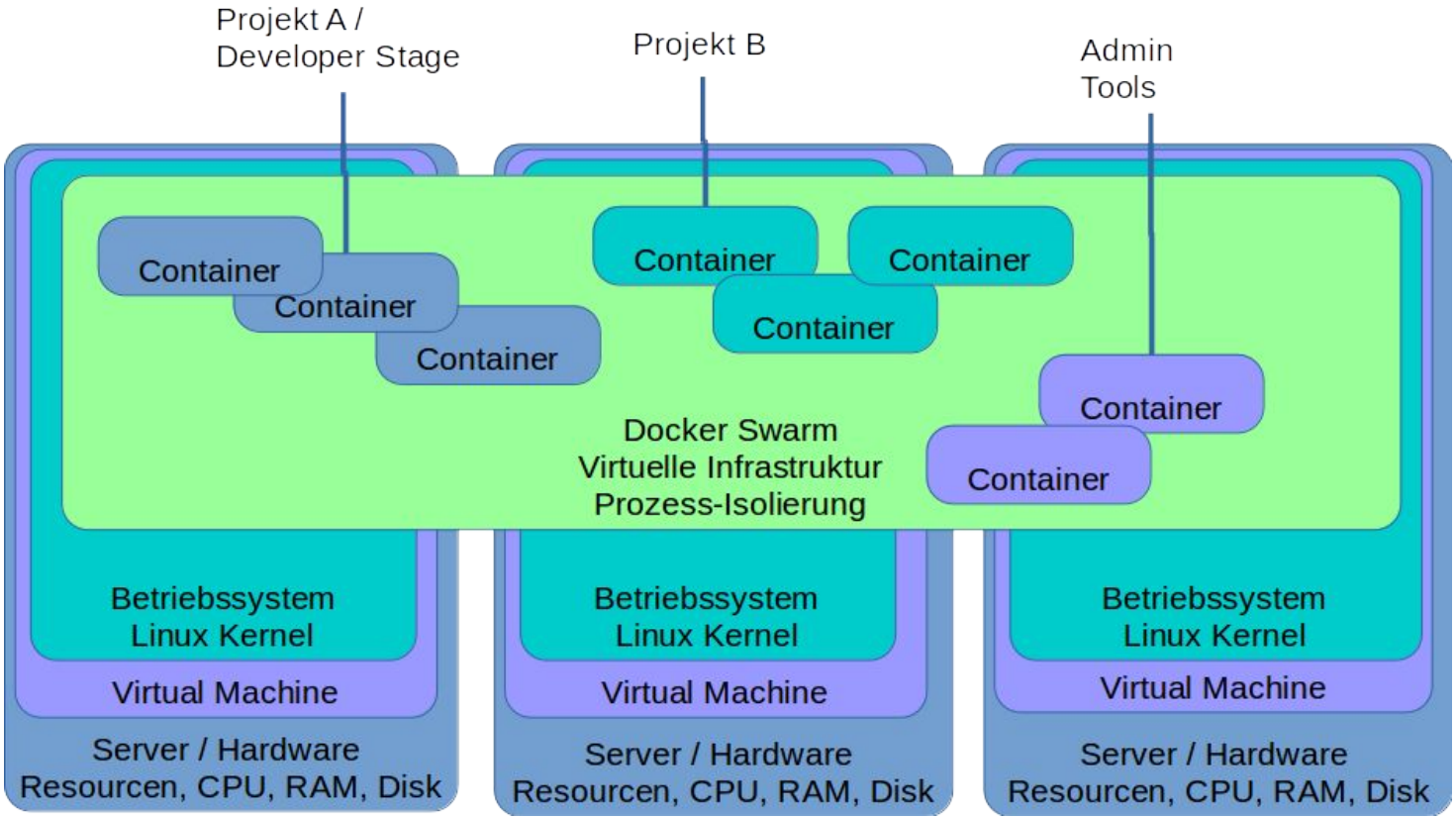
Docker Swarm Mode Integration



Docker Swarm Mode Integration



Docker Swarm Mode Integration



Docker Swarm Mode Installation

neofonie*
Data. Technology. Services.

los geht’s mit docker-machine …

Docker-Machine

neofonie*
Data. Technology. Services.

Für die Bereitstellung einer Swarm Umgebung bietet sich das Tool docker-machine an.

Was ist docker-machine ?

- ein Provisioning Tool
- installiert eine Docker Umgebung zum Betrieb von Containern
- verwaltet mehrere Hosts
- generiert TLS Zertifikate
- Tool zum Aufsetzen von Docker Nodes

Docker-Machine

neofonie*
Data. Technology. Services.

Installation der Docker Engine mit Hilfe von docker-machine

```
$ docker-machine create --driver generic --generic-ip-address=172.42.0.1
--generic-ssh-user=root wirt01

Running pre-create checks...
Creating machine...
...
Docker is up and running!
To see how to connect your Docker Client to the Docker Engine running on this virtual
machine, run: docker-machine env wirt01
```

Docker-Machine

Die Docker Engine ist nach der Installation über TCP erreichbar. Dafür gibt es einige Docker Environment Variablen, die über docker-machine gesetzt werden können:

```
$ docker-machine env wirt01
export DOCKER_TLS_VERIFY="1"
export DOCKER_HOST="tcp://172.42.0.1:2376"
export DOCKER_CERT_PATH="/home/trainer/.docker/machine/machines/wirt01"
export DOCKER_MACHINE_NAME="wirt01"
# Run this command to configure your shell:
# eval $(docker-machine env wirt01)
```

Docker-Machine

Setzen des Environment auf eine Docker-Engine

```
$ eval $(docker-machine env wirt01)
```

Alle weiteren Docker Kommandos werden dann auf dem Remote Host ausgeführt.

Docker-Machine

Zugriff auf den neu erstellten Docker Wirt:

Docker-Machine ermöglicht den Zugriff auf den Wirt per SSH. Dafür ist ein SSH-Key im Home-Verzeichnis des Root-Users hinterlegt. Weitere Installationsschritte können also sofort per docker-machine ausgeführt werden:

```
$ docker-machine ssh wirt01 “apt-get update && apt-get install dnsutils”
```

Docker-Machine

docker-machine unterstützt neben dem “generic” driver, welcher einen SSH Zugriff auf den Daemon Host benötigt, auch spezielle Umgebungen bekannter Hosting-Anbieter:

- Amazon Web Services
- Microsoft Azure
- Digital Ocean

Docker Swarm Mode Installation

neofonie*
Data. Technology. Services.

Nach erfolgreicher Installation einiger Nodes über docker-machine können wir nun einen Docker Swarm einrichten

Docker Swarm einrichten

Als ersten Schritt starten wir einen Swarm Manager Node.

```
~$ docker swarm init
Swarm initialized: current node (01r60w2ixzwrgr2fce7gwi6qp) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join \
    --token
SWMTKN-1-3yhmfl1tg4ni9kich8hezo3ebgf0dne2ou15pwvvebo6ihcom2-ccae85nvsq8xn8sxc8ta6fnk3
\
    1.2.3.4:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the
instructions.
```

Docker Swarm einrichten

Jetzt fügen wir einen Worker hinzu:

```
$ docker swarm join --token \
SWMTKN-1-3yhmfl1tg4ni9kich8hezo3ebgf0dne2ou15pwvvebo6ihcom2-ccae85nvsq8xn8sxc8ta6fnk3
<Manager IP Adresse hier eintragen>:2377
```

Docker Swarm einrichten

Es gibt jeweils einen Swarm Token für das Hinzufügen eines Manager Nodes und einen für den Worker Node. Auf dem Manager Node kann man den Token ausgeben lassen und als letzten Parameter worker bzw. manager wählen.

```
workshop@node01:~$ docker swarm join-token manager
To add a manager to this swarm, run the following command:

    docker swarm join \
    --token
SWMTKN-1-3nfxn8s3tbfyp3u35y26odx8sactj9u0w6636mvswtutykpwwu-ddumcn5dx1sc5jooqlu6f4m83
\
    138.68.83.108:2377
```

Docker Swarm einrichten

Liste aller Swarm Nodes ausgeben:

```
workshop@node01:~$ docker node ls
ID                            HOSTNAME   STATUS   AVAILABILITY   MANAGER STATUS
4c35vpyt3icxjfweyzkk578ij *   node01     Ready    Active         Reachable
84dpg6b1i14crty343jxvhdgt     node03     Ready    Active         Reachable
9jrbvyv33uv896b8bs745qtcs     node02     Ready    Active         Leader
```

In der letzten Spalte “MANAGER STATUS” sieht man einen Eintrag mit dem Status des jeweiligen Manager Nodes.

Swarm Worker haben in dieser Spalte keinen Eintrag (leer).

Docker Swarm einrichten

Wir können einen Manager Node zum Worker machen:

```
workshop@node01:~$ docker node ls
ID                            HOSTNAME  STATUS  AVAILABILITY  MANAGER STATUS
4c35vpyt3icxjfweyzkk578ij *   node01    Ready   Active        Reachable
84dpg6b1i14crty343jxvhdgt     node03    Ready   Active        Reachable
9jrbvyv33uv896b8bs745qtcs     node02    Ready   Active        Leader

workshop@node01:~$ docker node demote node03
Manager node02 demoted in the swarm.

workshop@node01:~$ docker node ls
ID                            HOSTNAME  STATUS  AVAILABILITY  MANAGER STATUS
4c35vpyt3icxjfweyzkk578ij *   node01    Ready   Active        Reachable
84dpg6b1i14crty343jxvhdgt     node03    Ready   Active
9jrbvyv33uv896b8bs745qtcs     node02    Ready   Active        Leader
```

Docker Swarm einrichten

und einen Worker Node zum Manager Node machen:

```
workshop@node01:~$ docker node promote  node03
Node node03 promoted to a manager in the swarm.

workshop@node01:~$ docker node ls
ID                            HOSTNAME  STATUS  AVAILABILITY  MANAGER STATUS
4c35vpyt3icxjfweyzkk578ij *   node01    Ready   Active        Reachable
84dpg6b1i14crty343jxvhdgt     node03    Ready   Active        Reachable
9jrbvyv33uv896b8bs745qtcs     node02    Ready   Active        Leader
```

Service Deklaration

Einen Service, also einen oder mehrere Container, starten wir mit dem Kommando “docker service”

Im Gegensatz zum Start eines Containers mit “docker run” sorgt die Service-Definition dafür, daß immer eine Instanz (default) oder mehrere Instanzen (z.B --replicas=3) im Swarm laufen auch wenn diese mit “docker stop <container_id>” beendet wird.

```
workshop@node01:~$ docker service create --name ping debian bash -c “ping -i 60 www.neofonie.de”
```

Service Deklaration

Eine Liste aller Services erhält man mit folgendem Befehl:

```
workshop@node01:~$ docker service ls
ID             NAME   REPLICAS   IMAGE    COMMAND
dzcstw4szy29   ping   0/1        debian   bash -c “ping www.neofonie.de”
```

Auf welchem Swarm Node wurde der Service gestartet ?

```
workshop@node01:~$ docker service ps ping
ID             NAME         IMAGE    NODE     DESIRED STATE   CURRENT STATE
ERROR
65cicbdpcl4ap1   ping.1   debian   node01   Running         Preparing 1 seconds ago
```

Service Deklaration

Die Logausgabe des Containers anschauen.

Im Swarm Mode kann auch `docker logs verwendet werden.` Dafür müßen wir auf dem selben Node eingeloggt sein auf dem auch der Container läuft.

```
ssh node01
workshop@node01:~$ docker logs ping.1.65cicbdpcl4ap1

64 bytes from 81.17.211.34: icmp_seq=50 ttl=57 time=15.436 ms
64 bytes from 81.17.211.34: icmp_seq=51 ttl=57 time=15.415 ms
64 bytes from 81.17.211.34: icmp_seq=52 ttl=57 time=15.669 ms
```

Service Deklaration

Die Logausgabe des Containers anschauen.

Die Log-Ausgabe aller Container eines Services sieht man mit:

```
~$ docker service logs ping

64 bytes from 81.17.211.34: icmp_seq=50 ttl=57 time=15.436 ms
64 bytes from 81.17.211.34: icmp_seq=51 ttl=57 time=15.415 ms
64 bytes from 81.17.211.34: icmp_seq=52 ttl=57 time=15.669 ms
```

Service Deklaration

Ein Software Update erfolgt durch die Definition eines neuen Image:

```
workshop@node01:~$ docker service ps wordpress
ID         NAME         IMAGE          NODE    DESIRED STATE CURRENT STATE         ERROR
57iyosqv… wordpress.1 wordpress:0.1 node01 Running       Running 18 hours

workshop@node01:~$ docker service update --image wordpress:0.2 wordpress
```

Service Deklaration

Die Ausgabe des Kommandos “docker service ps” zeigt uns eine History der Container für einen Service. In diesem Beispiel läuft Instanz Nr.1 des Wordpress Service mit dem Docker Image wordpress:0.2 (<image_name>:<tag>).

Die zuvor benutzte Version wordpress:0.1 wird mit dem Status “Shutdown” angezeigt. Logfiles und alle anderen persistenten Daten dieses “alten” Containers sind in dieser Phase bereits gelöscht.

```
workshop@node01:~$ docker service ps wordpress
ID      NAME             IMAGE          NODE    DESIRED STATE CURRENT STATE      ERROR
aevg2…  wordpress.1      wordpress:0.2 node01 Running        Running 3 seconds ago
57iyv…   \_ wordpress.1 wordpress:0.1 node03 Shutdown      Shutdown 2 seconds ago
```

Docker Volumes / Wordpress Beispiel

neofonie*
Data. Technology. Services.

Wir starten Wordpress und MySQL in einem Docker Swarm mit 3 Nodes:

```
docker network create -d overlay wordpress

docker service create --name mysql --network wordpress \
    --env MYSQL_ROOT_PASSWORD=1234 \
    mariadb

docker service create --name wordpress --network wordpress
    --env WORDPRESS_DB_PASSWORD=1234 \
    --publish 80:80 \
    wordpress
```

Docker Volumes / Wordpress Beispiel

Beispiel Wordpress und MySQL in einem Docker Swarm mit 3 Nodes:

```
workshop@node01:~$ docker service ls
ID              NAME         REPLICAS  IMAGE          COMMAND
4eozv9nhkyay  wordpress      1/1            wordpress
9qhml37it5b1  mysql          1/1       mariadb

workshop@node01:~$ docker service ps wordpress
ID        NAME         IMAGE     NODE   DESIRED STATE CURRENT STATE         ERROR
57iyosqv… wordpress.1 wordpress node01 Running       Running 18 hours ago

workshop@node01:~$ docker service ps mysql
ID        NAME         IMAGE     NODE   DESIRED STATE CURRENT STATE         ERROR
84wdx6e…  mysql.1      mariadb   node02 Running       Running 18 hours ago
```

Docker Volumes / Wordpress Beispiel

Wir machen einen Restart des Datenbank Node. Was passiert dann ?

```
workshop@node02: # poweroff
```

Docker Volumes / Wordpress Beispiel

Der Service mysql ist nun von node02 nach node03 “rescheduled” worden:

```
workshop@node02:~$ docker service ps mysql
ID        NAME         IMAGE    NODE     DESIRED STATE  CURRENT STATE           ERROR
99t3k8…   mysql.1      mariadb  node03   Running        Running 27 minutes ago
84wdx6e…   \_ mysql.1 mariadb  node02   Shutdown       Complete 14 minutes ago
```

Docker Volumes / Wordpress Beispiel

Wo sind meine Daten hin ?

```
workshop@node02:~$ docker inspect mysql.1.84wdx6e…
      ...
      "Mounts": [
           {
                "Name": "6dcdf785...",
                "Source": "/var/lib/docker/volumes/6dcdf785.../_data",
                "Destination": "/var/lib/mysql",
                ...
```

Docker Volumes / Wordpress Beispiel

neofonie*
Data. Technology. Services.

Wie kann ich das verhindern ?

```
workshop@node02:~$ docker service update mysql --restart-condition none
```

Leider nein !

Docker Volumes / Wordpress Beispiel

Wie kann ich das verhindern ? Wir definieren den Speicherort der Daten explizit.

```
workshop@node02:~$ docker service update --mount-add \
type=bind,source=/home/workshop/mysql,destination=/var/lib/mysql \
--constraint-add node.hostname==node02 mysql
```

Nun werden die Daten immer im Verzeichnis /home/workshop/mysql abgelegt.
Der Service wird immer auf Node “node02” gestartet.

Docker Registry einbinden

Möchte man ein Image aus seiner eigenen Registry starten, dann werden die Images nicht ohne weitere Konfiguration aus der Registry geholt.

```
workshop@node02:$ docker service create --name wordpress registry.neofonie.de/asp/wordpress
```

Der Service start schlägt fehl mit der Fehlermeldung:

```
No such image: registry.neofonie.de/asp/wordpress
```

Docker Registry einbinden

Die Login Informationen (User, Passwort) müßen an die Swarm Agents übergeben werden:

```
workshop@node02:~$ docker service update --with-registry-auth wordpress
```

Monitoring

- Agenten basiertes Monitoring ist nicht sinnvoll
- Resourcen CPU, RAM, SWAP
- Antwortzeiten, Anwendungs URLs
- Jeder Service braucht eine Healthcheck URL
- Healthcheck URL von außen zugänglich machen

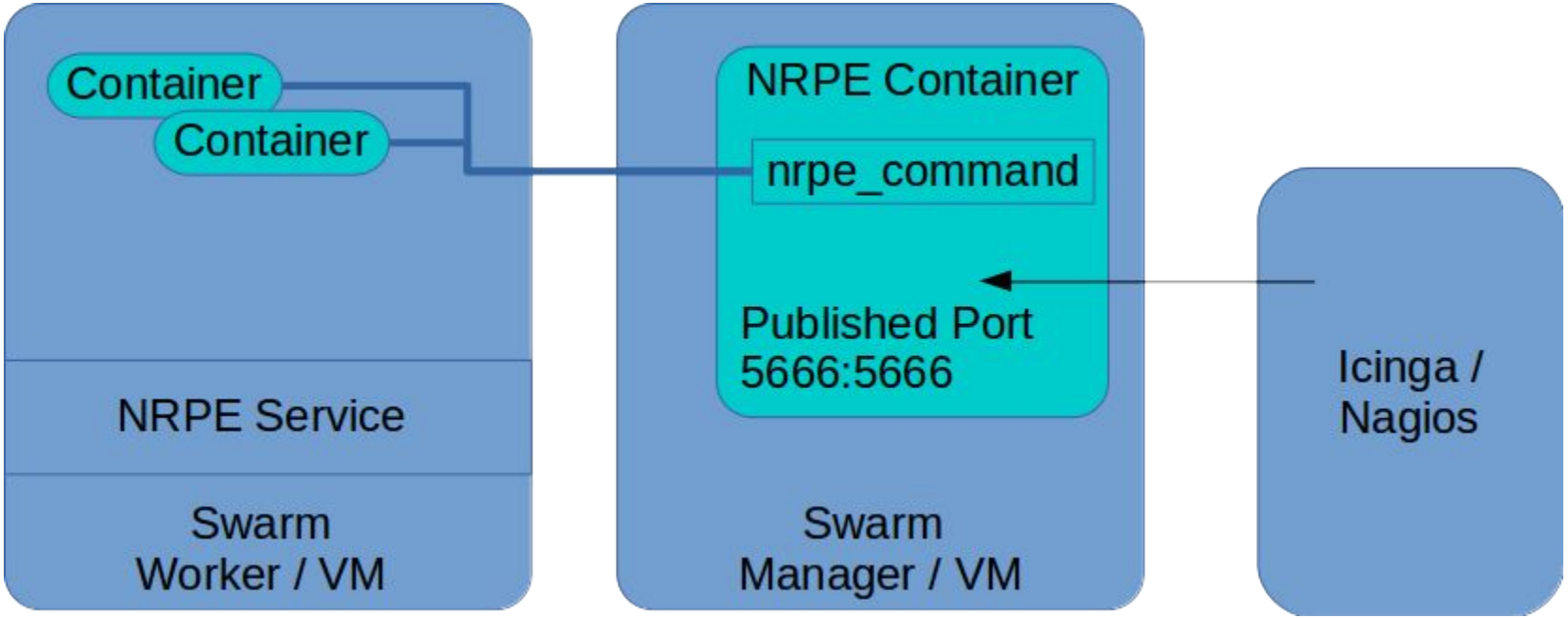
Monitoring / Statusinformationen

neofonie*
Data. Technology. Services.

	Hauptfunktion	SLA Reports	Alerts	Feature
Icinga	klassisches Monitoring mit zahlreichen checks und Plugins	JasperReports	script alerts sms, mail	Konfigurations API, push config
Prometheus.io	Metriken darstellen und Alerts erzeugen	nein	zahlreiche Schnittstellen slack, hipchat pager, push	proaktive Alerts, “morgen läuft die Platte voll”
Rancher / Portainer	Resourcenverbrauch der Container darstellen	nein	nein	Orchestration, Swarm Management

Icinga Integration

neofonie*
Data. Technology. Services.



Monitoring Integration

- NRPE Dienst läuft im Container
  - service namen können im Overlay Netzwerk verwendet werden
  - z.B. nslookup tasks.servicename gibt die Liste aller Service Instanzen zurück
  - Zugriff auf den Swarm möglich (tcp, docker socket)

- die Worker und Manager Nodes können mit einem NRPE Agent ausgestattet werden

- dynamische Anpassung der Icinga Konfiguration erforderlich
  - beim Hinzufügen neuer Worker oder Manager Nodes
  - beim Hinzufügen neuer Services
  - Automatisierung über API möglich (Icinga2)

weitere Links

neofonie*
Data. Technology. Services.

- https://www.netways.de/produkte/jasper/
- https://hub.docker.com/r/xforty/jasperserver/
- https://www.netways.de/produkte/icinga/icinga_2/features/reporting/

- https://blog.bugsnag.com/container-orchestration-with-docker-swarm-mode/
- https://github.com/spotify/docker-gc

Ops Learnings

neofonie*
Data. Technology. Services.

- https://blog.bugsnag.com/container-orchestration-with-docker-swarm-mode/
- https://github.com/spotify/docker-gc
- https://www.netways.de/produkte/jasper/