Pi Haufen mit LOL (Lot of LEDs)

Oder der **KNF-FabLab-Edu Pi Cluster** mit Docker Swarm und Blinkt!

Themen:

- Was ist ein Cluster
- Was ist Docker (Swarm) und warum?

PI-Cluster

Einen Cluster bezeichnet eine Anzahl von vernetzten Computern.

Einteilung in zwei unterschiedliche Aufgaben:

- die Erhöhung der Rechenkapazität (HPC-Cluster) und
- die Erhöhung der Verfügbarkeit (HA-Cluster, engl. high available - hochverfügbar).

Pi

Teach, Learn and Make with Raspberry Pi (https://www.raspberrypi.org/)

> Der im Vergleich zu üblichen Personal Computern sehr einfach aufgebaute Rechner wurde von der Stiftung (Raspberry Pi Foundation) mit dem Ziel entwickelt, jungen Menschen den Erwerb von Programmier- und Hardwarekenntnissen zu erleichtern. Entsprechend niedrig wurde der Verkaufspreis angesetzt, der je nach Modell etwa 5 bis 35 USD beträgt.

Die Motivation steigt.

Starterseminar im FabLab Nbg durch KNF möglich.

Tausend Pis

BitScope-Einschub mit 144 Raspberry Pi 3 auf 6 HE.

Scalable clusters make HPC R&D easy as Raspberry Pi (http://www.lanl.gov/discover/news-release-archive/2017/November/1113-raspberry-pi.php)

> It brings a powerful high-performance-computing testbed to system-software developers, researchers and others who lack machine time on the world's fastest supercomputers.

Gefunden auf heise.de (https://www.heise.de/newsticker/meldung/Tausend-Raspberry-Pi-im-Cluster-Rack-3891000.html) durch Schüler mitgeteilt.

KNF FabLab Edu Pi Cluster

Das fliegende Rechenzentrum.

Hat den gleichen Zweck wie die 1000 Pis oben. Hier kann einen Cluster betrieben werden, **ohne** dafür einige Tausend Euro in ein Rechenzentrum auszuzugeben. Es wurden ca. 500 € investiert.

Warum mach ich DevOps und Docker

- Auf DevOps-Camp 2014 (https://openspacer.org/60-devops-community/) kennengelernt
- KNF-Vortrag Kongress 2014 zu ansible (https://cloud.franken.de/owncloud/index.php/s/dbf8ab8c95016a68d49ebb289025d5a9)
- Stay on the cutting edge with Docker 1.6 and test drive it on your Raspberry Pi today (http://blog.hypriot.com/)
- Docker-Bamberg Meetup (https://www.meetup.com/de-DE/Docker-Bamberg)
- Serverspec auf DevOps-Camp 2015 (https://openspacer.org/60-devops-community/61-devops-camp-2015/) kennen gelernt
- Seit Sommer 2017 Raspberry Edu devops Setup (https://github.com/kraeml/raspberry-edu-devops)
- Seit Sommer 2017 heterogenes Virtuelles Netzwerk für DevOps (https://github.com/kraeml/packer-ubuntu-1604-de)

Docker

Docker	
 (/wiki /Datei:Docker_(container_engine)_logo.png)	
Entwickler	Docker, Inc.
Erscheinungsjahr	2013
Aktuelle Version	17.11.0 (20. November 2017)
Betriebssystem	Linux Windows MacOS
Programmiersprache	Go
Kategorie	Virtualisierung
Lizenz	Apache 2.0
deutschsprachig	nein
www.docker.com (http://www.docker.com/)	

In [1]:
```
# Docker version auf den Pi-Cluster
docker version
```

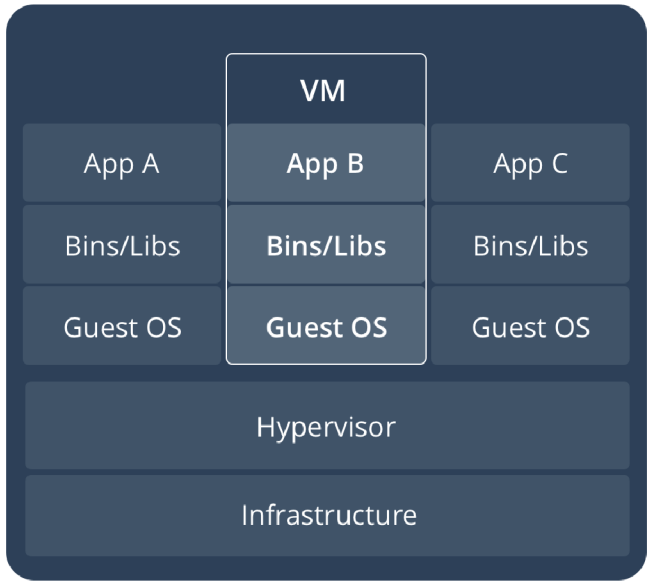
```
Client:
 Version:      17.09.0-ce
 API version:  1.32
 Go version:   go1.8.3
 Git commit:   afdb6d4
 Built:        Tue Sep 26 22:50:01 2017
 OS/Arch:      linux/arm

Server:
 Version:      17.09.0-ce
 API version:  1.32 (minimum version 1.12)
 Go version:   go1.8.3
 Git commit:   afdb6d4
 Built:        Tue Sep 26 22:44:09 2017
 OS/Arch:      linux/arm
 Experimental: false
```

Was ist Docker

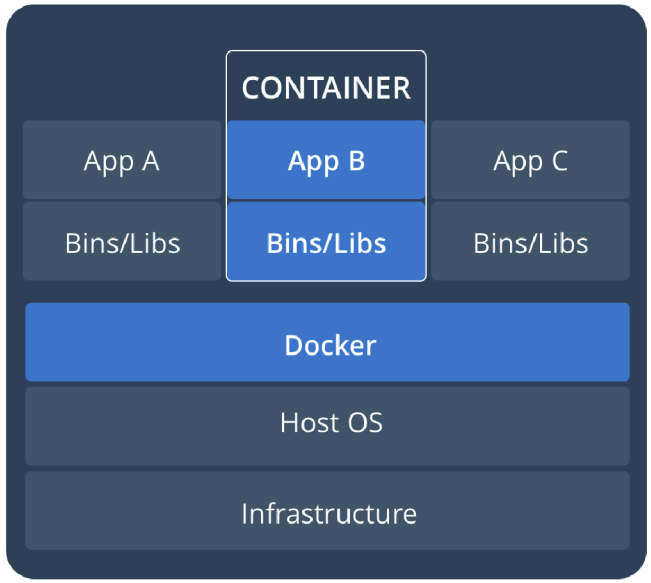
Docker ist eiene **Betriebssystemvirtualisierung** mittels OS-Container wie sie auch *LXC*, *LXD*, *Jail* (BSD), *Zone* (Solaris) und auch Windows Server Container (http://www.zdnet.de/88246989/docker-und-windows-server-2016-das-muessen-profis-wissen/?inf_by=5a143495671db8a11f8b4a68) (Docker for Windows) bieten.

Bei der Betriebssystemvirtualisierung läuft nur ein Host-Kernel.

Klassische Virtualisierung durch einen Hypervisor



Bitriebssystemvitualisierung mit Docker



Bei Virtualisierung auf Betriebssystemebene wird anderen Computerprogrammen eine **komplette Laufzeitumgebung** virtuell innerhalb eines geschlossenen Containers zur Verfügung gestellt. Es wird **kein weiteres Betriebssystem gestartet**;

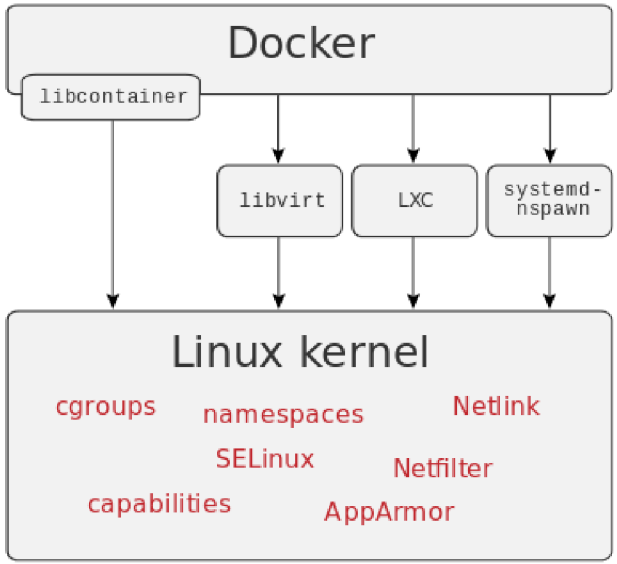
Nachteil:

- In Containern können keine Treiber geladen werden.

Vorteil:

- geht besonders effizient mit den Ressourcen um (insbesondere hinsichtlich der Prozessor-Last und des Haupt- und Massenspeicherbedarfs)

Überriss Docker



Docker Swarm

Docker Swarm	
 (/wiki /Datei:Docker_(container_engine)_logo.png)	
Entwickler	Docker, Inc.
Erscheinungsjahr	2015
Aktuelle Version	17.11.0 (1.12.0 or later) (20. November 2017)
Betriebssystem	Linux Windows MacOS
Programmiersprache	Go
Kategorie	Virtualisierung
Lizenz	Apache 2.0
deutschsprachig	nein
www.docker.com (http://www.docker.com/)	

Und wieder hypriot

DockerCon 2017 Demo: Monitoring Docker Swarm with LED's (http://blog.hypriot.com/post/dockerconaustin2017/)

Docker Swarm aufsetzen

Zunächst die IP-Adresse feststellen

Die Ip des AP (Leader) wird öfters benötigt und in IP_LEADER bzw. IP_LEADER_WLAN gespeichert.

In [2]:
```
ip addr show eth0
```

```
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether b8:27:eb:27:c2:35 brd ff:ff:ff:ff:ff:ff
    inet 172.24.2.1/24 brd 172.24.2.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::ba27:ebff:fe27:c235/64 scope link
       valid_lft forever preferred_lft forever
```

In [3]:
```
ip addr show eth0 | grep 'inet\s' | awk '{ print $2}' | cut -d '/' -f 1 | grep 172
```

```
172.24.2.1
```

In [4]:
```
IP_LEADER=$(ip a s eth0 | grep 'inet\s' | awk '{ print $2}' | cut -d '/' -f 1 | grep  172)
echo $IP_LEADER
```

```
172.24.2.1
```

In [5]:
```
IP_LEADER_WLAN=$(ip a s wlan0 | grep 'inet\s' | awk '{ print $2}' | cut -d '/' -f 1)
echo $IP_LEADER_WLAN
```

```
172.24.1.1
```

Docker Swarm Init

Feststellen ob es schon einen Docker Swarm gibt und wenn nicht, dann wird dieser angelegt und sein Status ausgegeben.

In [6]:
```
docker node ls || docker swarm init --advertise-addr ${IP_LEADER} && docker node ls
```

```
Error response from daemon: This node is not a swarm manager. Use "docker swarm init" or "docker swarm join" to connect this
node to swarm and try again.
Swarm initialized: current node (pqp74pxig8s3zs07eafeugjuq) is now a manager.

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

    docker swarm join --token SWMTKN-1-5k75rxbk2t9tf5m6bc1bfv5846jhgm15jffd4teh283cy8um9e-4znifs51sy43cdotqqlmsds0o 172.24.2
.1:2377

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

ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS
pqp74pxig8s3zs07eafeugjuq *   cluster-00          Ready               Active              Leader
```

Token

Damit die einzelnen Nodes dem Cluster beitreten können werden die Token für `worker` und `manger` gespeichert.

In [7]:
```
JOIN_TOKEN=$(docker swarm join-token -q worker)
echo $JOIN_TOKEN
```

```
SWMTKN-1-5k75rxbk2t9tf5m6bc1bfv5846jhgm15jffd4teh283cy8um9e-4znifs51sy43cdotqqlmsds0o
```

In [8]:
```
JOIN_TOKEN_MANAGER=$(docker swarm join-token -q manager)
echo $JOIN_TOKEN_MANAGER
```

```
SWMTKN-1-5k75rxbk2t9tf5m6bc1bfv5846jhgm15jffd4teh283cy8um9e-eoi1dzztzmb9sa373hewahl6w
```

SSH einrichten

Der Klassiker mit Schlüsselaustausch und allem drum und dran.

Gut geeignet für den Unterrichtsstoff.

ToDo: Dies kann man mit docker machine umgehen.

In [9]:
```
# Setup ssh-key for user if not exists
ls ~/.ssh/id_rsa || ansible -i localhost, -m shell -a 'ssh-keygen -b 2048 -t rsa -f ~/.ssh/id_rsa -q -N "" creates=~/.ssh/id
_rsa' --connection=local localhost
```

```
/home/pi/.ssh/id_rsa
```

Den SSH Zugang vorbereiten. Zunächst aus `knwon_hosts` löschen. Dann wieder bekannt geben und den ssh-key übertragen.

In [10]:
```
ping -c 1 cluster-01.local
```

```
PING cluster-01.local (172.24.2.2) 56(84) bytes of data.
64 bytes from 172.24.2.2: icmp_seq=1 ttl=64 time=0.905 ms

--- cluster-01.local ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.905/0.905/0.905/0.000 ms
```

In [11]:
```
ping -c 1 cluster-01.local | grep PING | cut -d '(' -f 2 | cut -d')' -f 1
```

```
172.24.2.2
```

In [12]:
```
for i in 01 02 03 04; do
    # Hostname only
    CL_HOSTNAME=cluster-${i}
    # Get Ip via local Domain (avahi)
    CL_IP=$(ping -c 1 ${CL_HOSTNAME}.local | grep PING | cut -d '(' -f 2 | cut -d')' -f 1)
    # Cluster Hostname only
    ssh-keygen -R ${CL_HOSTNAME}
    # Cluster with Domain local
    ssh-keygen -R ${CL_HOSTNAME}.local
    # IP Cluster
    ssh-keygen -R ${CL_IP}
    # Add Cluster into known_hosts
    ssh-keyscan -H ${CL_HOSTNAME}.local >> ~/.ssh/known_hosts
    ssh-keyscan -H ${CL_HOSTNAME} >> ~/.ssh/known_hosts
    ssh-keyscan -H ${CL_IP} >> ~/.ssh/known_hosts
    # SSH key copy into Cluster
    sshpass -p raspberry ssh-copy-id pi@${CL_HOSTNAME}.local
done
```

```
/home/pi/.ssh/known_hosts updated.
Original contents retained as /home/pi/.ssh/known_hosts.old
# Host cluster-01.local found: line 49 type ECDSA
# Host cluster-01.local found: line 50 type RSA
# Host cluster-01.local found: line 51 type ED25519
/home/pi/.ssh/known_hosts updated.
Original contents retained as /home/pi/.ssh/known_hosts.old
# Host 172.24.2.2 found: line 49 type RSA
# Host 172.24.2.2 found: line 50 type ECDSA
# Host 172.24.2.2 found: line 51 type ED25519
/home/pi/.ssh/known_hosts updated.
Original contents retained as /home/pi/.ssh/known_hosts.old
# cluster-01.local SSH-2.0-OpenSSH_6.7p1 Raspbian-5+deb8u3
# cluster-01.local SSH-2.0-OpenSSH_6.7p1 Raspbian-5+deb8u3
# cluster-01.local SSH-2.0-OpenSSH_6.7p1 Raspbian-5+deb8u3
getaddrinfo cluster-01: Name or service not known
# 172.24.2.2 SSH-2.0-OpenSSH_6.7p1 Raspbian-5+deb8u3
# 172.24.2.2 SSH-2.0-OpenSSH_6.7p1 Raspbian-5+deb8u3
# 172.24.2.2 SSH-2.0-OpenSSH_6.7p1 Raspbian-5+deb8u3
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed

/usr/bin/ssh-copy-id: WARNING: All keys were skipped because they already exist on the remote system.

/home/pi/.ssh/known_hosts updated.
Original contents retained as /home/pi/.ssh/known_hosts.old
# Host cluster-02.local found: line 49 type RSA
# Host cluster-02.local found: line 50 type ECDSA
# Host cluster-02.local found: line 51 type ED25519
/home/pi/.ssh/known_hosts updated.
Original contents retained as /home/pi/.ssh/known_hosts.old
# Host 172.24.2.3 found: line 49 type RSA
# Host 172.24.2.3 found: line 50 type ECDSA
# Host 172.24.2.3 found: line 51 type ED25519
/home/pi/.ssh/known_hosts updated.
Original contents retained as /home/pi/.ssh/known_hosts.old
# cluster-02.local SSH-2.0-OpenSSH_6.7p1 Raspbian-5+deb8u3
# cluster-02.local SSH-2.0-OpenSSH_6.7p1 Raspbian-5+deb8u3
# cluster-02.local SSH-2.0-OpenSSH_6.7p1 Raspbian-5+deb8u3
getaddrinfo cluster-02: Name or service not known
# 172.24.2.3 SSH-2.0-OpenSSH_6.7p1 Raspbian-5+deb8u3
# 172.24.2.3 SSH-2.0-OpenSSH_6.7p1 Raspbian-5+deb8u3
# 172.24.2.3 SSH-2.0-OpenSSH_6.7p1 Raspbian-5+deb8u3
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed

/usr/bin/ssh-copy-id: WARNING: All keys were skipped because they already exist on the remote system.
```

Huray was gelernt. Wie wichtig ist die Zeit in einem Cluster System mit x509?

Sehr wichtig!

In [13]:
```
for i in 00 01 02 03 04; do
    # Hostname only
    CL_HOSTNAME=cluster-${i}.local
    echo -n $CL_HOSTNAME
    ssh $CL_HOSTNAME "date"
done
```

```
cluster-00.localDo 23. Nov 14:40:52 CET 2017
cluster-01.localDo 23. Nov 14:40:53 CET 2017
cluster-02.localDo 23. Nov 14:40:53 CET 2017
cluster-03.localDo 23. Nov 14:40:54 CET 2017
cluster-04.localDo 23. Nov 14:41:06 CET 2017
```

Dem Swarm beitreten

In [14]:
```
# Noch ist unser Swarm allein
docker node ls
```

```
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS
pqp74pxig8s3zs07eafeugjuq *   cluster-00          Ready               Active              Leader
```

Die Worker

Achtung nicht nur die Worker verrichten die Arbeit, sondern auch die Manager :-))

Nur der Client 01 und 03 als Worker dem Swarm beitreten.

In [15]:
```
#set -x
for i in 01 03; do
    # Hostname only
    CL_HOSTNAME=cluster-${i}.local
    echo $CL_HOSTNAME
    ssh $CL_HOSTNAME "docker swarm join --token $JOIN_TOKEN $IP_LEADER:2377"
done
#set +x
```

```
cluster-01.local
This node joined a swarm as a worker.
cluster-03.local
This node joined a swarm as a worker.
```

In [16]:
```
docker node ls
```

```
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS
pqp74pxig8s3zs07eafeugjuq *   cluster-00          Ready               Active              Leader
0br7aj8dld5sr73vic82s3jo6     cluster-01          Ready               Active
qocjbg98xx6pr5ig1zc4k3dpk     cluster-03          Ready               Active
```

Die Manger

Diese haben neben der Aufgabe des Workers auch eine Ausfallsicherheit des Leaders zu gewährleisten.

Der Client 02 und 04 als Manger dem Swarm beitreten.

In [17]:
```
for i in 02 04; do
    # Hostname only
    CL_HOSTNAME=cluster-${i}.local
    echo $CL_HOSTNAME
    ssh $CL_HOSTNAME "docker swarm join --token $JOIN_TOKEN_MANAGER $IP_LEADER:2377"
done
```

```
cluster-02.local
This node joined a swarm as a manager.
cluster-04.local
This node joined a swarm as a manager.
```

In [18]:
```
docker node ls
```

```
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS
pqp74pxig8s3zs07eafeugjuq *   cluster-00          Ready               Active              Leader
0br7aj8dld5sr73vic82s3jo6     cluster-01          Ready               Active
v9odvkkycga6enf7ld6uok9vg     cluster-02          Ready               Active              Reachable
qocjbg98xx6pr5ig1zc4k3dpk     cluster-03          Ready               Active
c9rzz568a8lr2jzecbsveikxp     cluster-04          Ready               Active              Reachable
```

Visualizer anschauen

Einen graphischen Output als Service starten. Hierzu wird das Image alexellis2/visualizer-arm genutzt.

In [19]:
```
docker service create \
  --name=visualizer \
  --publish=8000:8080/tcp \
  --constraint=node.role==manager \
  --mount=type=bind,src=/var/run/docker.sock,dst=/var/run/docker.sock \
  --no-resolve-image \
  --detach=false \
  alexellis2/visualizer-arm
```

```
ohooxit3x6prghe14346a8sss

all progress: 0 out of 1 tasks
all progress: 1 out of 1 tasks alizer-arm:latest
fy: Service converged to verify that tasks are stable...
```

Dies wird als einmaliger Service auf einem Manager-Node gestartet. Die Hostdatei `/var/run/docker.sock` wird in dem Container eingefügt.

In [20]:
```
echo "http://${IP_LEADER_WLAN}:8000"
echo "http://${IP_LEADER}:8000"
```

```
http://172.24.1.1:8000
http://172.24.2.1:8000
```

In [21]:
```
docker service ls
```

```
ID                  NAME                MODE                REPLICAS            IMAGE                                PORTS
ohooxit3x6pr        visualizer          replicated          1/1                 alexellis2/visualizer-arm:latest     *:8000->8
080/tcp
```

Der LOL Monitor

Auf allen Knoten soll ein Monitor Dienst als Docker Container laufen. Dieser überwacht, wie viele *whoami* Container laufen und zeigt dies durch die Anzahl der LEDs an.

Grün: Ein whoami-Container wird gestartet.

Rot: Ein whoami-Container stoppt.

Gelb-Grün: Für die Version 1.1.0

Blau: Für die Version 1.2.0

In [22]:
```
docker service create --name monitor --mode global \
  --restart-condition any --mount type=bind,src=/sys,dst=/sys \
  --mount type=bind,src=/var/run/docker.sock,dst=/var/run/docker.sock \
  --no-resolve-image \
  --detach=false \
  stefanscherer/monitor:1.1.0
```

```
bql95rhwlmtnfopbxgeuoydgn

all progress: 0 out of 5 tasks asks
aj8dld5s: assigned
bg98xx6p: assigned
4pxig8s3: assigned
vkkycga6: assigned
all progress: 5 out of 5 tasks
fy: Service converged to verify that tasks are stable...
```

Es läuft noch kein whoami.

In [23]:
```
docker service ls
```

```
ID                  NAME                MODE                REPLICAS            IMAGE                                PORTS
bql95rhwlmtn        monitor             global              5/5                 stefanscherer/monitor:1.1.0
ohooxit3x6pr        visualizer          replicated          1/1                 alexellis2/visualizer-arm:latest     *:8000->8
080/tcp
```

Der GROSSE Moment der LED

Nun starten wir den Service `whoami` mti der Version 1.1.0 (Gelb-Grün).

In [24]:
```
docker service create --name whoami \
  --no-resolve-image \
  --detach=false \
  stefanscherer/whoami:1.1.0
```

```
gkhxl78gt5tk5vd77qpjfinua

all progress: 0 out of 1 tasks
all progress: 1 out of 1 tasks
fy: Service converged to verify that tasks are stable...
```

In [25]:
```
docker service ls
```

```
ID                  NAME                MODE                REPLICAS            IMAGE                                PORTS
bql95rhwlmtn        monitor             global              5/5                 stefanscherer/monitor:1.1.0
ohooxit3x6pr        visualizer          replicated          1/1                 alexellis2/visualizer-arm:latest     *:8000->8
080/tcp
gkhxl78gt5tk        whoami              replicated          1/1                 stefanscherer/whoami:1.1.0
```

Der Update auf 1.2.0

Die LED wechselt auf Blau.

In [26]:
```
docker service update \
  --detach=false \
  --image stefanscherer/whoami:1.2.0 whoami
```

```
image stefanscherer/whoami:1.2.0 could not be accessed on a registry to record
its digest. Each node will access stefanscherer/whoami:1.2.0 independently,
possibly leading to different nodes running different
versions of the image.

whoami

all progress: 0 out of 1 tasks
all progress: 1 out of 1 tasks
fy: Service converged to verify that tasks are stable...
```

In [27]:
```
docker service ls
```

```
ID                  NAME                MODE                REPLICAS            IMAGE                                PORTS
bql95rhwlmtn        monitor             global              5/5                 stefanscherer/monitor:1.1.0
ohooxit3x6pr        visualizer          replicated          1/1                 alexellis2/visualizer-arm:latest     *:8000->8
080/tcp
gkhxl78gt5tk        whoami              replicated          1/1                 stefanscherer/whoami:1.2.0
```

Wir skalieren auf 5

In [28]:
```
docker service scale --detach=false whoami=5
```

```
whoami scaled to 5

all progress: 0 out of 5 tasks
   K$<3>$<3>
   K$<3>$<3>
   K$<3>$<3>
   K$<3>$<3>
all progress: 5 out of 5 tasks
fy: Service converged to verify that tasks are stable...
```

In [29]:
```
docker service ls
```

```
ID                  NAME                MODE                REPLICAS            IMAGE                                PORTS
bql95rhwlmtn        monitor             global              5/5                 stefanscherer/monitor:1.1.0
ohooxit3x6pr        visualizer          replicated          1/1                 alexellis2/visualizer-arm:latest     *:8000->8
080/tcp
gkhxl78gt5tk        whoami              replicated          5/5                 stefanscherer/whoami:1.2.0
```

Downgrade

In [30]:
```
docker service update \
  --detach=false \
  --image stefanscherer/whoami:1.1.0 whoami
```

```
image stefanscherer/whoami:1.1.0 could not be accessed on a registry to record
its digest. Each node will access stefanscherer/whoami:1.1.0 independently,
possibly leading to different nodes running different
versions of the image.

whoami

all progress: 0 out of 5 tasks
   K$<3>$<3>
   K$<3>$<3>
   K$<3>$<3>
   K$<3>$<3>
all progress: 5 out of 5 tasks
fy: Service converged to verify that tasks are stable...
```

Auf 20 Saklieren

Jetzt können die Stecker gezogen werden.

In [31]:
```
docker service scale --detach=false whoami=20
```

```
whoami scaled to 20

all progress: 0 out of 20 tasks
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
0:   <3>$<3>
0:   <3>$<3>
0:   <3>$<3>
0:   <3>$<3>
0:   <3>$<3>
0:   <3>$<3>
0:   <3>$<3>
0:   <3>$<3>
0:   <3>$<3>
0:   <3>$<3>
0:   <3>$<3>
ll progress: 20 out of 20 tasks
fy: Service converged to verify that tasks are stable...
```

Versionswechsel

In [32]:
```
docker service update --update-parallelism 5 \
  --detach=false \
  --image stefanscherer/whoami:1.2.0 whoami
```

```
image stefanscherer/whoami:1.2.0 could not be accessed on a registry to record
its digest. Each node will access stefanscherer/whoami:1.2.0 independently,
possibly leading to different nodes running different
versions of the image.

whoami

all progress: 0 out of 20 tasks
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
0:   <3>$<3>
0:   <3>$<3>
0:   <3>$<3>
0:   <3>$<3>
0:   <3>$<3>
0:   <3>$<3>
0:   <3>$<3>
0:   <3>$<3>
0:   <3>$<3>
0:   <3>$<3>
ll progress: 20 out of 20 tasks
fy: Service converged to verify that tasks are stable...
```

In [33]:
```
docker service ls
```

```
ID                  NAME                MODE                REPLICAS            IMAGE                                   PORTS
bql95rhwlmtn        monitor             global              5/5                 stefanscherer/monitor:1.1.0
ohooxit3x6pr        visualizer          replicated          1/1                 alexellis2/visualizer-arm:latest        *:8000->8
080/tcp
gkhxl78gt5tk        whoami              replicated          20/20               stefanscherer/whoami:1.2.0
```

Alle LED

In [34]:
```
docker service scale --detach=false whoami=40
```

```
whoami scaled to 40

Operation continuing in background.
Use `docker service ps whoami` to check progress.
```

Versionswechsel

Allerdings fünf Maschinen gleichzeitig.

In [35]:
```
docker service update --update-parallelism 5 \
  --detach=false \
  --image stefanscherer/whoami:1.1.0 whoami
```

```
image stefanscherer/whoami:1.1.0 could not be accessed on a registry to record
its digest. Each node will access stefanscherer/whoami:1.1.0 independently,
possibly leading to different nodes running different
versions of the image.

whoami

all progress: 40 out of 40 tasks
fy: Service converged to verify that tasks are stable...
```

In [36]:
```
docker service ls
```

```
ID                  NAME                MODE                REPLICAS            IMAGE                                PORTS
bql95rhwlmtn        monitor             global              5/5                 stefanscherer/monitor:1.1.0
ohooxit3x6pr        visualizer          replicated          1/1                 alexellis2/visualizer-arm:latest     *:8000->8
080/tcp
gkhxl78gt5tk        whoami              replicated          40/40               stefanscherer/whoami:1.1.0
```

Down- and Upsize

In [37]:
```
docker service scale --detach=false whoami=1
```

```
whoami scaled to 1

all progress: 0 out of 1 tasks
all progress: 1 out of 1 tasks
fy: Service converged to verify that tasks are stable...
```

In [38]:
```
docker service scale --detach=false whoami=15
```

```
whoami scaled to 15

all progress: 0 out of 15 tasks
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
5:   <3>$<3>
5:   <3>$<3>
5:   <3>$<3>
5:   <3>$<3>
5:   <3>$<3>
ll progress: 15 out of 15 tasks
fy: Service converged to verify that tasks are stable...
```

In [39]:
```
docker service ls
```

```
ID                  NAME                MODE                REPLICAS            IMAGE                                PORTS
bql95rhwlmtn        monitor             global              5/5                 stefanscherer/monitor:1.1.0
ohooxit3x6pr        visualizer          replicated          1/1                 alexellis2/visualizer-arm:latest     *:8000->8
080/tcp
gkhxl78gt5tk        whoami              replicated          15/15               stefanscherer/whoami:1.1.0
```

In [40]:
```
docker service update --update-parallelism 5 \
  --detach=false \
  --image stefanscherer/whoami:1.2.0 whoami
```

```
image stefanscherer/whoami:1.2.0 could not be accessed on a registry to record
its digest. Each node will access stefanscherer/whoami:1.2.0 independently,
possibly leading to different nodes running different
versions of the image.

whoami

all progress: 0 out of 15 tasks
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
5:   <3>$<3>
5:   <3>$<3>
5:   <3>$<3>
5:   <3>$<3>
5:   <3>$<3>
ll progress: 15 out of 15 tasks
fy: Service converged to verify that tasks are stable...
```

In [41]:
```
docker service update --update-parallelism 5 \
  --detach=false \
  --image stefanscherer/whoami:1.1.0 whoami
```

```
image stefanscherer/whoami:1.1.0 could not be accessed on a registry to record
its digest. Each node will access stefanscherer/whoami:1.1.0 independently,
possibly leading to different nodes running different
versions of the image.

whoami

all progress: 0 out of 15 tasks
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
5:   <3>$<3>
5:   <3>$<3>
5:   <3>$<3>
5:   <3>$<3>
5:   <3>$<3>
ll progress: 15 out of 15 tasks
fy: Service converged to verify that tasks are stable...
```

In [42]:
```
docker service ls
```

```
ID                  NAME                MODE                REPLICAS            IMAGE                                PORTS
bql95rhwlmtn        monitor             global              5/5                 stefanscherer/monitor:1.1.0
ohooxit3x6pr        visualizer          replicated          1/1                 alexellis2/visualizer-arm:latest     *:8000->8
080/tcp
gkhxl78gt5tk        whoami              replicated          15/15               stefanscherer/whoami:1.1.0
```

Demomodus

In [43]:
```
END=1
for i in $(seq 1 $END); do
    docker service update --update-parallelism 5 \
        --detach=false \
        --image stefanscherer/whoami:1.1.0 whoami --detach=false
    docker service scale --detach=false whoami=1
    docker service update --update-parallelism 5 \
        --detach=false \
        --image stefanscherer/whoami:1.2.0 whoami --detach=false
    sleep 5
    docker service scale --detach=false whoami=15
    docker service update --update-parallelism 5 \
        --detach=false \
        --image stefanscherer/whoami:1.1.0 whoami --detach=false
    sleep 5
    docker service update --update-parallelism 5 \
        --detach=false \
        --image stefanscherer/whoami:1.2.0 whoami --detach=false
    sleep 5
    docker service scale --detach=false whoami=40
    sleep 5
    docker service update --update-parallelism 5 \
        --detach=false \
        --image stefanscherer/whoami:1.1.0 whoami --detach=false
    sleep 5
    docker service update --update-parallelism 5 \
        --detach=false \
        --image stefanscherer/whoami:1.2.0 whoami --detach=false
    sleep 5
done
```

```
image stefanscherer/whoami:1.1.0 could not be accessed on a registry to record
its digest. Each node will access stefanscherer/whoami:1.1.0 independently,
possibly leading to different nodes running different
versions of the image.

whoami

all progress: 0 out of 15 tasks
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
5:   <3>$<3>
5:   <3>$<3>
5:   <3>$<3>
5:   <3>$<3>
5:   <3>$<3>
ll progress: 15 out of 15 tasks
whoami scaled to 1ged to verify that tasks are stable...

all progress: 0 out of 1 tasks
all progress: 1 out of 1 tasks
image stefanscherer/whoami:1.2.0 could not be accessed on a registry to record
its digest. Each node will access stefanscherer/whoami:1.2.0 independently,
possibly leading to different nodes running different
versions of the image.

whoami

all progress: 0 out of 1 tasks
all progress: 1 out of 1 tasks
whoami scaled to 15ed to verify that tasks are stable...

all progress: 0 out of 15 tasks
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
:   $<3>$<3>
```

Aufräumen

In [44]:
```
docker service rm whoami
```

```
whoami
```

In [45]:
```
docker service ls
```

```
ID                  NAME                MODE                REPLICAS            IMAGE                                PORTS
bql95rhwlmtn        monitor             global              5/5                 stefanscherer/monitor:1.1.0
ohooxit3x6pr        visualizer          replicated          1/1                 alexellis2/visualizer-arm:latest     *:8000->8
080/tcp
```

In [46]:
```
docker service rm monitor
```

```
monitor
```

In [47]:
```
docker service ls
```

```
ID                  NAME                MODE                REPLICAS            IMAGE                                PORTS
ohooxit3x6pr        visualizer          replicated          1/1                 alexellis2/visualizer-arm:latest     *:8000->8
080/tcp
```

In [48]:
```
docker node ls
```

```
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS
pqp74pxig8s3zs07eafeugjuq *   cluster-00          Ready               Active              Reachable
0br7aj8dld5sr73vic82s3jo6     cluster-01          Ready               Active
v9odvkkycga6enf7ld6uok9vg     cluster-02          Ready               Active              Reachable
qocjbg98xx6pr5ig1zc4k3dpk     cluster-03          Ready               Active
c9rzz568a8lr2jzecbsveikxp     cluster-04          Ready               Active              Leader
```

In [49]:
```
docker service rm visualizer
```

```
visualizer
```

In [50]:
```
docker service ls
```

```
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS
```

In [51]:
```
for i in 01 02 03 04; do
    ssh cluster-${i}.local "docker swarm leave --force"
done
```

```
Node left the swarm.
Node left the swarm.
Node left the swarm.
Node left the swarm.
```

In [52]:
```
docker node ls
```

```
Error response from daemon: rpc error: code = DeadlineExceeded desc = context deadline exceeded
```

In [53]:
```
docker swarm leave --force
```

```
Node left the swarm.
```

Reboot Cluster

In [ ]:
```
for i in 01 02 03 04; do
    ssh cluster-${i}.local "sudo reboot &"
done
```