ISOLATING MULTIPLE PHP VERSIONS/APPS WITH DOCKER

THE GOAL OF THIS PRESENTATION

- Highlight a specific usecase
- Using Docker for our benefit
- Inspire you to research more about Docker and co.

WHAT THIS PRESENTATION IS NOT

- This is NOT an introduction into Docker
- This is NOT a security talk
- This is NOT an in depth talk about all the features Docker has

A VERY SHORT INTRODUCTION OF DOCKER

Docker is a container system, it uses resource isolation features of the Linux kernel such as cgroups and kernel namespaces. This is used to isolate the process and its software stack from the rest of the system without the overhead of a virtual machine. This will allow us to have a predictable outcome of the process running.

LEARN MORE ABOUT DOCKER

- docker website
- docker blog

THE GOAL

- Single Server Machine
- Multiple PHP installations
- Multiple PHP apps

THE SETUP

ARCHLINUX

Why ArchLinux as base OS ? Isn't Everyone using Debian based OS'es these days ?

1. I'm very familiar with it
2. systemd

NGINX

We'll use nginx because it is low hassle in configuration and works a lot better with fpm versus some other webservers.

PERCONA

Just because MySQL is no longer officially supported on ArchLinux and MariaDB gave me issues some time ago.

PHP VERSIONS

- PHP 5.3 dockerized based on ubuntu 12.04
- PHP 5.5 dockerized based on ubuntu 14.04
- PHP 5.6 dockerized based on ArchLinux

PHP APPS

- Wordpress: because it seems to be very widely used
- MyBB: nostalgia to a long forgotten past
- ownCloud: because it gives you back the control over your data


PHP
wvl
-wvl.be

BUILDING THE CONTAINERS

NOTE:

Newlines will be added for presentation pruposes

Only the relevant parts of the Dockerfile's is displayed

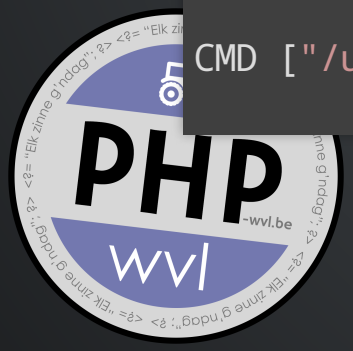
NGINX

```
FROM blackikeeagle/archlinux

RUN pacman -Syu --noconfirm nginx
RUN pacman -Scc --noconfirm
RUN mkdir -p /etc/nginx/conf.d
RUN mkdir -p /etc/nginx/sites-available
RUN mkdir -p /etc/nginx/sites-enabled
ADD ./config/nginx.conf /etc/nginx/nginx.conf
ADD ./config/01-wordpress.conf /etc/nginx/sites-available/01-wordpress.conf
ADD ./config/02-mybb.conf /etc/nginx/sites-available/02-mybb.conf
ADD ./config/03-owncloud.conf /etc/nginx/sites-available/03-owncloud.conf
RUN ln -s /etc/nginx/sites-available/01-wordpress.conf
    /etc/nginx/sites-enabled/01-wordpress.conf
RUN ln -s /etc/nginx/sites-available/02-mybb.conf
    /etc/nginx/sites-enabled/02-mybb.conf
RUN ln -s /etc/nginx/sites-available/03-owncloud.conf
    /etc/nginx/sites-enabled/03-owncloud.conf

EXPOSE 80

CMD ["/usr/sbin/nginx"]
```



PERCONA

```
FROM blackikeeagle/archlinux

RUN pacman -Syu --noconfirm percona-server
RUN pacman -Scc --noconfirm
RUN rm /etc/mysql/my.cnf
ADD ./my.cnf /etc/mysql/my.cnf
ADD ./create-mysql-structure.sh /opt/create-mysql-structure.sh
RUN chmod +x /opt/create-mysql-structure.sh

EXPOSE 3306

USER mysql

CMD ["/usr/bin/mysqld_safe", "--pid-file=/run/mysqld/mysqld.pid",
    "--bind-address=0.0.0.0", "--skip-name-resolve", "--datadir=/var/lib/mysql
```

PHP 5.3

```
FROM ubuntu:12.04

# wordpress php deps
RUN apt-get -y install php5-fpm php5-mysql php5-curl php5-gd php5-intl php-pea

RUN sed -e 's/.*daemonize.*/daemonize = no/' -i /etc/php5/fpm/php-fpm.conf
RUN sed -e 's/.*cgi.fix_pathinfo.*/cgi.fix_pathinfo=0/'
    -e 's/upload_max_filesize.*/upload_max_filesize = 100M/'
    -e 's/post_max_size.*/post_max_size = 100M/' -i /etc/php5/fpm/php.ini
RUN sed 's/.*listen =.*/listen = 0.0.0.0:9000/' -i /etc/php5/fpm/pool.d/www.co
RUN sed -e 's/.*listen =.*/listen = 0.0.0.0:9000/'
    -e 's/.*catch_workers_output.*/catch_workers_output = yes/'
    -i /etc/php5/fpm/pool.d/www.conf
RUN echo "php_admin_value[error_log] = stdout\nphp_admin_flag[log_errors] = on
    >> /etc/php5/fpm/pool.d/www.conf

EXPOSE 9000

CMD ["/usr/sbin/php5-fpm"]
```

PHP 5.5

```
FROM ubuntu:14.04

# mybb php deps
RUN apt-get -y install php5-fpm php5-mysql php5-gd php5-intl php-pear php5-sql

RUN sed -e 's/.*cgi.fix_pathinfo.*/cgi.fix_pathinfo=0/' -i /etc/php5/fpm/php.i
RUN sed -e 's/.*listen =.*/listen = 0.0.0.0:9000/'
    -e 's/.*catch_workers_output.*/catch_workers_output = yes/'
    -i /etc/php5/fpm/pool.d/www.conf
RUN echo "php_admin_value[error_log] = stdout\nphp_admin_flag[log_errors] = on
    >> /etc/php5/fpm/pool.d/www.conf

EXPOSE 9000

CMD ["/usr/sbin/php5-fpm", "-F"]
```

PHP 5.6

```
FROM blackikeeagle/archlinux

RUN pacman -Syu --noconfirm php-fpm php-gd php-mcrypt php-sqlite php-apcu
RUN pacman -Scc --noconfirm

RUN sed -e 's/.*cgi.fix_pathinfo.*/cgi.fix_pathinfo=0/' -i /etc/php/php.ini
RUN sed -e 's/.*listen =.*/listen = 0.0.0.0:9000/'
    -e 's/.*catch_workers_output.*/catch_workers_output = yes/'
    -i /etc/php/php-fpm.conf
RUN echo -e "php_admin_value[error_log] = stdout\nphp_admin_flag[log_errors] =
    >> /etc/php/php-fpm.conf
ADD ./enable_extensions.sh /tmp/enable_extensions.sh
RUN /bin/sh /tmp/enable_extensions.sh && rm /tmp/enable_extensions.sh
ADD ./apcu.ini /etc/php/conf.d/apcu.ini

EXPOSE 9000

CMD ["/usr/bin/php-fpm", "-F"]
```

WHAT IS IMPORTANT WHILE BUILDING

There is actually one thing very important to keep in mind while building containers. The processes running inside the container MUST NOT fork. So the process must stay in the foreground, if it forks docker will assume the process is done and quit the container.

THE ACTUAL BUILDING

Lets just show one example of building, basically it is always the same.

```
$ cd /var/docker/percona
#./build.sh

$ docker build -t blackikeeagle/percona .
```

for all options: docker build --help

CHECK WHAT IMAGES WE HAVE

After building all our base containers we might want to find out what their names are and maybe if we want to completely rebuild throw away an existing container and start 'from scratch'.

for all options: docker images --help

FOR OUR USECASE:

```
$ docker images
REPOSITORY                TAG                 IMAGE ID            CREATED
sample/owncloud-fpm       latest              5911540f5415        40 minutes a
sample/mybb-fpm           latest              143668a193b2        42 minutes a
sample/wordpress-fpm      latest              aa05762f4452        45 minutes a
blackikeeagle/percona     latest              fd2265e8f37d        49 minutes a
blackikeeagle/nginx       latest              ca827303f7fd        50 minutes a
ubuntu                    14.04               6b4e8a7373fe        45 hours ago
ubuntu                    12.04               b45a6cbea6d1        45 hours ago
blackikeeagle/archlinux   latest              b4bbf04b414e        5 days ago
```

HANDLING THE IMAGES

- run
- update
- remove

RUNNING A CONTAINER

for all options: docker run --help

TESTDRIVE A CONTAINER

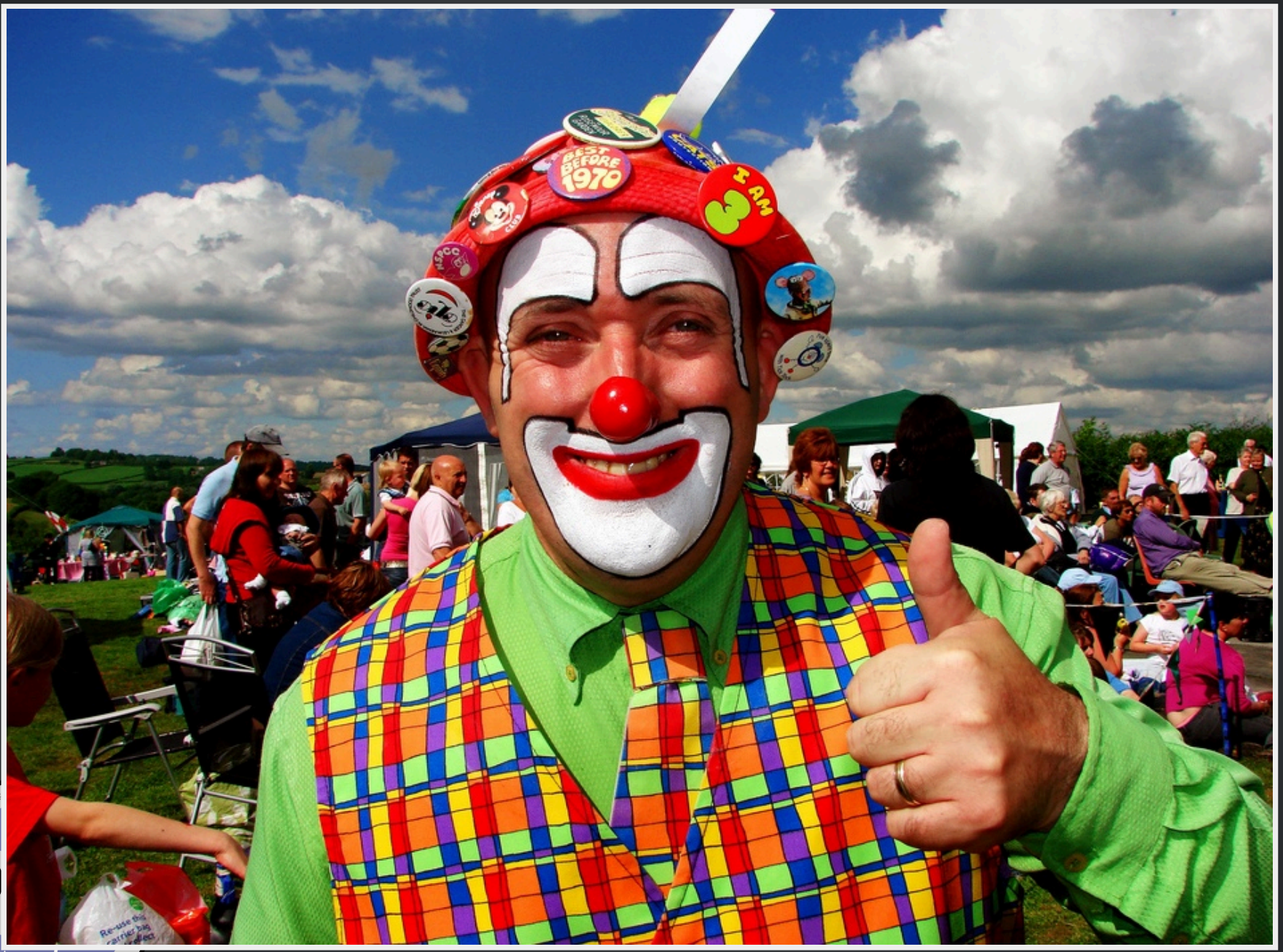
```
docker run blackikeeagle/percona
```

We'll see if it stays up. If so we've done superb

MAYBE DEBUG

```
docker run -i -t blackikeeagle/percona /bin/bash
```

We override the default CMD here with /bin/bash. This way we might be able to find out why a container is not running as we were expecting.


BEST BEFORE 1970
I AM 3

GETTING THERE

OUR APPS

To make it easy we just 'install' the apps on our host in subfolders of /srv

- /srv/wordpress
- /srv/mybb
- /srv/owncloud

PERCONA (MYSQL)

Since docker containers are stateless we'll have to store our mysql data in a volume. This way we can persist is between 'runs'.

- /var/lib/mysql

PERCONA (MYSQL)

We'll run our percona container with a volume mounted to the host

```
$ docker -v /var/lib/mysql:/var/lib/mysql blackikeeagle/percona
```

NOTE: we need to initialize mysql in /var/lib/mysql

THE DOCKER NETWORK SETUP

- Docker internal network: 192.168.1.0/24
- it is similar to host only networking with vagrant
- 'Host' will get 192.168.1.1
- the hosts exposed services will be reachable on 192.168.1.1

THE FPM'S

To make it easy for ourself we'll just run the fpm's on different ports, lets say 9001, 9002, 9003 :)

AN FPM

```
$ docker run \
    -v /srv/owncloud:/srv/owncloud \
    -p 192.168.1.1:9003:9000 \
    sample/owncloud-fpm
```

BRINGING IT ALL TOGETHER WITH PORTS

RUNNING PERCONA

```
$ docker run \
    -v /var/lib/mysql:/var/lib/mysql \
    -p 192.168.1.1:3306:3306 blackikeeagle/percona
```

RUNNING THE FPM'S

```
$ docker run \
    -v /srv/wordpress:/srv/wordpress \
    -p 192.168.1.1:9001:9000 sample/wordpress-fpm
```

```
$ docker run \
    -v /srv/mybb:/srv/mybb \
    -p 192.168.1.1:9002:9000 sample/mybb-fpm
```

```
$ docker run \
    -v /srv/owncloud:/srv/owncloud \
    -p 192.168.1.1:9003:9000 sample/owncloud-fpm
```

Here you see a one-to-one mapping of the volume's, this is not required, but handy

NGINX FPM CONFIG

```
# Configure PHP-FPM stuff
location ~ ^(?<script_name>.+?\.php)(?<path_info>/.*)?$ {
    try_files $script_name = 404;
    fastcgi_pass  192.168.1.1:9003;
    fastcgi_param PATH_INFO $path_info;
    fastcgi_param HTTPS $https;
    fastcgi_intercept_errors on;

    fastcgi_param PHP_VALUE
        "upload_max_filesize = 1024M \n post_max_size = 1024M";
    fastcgi_param SCRIPT_FILENAME /srv/owncloud$script_name;
    include fastcgi_params;
}
```

RUNNING NGINX

```
docker run \
    -v /srv/wordpress:/srv/wordpress \
    -v /srv/mybb:/srv/mybb \
    -v /srv/owncloud:/srv/owncloud \
    -p 0.0.0.0:80:80 blackikeeagle/nginx
```

BRINGING IT ALL TOGETHER WITH --LINK

RUNNING PERCONA

```
$ docker run \
    -v /var/lib/mysql:/var/lib/mysql \
    -p 192.168.1.1:3306:3306 blackikeeagle/percona
```

RUNNING THE FPM'S

```
$ docker run \
    -v /srv/wordpress:/srv/wordpress \
    --link percona:mysql sample/wordpress-fpm
```

```
$ docker run \
    -v /srv/mybb:/srv/mybb \
    --link percona:mysql sample/mybb-fpm
```

```
$ docker run \
    -v /srv/owncloud:/srv/owncloud \
    --link percona:mysql sample/owncloud-fpm
```

Here you see a one-to-one mapping of the volume's, this is not required, but handy

NGINX FPM CONFIG

```
# Configure PHP-FPM stuff
location ~ ^(?<script_name>.+?\.php)(?<path_info>/.*)?$ {
    try_files $script_name = 404;
    fastcgi_pass  owncloud:9000;
    fastcgi_param PATH_INFO $path_info;
    fastcgi_param HTTPS $https;
    fastcgi_intercept_errors on;

    fastcgi_param PHP_VALUE
        "upload_max_filesize = 1024M \n post_max_size = 1024M";
    fastcgi_param SCRIPT_FILENAME /srv/owncloud$script_name;
    include fastcgi_params;
}
```

RUNNING NGINX

```
docker run \
    -v /srv/wordpress:/srv/wordpress \
    -v /srv/mybb:/srv/mybb \
    -v /srv/owncloud:/srv/owncloud \
    --link wordpress-fpm:wordpress \
    --link mybb-fpm:mybb \
    --link owncloud-fpm:owncloud \
    -p 0.0.0.0:80:80 blackikeeagle/nginx
```

ENVIRONMENT VARIABLES

```
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=d5f803ba46fe
MYBB_PORT=tcp://192.168.1.4:9000
MYBB_PORT_9000_TCP=tcp://192.168.1.4:9000
MYBB_PORT_9000_TCP_ADDR=192.168.1.4
MYBB_PORT_9000_TCP_PORT=9000
MYBB_PORT_9000_TCP_PROTO=tcp
MYBB_NAME=/clever_yonath/mybb
MYBB_ENV_DEBIAN_FRONTEND=noninteractive
OWNCLOUD_PORT=tcp://192.168.1.5:9000
...
WORDPRESS_NAME=/clever_yonath/wordpress
WORDPRESS_ENV_DEBIAN_FRONTEND=noninteractive
HOME=/root
```

HOSTS ENTRIES

```
192.168.1.8     040a554b4ed3
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
127.0.0.1       localhost
192.168.1.3     wordpress
192.168.1.4     mybb
192.168.1.5     owncloud
```

USING --LINK

For this kind of setup using --link will be more error prone than using the ports method. When an fpm container accidentally restarts, the link from nginx to it will be gone. With the port method it will be just fine when fpm is back up.

USING --LINK

If you want to use docker for developemnt boxes, I personally would prefer using --link, especially when you have a lot of different projects.

REAL LIFE

Yeah, you know, docker is cool and all, but in a VM or bare metal server we just do `systemctl enable nginx.service` and we are sure if the machine reboots everything is up

on debian it would be `service nginx start`

WE CAN DO THAT

We can easily write unitfiles for our docker containers so they will be started automatically at boot.

FPM

```
[Unit]
Description=mybb-fpm
After=docker.service

[Service]
TimeoutStartSec=0
ExecStartPre=-/usr/bin/docker kill mybb-fpm
ExecStartPre=-/usr/bin/docker rm mybb-fpm
ExecStart=/usr/bin/docker run --name mybb-fpm
    -v /srv/mybb:/srv/mybb -p 192.168.1.1:9002:9000 sample/mybb-fpm
ExecStop=/usr/bin/docker stop mybb-fpm

[Install]
WantedBy=multi-user.target
```

PERCONA

```
[Unit]
Description=percona
After=docker.service

[Service]
TimeoutStartSec=0
ExecStartPre=-/usr/bin/docker kill percona
ExecStartPre=-/usr/bin/docker rm percona
ExecStart=/usr/bin/docker run --name percona
    -v /var/lib/mysql:/var/lib/mysql -p 192.168.1.1:3306:3306 blackikeeagle/pe
ExecStop=/usr/bin/docker stop percona

[Install]
WantedBy=multi-user.target
```

NGINX

```
[Unit]
Description=nginx
After=docker.service

[Service]
TimeoutStartSec=0
ExecStartPre=-/usr/bin/docker kill nginx
ExecStartPre=-/usr/bin/docker rm nginx
ExecStart=/usr/bin/docker run --name nginx
    -v /srv/wordpress:/srv/wordpress -v /srv/mybb:/srv/mybb
    -v /srv/owncloud:/srv/owncloud -p 0.0.0.0:80:80 blackikeeagle/nginx
ExecStop=/usr/bin/docker stop nginx

[Install]
WantedBy=multi-user.target
```

WHY SYSTEMD

systemd is by now the most spread next generation init system for linux. Even debian and ubuntu have voted to use it in the future. Although these still use upstart, we need to look at the future.

ENABLE ON BOOT

```
$ systemctl enable wordpress-fpm.service \
    mybb-fpm.service owncloud-fpm.service \
    percona.service nginx.service
```

DONE

We now have a working minimal system where all our services are running in docker containers.

DEMO (CODE)

There is a working demo for this presentation, the code can be found on github:
https://github.com/BlackIkeEagle/docker-php-samples

NOTES

DOCKERIZED VOLUMES

Instead of mounting volumes on the host system we could have used dockerized volumes. The reason not to do that is, it is very hard to persist your data between boots that way.

SITES-ENABLED VOLUME

In this usecase the sites-enabled is pre-filled with the configurations we wanted to have. A more flexible approach could be to add a volume to /etc/nginx/sites-enabled so you can more dynamically update your setup.

'LOG' INTO A RUNNING CONTAINER

Before v 0.9 of docker it was impossible to get into a running container and do some additional stuff with it. If you have a recent linux distribution with util-linux > 2.23, you can use nsenter.

```
$ nsenter --target $(docker inspect --format "{{ .State.Pid }}" nginx) \
    --mount --ipc --uts --net --pid
```

NGINX RELOAD

Enter the container with nsenter and ...

```
$ /usr/bin/kill -HUP 1
```

INTERESTING PROJECTS USING DOCKER

COREOS

CoreOS is a operating system specifically designed around docker. With some additional tools it is the OS that can be run as a cluster and be setup for High Availabilty environments

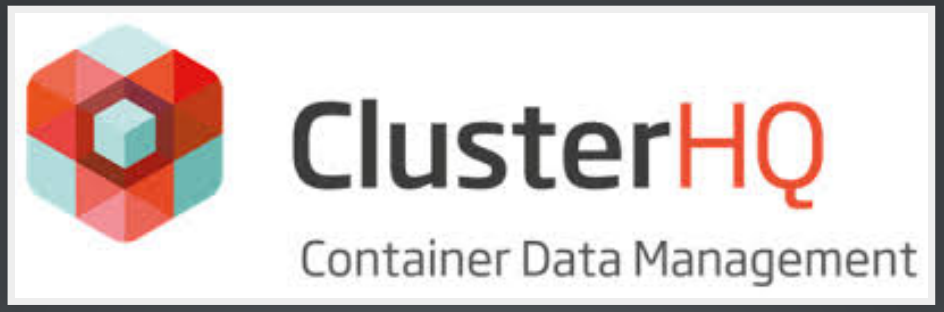
MESOS

Apache Mesos is a cluster manager that simplifies the complexity of running applications on a shared pool of servers.

FIG

Fast, isolated development environments using Docker.

FLOCKER



Flocker is a data volume manager and multi-host Docker cluster management tool.

FLYNN

Flynn is a PaaS (Platform as a Service) based upon container technologies. And it can easily handle statefull applications.

QUESTIONS ?

WARNING: you might have a lot of questions I don't know the answer to.

USEFULL LINKS

- docker.com
- docker blog
- awesome docker
- nkode valuable docker links
- CoreOS
- Apache Mesos
- fig
- ClusterHQ
- flynn.io
- centurylinklabs docker filesystem persistence with flocker
- centurylinklabs flyn vs deis

THANKS.

https://joind.in/talk/view/11934

IKE DEVOLDER

@BlackIkeEagle

Senior Webdeveloper - Studio Emma

Archlinux Trusted User

enthousiast about docker


PHP
-wvl.be
wvl