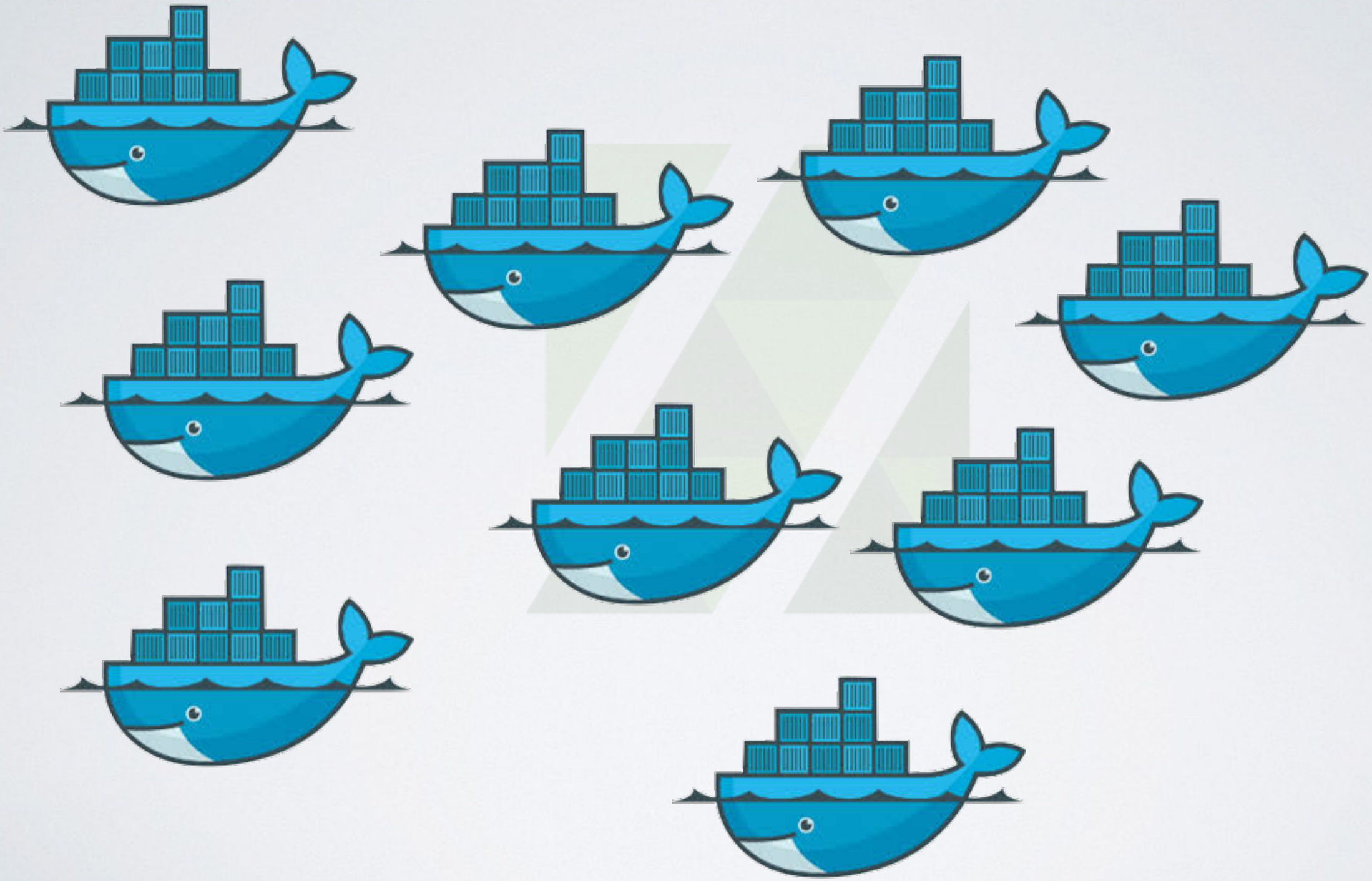
DOCKER 101 FOR JS AFFICIONADOS

Christian Ulbrich, Zalari UG

AGENDA

- Docker what is all the craze about?
- Docker is hard
- One-Liners
- Orchestration
- Outlook
- Links

DOCKER WTF?



DOCKER WTF?

- Docker is light-weight virtualization
- Isolated full-blown applications with their own process hierarchy can be started as a user processes (called *container*) sharing the same (Linux) kernel
- -> software running on top of Linux can be run as *apps*

DOCKER WTF?

- templates for containers are called *images*
- images can be (automagically) *built* from a *Dockerfile*
- images can be *tagged* (versioned)
- there is a central *registry* for images (DockerHub)
- you *pull* and *push* images to the registry

DOCKER WHY?

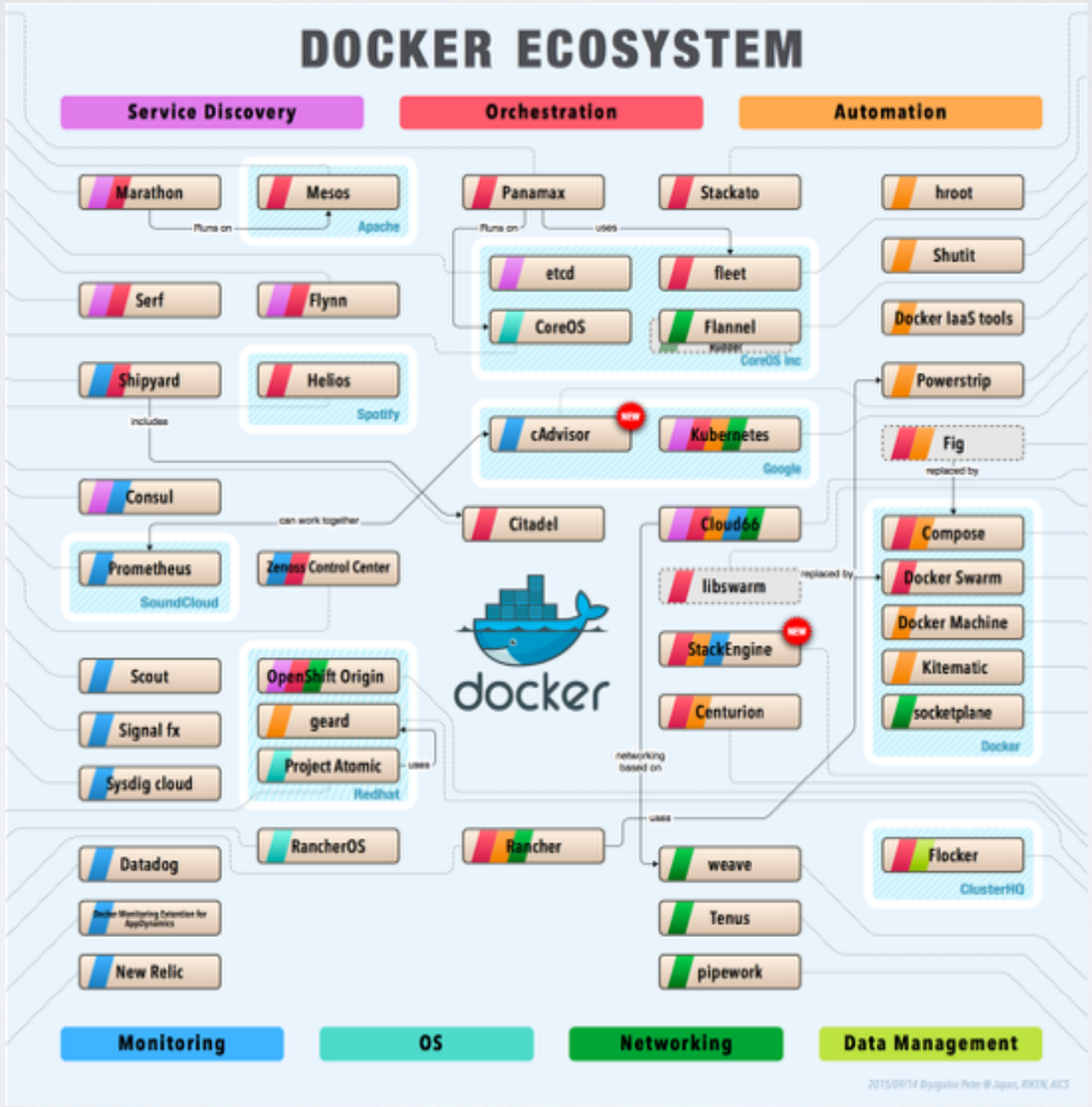
- high scalability -> much more concurrent docker processes (and thus kernels) than conventional virtualization
- no more software dependencies!
- a good candidate for a microservice architecture
- feature-rich eco system
- DevOps!

DOCKER WHY?

- no more software dependencies -> perfect for developer bootstrapping
- you only need the docker engine on your PC, no more Java, Gradle, Maven, …
- software is via pulling images from a registry as a blob

DOCKER IS HARD

DOCKER IS HARD



DOCKER IS HARD

- full-blown application stack with docker is a challenge
- very complex and verrrry dynamic eco systems -> you need to learn a lot
- logging, monitoring
- security
- managing file data
- resilence of applications

DOCKER IS HARD

- your good to go, if you already have:
  - automated deployments
  - automated provisioning
  - resilent services
  - working monitoring
  - working, secure offsite backups

DOCKER IS HARD

- „Docker is a great optimization - but it needs a firm foundation to live on“ (Matt Jaynes)

ONE-LINERS

- commands can be packaged as apps
- Maven, Gradle…

```
docker run -v $PWD:/tmp/work -w /tmp/work --rm maven mvn clean install


docker run -v $PWD:/tmp/work -w /tmp/work --rm zalari/swagger-codegen \
      generate -i http://localhost:8080/api/v1/swagger.json \
      -l typescript-angular2 -o .tmp/generated
```

ONE-LINERS

- *docker run* for running a container
- *-v $PWD:/tmp/work* mounts the current dir into the container and *-w /tmp/work* sets the current dir in the container
- —*rm* removes the container, after it has finished
- maven is the name of the *image* used to bootstrap the container
- *mvn clean install* is the command to run inside the container, which works on stuff outside because of the mount

DOCKER BUILD

- or even node itself for running your builds:

```
docker run --rm -v "$PWD":/usr/src/app -w /usr/src/app node:argon \
 /bin/bash -c "npm rebuild node-sass && npm run package"
```

- against a defined version (*argon*)
- cheat by using our *local* node_modules and *rebuild node-sass* downloads the (Linux) binaries for SASS
- */bin/bash -c* starts a shell *inside* the container for chaining of commands

ONE-LINERS

- deploy frontend and backend as images
- frontend is static, use nginx to serve
- backend is served using node
- images are build with Dockerfiles

ONE-LINERS

- or even node itself for running your builds:

Dockerfile

```
FROM nginx

#statically copy all code
COPY ./dist /usr/share/nginx/html/
```

```
docker build -t dresdenjs/frontend .
docker run --rm -p 9000:80 dresdenjs/frontend
```

ONE-LINERS

- *docker build* for building an image
- *-t dresdenjs/frontend* „gives“ the image a name
- . is the path to the Dockerfile
- *docker run* for running a container of an image
- *-p 9000:80* exposes the inside port 80 to the outside port 9000

ONE-LINERS

- Docker becomes a building block for your CI
- CI only needs to allow to run Docker
  - homegrown (such as GitLab)
  - … Wercker, Shippable, Drone, Travis, …

SOME ORCHESTRATION

- You might want to *orchestrate* containers, i.e. a group of container forms an app
- Orchestration is a **huge** topic

SOME ORCHESTRATION

- docker-compose is your f(r)iend
- it basically allows for setting the abundance of parameters for the docker demon in a YAML file

SOME ORCHESTRATION

```yaml
version: '2'

services:

  db:
    image: mongo
    volumes_from:
      - db_data

  db_data:
    image: mongo
    command: /bin/true
    volumes:
      - /data/db

  backend:
    image: dresdenjs/backend

  frontend:
    image: dresdenjs/frontend

  proxy:
    image: nginx
    volumes:
      - ./conf.d:/etc/nginx/conf.d
    ports:
      - '80:80'
    restart: always
```

SOME ORCHESTRATION

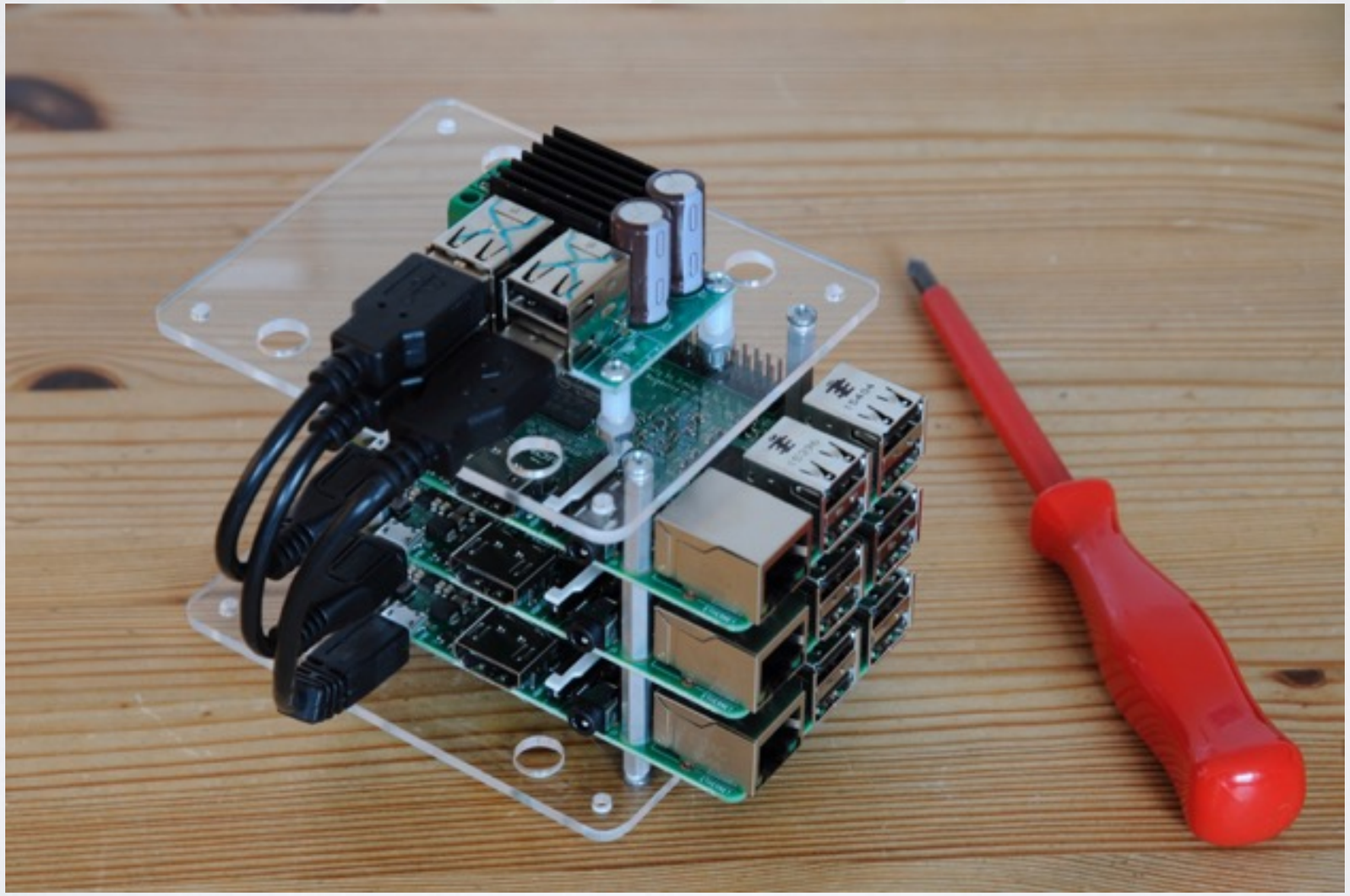
- you need to think about setting up your docker images, what should they contain?
- how are they parametrized?
- how do they match in version numbers?
- where do the version numbers come from?
- very soon you will need a private docker registry

OUTLOOK

- Docker is *addictive* - everything is a container may become your *Mantra*!
- Docker is *contagious* and *infectious* (especially for managers)
- *native* Docker for OS X + Windows
- Docker without serious provisioning is hard
- Multi-Host, i.e. high-availability, load-balanced, scalable deployments are complex and there are so many tools to choose from!

OUTLOOK

- clustered Docker on Rasberry Pi



LINKS

- Docker Misconceptions http://bit.ly/1KCdllk
- The Docker Book https://www.dockerbook.com/
- Docker-Awesome http://bit.ly/2aNAL0A
- Docker Pirates ARMed… http://bit.ly/1ZDOyGo