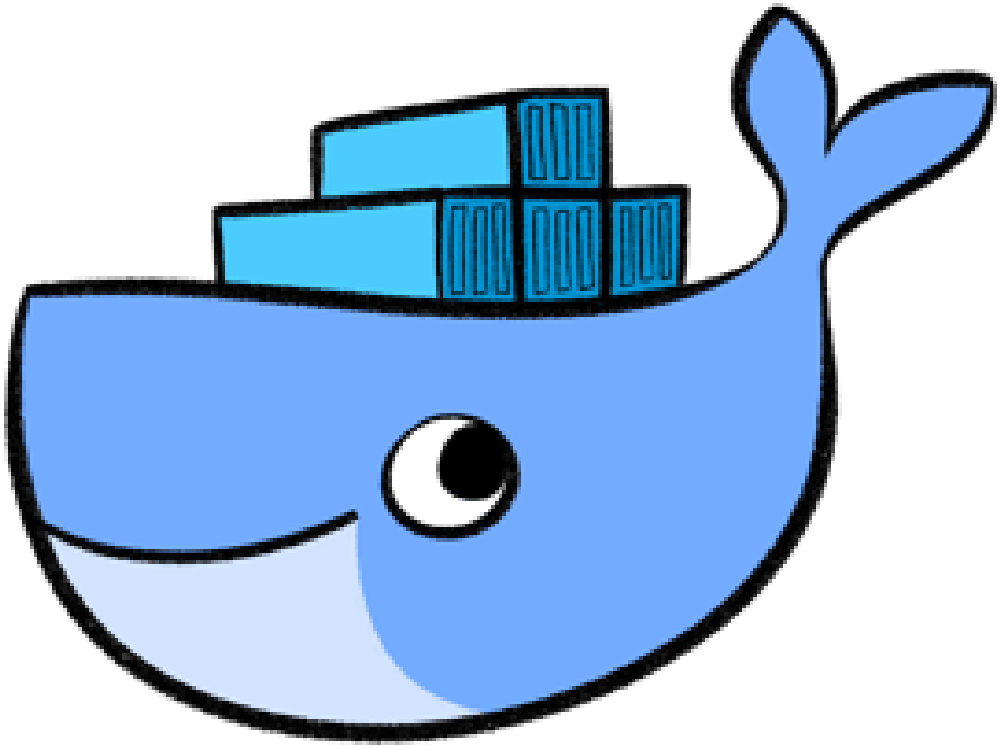
Induction to Docker



by Maciek Plachta

Agenda

- Demistifying containers - what they are and what they are not
- Docker - what it is and what technology stack made it number one on the market
- What is its purpose, and when it is useful
- Docker ecosystem
- Real life scenarios and aplications

Presentation outline

A story of three acts:

I. Concept behind containers

(presentation warm-up)

II. Case study

(learn by example)

III. Diving into details

(what the heck just happened?)

Spaces for applying containers:

- highly coupled systems with a huge dependencies chain,
- having the same environments for development, CI and production,
- systems which scales both ways,
- microservices,
- design requires strong abstraction,
- reduce provisioning (make things portable),

Spaces for applying Containers (continued):

- request of port and service mapping,
- ‘*it works for me*’,
- ‘*you want to run what?*’,
- ‘*you did what ?!*’,
- reproducibility is of an essence,
- sandboxing and doing experiments with a different tech stack, where one wants to preserve the current environment unaffected.

Container X-ray

"A Container is not a lightweight VM"

Container *acts like* VM, but they are only *a self contained processes*. They use Linux Kernel features to limit their resources and access rights. Container lives as long as their root process and exits immediately when it stops.

Container X-ray

Container has:

- binaries
- libraries
- file system

Container use:

- host network
- host kernel
- host resources

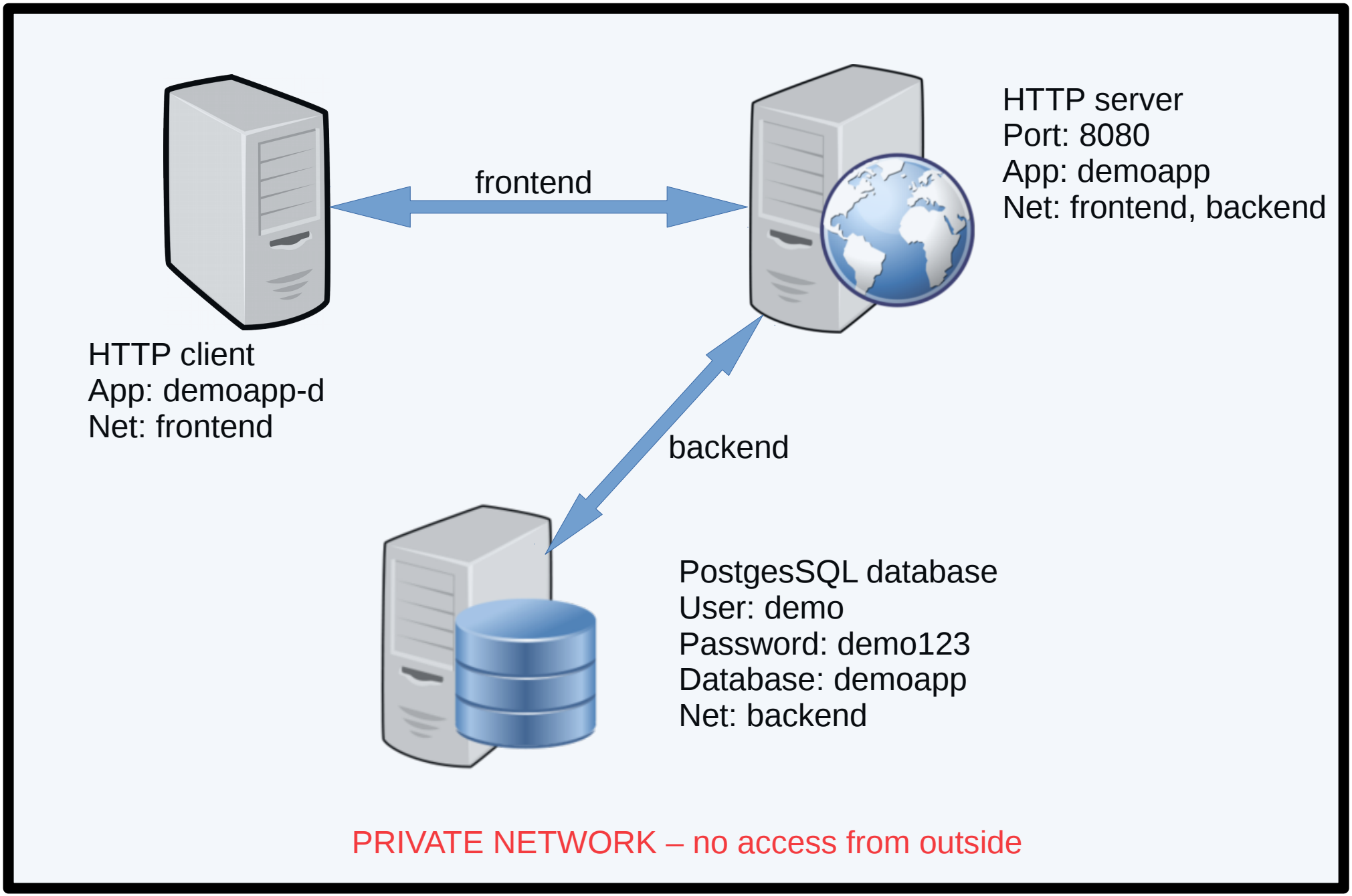
Docker

In short. It is one of the container standards which has a large ecosystem around it, strong community and business model built on top of an open project codebase.

Launched at March 2013, hit v1.0 in a late 2014, and had over 100 millions downloads till 2015.

...Oh and it was written in **Go**. An accident ?
I DON’T THINK SO !!! :-)

Case study: small, distributed system



Solution: Slice this cake!

- Stage 1: Let’s make services deployable
- Stage 2: Configure database
- Stage 3: Configure networks
- Stage 4: Link everything together
- Stage 5: Test it

Demystifying Docker

Over time Docker has scaled up rapidly. So it’s model evolved. The current state is:

- Moby project (open source community driven)
- Docker project (based on Moby core but also containing proprietary content).
  - Docker CE (stable, edge) – free of charge fully supported ecosystem,
  - Docker EE (stable, edge) – paid, extra support.

Docker Ecosystem

- Docker Engine (also Docker daemon/server)
- Docker CLI
- Docker Swarm
- Docker Compose
- Docker Machine
- Docker for Mac / Docker for Windows (port)
- Docker Registry aka Docker Hub
- Docker Cloud
- Docker Store

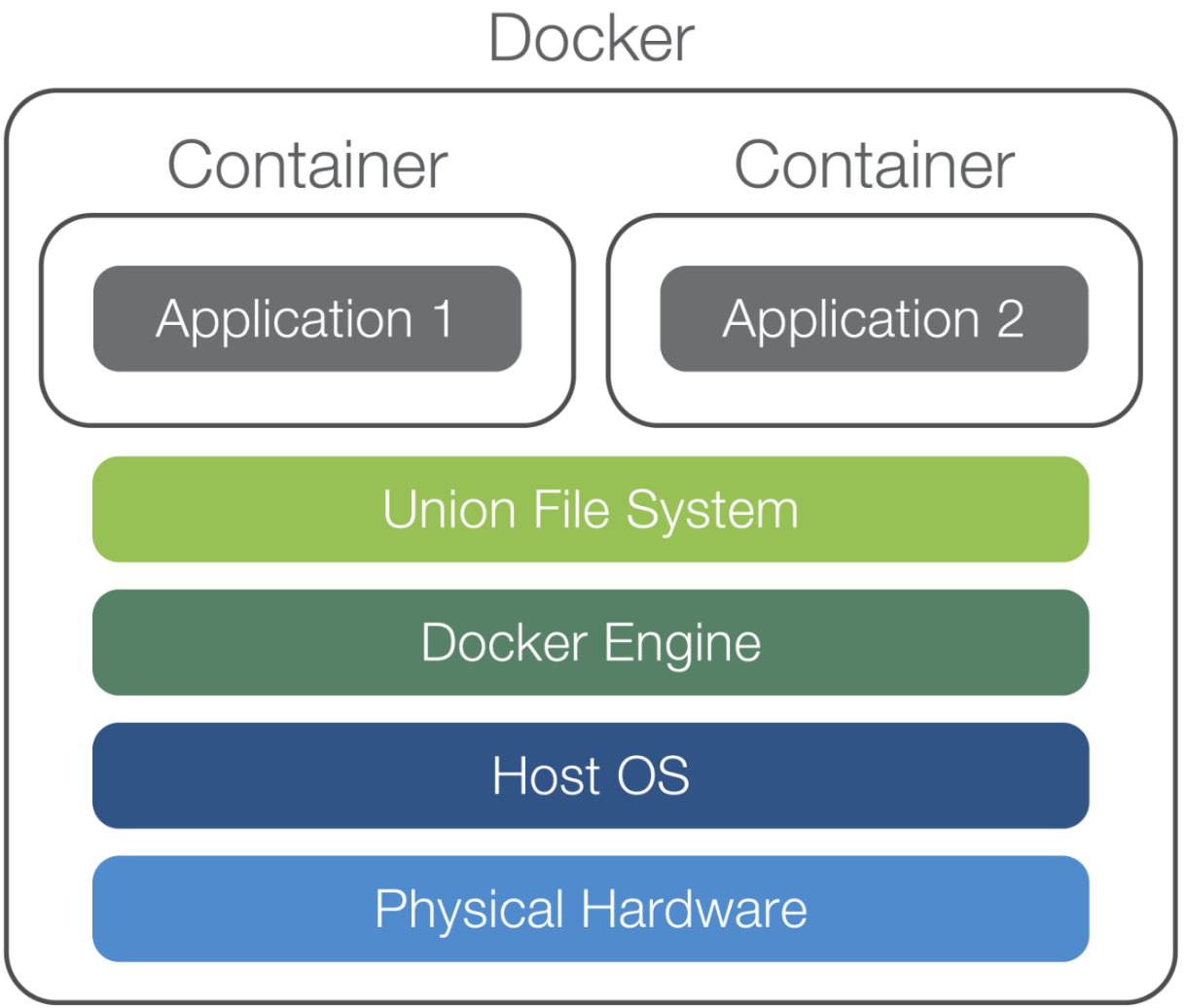
Docker Image

- Application binaries and dependencies
- Metadata about the image data and how to run it
- Not a complete OS. No kernel modules(drivers). The host provides its kernel.
- By definition: ‘An image is an ordered collection of root filesystem changes and the corresponding execution parameters for use within a container runtime’

Docker Container

A textbook definition explained before plus instantiation of a docker image. It is an additional layer build on top of an image with read/write privileges (immutable, ephemeral by default). You can have multiple containers spawned from just one image.

Docker layout



Docker Virtual Network Model
(“Battery included but removable”)

Three types of network drivers:

- *bridge* (default) – secure, different subnet, behind NAT firewall
- *host* – no NAT, container use host network,
- *none* – only localhost,

….. plus various plugins

Container lifetime & Persistent Data

By default everything within a container lives and dies with it. When one want data to outlive container it has to options:

- *bind mounts* – map host file/directory to the container space.
- *volume* – makes special memory location outside of a container File System layer, handled by docker storage driver

Summary

- Container is just a process
- Containers are stateless by default
- Docker image is just an app we want to run
- Docker container is an instance of an image running as a process
- You can spawn multiple containers from just one image
- Docker API is easy to use
- Docker Hub is a mine of official images, as well as custom ones
- Build, ship and run application in one environment to minimalize failure

References

1. Bret Fisher docker course on Udemy
2. docs.docker.com
3. store.docker.com
4. get.docker.com
5. hub.docker.com

Thanks