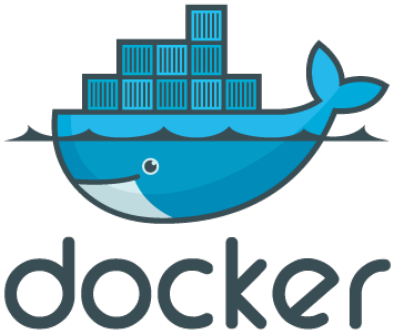
Docker Fundamentals

What? Where? How?



DATAART
Enjoy IT

About instructor

Yuriy Kushch

- Works @DataArt
- Over 5 years of experience.
- A big fan of high performance stuff, interested in JVM related things and front-end.

@YuriKushch

DATAART
Enjoy IT

Agenda

- What is Docker?
- Docker vs Virtual Machines
- Docker platform overview
- First steps with containers

DATAART
Enjoy IT

Requirements

1. Linux machine (Ubuntu will be great)
2. Cloud provider:
    a. Digital Ocean
    b. Amazon EC2
3. Familarity with command line.

DATAART
Enjoy IT

What is Docker?

Docker is an open platform for building, shipping and running distributed applications.

It gives programmers, development teams and operations engineers the common toolbox they need to take advantage of the distributed and networked nature of modern applications. ©

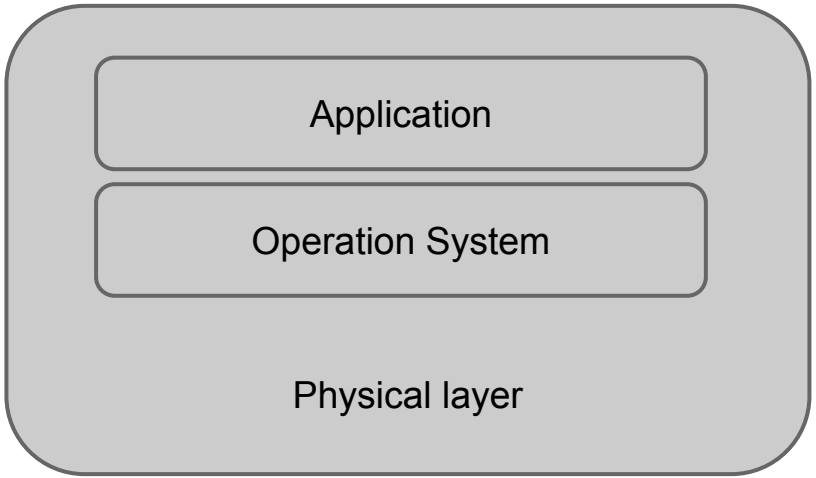
DATAART
Enjoy IT

Docker Platform

- Docker Engine
- Docker Hub
- Docker Machine
- Docker Swarm
- Docker Compose
- Kitematic

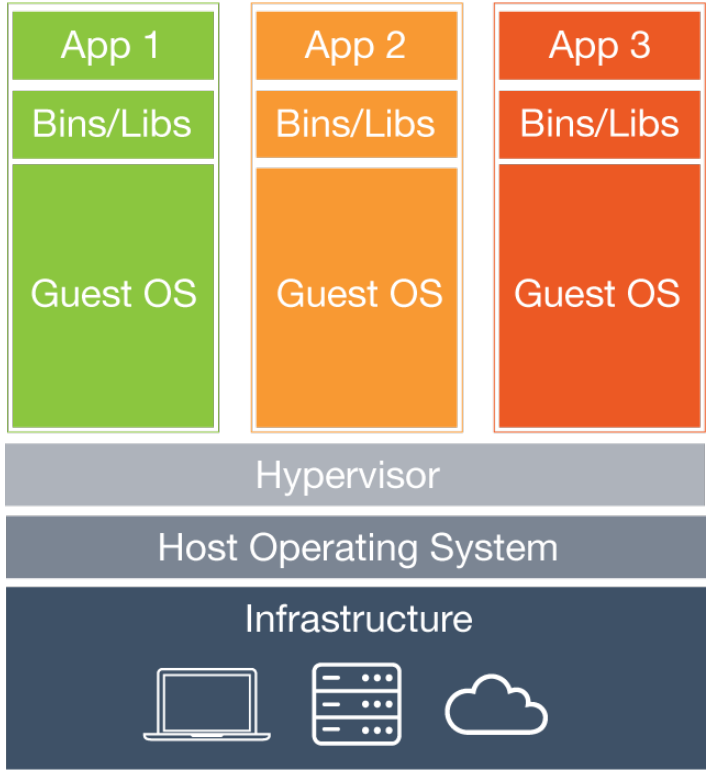
In the past

- Slow deployments
- Difficult to scale
- Difficult to migrate
- Vendor lock in



Hypervisor based solution

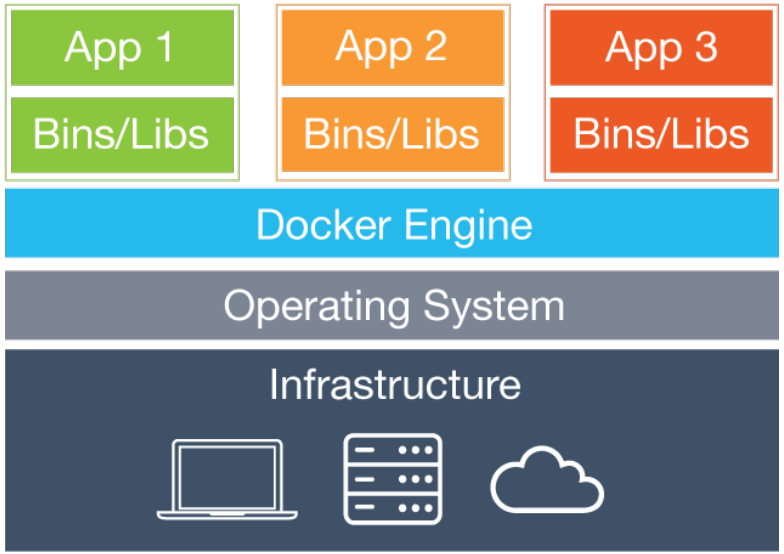
- 1 server - multiple VMs
- Each application runs on VM



Containers

Containers include the application and all of its dependencies, but share the kernel with other containers.

- Run as an isolated process
- Not tied to specific infrastructure



DATAART
Enjoy IT

Docker vs VMs

- Lightweights containers
- No need to install the required set of software
- Less CPU, RAM required
- Great portability

DATAART
Enjoy IT

Docker Engine

- A lightweight runtime and robust tooling that builds and runs Docker containers.
- Runs on Linux to create the operating environment for your distributed applications.
- Execute commands to build, ship and run containers.

DATAART
Enjoy IT

Docker Client and Daemon

- Docker client send user input to docker daemon
- Daemon could be remote
- Kitematic as GUI

Containers & Images

Image

- Docker images are the build component of Docker.
- A read-only template.
- Images are used to create Docker containers.

Container

- Docker containers are similar to a directory.
- Holds everything that is needed for an application to run.
- Each container is created from a Docker image.
- Docker containers can be run, started, stopped, moved, and deleted.
- Each container is an isolated and secure application platform.

DATAART
Enjoy IT

Docker Registry/HUB

- Docker registries hold images.
- Public or private stores from which you upload or download images.
- The public Docker registry is called Docker Hub.

DATAART
Enjoy IT

Docker PROS

- Fast Development cycle
- Portability
- Scalability
- System Admins are focused on deployment and developers on development

DATAART
Enjoy IT

Docker in Action

Install Docker

```
sudo aptitude    update
sudo aptitude -y upgrade
```

Make sure aufs support is available:

```
sudo aptitude install linux-image-extra-`uname -r`
```

Add docker repository key to apt-key for package verification:

```
sudo sh -c "wget -qO- https://get.docker.io/gpg | apt-key add -"
```

Install Docker - 2

Add the docker repository to aptitude sources:

```
sudo sh -c "echo deb http://get.docker.io/ubuntu docker main\
> /etc/apt/sources.list.d/docker.list"
```

Update the repository with the new addition:

```
sudo aptitude update
```

Download and install docker:

```
sudo aptitude install lxc-docker
```

docker images

Lists all images that are available on host.

```
$ docker images
REPOSITORY      TAG      IMAGE ID       CREATED      VIRTUAL SIZE
training/webapp latest   fc77f57ad303   3 weeks ago  280.5 MB
ubuntu          13.10    5e019ab7bf6d   4 weeks ago  180 MB
ubuntu          saucy    5e019ab7bf6d   4 weeks ago  180 MB
ubuntu          12.04    74fe38d11401   4 weeks ago  209.6 MB
ubuntu          precise  74fe38d11401   4 weeks ago  209.6 MB
```

docker run

Command that gives ability to run a container.

- `$ docker run [options] [image] [command] [args]`

```
$ docker run container:123.1 echo “Hello World”
```

DATAART
Enjoy IT

[Example] Run simple container

Fetch container (if not done yet) and run it:

```
docker run ubuntu echo “Hello World”
```

1. Run container and attach to STDIN

```
docker run -i -t ubuntu:14.04 /bin/bash
```

2. Create file

```
touch file1
```

3. Exit from container and start again

```
exit
```

```
docker run -i -t ubuntu:14.04 /bin/bash
```

Container ID

ID usually long but when list - they’re short.

- Long ID could be found via

```
docker inspect CONTAINER_ID
```

- Short ID could found via

```
docker ps
```

DATAART
Enjoy IT

Container as daemon

1. To run container in background as daemon, consider using the `-d` flag
2. To see output use

```
docker logs CONTAINER_ID
```

DATAART
Enjoy IT

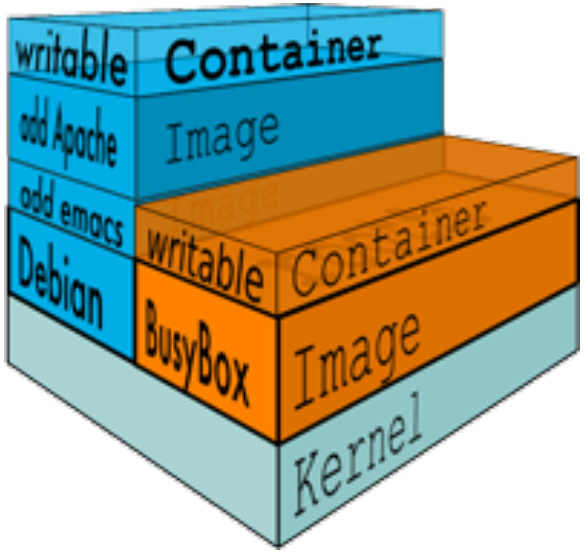
Using docker to run Tomcat

- Run a web application inside a container
- Map **ALL** ports from container to host `-P`

```
docker run -d -P tomcat:7
```

How it looks?

Docker creates new writeable layer



docker commit

Saves changes (current state) to the new image

```
$ docker commit <container_id> <some_name>
```

DATAART
Enjoy IT

DOCKERFILE

DATAART
Enjoy IT

What is DOCKERFILE

- A text document that contains all the commands you would normally execute manually in order to build a Docker image.
- By calling docker build from your terminal, you can have Docker build your image step by step, executing the instructions successively.

DOCKERFILE Instructions

- `FROM` instruction for specifying the base image
- `MAINTAINER` specify who maintains this dockerfile
- `RUN` instruction for executing command

```
FROM phusion/baseimage

MAINTAINER YOUR_NAME <your.email@gmail.com>

RUN \
  sed -i 's/# \(.*multiverse$\)/\1/g' /etc/apt/sources.
list && \
  apt-get update && \
  apt-get -y upgrade && \
  locale-gen en_US.UTF-8
```

DOCKERFILE Instructions - 2

- `ENV` instruction sets the environment variable. This value will be in the environment of all “descendent” Dockerfile commands and can be replaced inline in many as well.

```
ENV JAVA_HOME /usr/lib/jvm/java-8-oracle
```

docker build

Build your image step by step, executing the instructions successively.

```
docker build -t myimage .
```

CMD command

Possible forms of execution:

- `CMD ["executable","param1","param2"]`
- `CMD ["param1","param2"]`
- `CMD command param1 param2`

1. There can only be one `CMD` instruction in a `Dockerfile` (if multipe occur then only the last one will be executed).
2. **The main purpose of a `CMD` is to provide defaults for an executing container.**

```
CMD [“ping”, “127.0.0.1”, “-c”, “30”]
```

ENTRYPOINT

Possible forms of execution:

- ENTRYPOINT ["executable", "param1", "param2"] (the preferred *exec* form)
- ENTRYPOINT command param1 param2 (*shell* form)

1. Run-time arguments and CMD instruction are passed as parameters to the ENTRYPOINT instruction
2. Shell and EXEC forms are possible

```
ENTRYPOINT [“ping”]
```

DATAART
Enjoy IT

Start & Stop containers

- `docker stop CONTAINER_ID` - stop the container
- `docker start CONTAINER_ID` - start the stopped container

DATAART
Enjoy IT

Quick stop & remove ALL

One liner to stop / remove all of Docker containers:

```
docker stop $(docker ps -a -q)
docker rm $(docker ps -a -q)
```

DATAART
Enjoy IT

The End

http://bit.ly/1eUna6l