The age of orchestration
From Docker basics to cluster management

NICOLA PAOLUCCI • DEVELOPER INSTIGATOR • ATLASSIAN • @DURDN
Three minute Docker intro?

Time me and ring a bell if I am over it. Just kidding I’ll be over by a bit but it’s ok. We’re friends.
DEFINITIONS

Traditional Virtual Machines

APP A
BINS/LIBS
GUEST OS
HYPervisor
HOST OS
SERVER

APP B
BINS/LIBS
GUEST OS

ROBUST MONOLITHIC
SLOW TO BOOT
HEAVY OVERHEAD
DEFINITIONS

Docker Containers

APP A
BINS/LIBS

APP B
BINS/LIBS

DOCKER ENGINE
HOST OS
SERVER

LEAN
LIGHTWEIGHT
ISOLATED
PORTABLE
EFFICIENT
DEFINITIONS

Seen from another angle, the core of Docker is four distinct things:

- A standard format to package applications
- Clearly defined interfaces
- Caching mechanism to re-use steps
- Central registry of ready images
DO WE USE IT?

We have embraced Docker on two fronts

For our internal PaaS

In our products
Our internal PaaS, called Micros
Overall Micros Numbers

600+ Microservices

40% Docker containers

Java, Node.js, Python

Rest is pre-made stacks
Our Conversion Stack

ImageMagick

ghostscript

node.js

LibreOffice

FFmpeg
Media Services Numbers

- **10** Microservices
- **TBs** Processed every month
- **6M** containers spun per month
CONCLUSIONS

Docker has been a great fit for our Media Services team

- Easily scale horizontally
- Worked around tools not easy to parallelise
- Isolating data from different customers
- Manage resource control
Orchestration is the next arena
What’s Orchestration?

Services
YOUR APPLICATION

Orchestration
FRAMEWORKS

Data Center
PHYSICAL INFRA
Manage a cluster of Linux containers as a single system to accelerate Dev and simplify Ops.
An ocean of user containers

Scheduled and packed dynamically onto nodes
Program against your datacenter like it’s a single pool of resources

Apache Mesos abstracts CPU, memory, storage, and other compute resources away from machines (physical or virtual), enabling fault-tolerant and elastic distributed systems to easily be built and run effectively.

Download Mesos 0.25.0 or learn how to get started
Docker’s Own Orchestration Tools

- Docker machine
- Docker compose
- Docker swarm
- Docker network
Where is the DEMO Lebowski?
**TOOL NR.1**

**Docker machine**

Simple command line tool to provision local and remote hosts with Docker installed. Fantastic to get up and running fast. It has drivers for many Internet service providers and PaaS.

```bash
$ docker-machine create -d virtualbox dev
```

```
INFO[0000] Downloading boot2docker.iso from...
INFO[0001] Creating SSH key...
INFO[0001] Creating VirtualBox VM...
INFO[0006] Starting VirtualBox VM...
INFO[0041] "dev" has been created and is now active
```
Docker machine DEMO

- Provision a machine with Docker installed and ready
- Pull a minimal image
- Run a few docker commands
- Tear down the machine

```
$ docker-machine create -d virtualbox dev
INFO[0000] Downloading boot2docker.iso from...
INFO[0001] Creating SSH key...
INFO[0001] Creating VirtualBox VM...
INFO[0006] Starting VirtualBox VM...
INFO[0007] Waiting for VM to start...
INFO[0041] "dev" has been created and is now active
```
Recap of what you saw

- “docker-machine create” to provision the host, locally or remotely
- “docker-machine ls” to list the machines
- “docker-machine stop/rm” to stop and remove

```
$ docker-machine create -d virtualbox dev
INFO[0000] Downloading boot2docker.iso from...
INFO[0001] Creating SSH key...
INFO[0001] Creating VirtualBox VM...
INFO[0006] Starting VirtualBox VM...
INFO[0007] Waiting for VM to start...
INFO[0041] "dev" has been created and is now active
```
Describe the relation of your components in a simple YAML file called `docker-compose.yml` and docker-compose takes care of starting them and linking them in order.
Docker compose DEMO

- Provision a machine on a PaaS
- Pull PostgreSQL and a Java app from the Registry
- Use Compose to start the app
- Tear down the machine

```bash
$ docker-compose up -d
```
Recap of what you saw

- “docker-machine create” to provision the host
- Edit “docker-compose.yml” to describe our app
- “docker-compose up -d” to start our application
- “docker-machine rm compose-demo” to remove it

$ docker-compose up -d
Deploy images and run containers on a full clusters as if you’re handling a single machine.
Docker swarm

High level architecture

swarm master

scheduler

discovery service

swarm node

container

container

container

container

container

container
Swarm comes with strategies and filters

- Strategies
  - Spread
  - Binpack
  - Random

- Filters
  - Constraint
  - Affinity
  - Port
  - Dependency
  - Health

$ docker run -e \n  constraint:instance==database --name db
HELPER TOOL
Discovery Service

Consul from HashiCorp

For our Swarm to know which nodes are added to the infrastructure and store information about them we need to use a key-value discovery service, like Consul.
Docker network

New Docker command to manage advanced and transparent networking, like creating VXLAN-based overlay networks that span across data centers.

$ docker network create --driver overlay mynet
The plan for the Swarm DEMO

• Provision a Docker swarm
• Made up of three hosts
  • Master node
  • Node with 2gb of RAM
  • simple Node
• Use labels to deploy to nodes
• Run Java app and PostgreSQL on different nodes

$ docker pull swarm
$ docker run --rm swarm create 6856663cdefdec325839a4b7e1de38e8
After Part 1

Current Architecture

- **demo-master**: Consul: discovery service
- **scheduler**
- **node1**: label: java, RAM: 2GB
- **node2**: label: database, RAM: 512MB
bitbucket:
  image: atlassian/bitbucket-server
  ports:
    - "7990:7990"
    - "7999:7999"
  volumes_from:
    - license
  user: root
  privileged: true
  environment:
    - "constraint:instance==java"

db:
  image: postgres
  ports:
    - "5432:5432"
  environment:
    - "POSTGRES_PASSWORD=somepassword"
    - "constraint:instance==db"
  license:
  build: .
What we did

Final Architecture

demo-master

scheduler

Java App
node1
- label: java
- RAM: 2GB

Consul: discovery service

PostgreSQL
node2
- label: database
- RAM: 512MB
Recap of Swarm DEMO

• We created a 3-node cluster with “docker-machine”
• We tagged the nodes with labels
• We started our components using label constraints and not IP addresses
I hope you are hyped as I am for all this coolness, come talk to me afterwards!
Please Remember to rate session

Thank you!