

CONTAINERS

G. Molines
2017-2018



SOFTWARE CONTAINERS

App servers

- A place to run your applications
- What does it provide?
 - Orthogonal services:
 - Transaction support, resource management, isolation, clustering, ...
 - Support services
 - Messaging, binding, naming, ...

App servers

- Constraints
 - Package format
 - Rules of the game:
 - No thread, no file access
 - Resource lookups
 - No static

Running your app

- Deployment
- Test in the context of the container
 - Test frameworks: eg: Arquillian
- Setting-up the container:
 - Part of the test setup?
 - Or assumed as done beforehand?

Types of containers

- Servlet containers
 - Hosting servlets, JSPs
 - Example: Tomcat
- EJB containers
 - Hosting EJB, J2EE standard
- → Application servers
- TomEE vs Tomcat

Hosted services

- Eg: Amazon AWS, Google Web Toolkit
- Dev of java / javascript
- Execution in the cloud
- Limited to some APIs

VIRTUAL MACHINES

Virtual machine

- Emulate a machine on another
- Host and Guest
- Provides enough function to run a guest OS
- Guest OS can be different from Host'
- Resources binding
- Network bridge

VM Hypervisor

- Manage VM resources
- Start, Stop, Pause VMs
- File operations
 - Clone VM, export to file, import, etc.
- Snapshot managements
 - Great for restoring known good start state for testing

Virtual machine

- But....
- Performance
- Large disk size
- Heavyweight
 - Run your app in an OS which runs in a VM which runs in another OS. Do you really need the OS twice?

LIGHT CONTAINERS

Light container

- Like a VM, but without the extra OS
- Shares CPU, network, disk with host
 - Leverage host features for proper isolation
 - Control groups (cpu, memory)
 - Namespaces (resources, network)
 - Union file system (disk)
 - Can't switch to another OS: linux on linux only
- Lightweight

Images vs containers

- An image is
 - a reference template
 - Static
 - Inert: file on disk
- A container is
 - a running instance
 - Dynamic
 - can be modified
 - Where you deploy your app

Building an image

- Manually:
 - `docker run -t -i someImage /bin/bash`
 - `-t` → terminal
 - `-i` → interactive
 - You get a shell prompt
 - Perform your actions: file operations, package installation, configuration, etc.
 - When done, Ctrl+C, then:
 - `docker commit <containerID>`
`imageName:vX`

Building an image

- Dockerfiles:
 - Series of instructions to build the image
 - ADD, RUN, CMD, COPY, etc.
 - Start FROM a base image
- Docker uses smart image cache for common parts
 - Image composition
 - Tree of images

Docker registry

- Local storage of all your available images
 - `docker images` gets you the list
 - Contains images and their tags
- If you host images, you need to add them to the registry of the host
 - Example on Bluemix:

```
MacBook-Pro-4:dockerData gmlines$ cf ic images
```

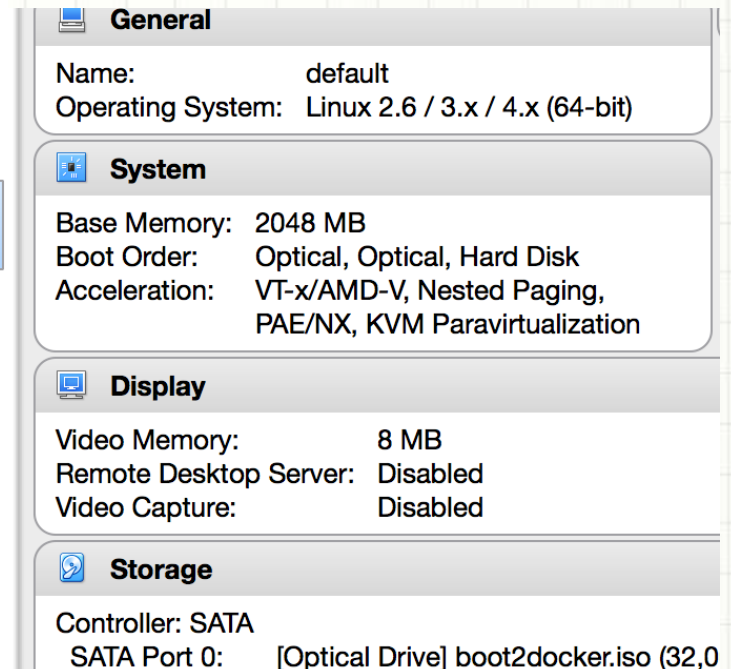
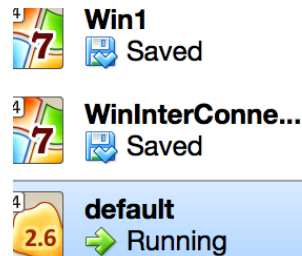
REPOSITORY	TAG	IMAGE ID
registry.eu-gb.bluemix.net/gmpolytech/trainingwebapp	v1	02a8815912ca
registry.eu-gb.bluemix.net/ibm-mobilefirst-starter	latest	a100524c96cb
registry.eu-gb.bluemix.net/ibm-node-strong-pm	latest	3e2373877cf5
registry.eu-gb.bluemix.net/ibm/liberty	latest	33fdda9431c7
registry.eu-gb.bluemix.net/ibm/node	latest	a4964fd52b4f
registry.eu-gb.bluemix.net/ibm/node	v4	a4964fd52b4f
registry.eu-gb.bluemix.net/ibm/node	v1.1	e4812bb29c8e

Docker on Win, Mac

- Docker on linux: host == native system
- ~~Docker in Win, MacOS:~~
 - ~~Host == dedicated VM~~

```
MacBook-Pro-4:teamserver gmolines$ docker-machine inspect
```

```
{  
  "ConfigVersion": 3,  
  "Driver": {  
    "IPAddress": "192.168.99.100",  
    "MachineName": "default",  
    "SSHUser": "docker",  
    "SSHPort": 60066,  
    "SSHKeyPath": "/Users/gmolines/.do  
    "StorePath": "/Users/gmolines/.doc  
    "SwarmMaster": false,  
    "SwarmHost": "tcp://0.0.0.0:3376",  
    "SwarmDiscovery": "",  
    "VBoxManager": {},  
    "CPU": 1,  
    "Memory": 2048,  
    "DiskSize": 204800.
```



Docker ports

- Within the container, you can start services
 - They'll listen onto a port
 - Eg: tomcat on 80
 - By default, these ports are not visible from outside the container
- You have to map them (`docker run ... -P`)
- ~~Win and MacOS: access the port on your docker machine (= VM) !~~
- Eg: <http://0.0.0.0:32012> → a port inside your container



QUESTIONS?

Next

- TD Docker 7/4
 - Play with docker and images
 - Create Docker files for client, server, payment
 - Run your app on docker (separate containers
 - Tests still work 😊!
 - Build images from Jenkins
 - Docker compose
 - Tests still work again 😊!



APPENDIX