Containers

G. Molines

2020-2021

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: Arquilian
- Setting-up the container:
  - Part of the test setup?
  - Or assumed as done beforehands?

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

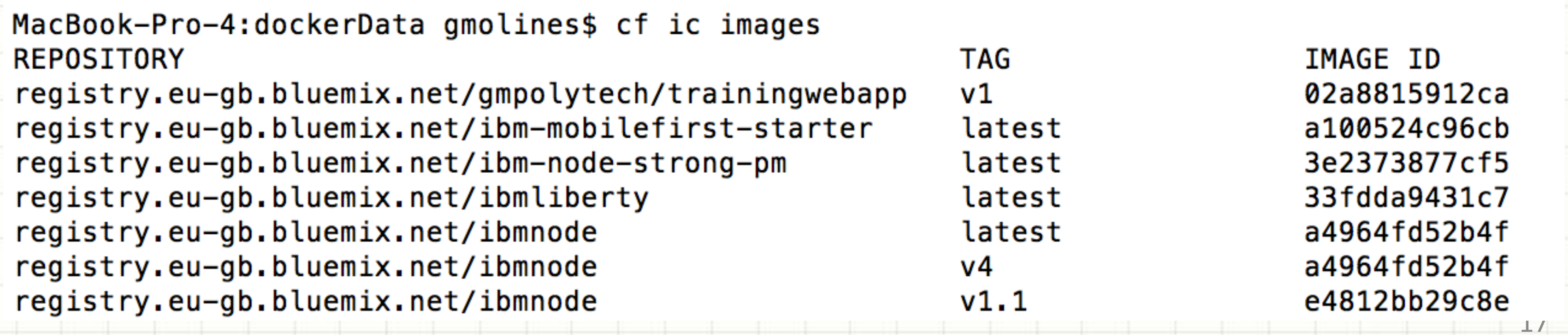
- Manually:
  - `docker run -t -i someImage /bin/bash`
    - -t → terminal
    - -I → interactive
  - You get a shell prompt
  - Perform your actions: file operations, package installation, configuration, etc.
  - When done, Ctrl+C, then:
  - `docker commit <containerID> imageName:vX`

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

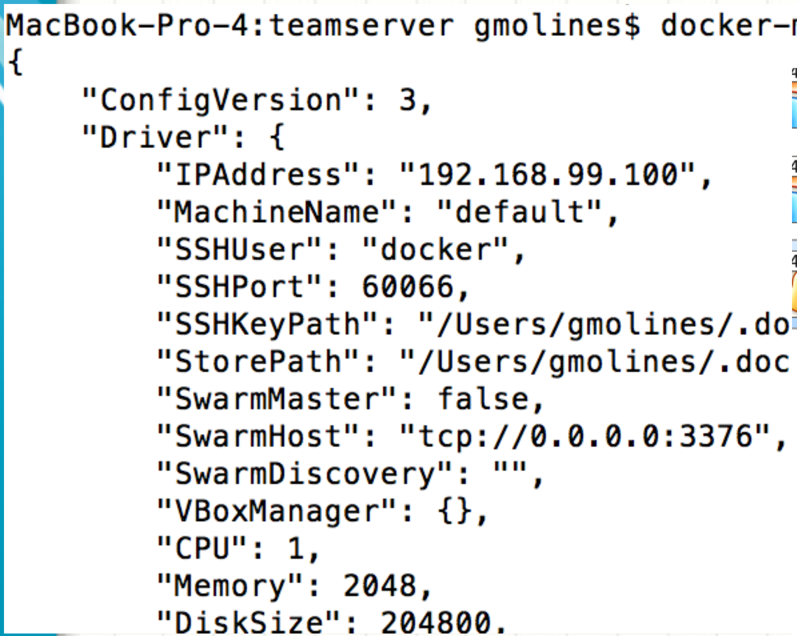
- Local storage of all your available images
  - `docker images` gets you the list
  - Contains images and their tags
- If you host images, you need to add them to the registry of the host
  - Example on Bluemix:



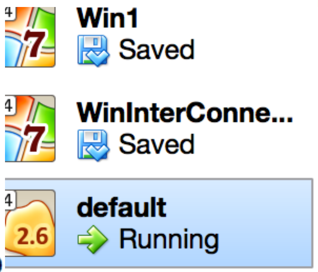
```
MacBook-Pro-4:dockerData gmolines$ cf ic images
REPOSITORY                                          TAG         IMAGE ID
registry.eu-gb.bluemix.net/gmpolytech/trainingwebapp v1          02a8815912ca
registry.eu-gb.bluemix.net/ibm-mobilefirst-starter  latest      a100524c96cb
registry.eu-gb.bluemix.net/ibm-node-strong-pm       latest      3e2373877cf5
registry.eu-gb.bluemix.net/ibmliberty               latest      33fdda9431c7
registry.eu-gb.bluemix.net/ibmnode                  latest      a4964fd52b4f
registry.eu-gb.bluemix.net/ibmnode                  v4          a4964fd52b4f
registry.eu-gb.bluemix.net/ibmnode                  v1.1        e4812bb29c8e
```

Docker on Win, Mac

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



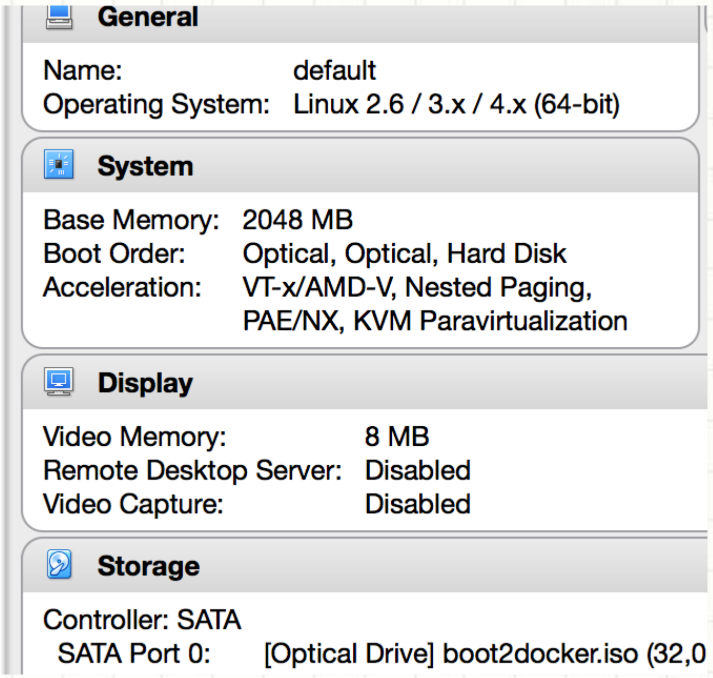
```
MacBook-Pro-4:teamserver gmolines$ docker-machine inspect
{
    "ConfigVersion": 3,
    "Driver": {
        "IPAddress": "192.168.99.100",
        "MachineName": "default",
        "SSHUser": "docker",
        "SSHPort": 60066,
        "SSHKeyPath": "/Users/gmolines/.do
        "StorePath": "/Users/gmolines/.doc
        "SwarmMaster": false,
        "SwarmHost": "tcp://0.0.0.0:3376",
        "SwarmDiscovery": "",
        "VBoxManager": {},
        "CPU": 1,
        "Memory": 2048,
        "DiskSize": 204800.
```



Win1
Saved

WinInterConne...
Saved

default
Running



General

Name:	default
Operating System:	Linux 2.6 / 3.x / 4.x (64-bit)

System

Base Memory:	2048 MB
Boot Order:	Optical, Optical, Hard Disk
Acceleration:	VT-x/AMD-V, Nested Paging, PAE/NX, KVM Paravirtualization

Display

Video Memory:	8 MB
Remote Desktop Server:	Disabled
Video Capture:	Disabled

Storage

Controller: SATA
SATA Port 0: [Optical Drive] boot2docker.iso (32,0

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

Docker Volumes

- A container is volatile. Once stopped, content is lost.
- You can map an internal path to an external volume

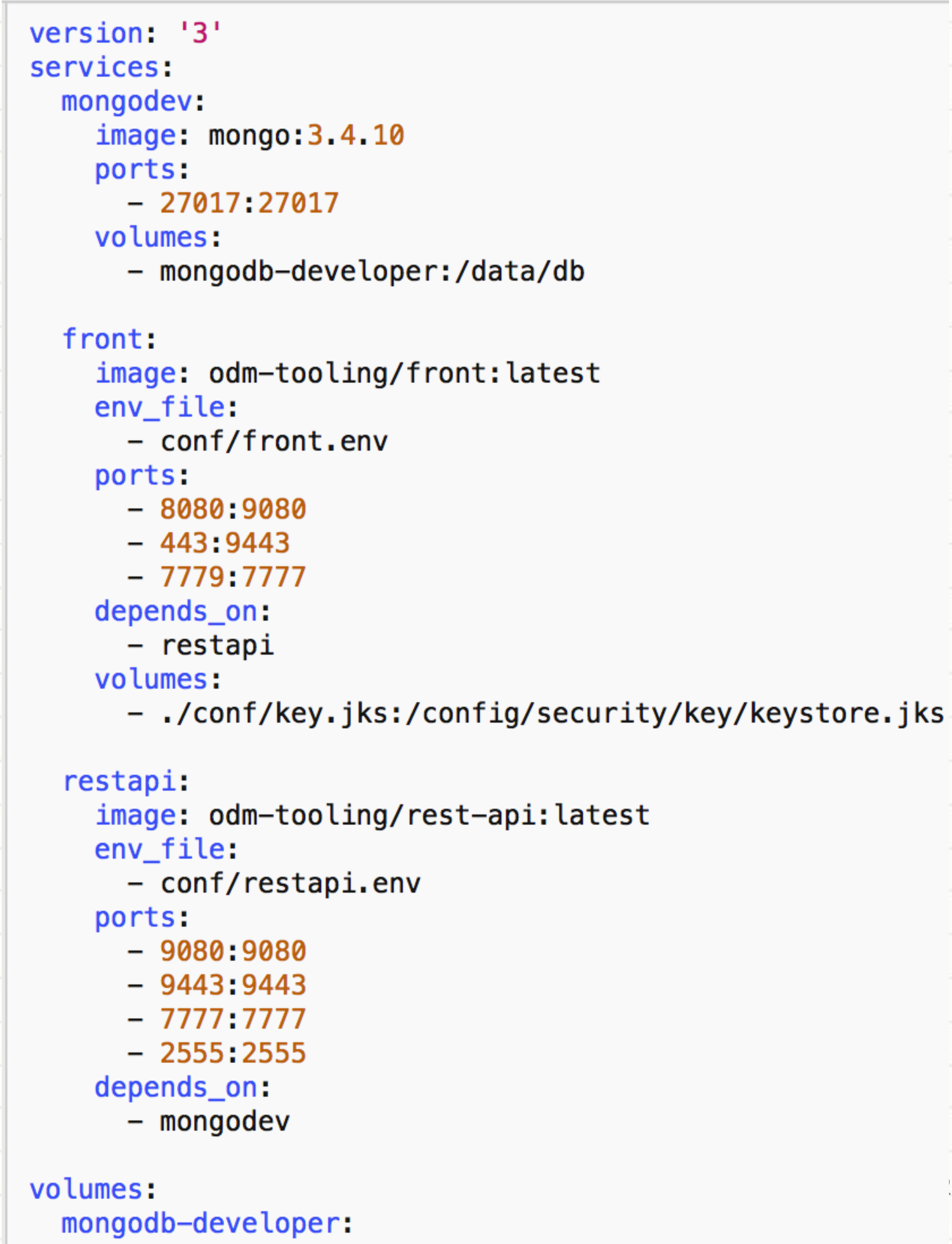
```
docker run -v
/path/on/host:/path/in/container
```

- Allows to persist state elsewhere, and thus to find it again upon next execution

Docker Compose

- Starts / Stops several images at the same time
- Virtual network between those images
- Shared resources

Docker Compose Example



```yaml
version: '3'
services:
  mongodev:
    image: mongo:3.4.10
    ports:
      - 27017:27017
    volumes:
      - mongodb-developer:/data/db

  front:
    image: odm-tooling/front:latest
    env_file:
      - conf/front.env
    ports:
      - 8080:9080
      - 443:9443
      - 7779:7777
    depends_on:
      - restapi
    volumes:
      - ./conf/key.jks:/config/security/key/keystore.jks

  restapi:
    image: odm-tooling/rest-api:latest
    env_file:
      - conf/restapi.env
    ports:
      - 9080:9080
      - 9443:9443
      - 7777:7777
      - 2555:2555
    depends_on:
      - mongodev

volumes:
  mongodb-developer:
```

QUESTIONS?

Next

- TD Docker 9/4
    - 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