RUG 2015/2016

Software Patterns

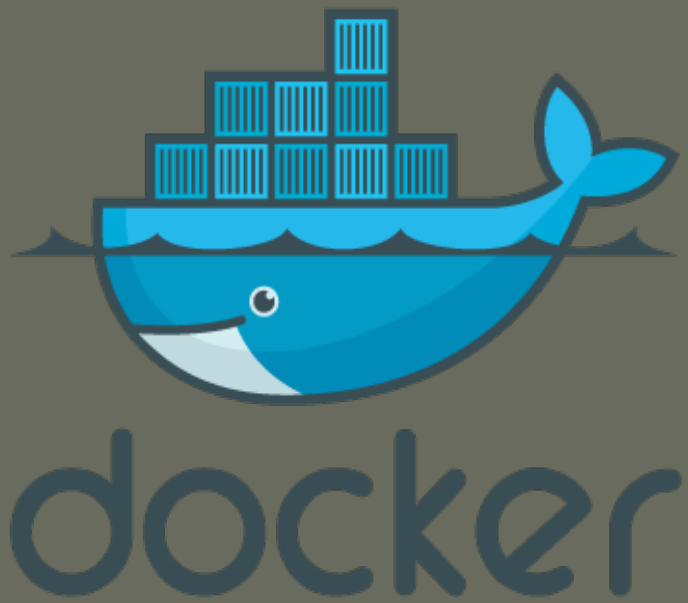
Second Presentation : Pattern-Based Recovery

Docker

Joris Schaeffers
Wouter Menninga
Guntur Dharma Putra
Aurélie Fakambi

Introduction

- « **Docker** is an open-source project that automates the deployment of applications inside software containers, by providing an additional layer of abstraction and automation of operating-system-level virtualization on Linux » Wikipedia

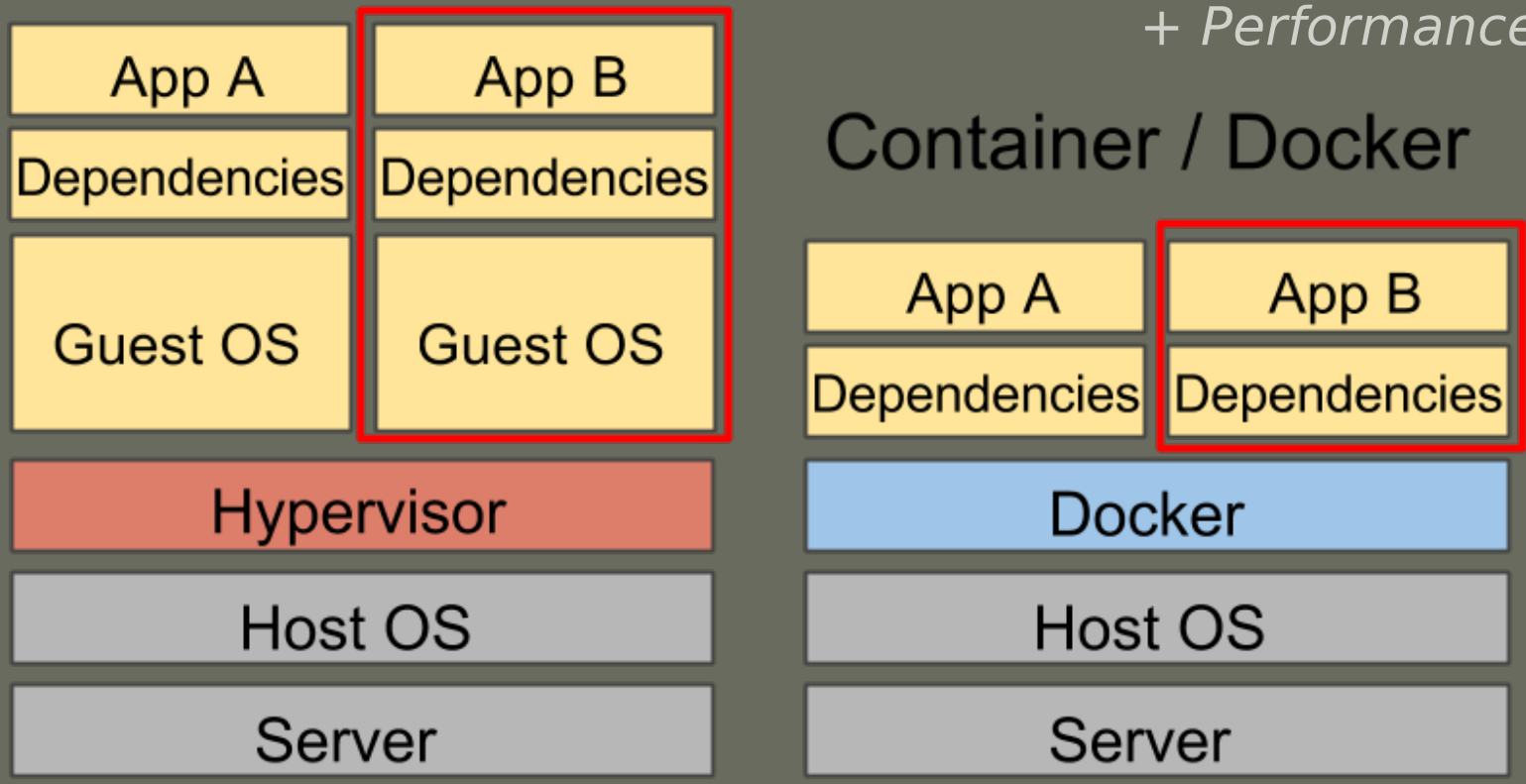


- Lightweight, fast and secure

Hardware Emulation virtualization

Operating system level of virtualization

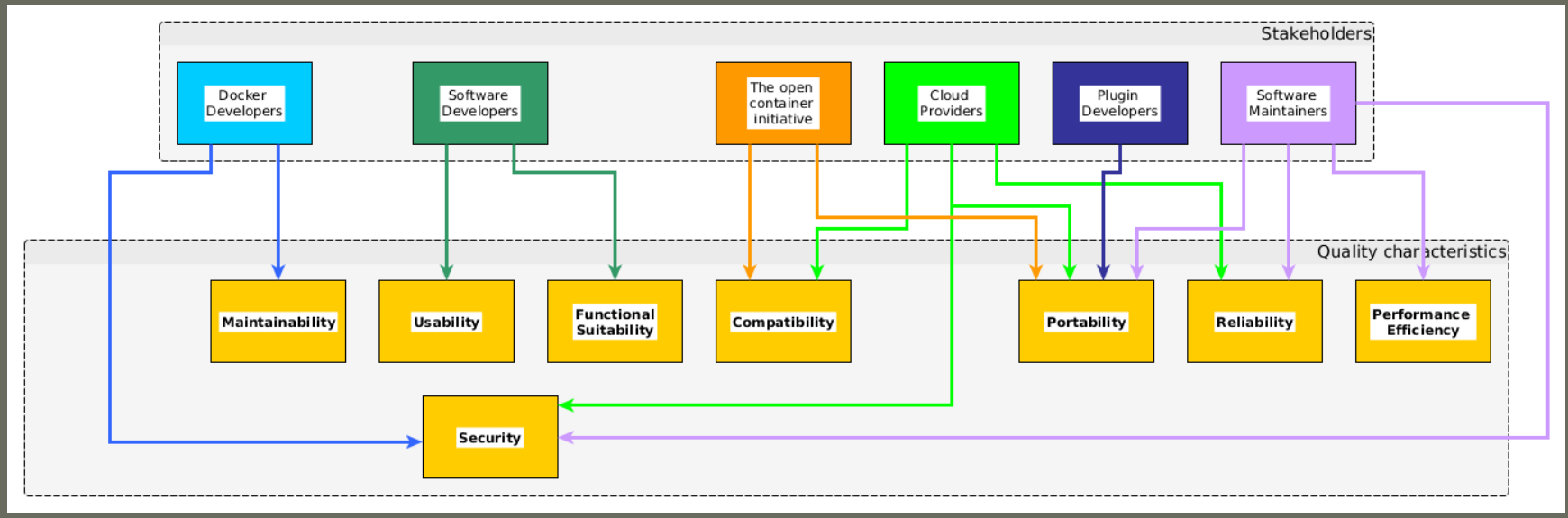
+ Performance



Stakeholders

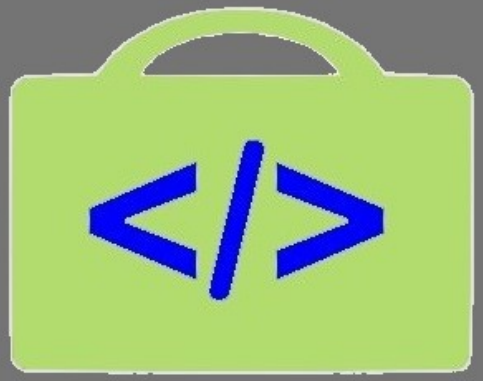
- Software developers
- Software Maintainers
- Cloud providers
- The open container initiative
- Docker developers
- Plugin developers

…& their concerns



Key Drivers

Portability



Platform and hardware agnostic

Reliability

Test

Security

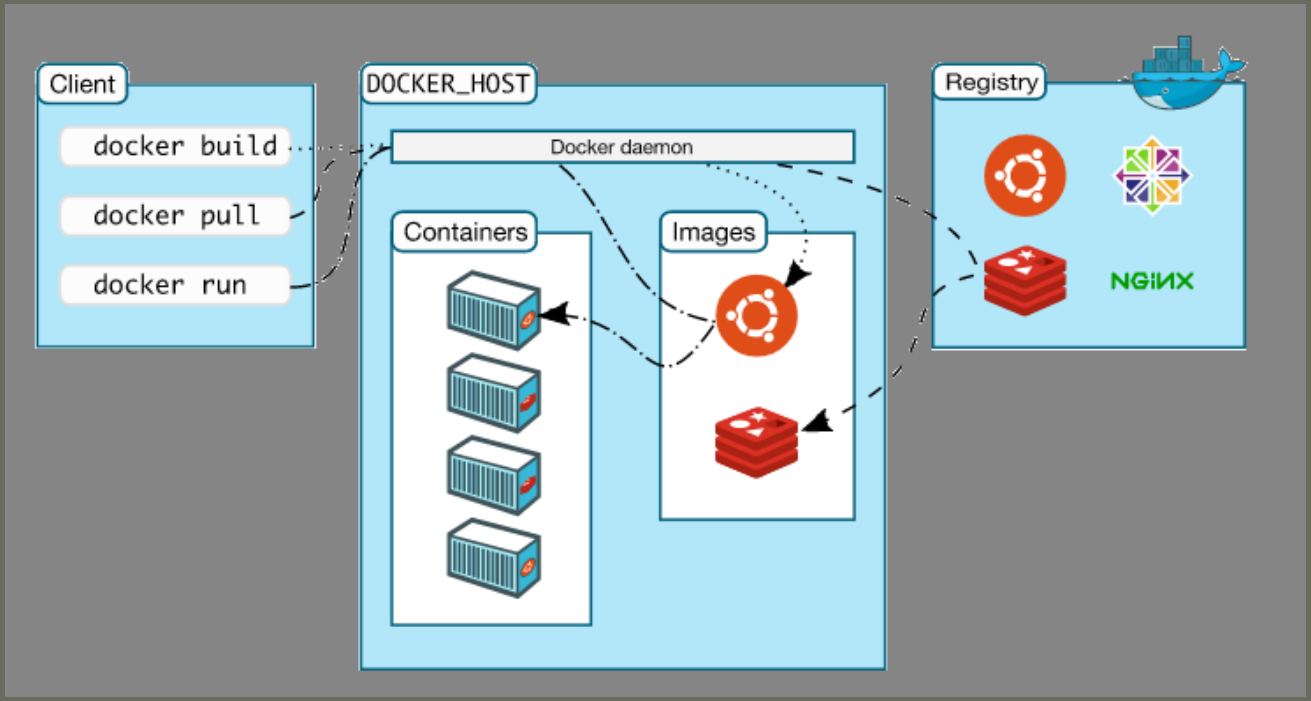
API and Containers isolation

Docker Architecture

Three main inside components :

- *Images to build*
- *Containers to run*
- *Registries to distribute*

Logical view



Patterns

- Client-Server
- Layers
- Shared repository
- Publish-Subscribe
- Brokered authentication
- Plugin
- Proxy

Layers

- + Security
- + Reliability
- + Portability

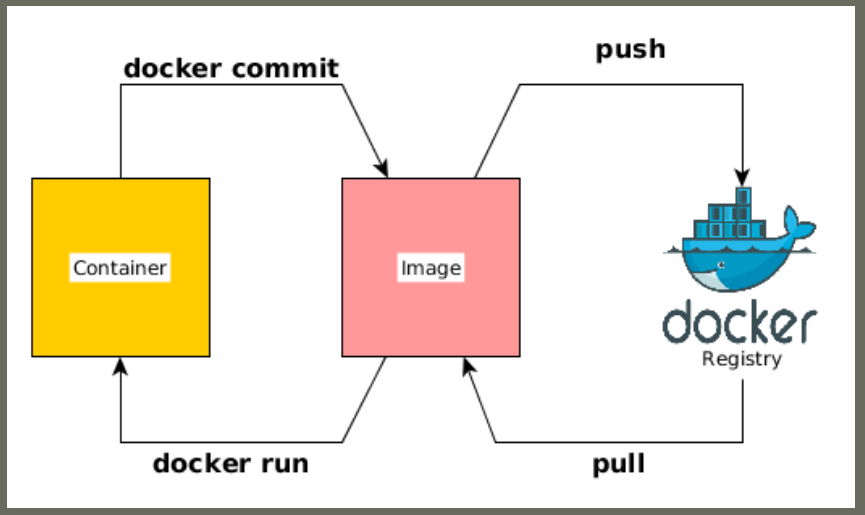
Client-Server

- + Portability
- + Reliability

Shared/Active Repository

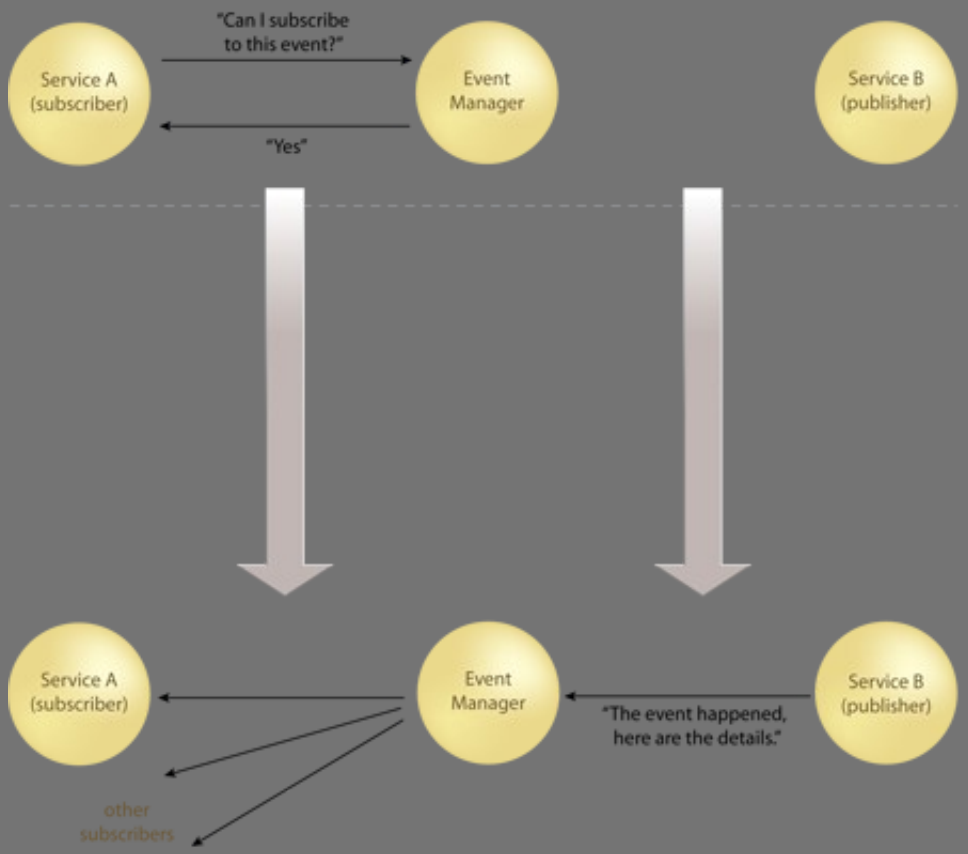
- *Through the Docker registry : store and distribution of images*

- + Security



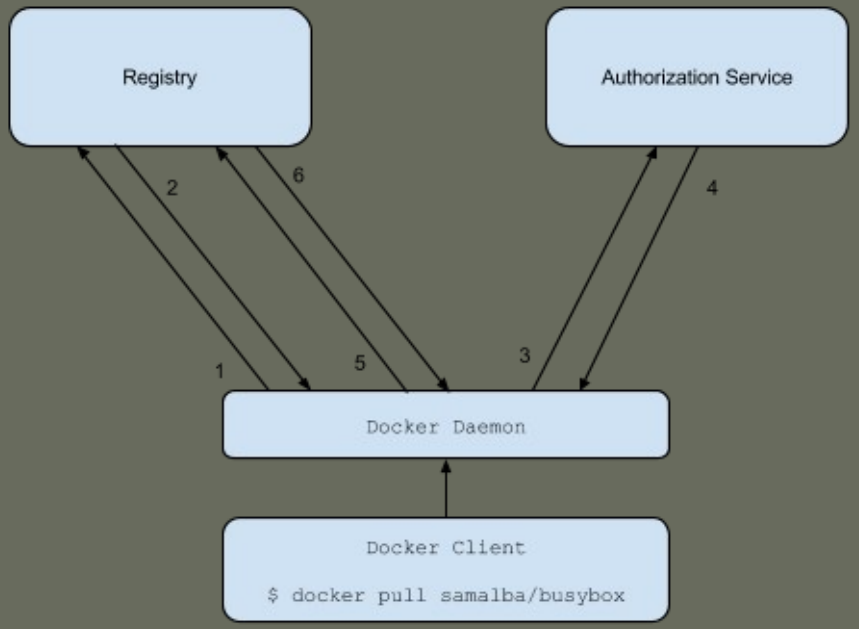
Publish Subscribe

- The notification mechanism in the Active Repository
- Users subscribe for webhook notifications about events occuring in the registry



Brokered Authentication

- + Security



Broker

- + Portability
- + Security
- - Performance

Plugin

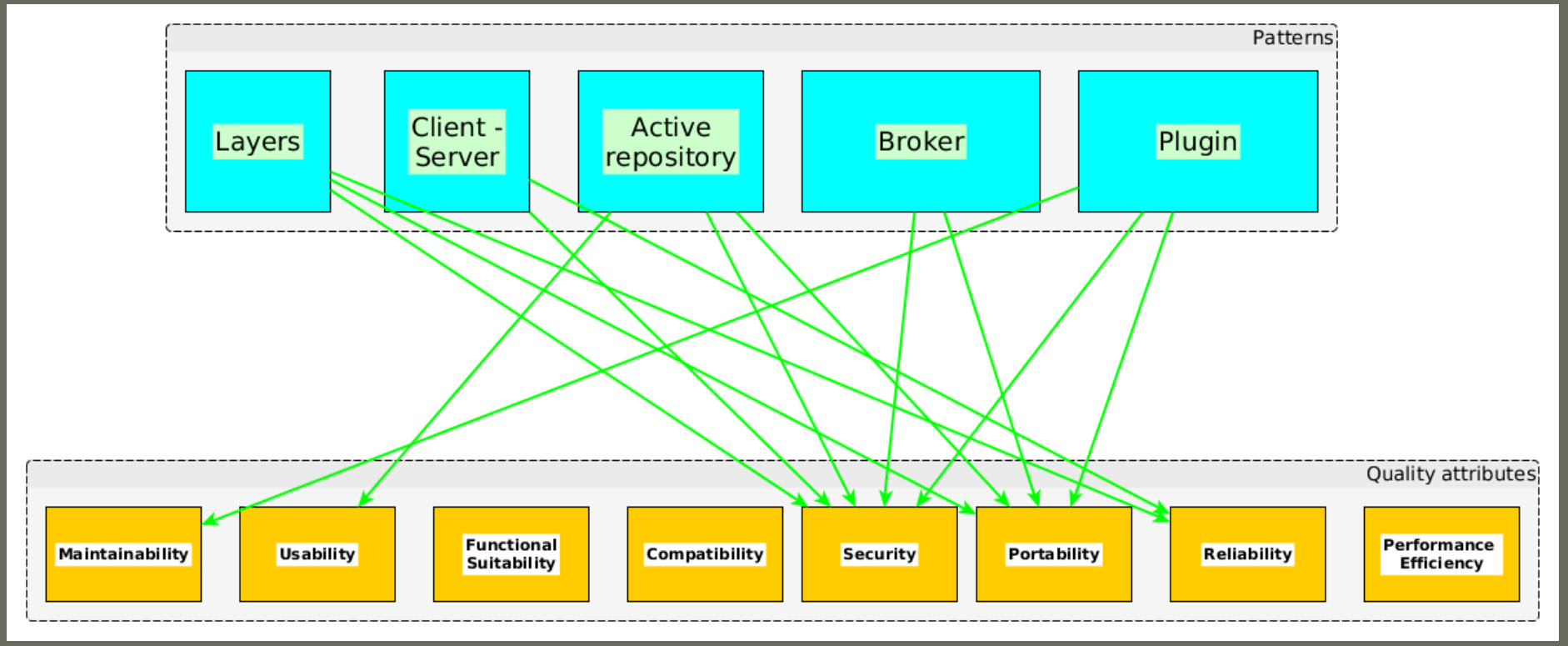
- + Portability
- + Maintainability
- + Security

Evaluation

- *Are the key drivers validated by the patterns ?*

- Goal of the second part : matching KD and Patterns

Evaluation graph



- *Thanks for you attention*

Sources