The rise of containers

Jorge Dias

Introduction

salut Bucuresti

Devops Enginner @ XING

mrdias.com

@dias_jorge

Containers

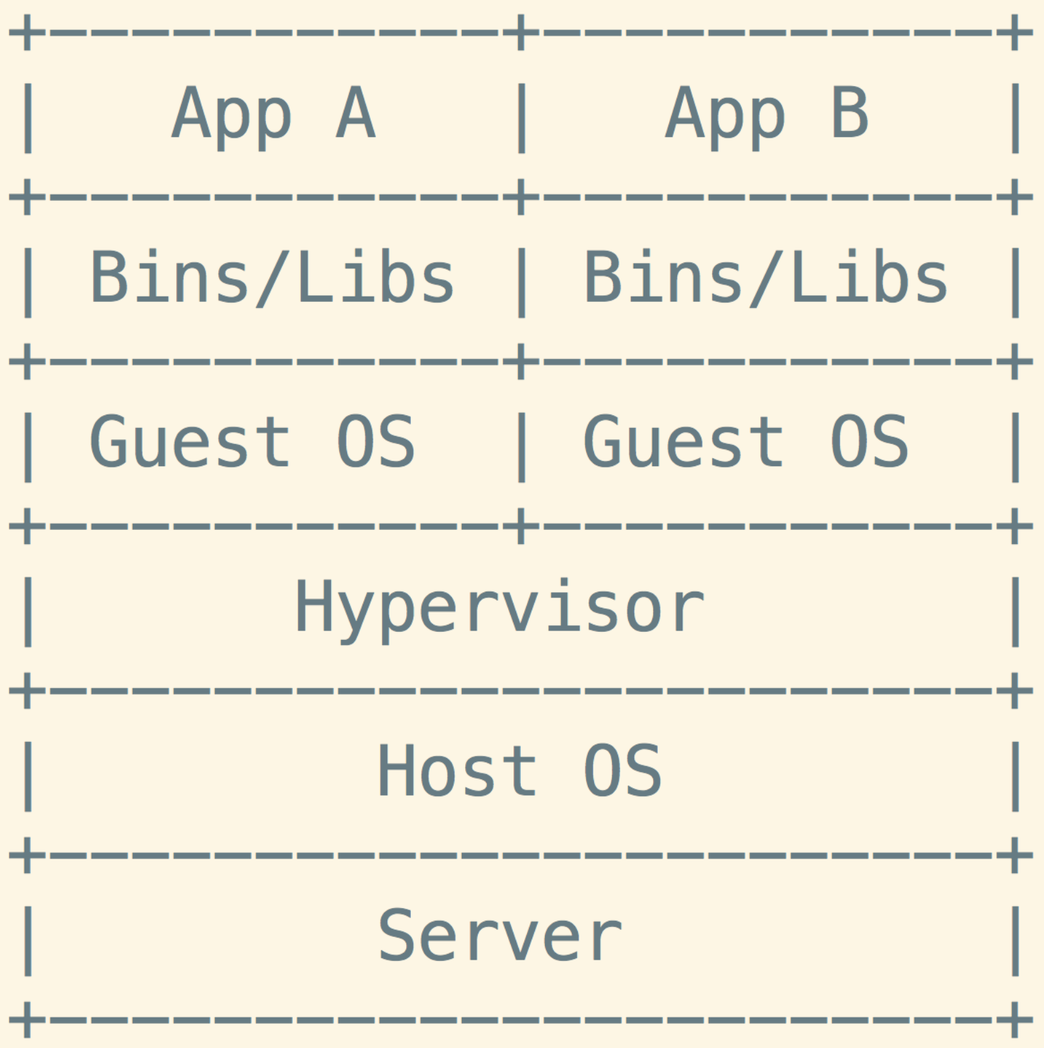
Virtualization

Hardware level virtualization

Started at the 60s

Way of running old code on new hardware

Method to divide system resources


App A
App B
Bins/Libs
Bins/Libs
Guest OS
Guest OS
Hypervisor
Host OS
Server

Advantages

Can run legacy stack unmodified

Can mix and match operative systems

Disadvantages

Big performance impact

Virtual Machine includes application, dependencies and Guest OS

Containers

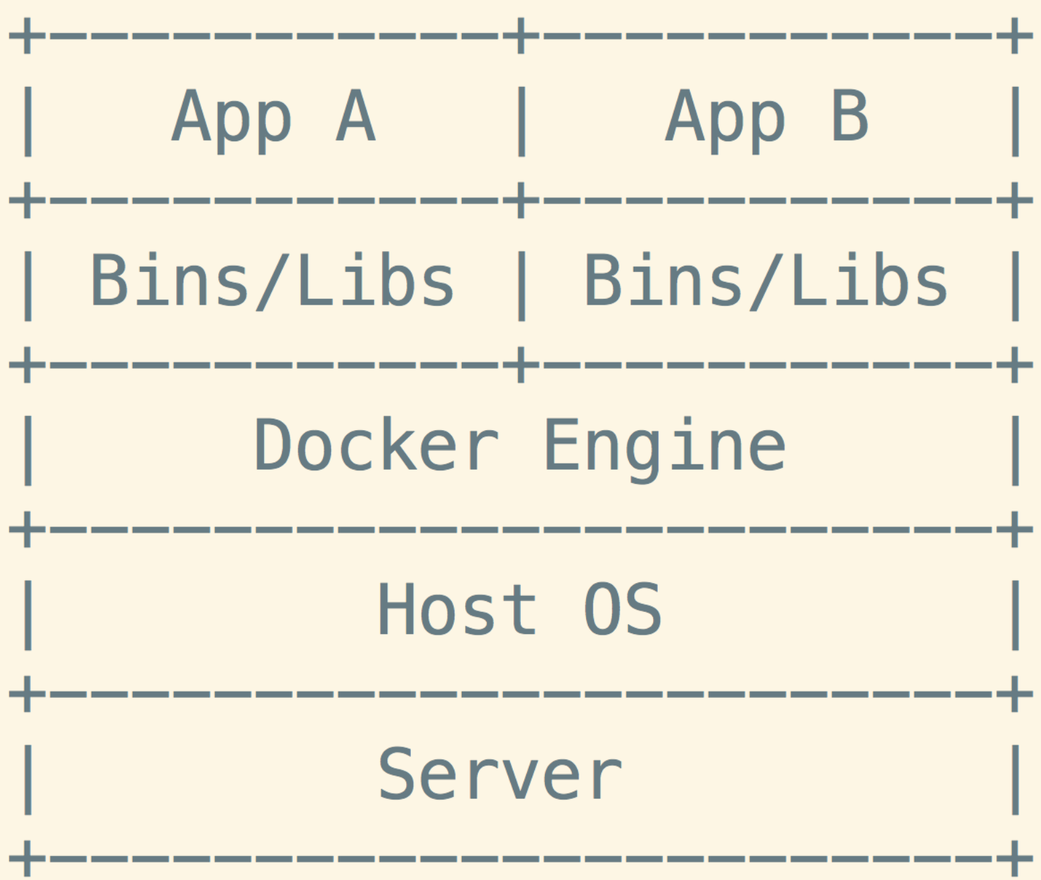
OS level virtualization

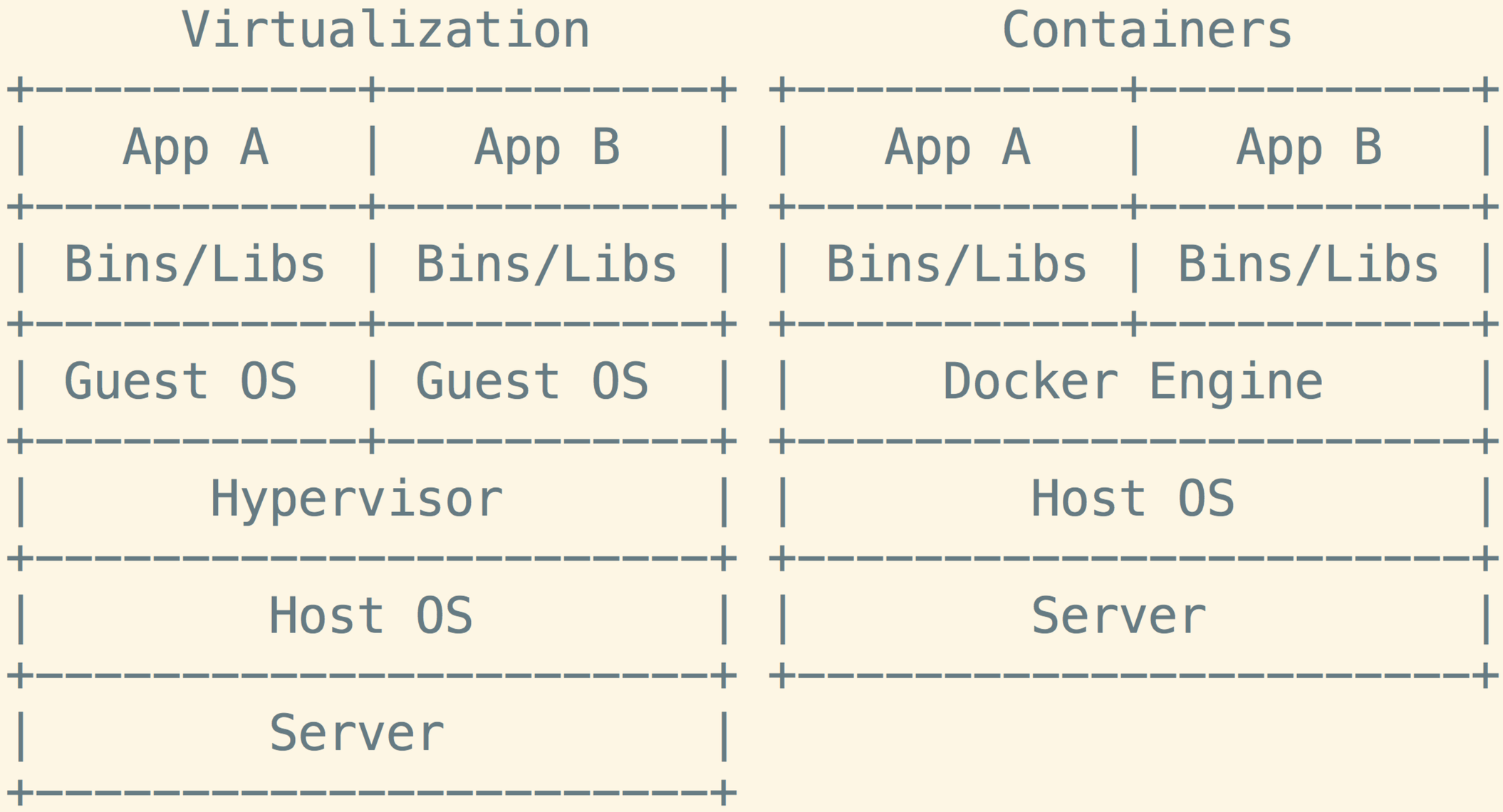
Free BSD Jails – chroot

Solaris zones

Open-VZ

LXC


App A
App B
Bins/Libs
Bins/Libs
Docker Engine
Host OS
Server


Virtualization
Containers
App A
App B
Bins/Libs
Guest OS
Hypervisor
Host OS
Server
App A
App B
Bins/Libs
Docker Engine
Host OS
Server

Advantages

Full Speed

Better sharing of resources like RAM, CPU

Container includes only application and dependencies

Disadvantages

Only can run one OS

(But you can mix Linux distributions)

Docker introduction

An open platform for distributed applications
for developers and sysadmins.

Docker enables apps to be quickly assembled from components

Run the same app, **UNCHANGED,**
on laptops, data center VMs, and any cloud.

Eliminates the friction between development,
QA, and production environments.

Docker for developers

Why is it so popular?

Good user experience

Build any app in any language
using any toolchain

Forget installing and compiling
libraries locally

Easy to use exact versions
of external services

Dockerized apps are completely portable
and can run anywhere

Easy to compose and integrate different services

The distribution model

Apt for applications

Docker for operations

Standardized environments

Dev, QA, staging, production

Flexibility to distribute

where things run

Better resources utilization

Easily scale up and down

Dockerizing my apps

What should I do?

The Twelve-Factor app

Store configuration in the environment

Treat backing services as attached resources

Execute the app as one or more
stateless processes

Maximize robustness with fast startup and graceful shutdown

Treat logs as event streams

Challenges

Service discovery and registration

Where are things running?

Networking

How do containers talk to each other?

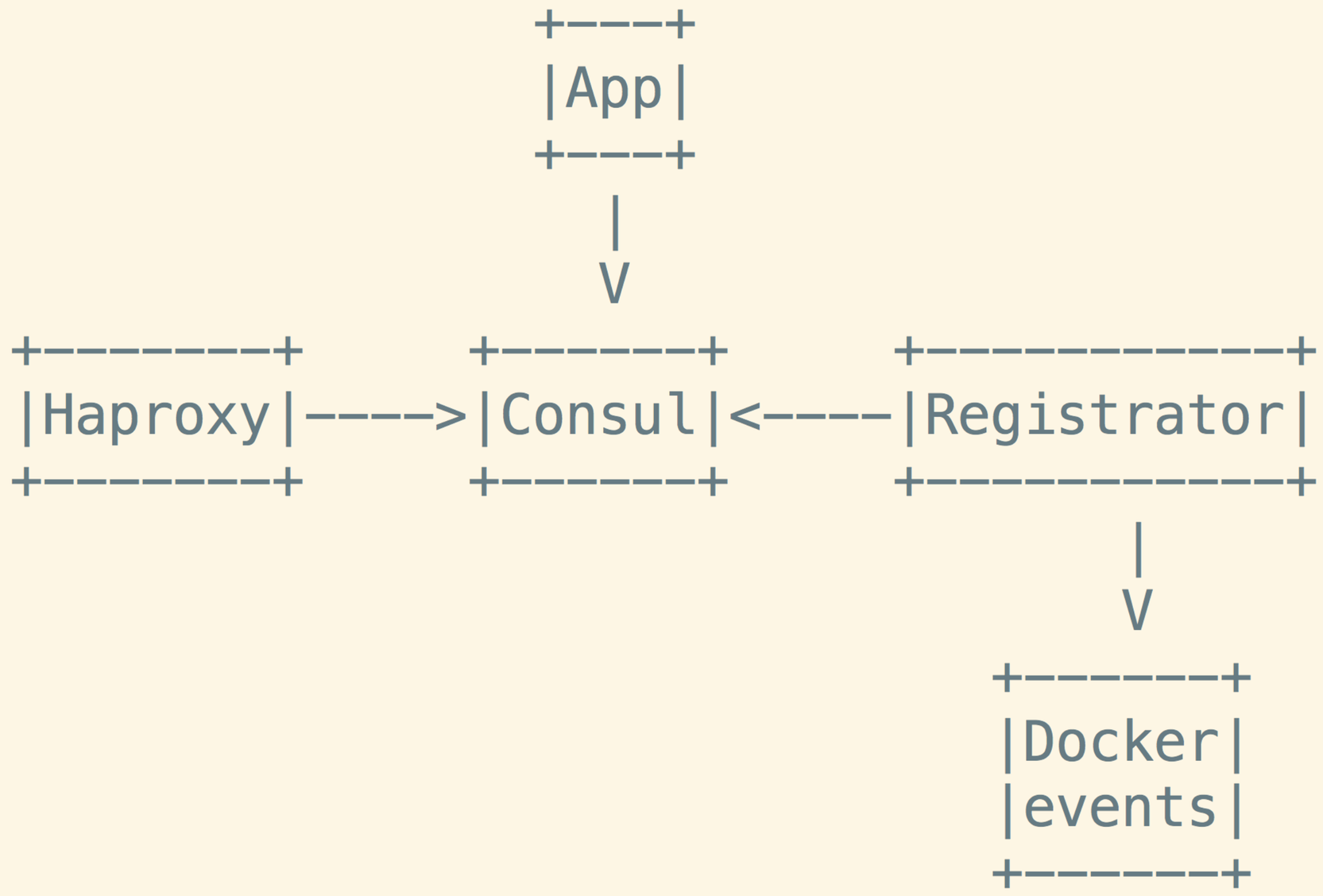
Persistence

How do we store data?

Security

How do we isolate containers from each other?

Service discovery demo


App
Haproxy
Consul
Registrator
Docker
events

Show time!

That's all, thanks!