HYBRID CLOUD-BASED APPLICATIONS

Luciano Baresi
luciano.baresi@polimi.it

LUCIANO BARESI

Professor @ DEIB – Politecnico di Milano

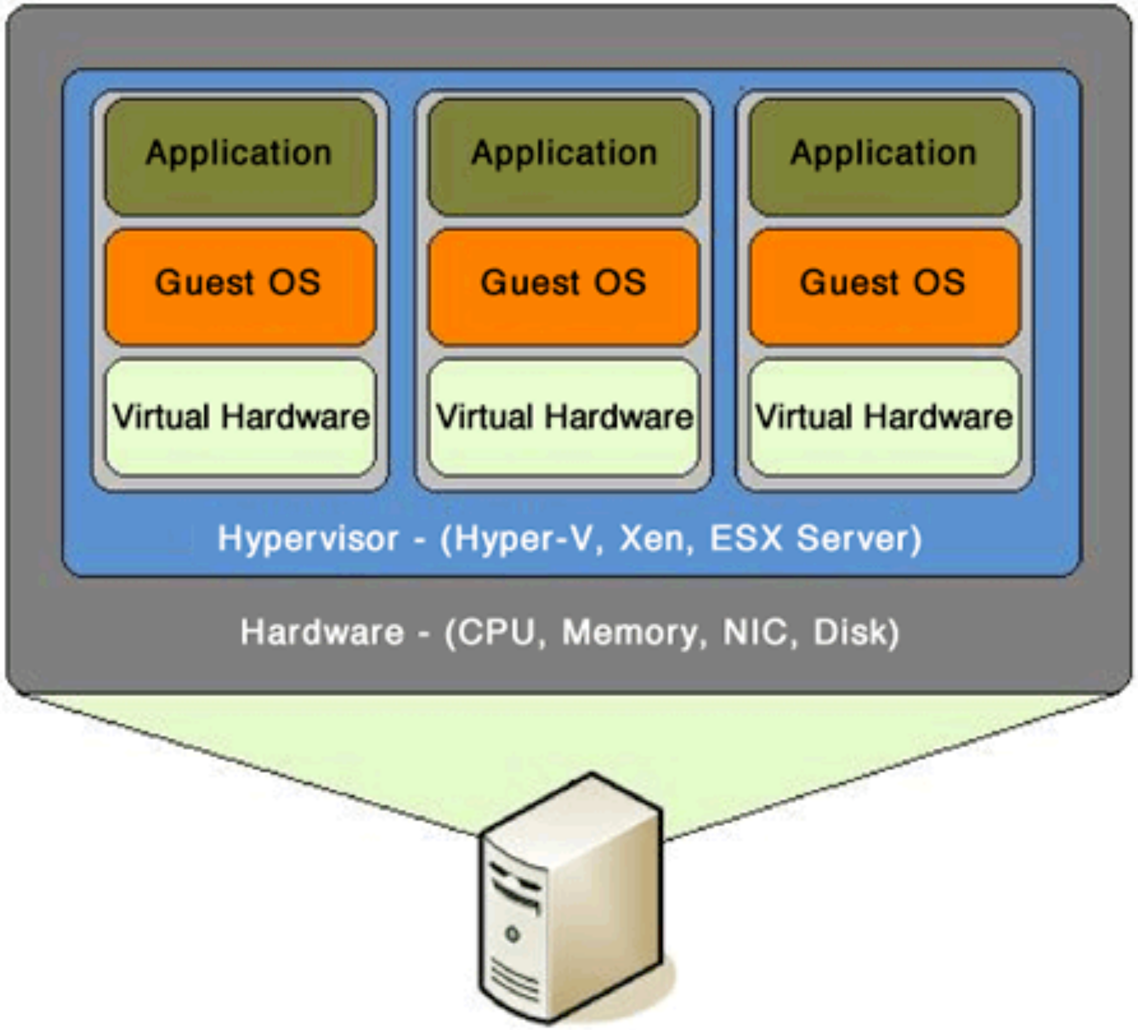
- Researcher at Cefriel
- Visiting researcher
  - University of Oregon (USA)
  - University of Paderborn (Germany)

Research interests

- Software engineering
  - Dynamic software architectures
  - Service-oriented applications
  - Mobile applications
  - Cloud computing

home.deib.polimi.it/baresi

VIRTUAL MACHINES

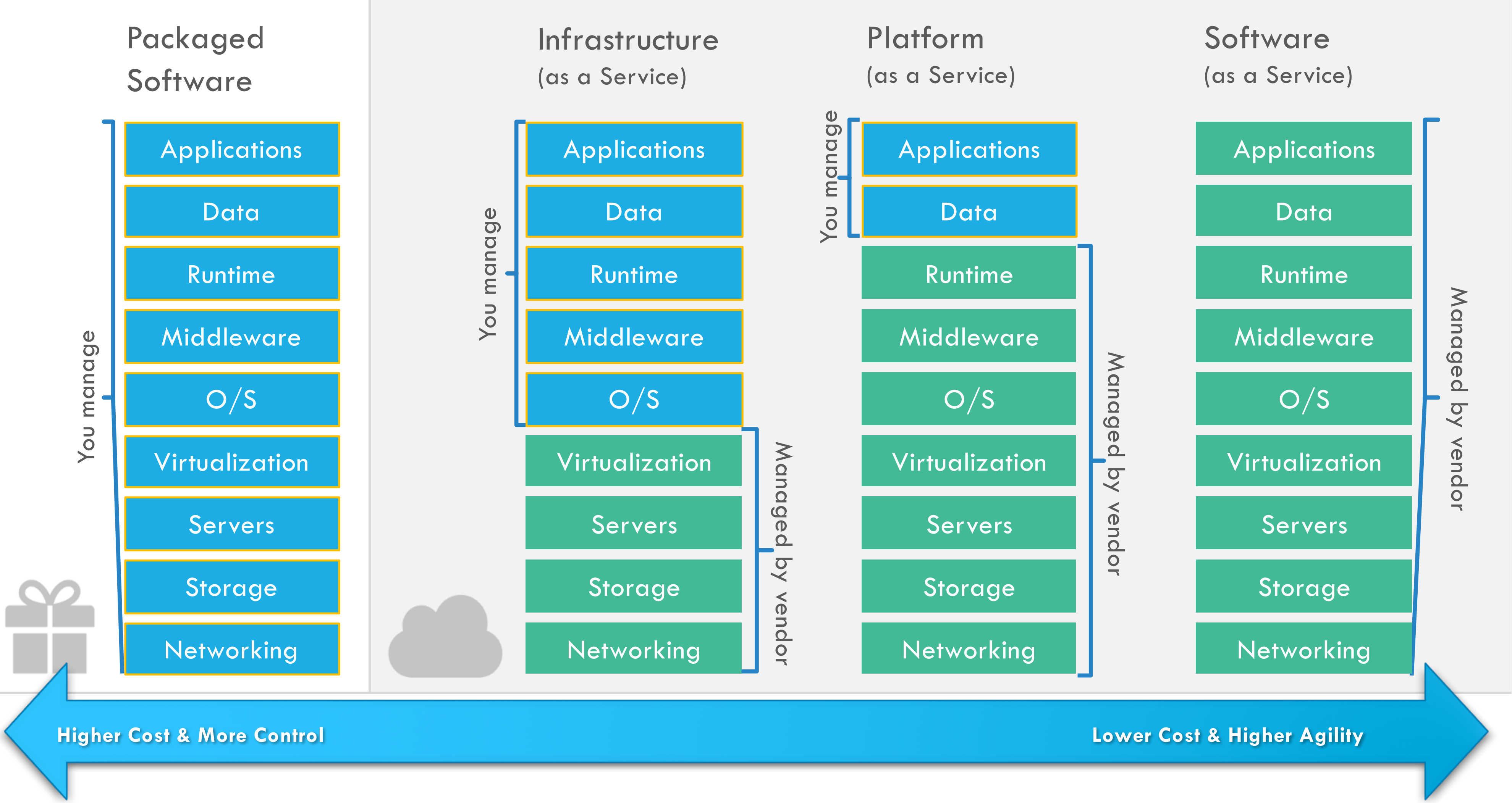


CLOUD COMPUTING

"The interesting thing about cloud computing is that we've redefined cloud computing to include everything that we already do. […] The computer industry is the only industry that is more fashion-driven than women's fashion. Maybe I'm an idiot, but I have no idea what anyone is talking about. What is it? It's complete gibberish. It's insane. When is this idiocy going to stop?

Larry Ellison
during Oracle's Analyst Day

CLOUD COMPUTING TAXONOMY



Packaged Software

You manage

Applications

Data

Runtime

Middleware

O/S

Virtualization

Servers

Storage

Networking

Infrastructure (as a Service)

You manage

Applications

Data

Runtime

Middleware

O/S

Managed by vendor

Virtualization

Servers

Storage

Networking

Platform (as a Service)

You manage

Applications

Data

Managed by vendor

Runtime

Middleware

O/S

Virtualization

Servers

Storage

Networking

Software (as a Service)

Managed by vendor

Applications

Data

Runtime

Middleware

O/S

Virtualization

Servers

Storage

Networking

Higher Cost & More Control

Lower Cost & Higher Agility

Amazon Web Services Are...

A set of APIs and business models which give developers access to Amazon technology and content

- Data As a Service
  - Amazon E-Commerce Service
  - Amazon Historical Pricing
- Infrastructure As a Service
  - Amazon Simple Queue Service
  - Amazon Simple Storage Service
  - Amazon Elastic Compute Cloud
- Search As a Service
  - Alexa Web Information Service
  - Alexa Top Sites
  - Alexa Site Thumbnail
  - Alexa Web Search Platform
- People As a Service
  - Amazon Mechanical Turk

WHAT DOES HYBRID MEAN HERE?

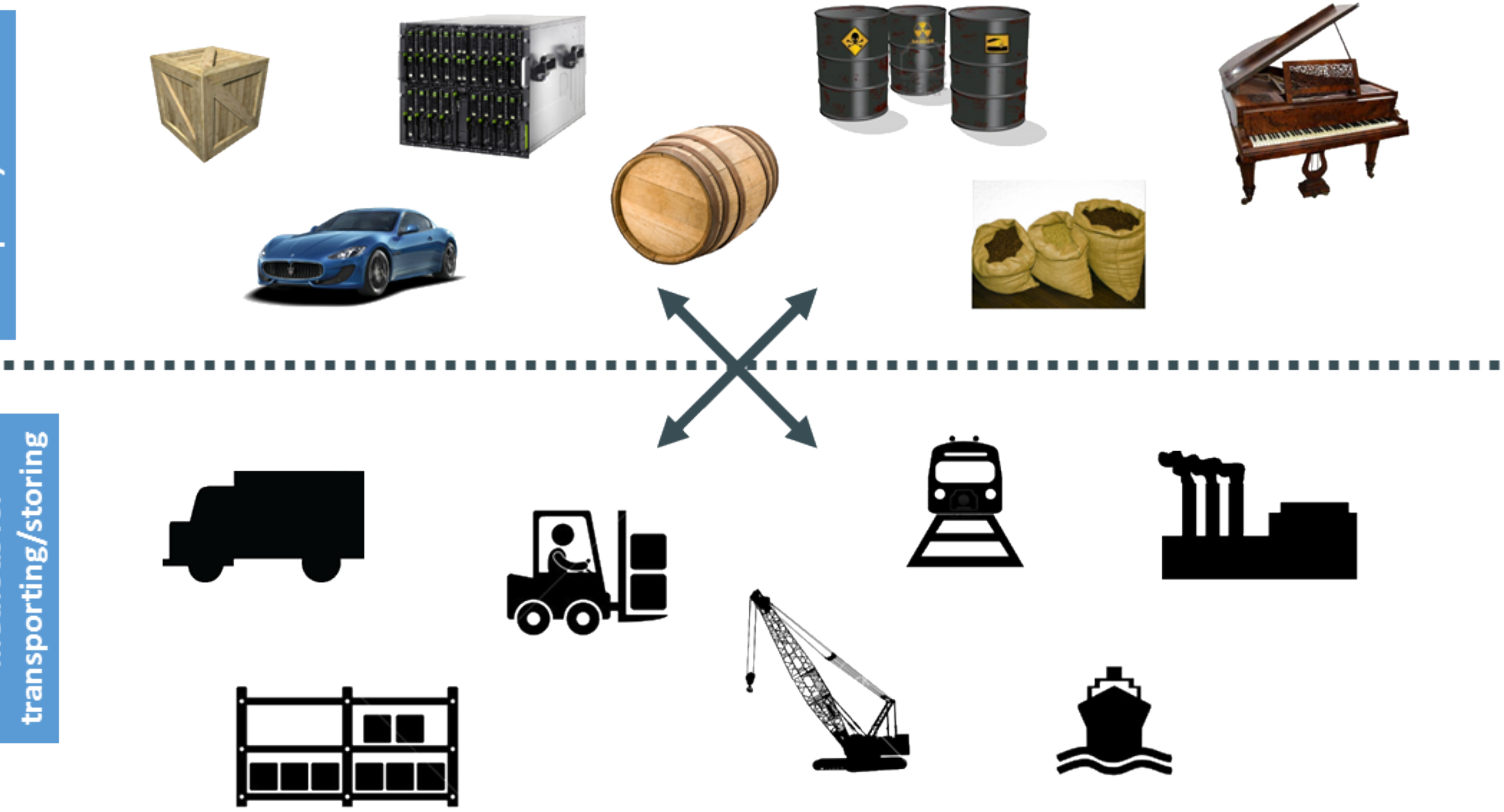
Some components stay local and others go on the cloud

Some virtual machines on different cloud infrastructures

Systems assembled by means of services provided by different cloud solutions

HOW ABOUT CONTAINERS?

CARGO TRANSPORT



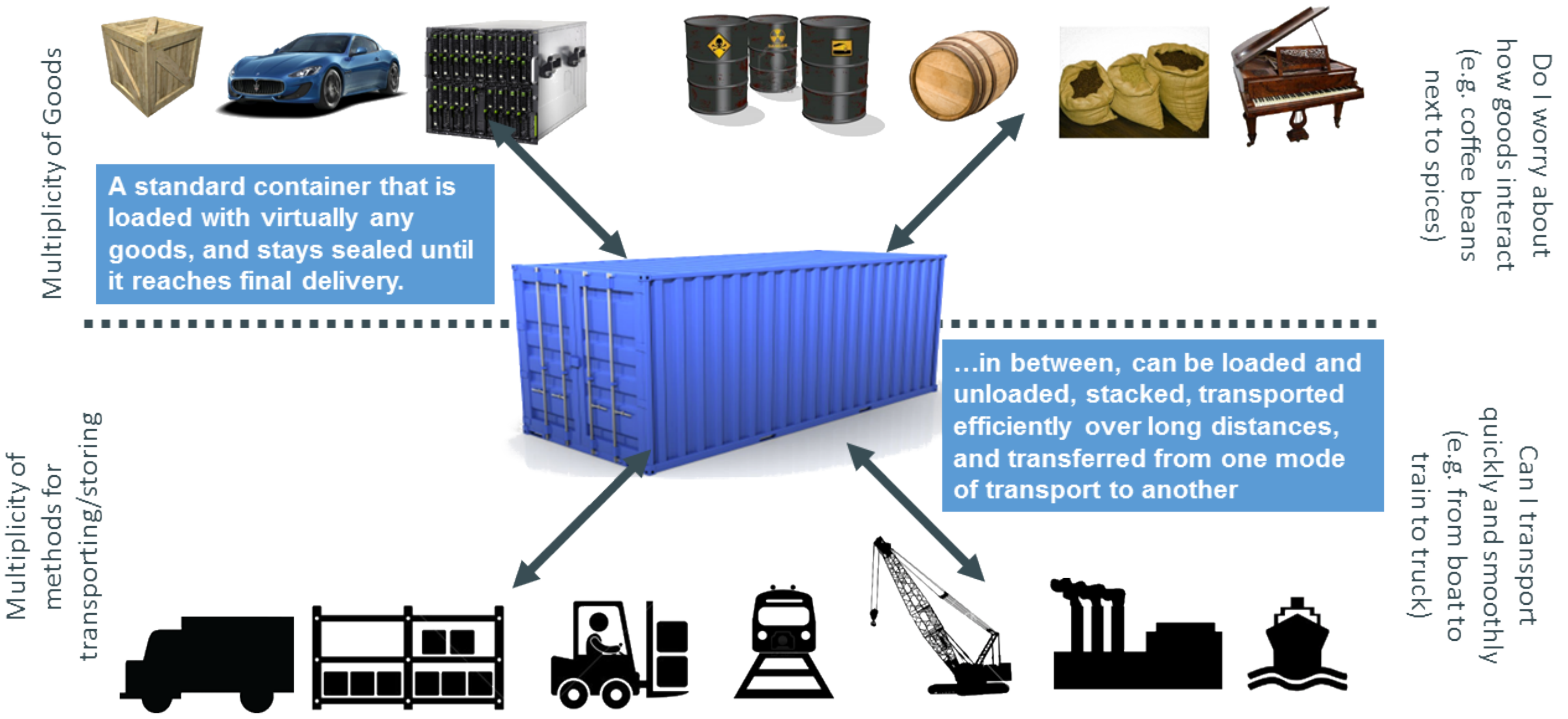
Multiplicity of Goods

Do I worry about how goods interact (e.g. coffee beans next to spices)

Multiplicity of methods for transporting/storing

Can I transport quickly and smoothly (e.g. from boat to train to truck)

CARGO CONTAINERS



Multiplicity of Goods

A standard container that is loaded with virtually any goods, and stays sealed until it reaches final delivery.

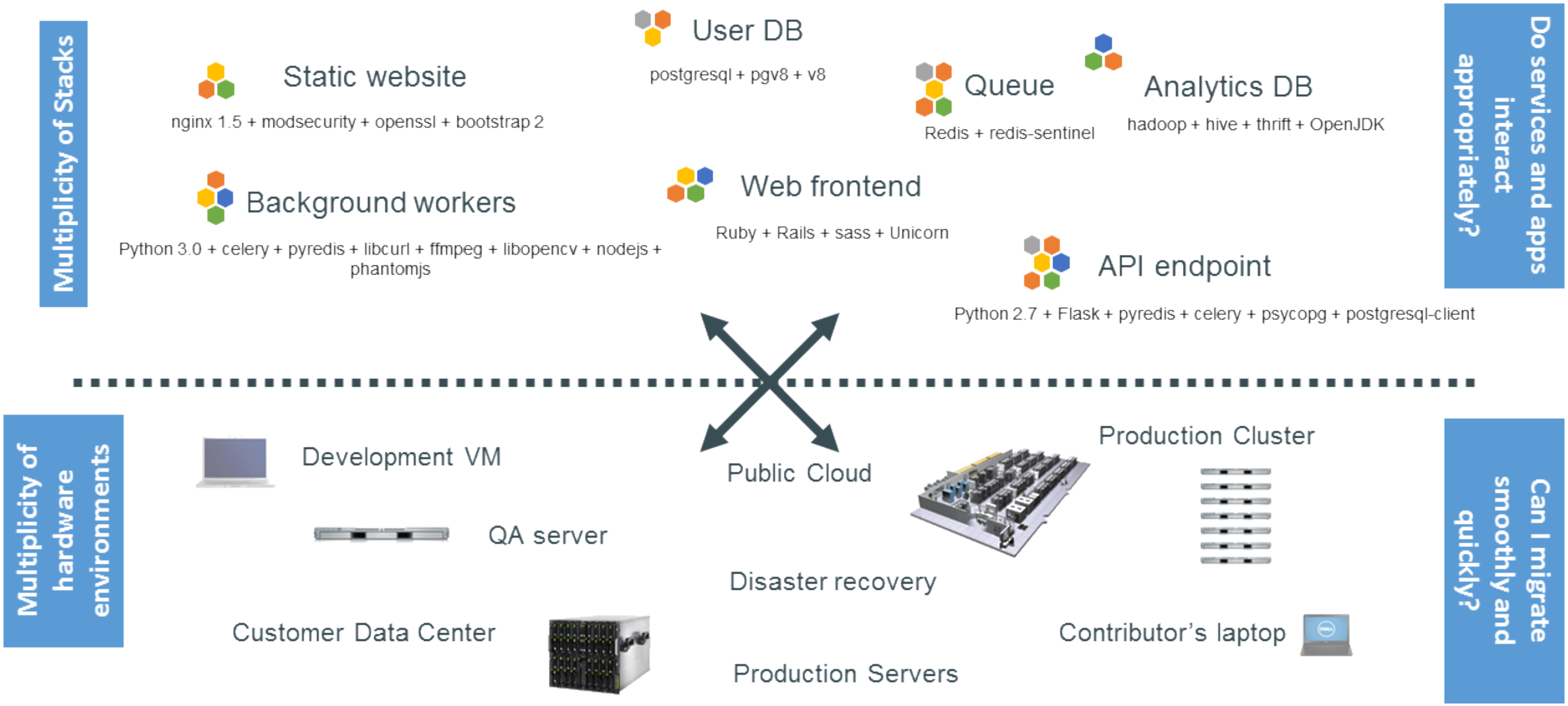
Do I worry about how goods interact (e.g. coffee beans next to spices)

Multiplicity of methods for transporting/storing

…in between, can be loaded and unloaded, stacked, transported efficiently over long distances, and transferred from one mode of transport to another

Can I transport quickly and smoothly (e.g. from boat to train to truck)

BACK TO CLOUD COMPUTING



Multiplicity of Stacks

Static website

nginx 1.5 + modsecurity + openssl + bootstrap 2

User DB

postgresql + pgv8 + v8

Queue

Redis + redis-sentinel

Analytics DB

hadoop + hive + thrift + OpenJDK

Background workers

Python 3.0 + celery + pyredis + libcurl + ffmpeg + libopencv + nodejs + phantomjs

Web frontend

Ruby + Rails + sass + Unicorn

API endpoint

Python 2.7 + Flask + pyredis + celery + psycopg + postgresql-client

Do services and apps interact appropriately?

Multiplicity of hardware environments

Development VM

QA server

Customer Data Center

Public Cloud

Disaster recovery

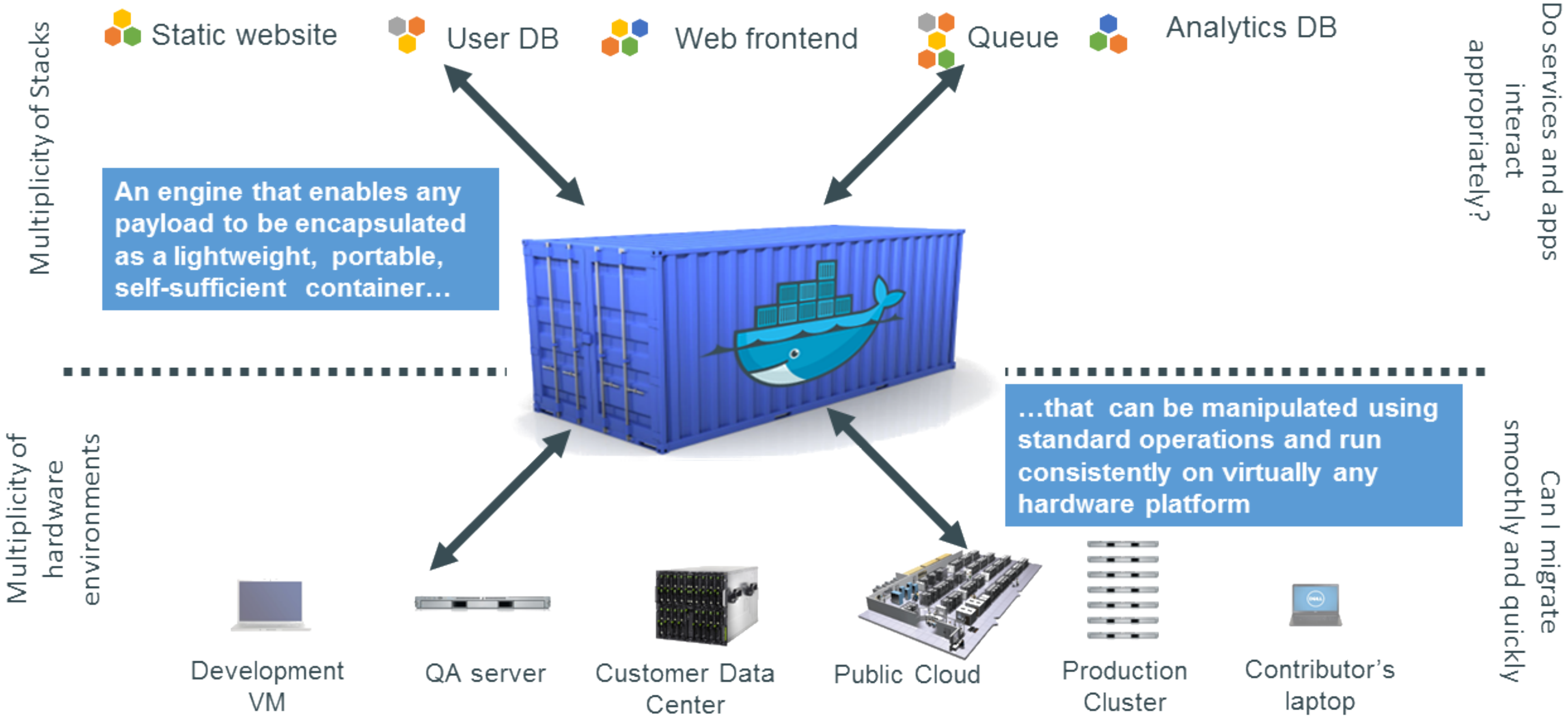
Production Servers

Production Cluster

Contributor’s laptop

Can I migrate smoothly and quickly?

CONTAINER SYSTEM FOR CODE……



Multiplicity of Stacks

Static website

User DB

Web frontend

Queue

Analytics DB

Do services and apps interact appropriately?

An engine that enables any payload to be encapsulated as a lightweight, portable, self-sufficient container…

…that can be manipulated using standard operations and run consistently on virtually any hardware platform

Multiplicity of hardware environments

Development VM

QA server

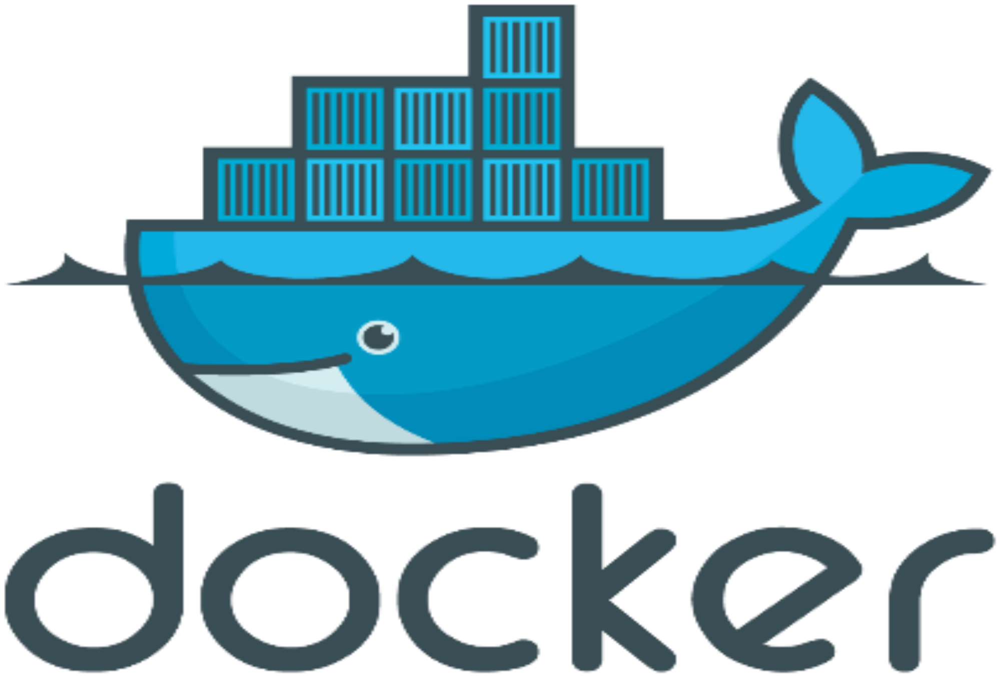
Customer Data Center

Public Cloud

Production Cluster

Contributor's laptop

Can I migrate smoothly and quickly



based on material by Patel Jitendra

WHAT IS DOCKER?

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.

[Source: en.wikipedia.org]

Open platform for developers and sysadmins to build, ship and run distributed applications

Can run on popular 64-bit Linux distributions with kernel 3.8 or later

Supported by several cloud platforms including Amazon EC2, Google Compute Engine, and Rackspace

FEATURES….

Light-Weight

- Minimal overhead (cpu/io/network)
- Based on Linux containers
- Uses layered filesystem to save space (AUFS/LVM)
- Uses a copy-on-write filesystem to track changes

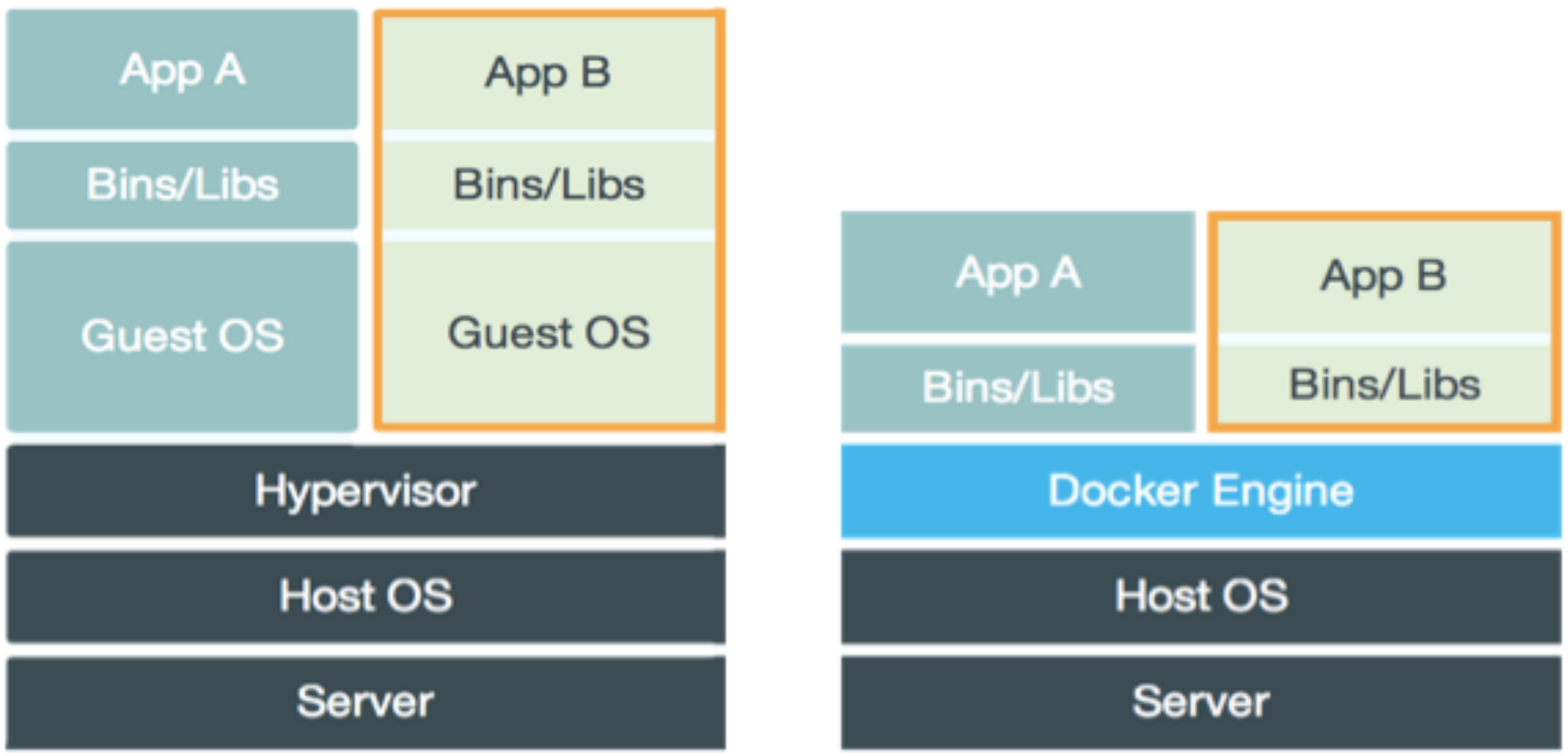
Portable

- Can run on any Linux system that supports LXC (today)
- 0.7 release includes support for RedHat/Fedora family
- Raspberry pi support
- Future plans to support other container tools (lmctfy, etc.)
- Possible future support for other operating systems (Solaris, OSX, Windows?)

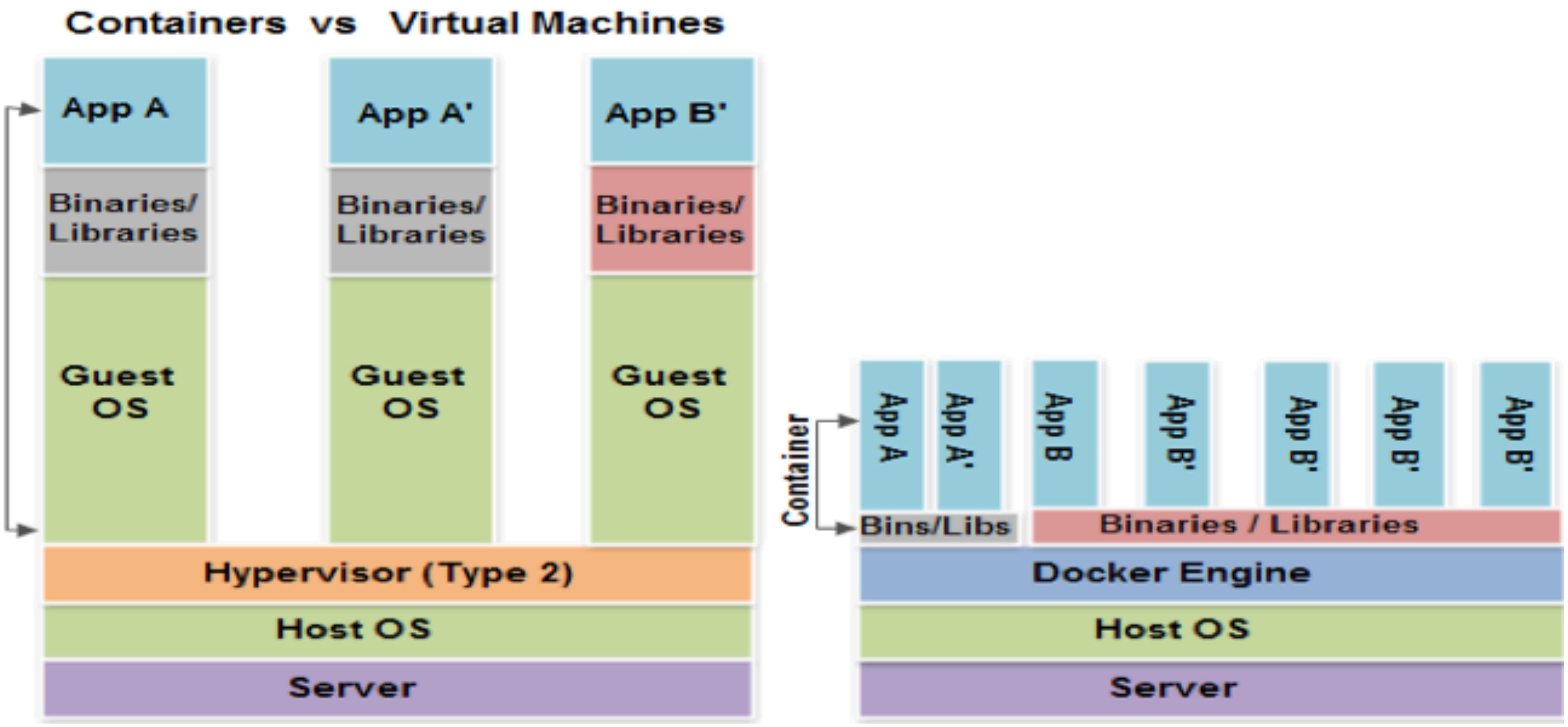
Self-sufficient

- A Docker container contains everything it needs to run
- Minimal Base OS
- Libraries and frameworks
- Application code
- A docker container should be able to run anywhere that Docker can run.

VIRTUAL MACHINE VERSUS CONTAINER



VIRTUAL MACHINE VERSUS CONTAINER

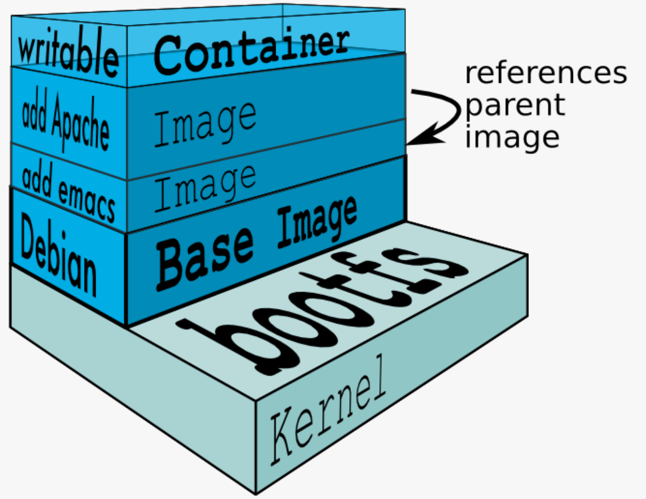


TECHNOLOGY

libvirt: Platform Virtualization

LXC (LinuX Containers): Multiple isolated Linux systems (containers) on a single host

Layered File System



[Source: https://docs.docker.com/terms/layer/]

Docker images

Docker containers

HOW DOES IT WORK ?

You can build Docker images that hold your applications

You can create Docker containers from those Docker images to run your applications.

You can share those Docker images via Docker Hub or your own registry

THEN …

Easy to build, run & share containers

Rapidly expanding ecosystem

Better performance vs. VMs

Layered file system gives us git-like control of images

Reduces complexity of system builds

MICROSERVICES

Many smaller (fine grained), clearly scoped services

- Single Responsibility Principle
- Domain Driven Development
- Bounded Context
- Independently Managed

Clear ownership for each service

- Typically need/adopt the “DevOps” model

Attribution: Adrian Cockroft, Martin Fowler …

COMPOSABILITY– UNIX PHILOSOPHY

Write programs that do one thing and do it well

Write programs to work together

WHY?

Faster and simpler deployments and rollbacks

- Independent Speed of Delivery (by different teams)

Right framework/tool/language for each domain

- Recommendation component using Python?, Catalog Service in Java ..

Greater Resiliency

- Fault Isolation

Better Availability

- If architected right

RESOURCE MANAGEMENT

Luciano Baresi, Sam Guinea, Alberto Leva, and Giovanni Quattrocchi.
Advanced Self-adaptation of cloud applications with containerization and control theory. FSE'16, to appear

RUNTIME MANAGEMENT OF CLOUD APPLICATIONS

Satisfy functional and non-functional requirements

Dynamic resource allocation to manage a changing workload (number of requests per time unit)

Runtime management through autonomic adaptation

Focus on Virtual Machines both in academia and industry

Many kind of adaptations, many things to adapt

ELASTIC PROVISIONING

Containers enable adaptation at the Operating System layer

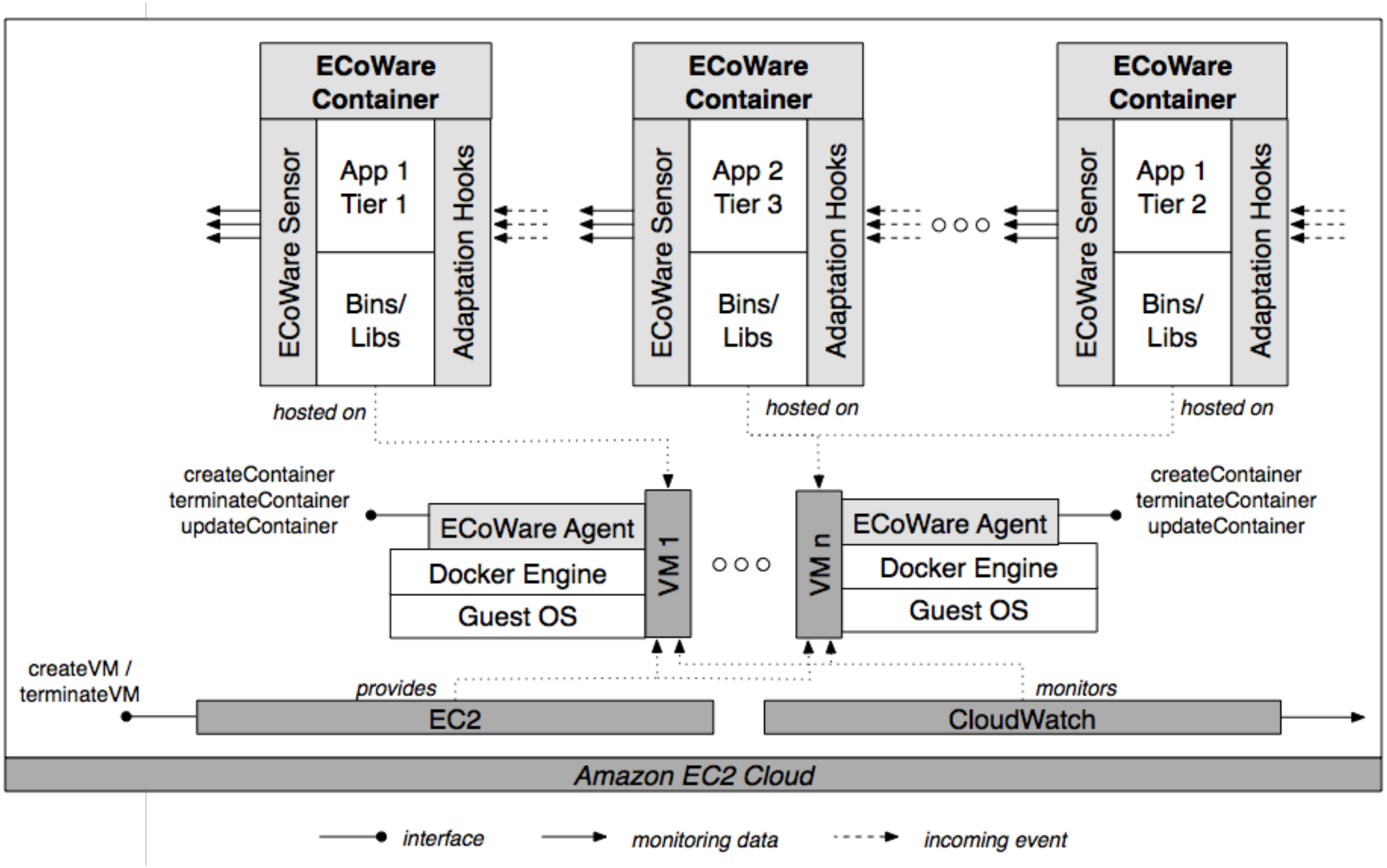
Instant reaction to workload changes by dynamically starting, terminating or updating (vertical scaling) containers

We deploy many containers inside each VM

Finer granularity than working only with VMs

Resources allocated to a middleware (e.g., MySQL, JBoss AS) can change during execution

SUPPORTED ARCHITECTURE



PLANNER

Control-theoretical design, discrete-time

Each application tier is endowed with a local controller devoted to maintaining a desired response time

The controller computes the resources (i.e., CPU cores and memory) that need to be made available to that tier

Five generic actions

- Container actions (create, update, terminate)
- VM actions (create, terminate)

CONTROL-THEORETICAL DESIGN (I)

$$f\left(\frac{c(k)}{r(k)}\right) = c_1 + \frac{c_2}{1 + c_3\,\frac{c(k)}{r(k)}}$$

Grey-box approach

The characteristic function is monotonically decreasing towards a possible lower horizontal asymptote (finite parallelism degree)

Not linear, depends on c (core) and r (workload); c1,2,3 obtained through profiling

Invertible in the signal range of interest

ACTIONS

Each controller emits the requested resource allocation for a tier

These data are then passed to an ILP solver to be translated into actions (create VM, create container, …)

The ILP formulation is a variation of the 2-dimensional bin packing problem: the bins are the VMs and we need to pack containers inside them

Each of the five actions has a weight w, such that containers actions are preferred over VM actions

PLATFORM ADAPTATION

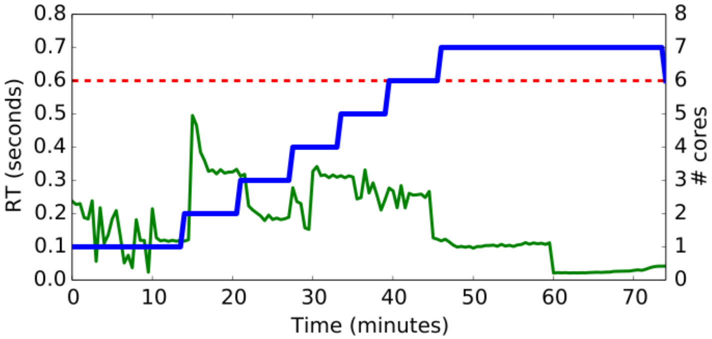
Scripts declared inside the Applications Description file and mounted into containers when launched

Separated from the planner that we keep technology agnostic

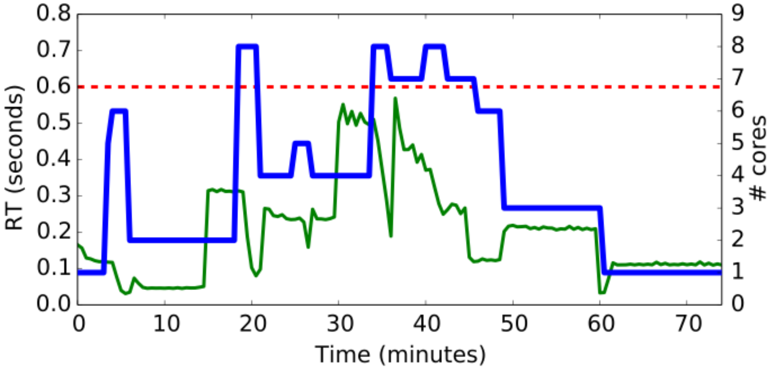
When some events happen we invoke the specific hook that manages the particular event

SOME RESULTS

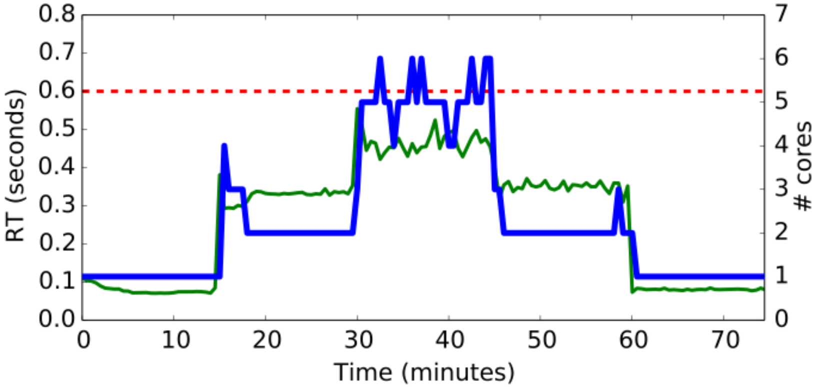
AWS autoscaling



ECoWare VM only



ECoWare VM + Containers



RT 🟩 **Core** 🟦 **SLA** - - -

Q&A
You have
Questions
We have
Answers