Pivotal

Intro to Containers

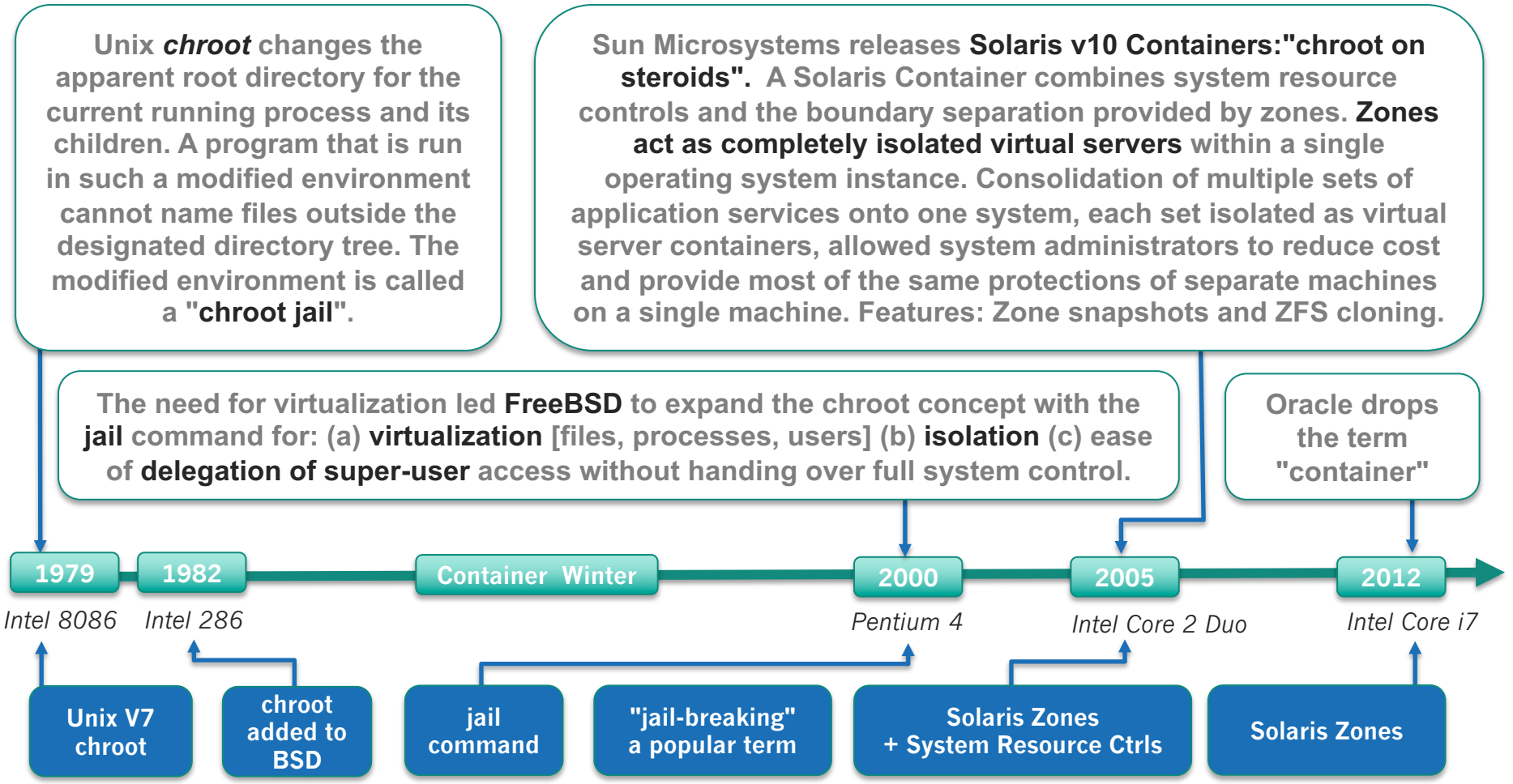
April 2019

Ralph Meira – Platform Architect

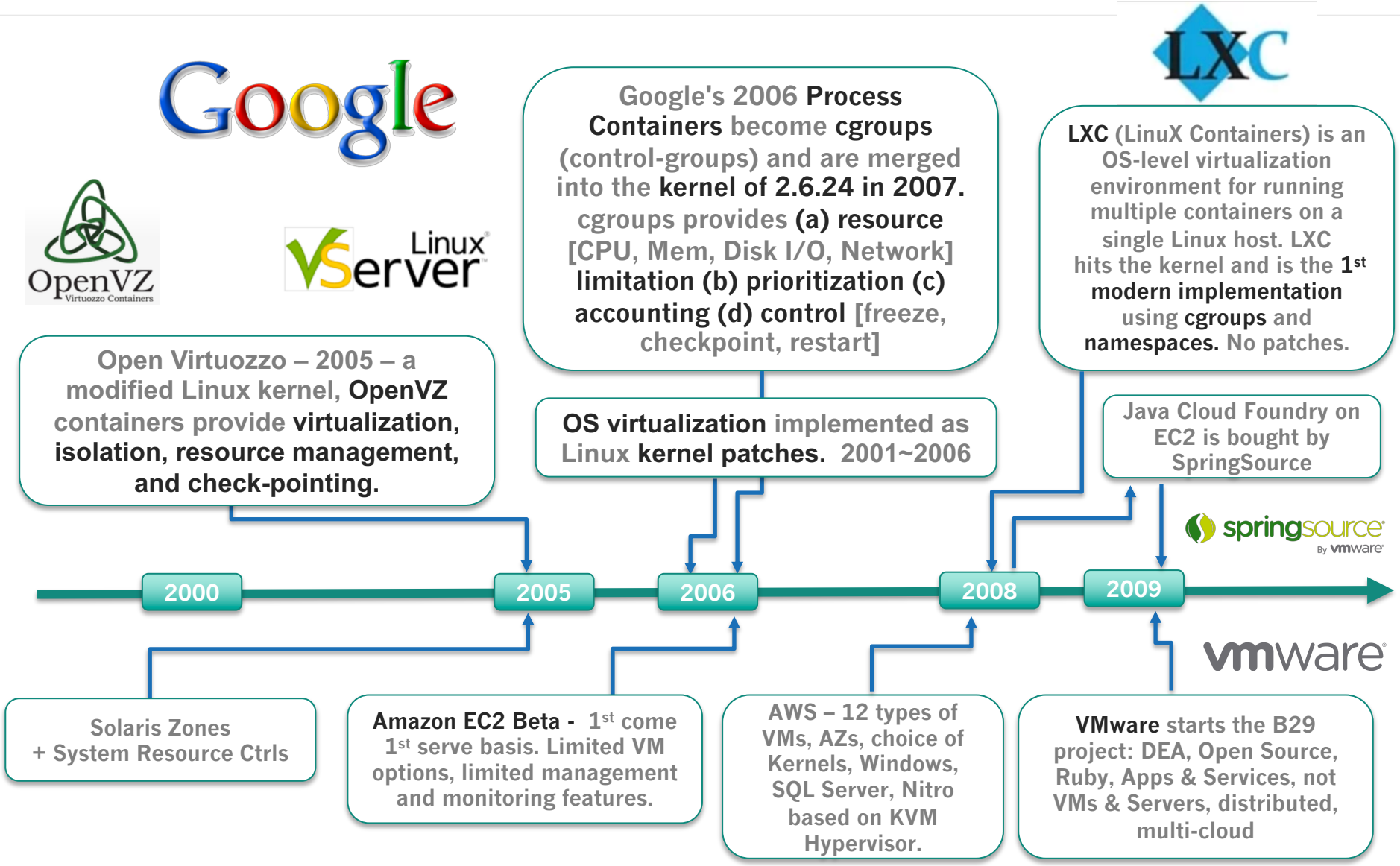
Agenda

- What are containers?
- Why are containers suddenly so important?
- Docker, Kubernetes, CNCF, … Who’s who?
- Container Orchestration
- IaaS, CaaS, PaaS, FaaS and SaaS & Workloads
- DevOps
- The bottom line

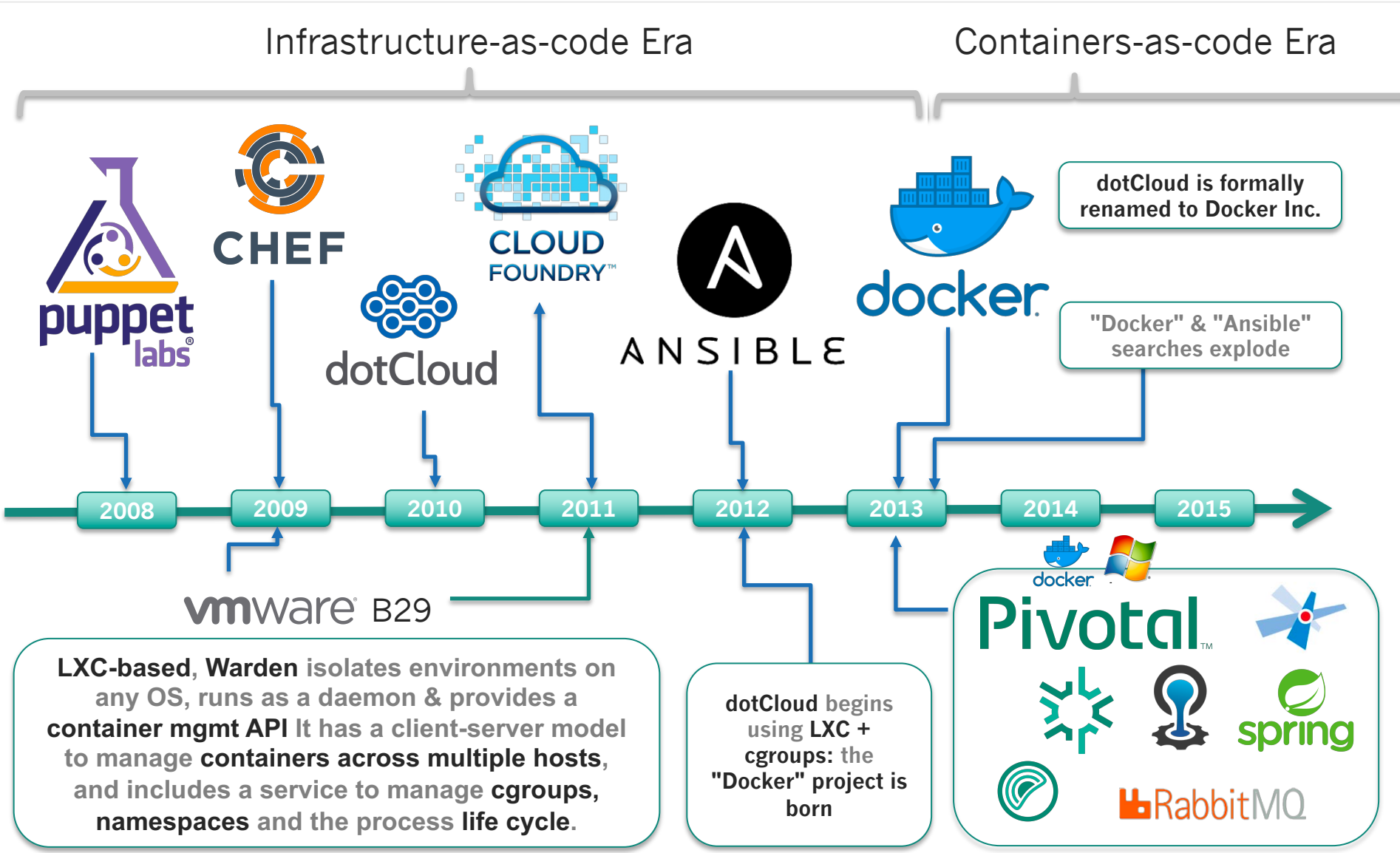
Chronology – Orchestration & Containers



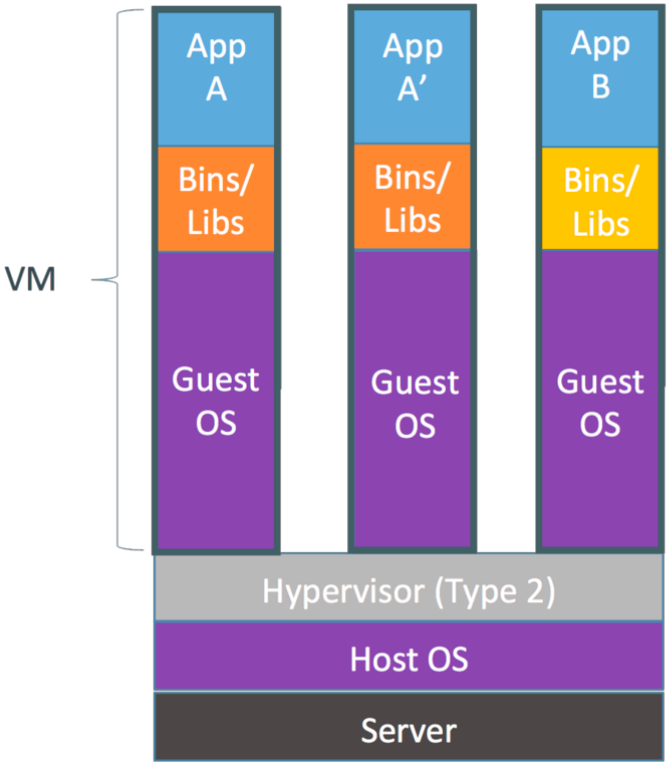
Chronology – Orchestration & Containers



Chronology – Orchestration & Containers

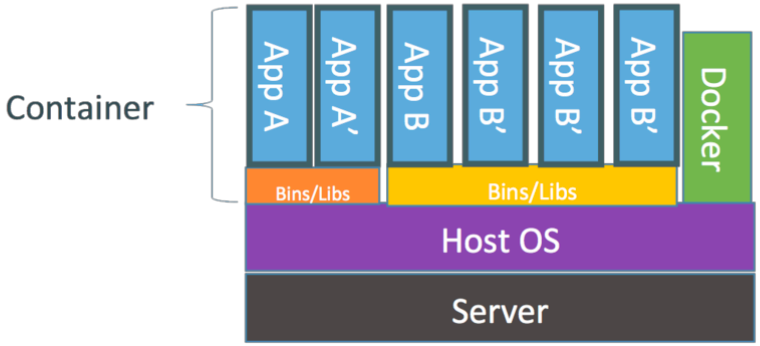


App Containers vs. VMs (Virtual Machines)



Containers are isolated, but share OS and, where appropriate, bins/libraries

...result is significantly faster deployment, much less overhead, easier migration, faster restart



CITRIX XenServer

vmware vSphere

docker

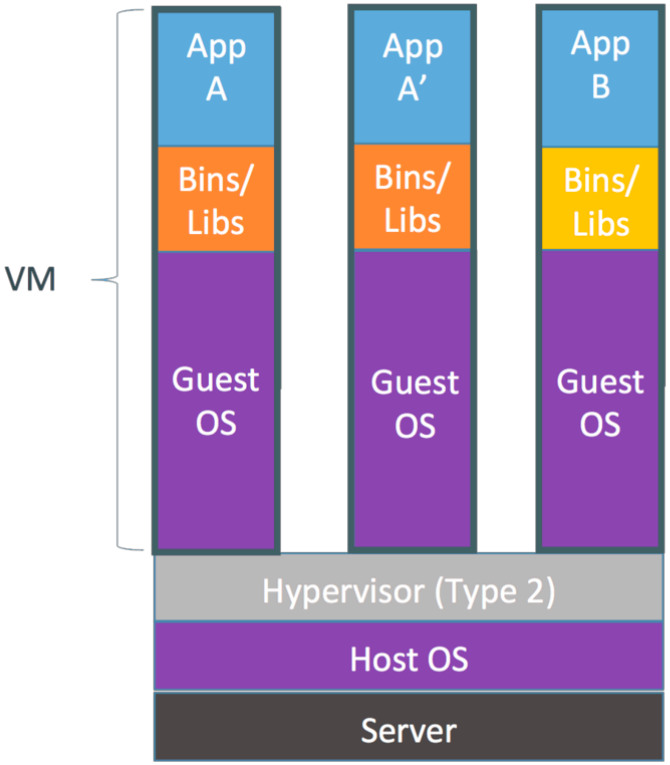
rkt

Microsoft Hyper-v

KVM

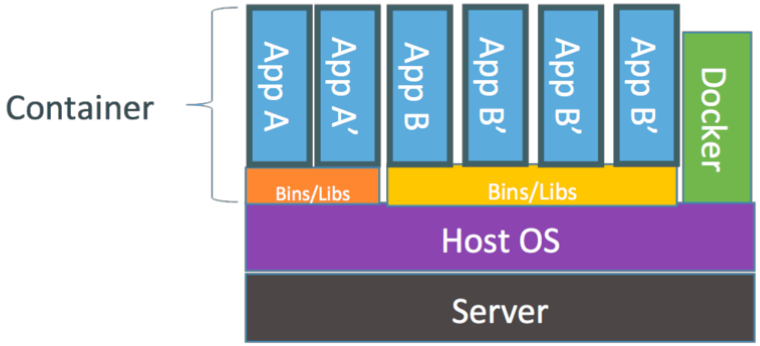
OPEN CONTAINER INITIATIVE

App Containers vs. VMs (Virtual Machines)



Containers are isolated, but share OS and, where appropriate, bins/libraries

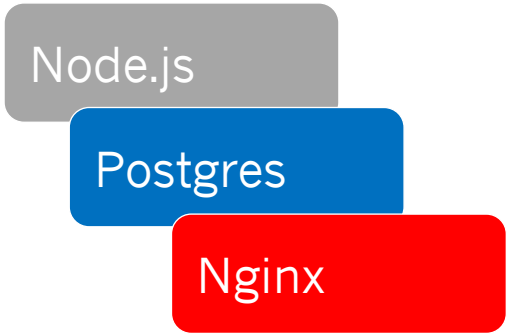
...result is significantly faster deployment, much less overhead, easier migration, faster restart



- Abstraction of H/W Infrastructure
- Secure Isolation of workloads
- 20 years of well understood tech
- Resource partitioning & mgmt.
- Massive installed base & skilled workforce

- App Packaging
- App distribution Ecosystem
- Orchestration of App microservices
- Dev, Test, Stage, Prod flows
- App-centric management

OS Containers vs. App Containers



OS Containers

- Meant to be used as an OS running multiple services
- By default, no layered filesystems
- Built on cgroups, namespaces, native process resource isolation
- Examples: LXC, OpenVZ, Linux Vserver, BSD Jails, Solaris Zones

App Containers

- Meant to run a single service
- Layered filesystems
- Built on top of OS container technologies
- Examples: Docker, Rocket, RunC

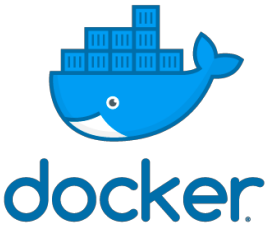
Container – “feels like a lightweight VM”

- I can* SSH into a container
- It has its own process space
- It has its own network interface
- I can* run commands and apps as root
- I can install packages
- I can run services
- I can mess up routing, iptables ...

(*) *if allowed*

Container – “but it's not quite like a VM”

- It uses the host kernel
- It can't boot a different OS
- It can't have its own modules
- It doesn't need init as PID 1
- It doesn't need syslogd, cron...
- It's just normal processes on the host machine
- Very fast to start & stop

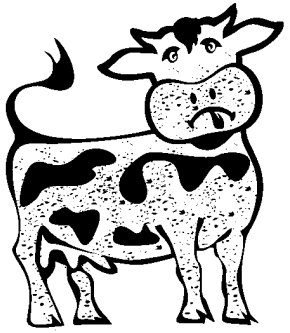


also uses Namespaces and cgroups

When **Docker** emerged in **2013**, containers exploded in popularity. Just as Warden did, Docker also **used LXC** in its initial stages and **later** replaced it with **libcontainer** – its own container management library. **Docker** separated itself from the pack by offering an entire ecosystem for container management that **was easy to use**.

2016: The importance of Container Security is Revealed

With the wide adoption container-based applications, systems became more complex, increasing the surface of attack and laying the groundwork for Container Security. Vulnerabilities like **Dirty Cow** only furthered this thinking. This led to a shift left in security along with the software development lifecycle, making security a key part of each stage in container-based App development, also known as **DevSecOps**. The goal is to build secure containers from the ground up without reducing time to market.



DIRTY COW

MITRE **CVE-2016-5195**
a privilege escalation vulnerability in the Linux Kernel

Containers Mature → Container Orchestration

2017 - Hundreds of tools have been developed to ease container management. While these types of tools have been around for years, 2017 is the year that many of them earned their stripes. Just look at Kubernetes; since its adoption into the Cloud Native Computing Foundation (CNCF) in 2016, Pivotal, VMWare, Azure, AWS, and even Docker have announced their support, on top of their infrastructures.
While the marketplace is still growing, some tools have come to define certain functions in the container community. Ceph and REX-Ray set standards for container storage, while Flannel connects millions of containers across datacenters. And in CI/CD, Jenkins, CircleCI, Concourse are completely changing the way we build and deploy apps.

Adoption of rkt and Containerd by CNCF
The container ecosystem is unique as it is powered by a community-wide effort and commitment to open source projects. Docker's donation of the Containerd project to the CNCF in 2017 is emblematic of this concept, as well as the CNCF's adoption of the rkt ("rocket") container runtime around the same time. This has led to greater collaboration between projects, more choices for users, and a community centered around improving the container ecosystem.

So, why are containers so popular?

- **Containers enable developers to package an App** and all its dependencies (OS, database, etc.) on a laptop and then deploy it to a server unchanged. It's simple, easy and repeatable.
- Applications, broken down into **microservices** (component parts), which are then managed individually, have proven to be a huge **improvement** over monolithic architectures in terms of **development velocity**, manageability, scalability, deployment, resilience to failure, and cloud infrastructure friendliness.
- **Containers** became the default **packaging** for tenths to thousands of **microservices** per App. Multiply that across an organization and you have a management problem: containers and services with various properties (duration, function, permissions, etc.) and relationships to each other. To manage it all you need a container orchestration solution.
- From an infrastructure perspective, **containers** are a more **granular and efficient** form of application virtualization. They support a more traditional lift-and-shift (as opposed to re-architecture) approach to adopting new technologies. But, they require an orchestration layer.
- In true open source fashion, **network effects** came into play as the community of developers contributing to **Kubernetes** grew, software vendors and cloud providers followed along because developing their own **container orchestration platforms** did not make financial sense.

What is Kubernetes (K8s)?

- a system for managing application containers
- a container orchestration platform: deploy, scale & operate
- κυβερνήτης: greek for "governor", "helmsman" or "captain"

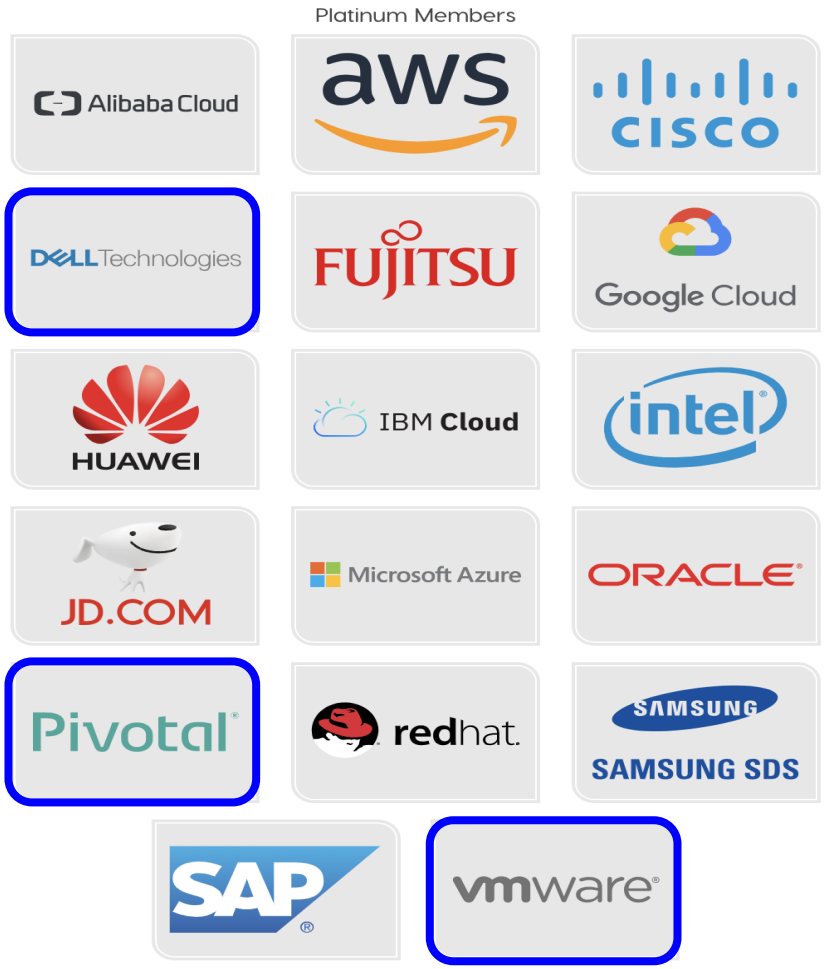
Kubernetes was originally created and released as an open source project (2014) by Google. It is based on Google's Borg and Omega cluster-management systems and its original code name was Seven, because Seven-of-Nine was a friendlier Borg.

Borg

Who are the stewards of K8s?

CLOUD NATIVE COMPUTING FOUNDATION

- K8s was originally created by Joe Beda, Brendan Burns and Craig McLuckie using C++, and later re-written in Go.
- Google donated K8s to the Linux Foundation as a seed project for the creation of the **Cloud Native Computing Foundation** (2015).

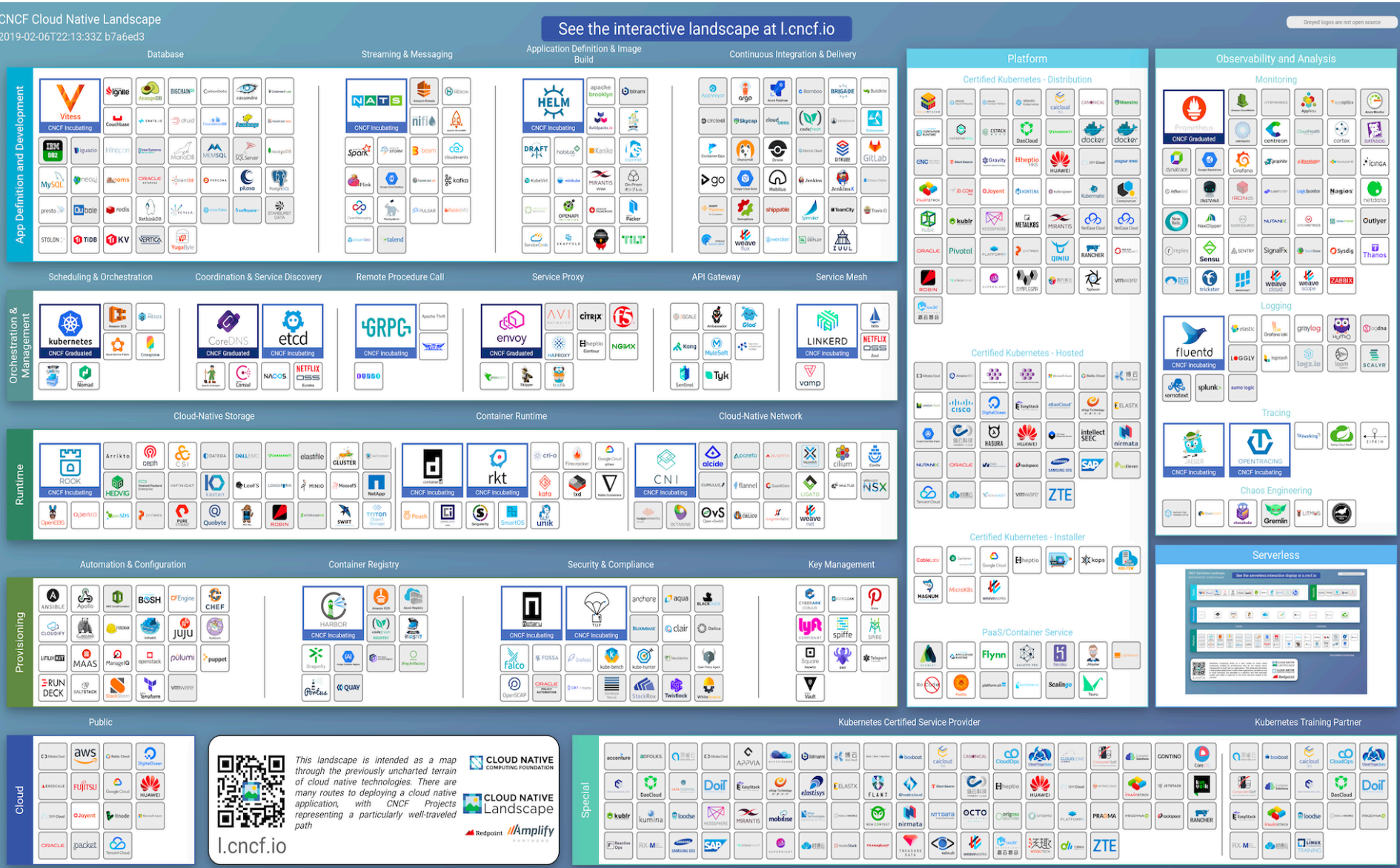


What is the CNCF charter?

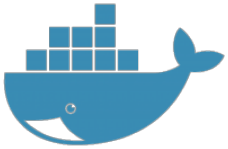
CLOUD NATIVE COMPUTING FOUNDATION

- to build sustainable ecosystems and foster a community around a constellation of ***high-quality projects*** that orchestrate ***containers*** as part of a ***microservices architecture***.

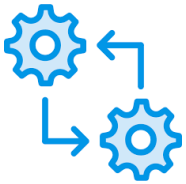
How complex is the Cloud Native space?



Companies have many ways to package and run their workloads in the cloud



CONTAINERS



Batches

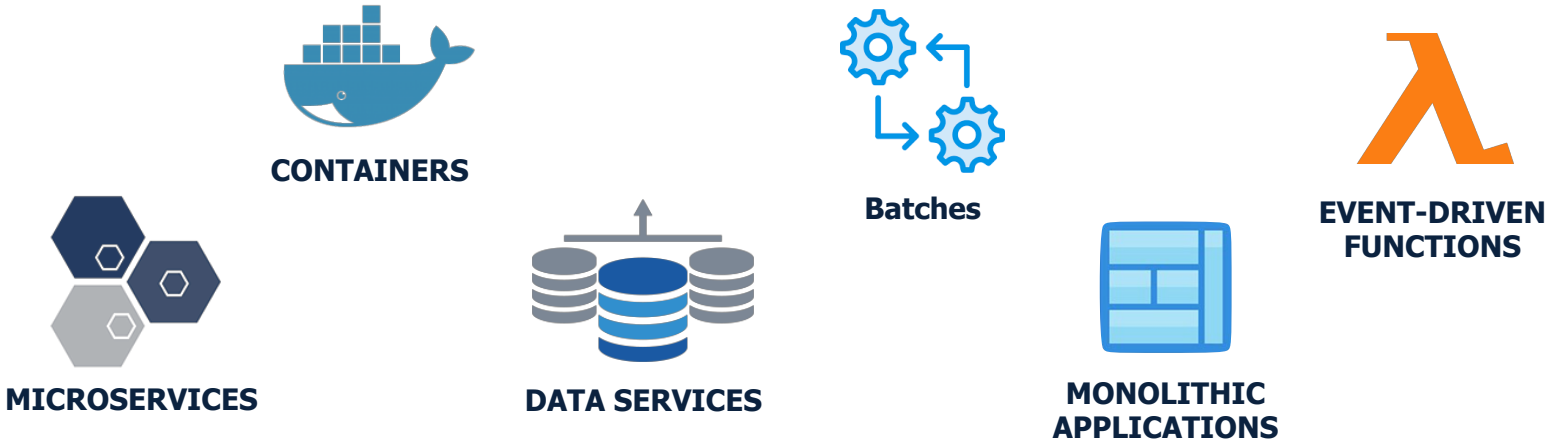
EVENT-DRIVEN FUNCTIONS

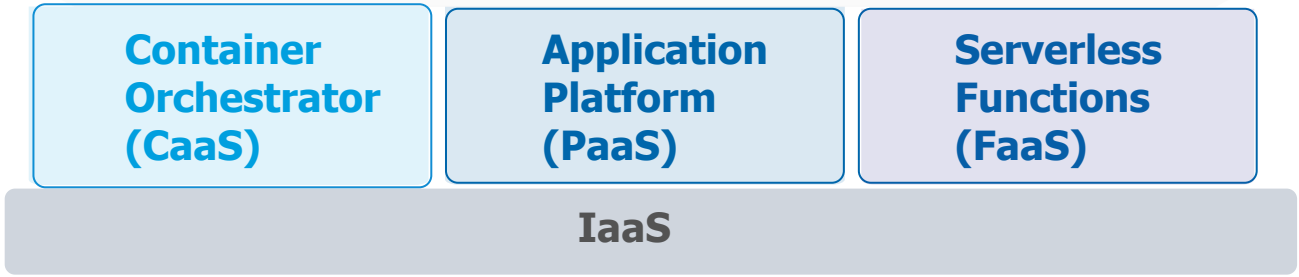
MICROSERVICES

DATA SERVICES

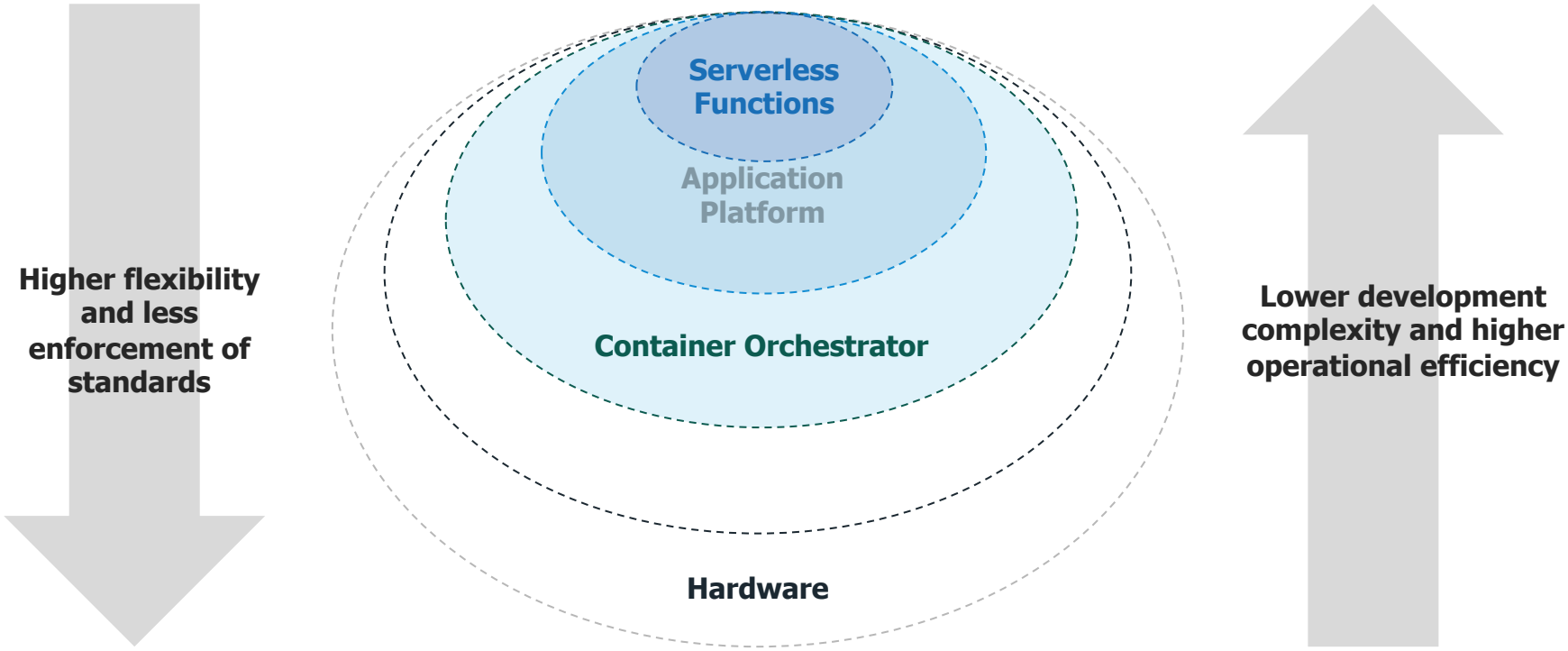
MONOLITHIC APPLICATIONS

Their goal: pick the right runtime for each workload



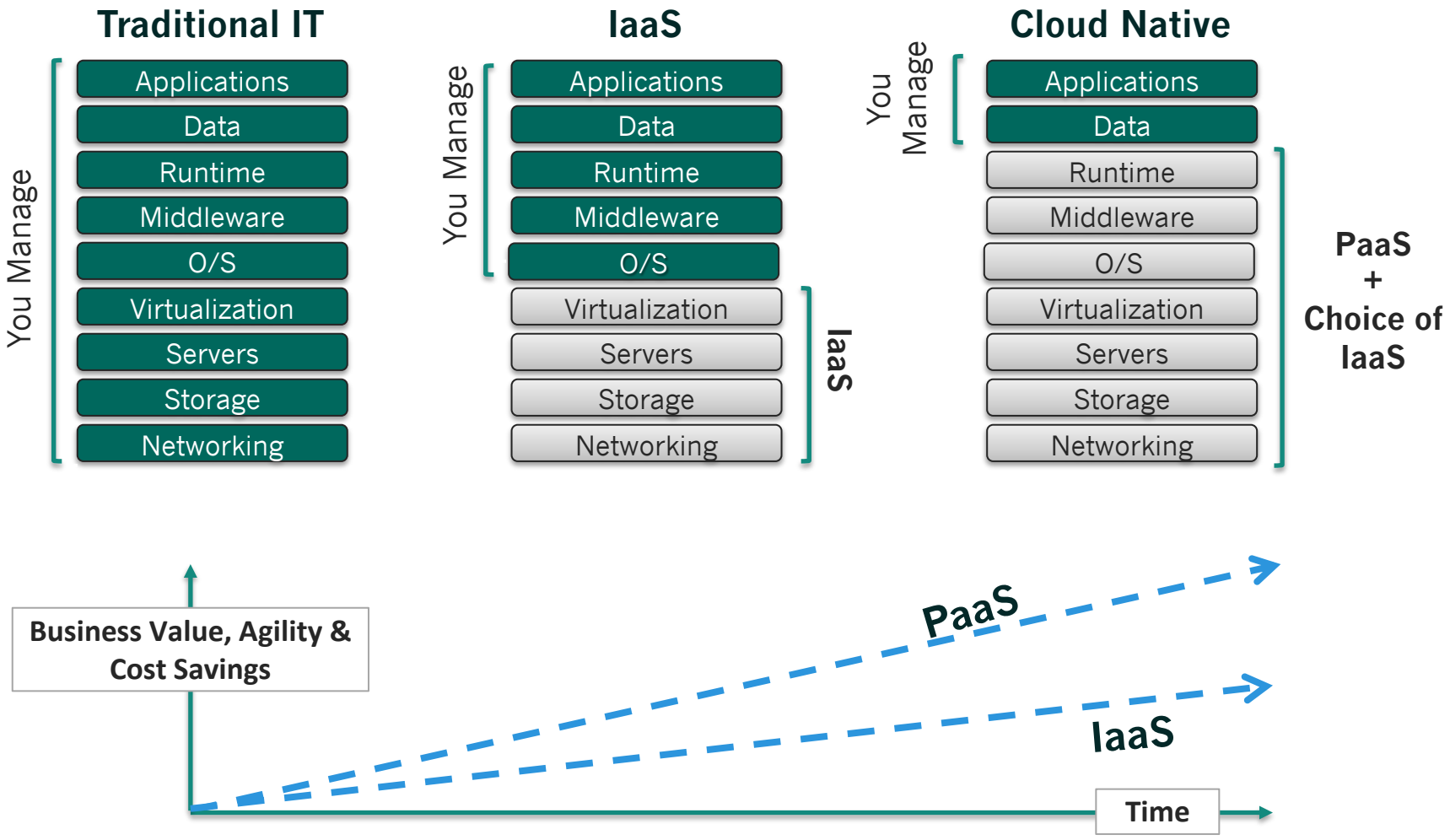


Use the highest abstraction possible



Strategic goal: Push as many workloads as technically feasible to the top of the platform hierarchy

Another way of looking at this evolution to Cloud Native Application Platforms

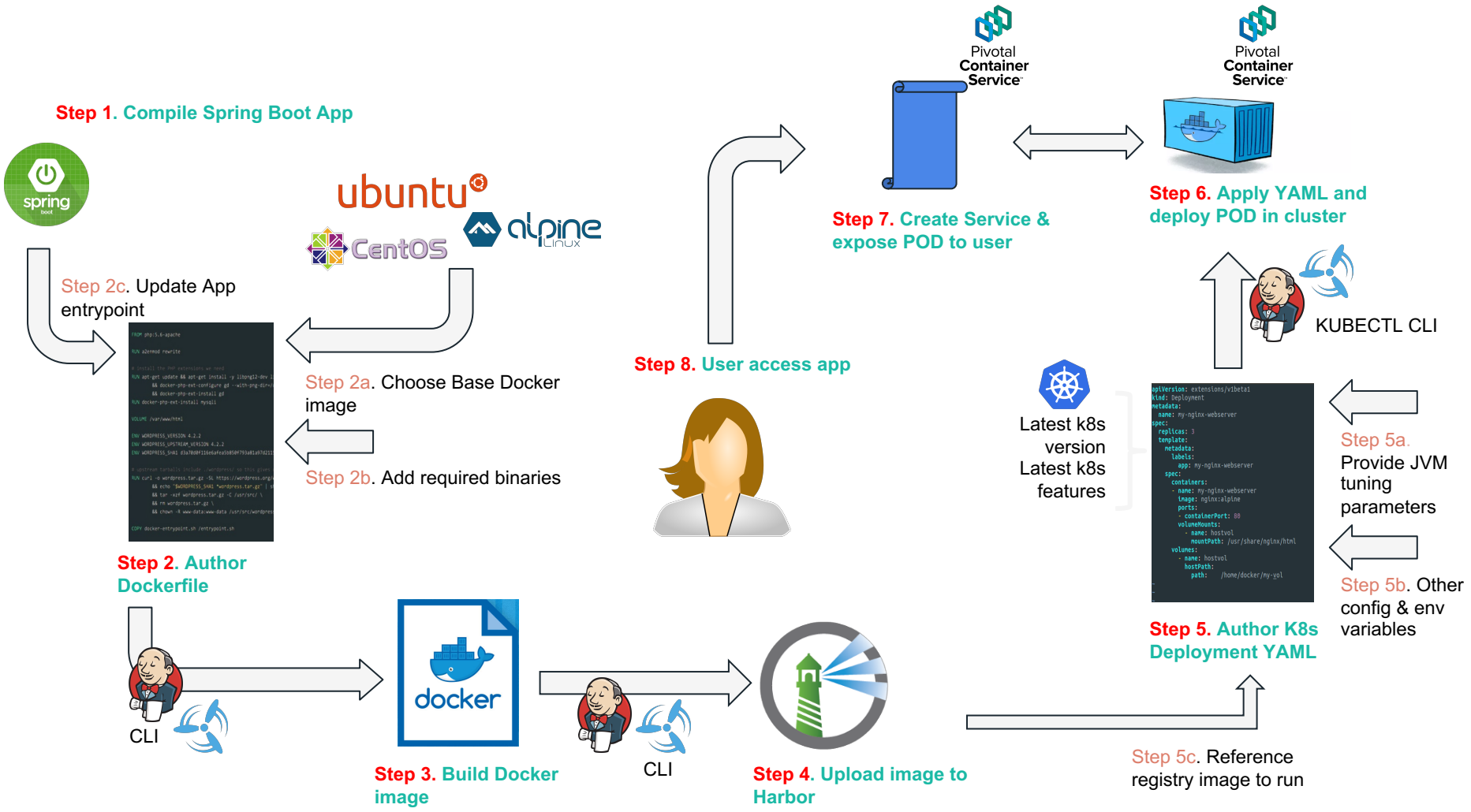


Key Insight about Cloud Computing

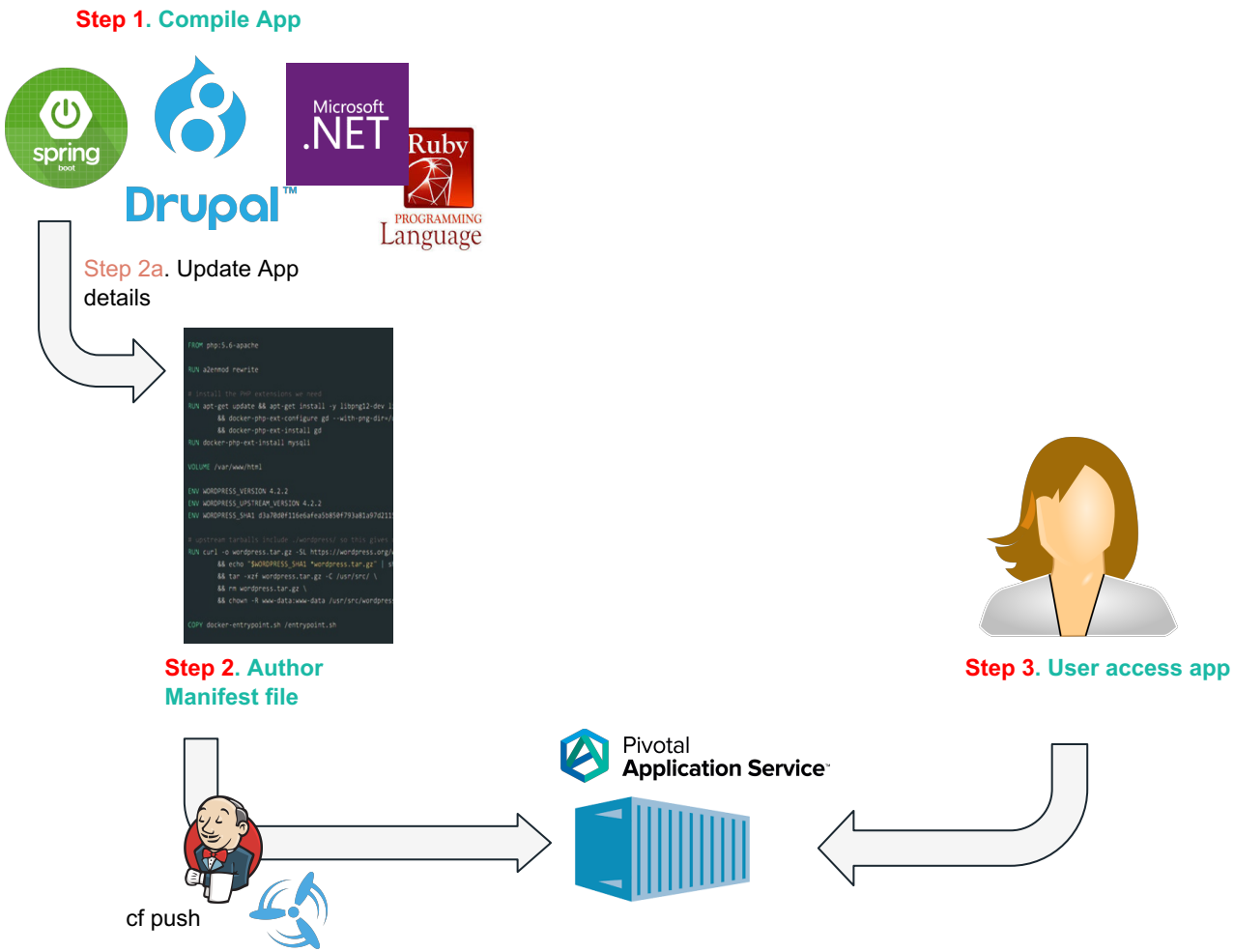
“Cloud is about how you do computing, not where you do computing.”

Paul Maritz – Chairman, Pivotal

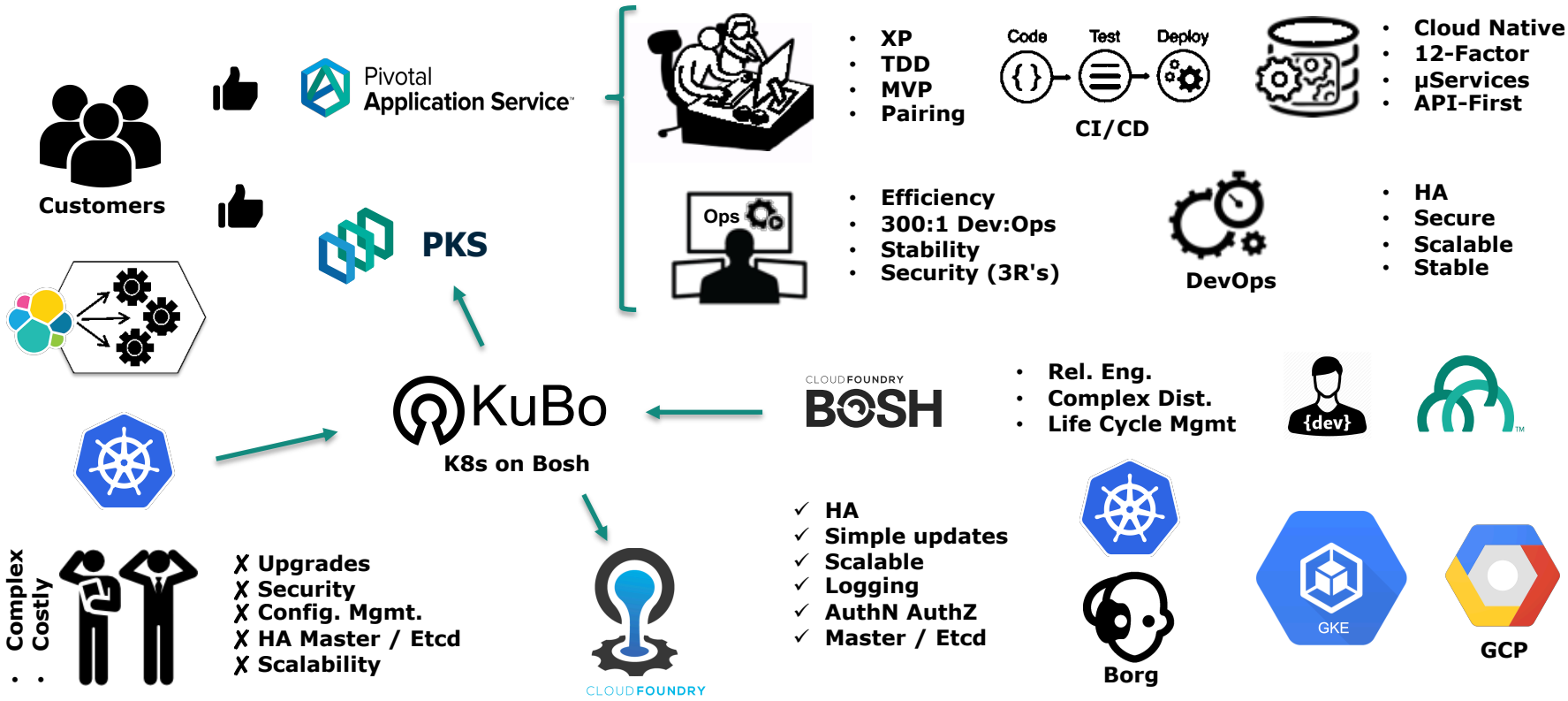
Spring Boot K8s deployment - CaaS



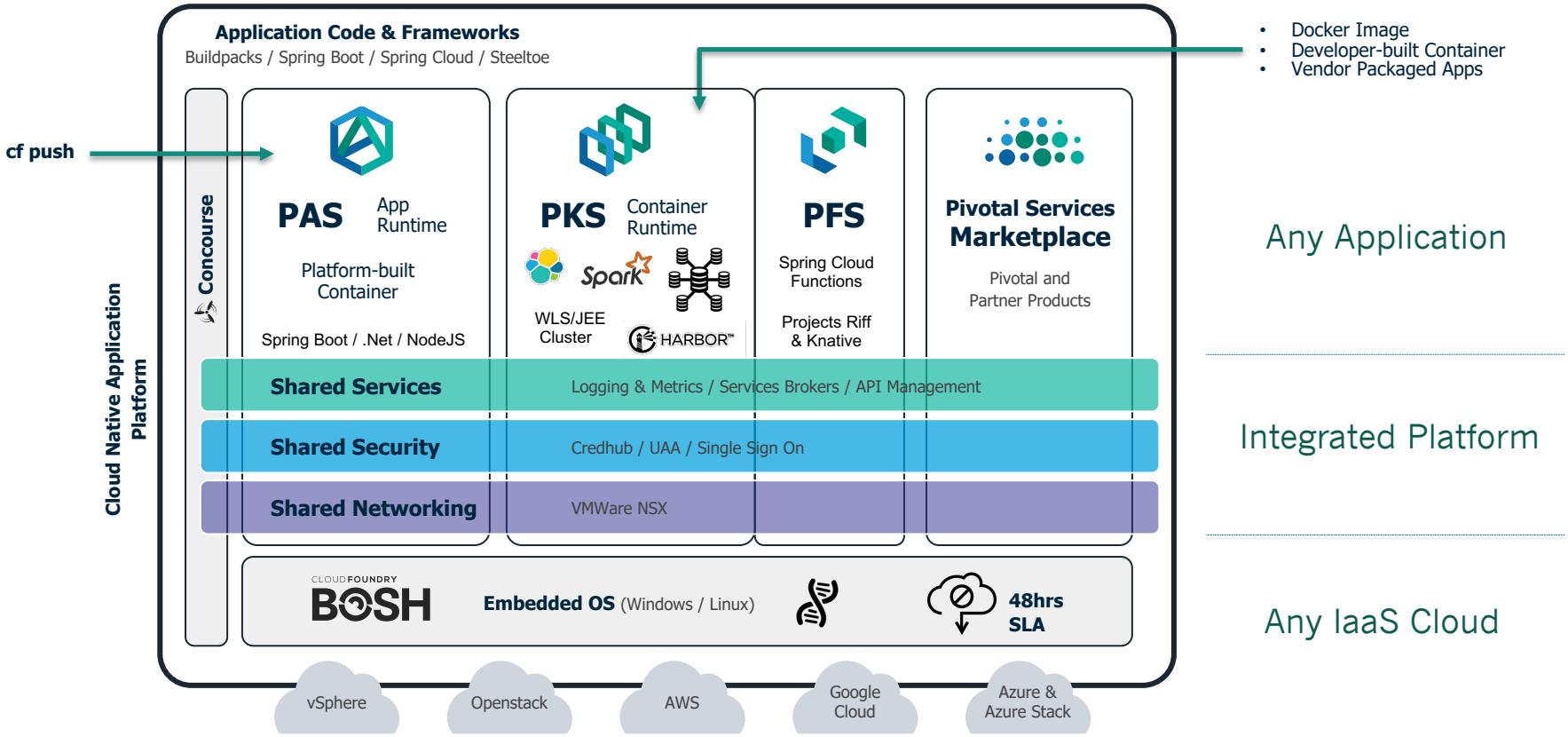
Spring Boot or .NET Core PaaS deployment



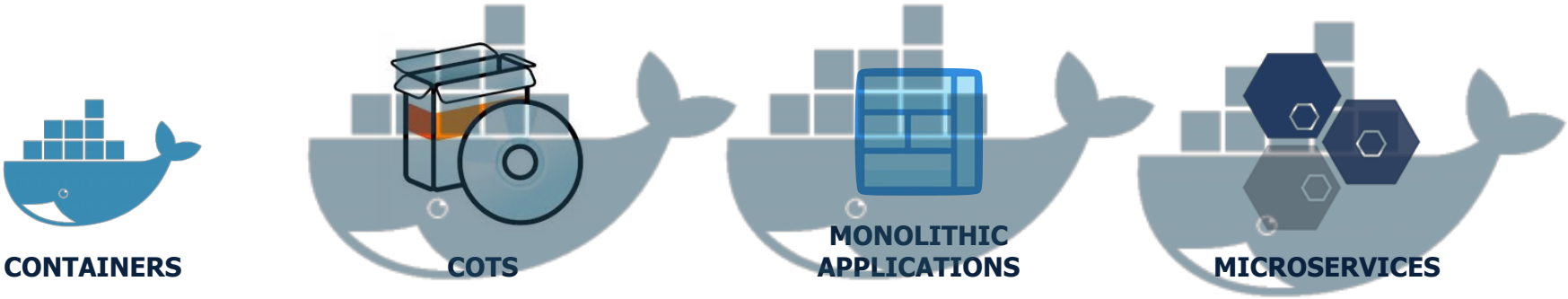
CaaS and PaaS



CaaS, PaaS, FaaS and OSBAPI Services

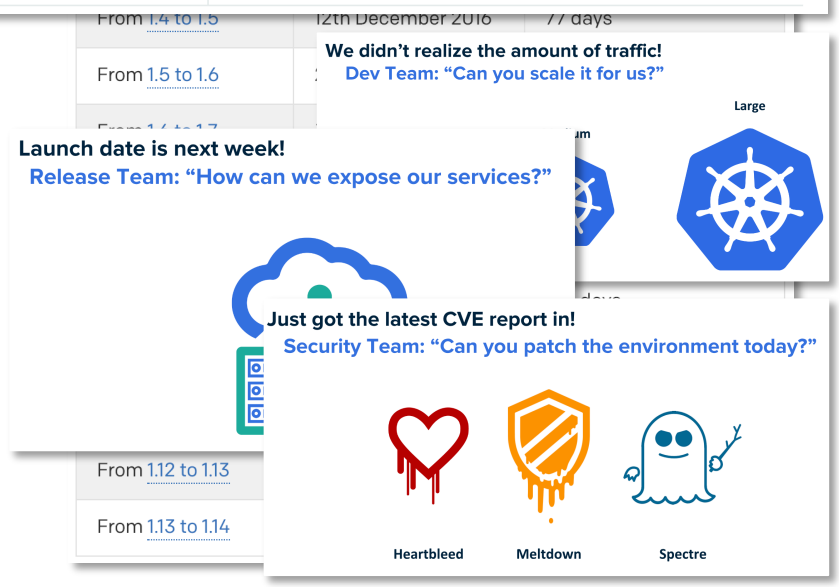


Containerized workloads



Kubernetes

K8s in the Enterprise

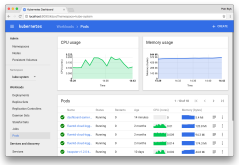


Basic Kubernetes Deployment

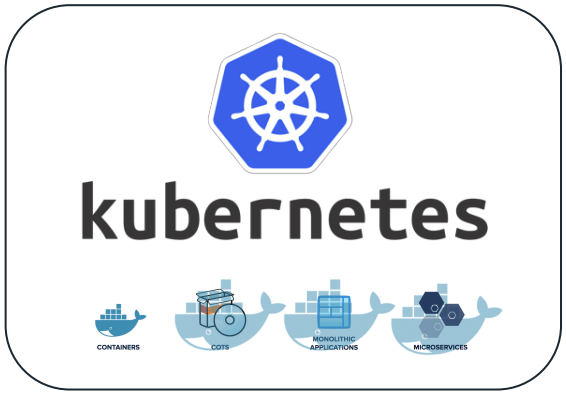
App User

Dev / Apps

```
> kubectl
```



Kubernetes Dashboard



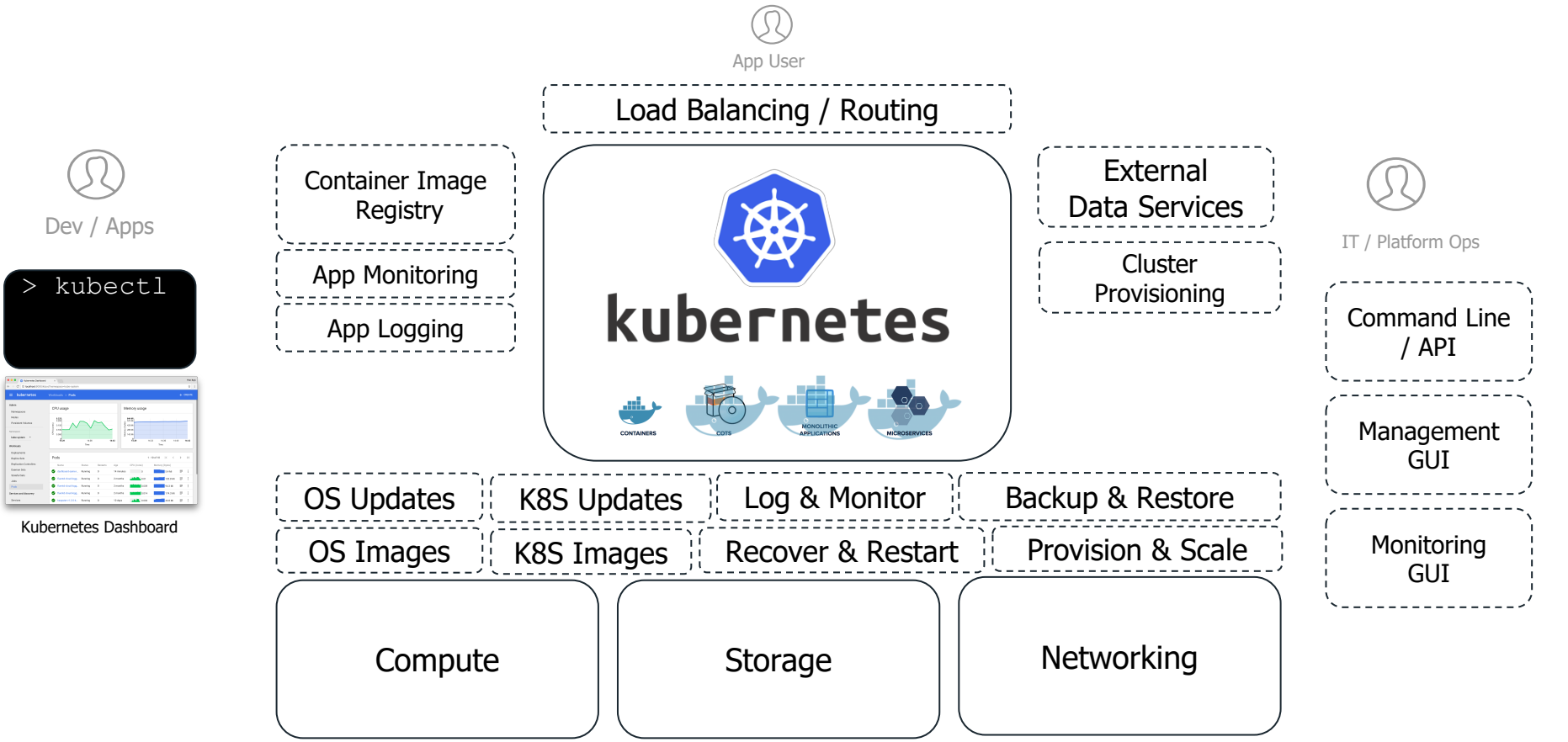
IT / Ops

Compute

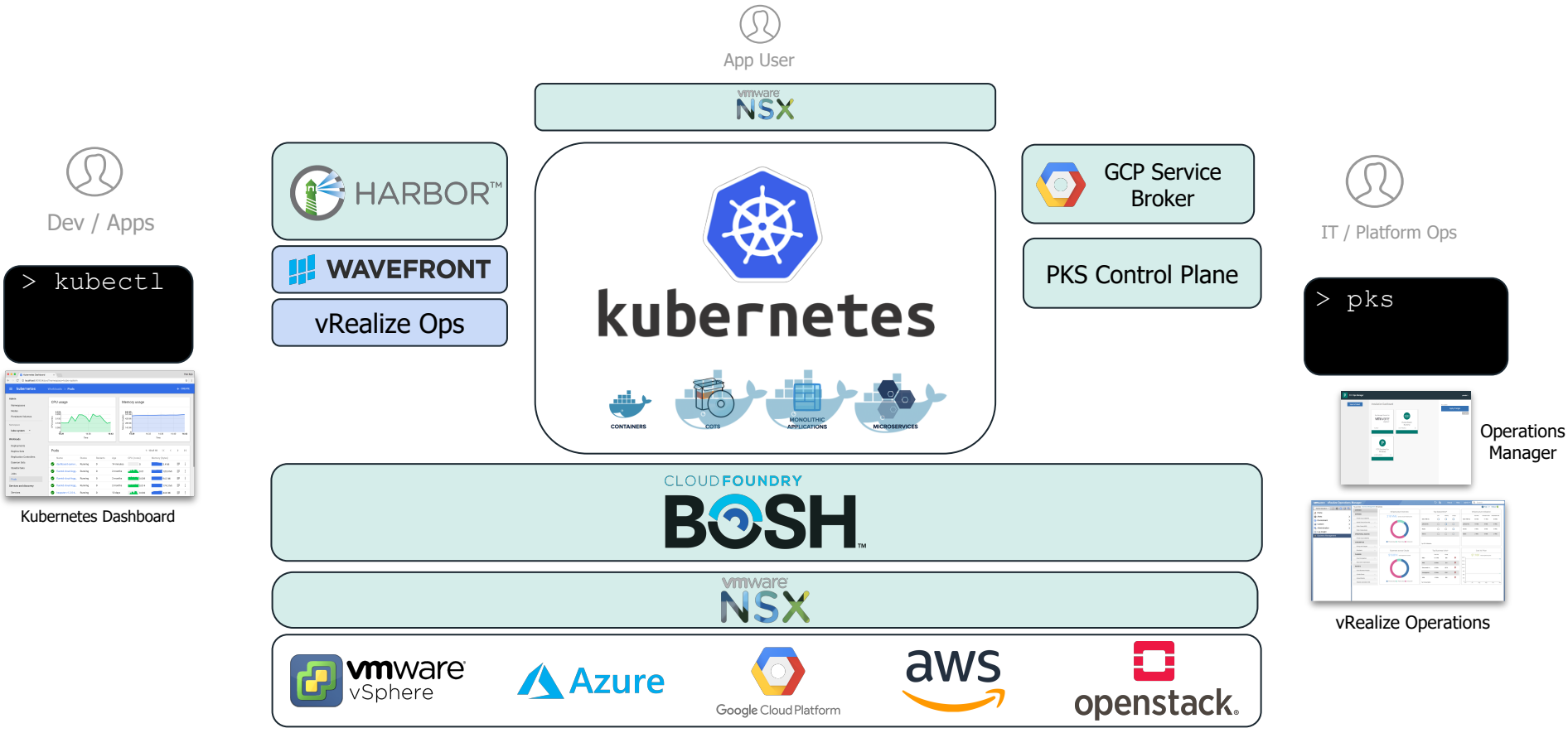
Storage

Networking

Kubernetes in Production



Enterprise PKS



The Platform Promise for all your workloads

Developer Productivity

- Accelerate feedback loops by improving delivery velocity
- Focus on applications, not infrastructure
- Give developers the tools and frameworks to build resilient apps

Operational Efficiency

- Employ 300:1 developer to operator ratio
- Perform zero-downtime upgrades
- Runs the same way on every public/private cloud

Comprehensive Security

- Adopt a defense-in-depth approach
- Continuously update platforms to limit threat impact
- Apply the 3 R's → repair, repave, rotate

High Availability

- Run platforms that stays online under all circumstances
- Scale up and down, in and out, through automation
- Deploy multi-cloud resilience patterns


T
H
A
N
K
Y
O
U