

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

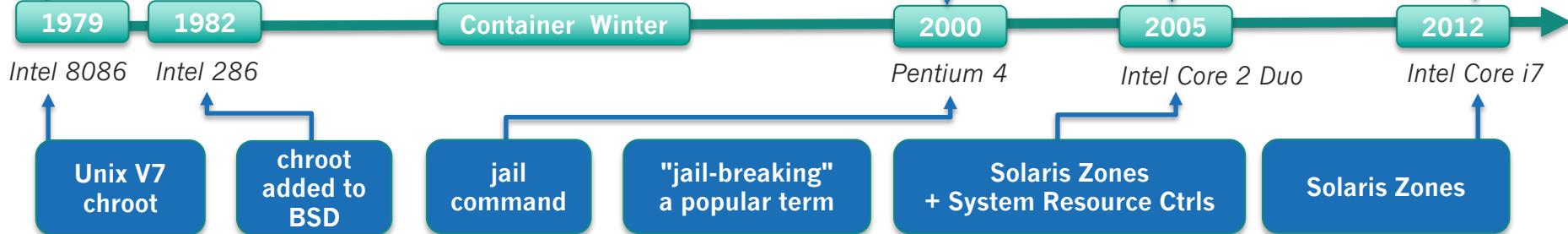
Chronology – Orchestration & Containers

Unix ***chroot*** changes the apparent root directory for the current running process and its children. A program that is run in such a modified environment cannot name files outside the designated directory tree. The modified environment is called a "***chroot jail***".

Sun Microsystems releases **Solaris v10 Containers**: "chroot on steroids". A Solaris Container combines system resource controls and the boundary separation provided by zones. **Zones act as completely isolated virtual servers** within a single operating system instance. Consolidation of multiple sets of application services onto one system, each set isolated as virtual server containers, allowed system administrators to reduce cost and provide most of the same protections of separate machines on a single machine. Features: Zone snapshots and ZFS cloning.

The need for virtualization led **FreeBSD** to expand the chroot concept with the **jail command** for: (a) **virtualization** [files, processes, users] (b) **isolation** (c) **ease of delegation of super-user access** without handing over full system control.

Oracle drops the term "**container**"



Chronology – Orchestration & Containers



Open Virtuozzo – 2005 – a modified Linux kernel, **OpenVZ** containers provide virtualization, isolation, resource management, and check-pointing.

Google's 2006 Process
Containers become cgroups
(control-groups) and are merged
into the kernel of 2.6.24 in 2007.
cgroups provides (a) resource
[CPU, Mem, Disk I/O, Network]
limitation (b) prioritization (c)
accounting (d) control [freeze,
checkpoint, restart]

LXC (LinuX Containers) is an OS-level virtualization environment for running multiple containers on a single Linux host. LXC hits the kernel and is the 1st modern implementation using cgroups and namespaces. No patches.

OS virtualization implemented as Linux kernel patches. 2001~2006

Java Cloud Foundry on EC2 is bought by SpringSource



2000

2005

2006

2008

2009

Solaris Zones + System Resource Ctrl

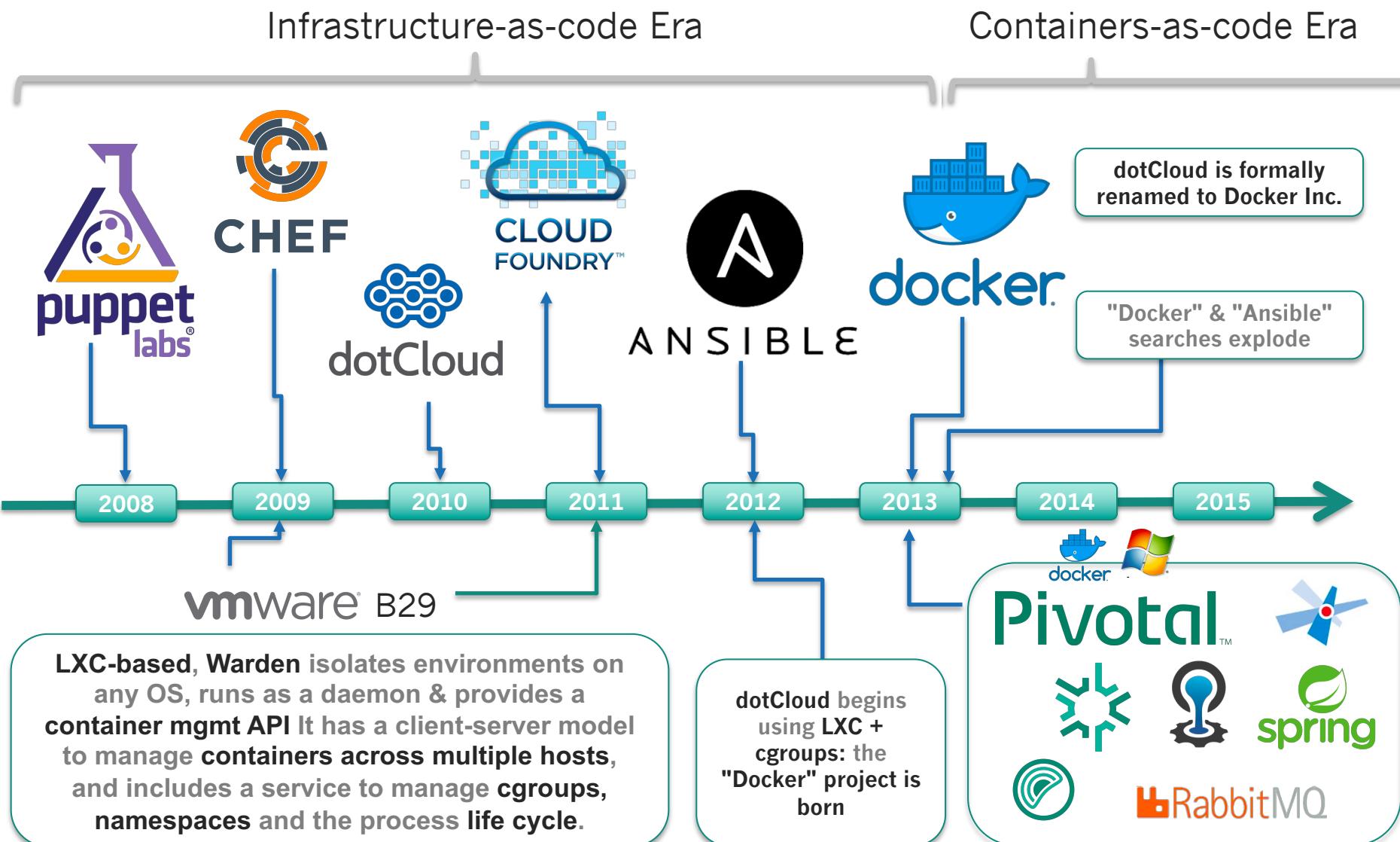
Amazon EC2 Beta - 1st come 1st serve basis. Limited VM options, limited management and monitoring features.

AWS – 12 types of VMs, AZs, choice of Kernels, Windows, SQL Server, Nitro based on KVM Hypervisor.

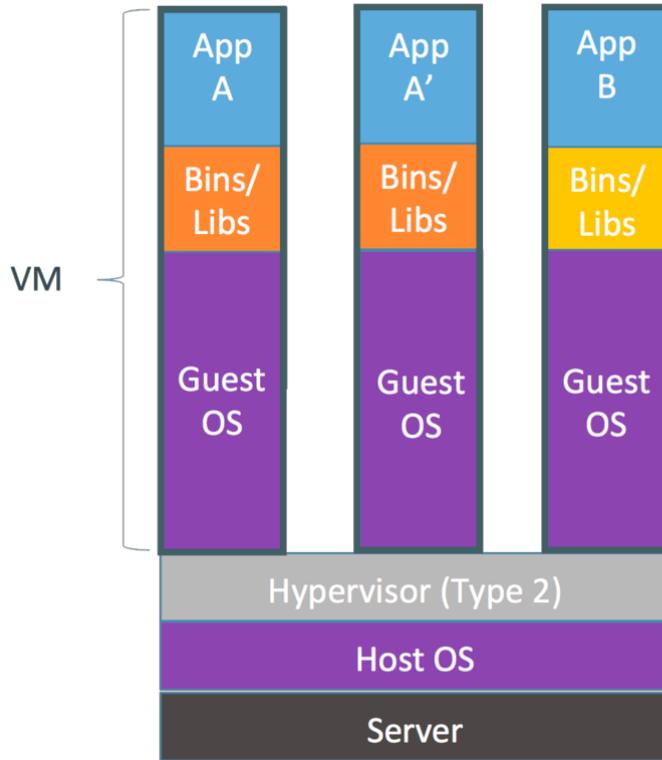
VMware starts the B29 project: DEA, Open Source, Ruby, Apps & Services, not VMs & Servers, distributed, multi-cloud



Chronology – Orchestration & Containers

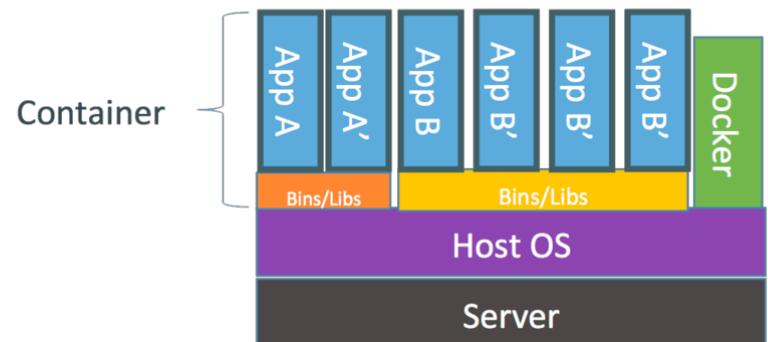


App Containers vs. VMs (Virtual Machines)

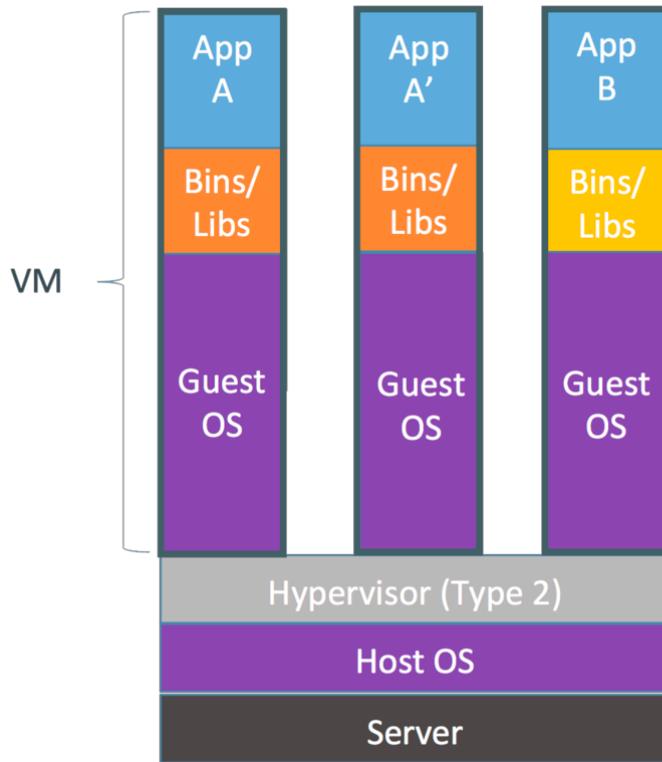


Containers are isolated, but share OS and, where appropriate, bins/libraries

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

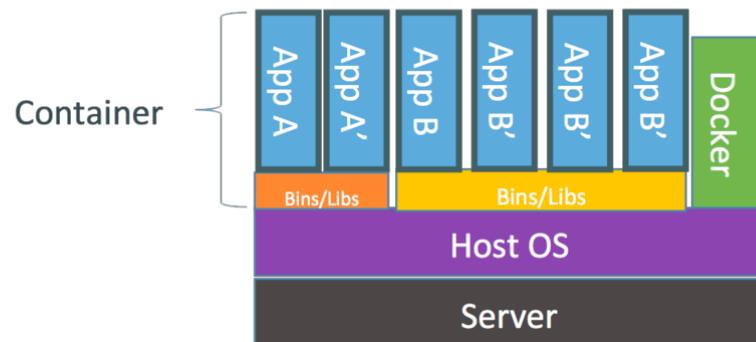


App Containers vs. VMs (Virtual Machines)



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

- Node.js
- Postgres
- Nginx

Node.js

Postgres

Nginx

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

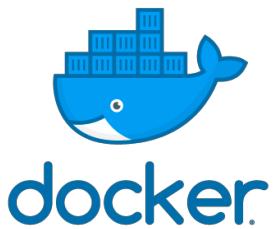
- Meant to run a single service
- Layered filesystems
- Built on top of OS container technologies
- Examples: Docker, Rocket, RunC

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 ...

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



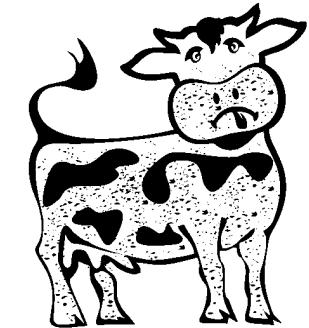
also uses Namespaces and cgroups

docker

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 of 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)?



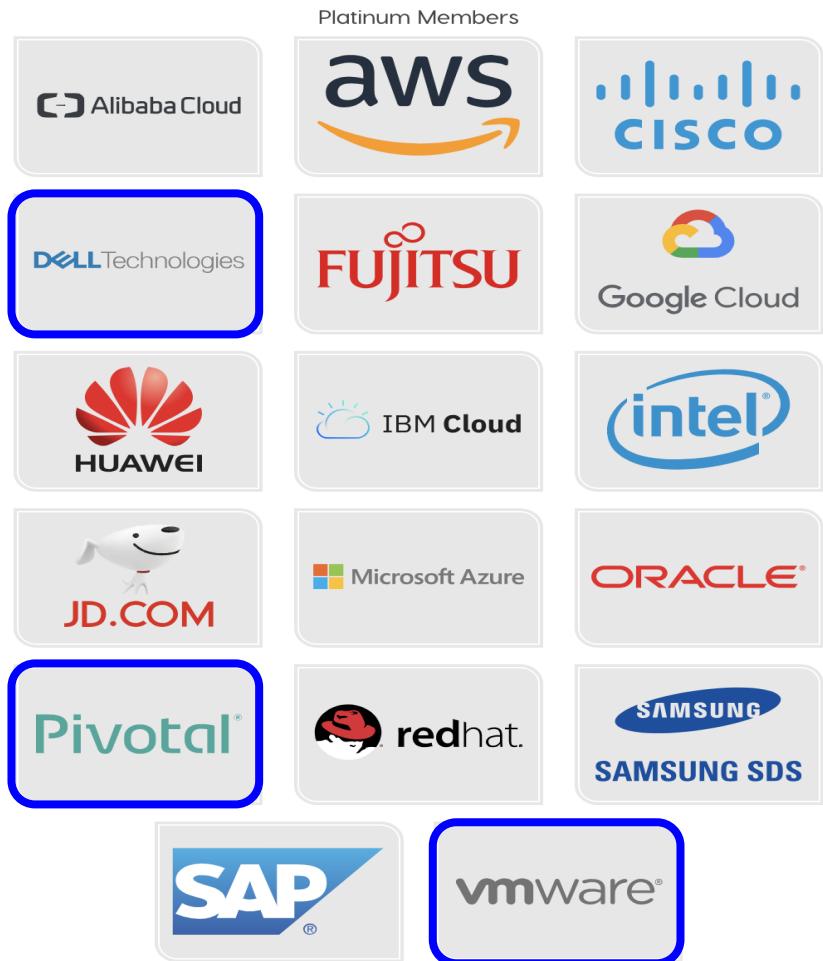
- a system for managing application containers
- a container orchestration platform: deploy, scale & operate
- κυβερνήτης: greek for "governor", "helmsman" or "captain"

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.



Who are the stewards of K8s?

- 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?

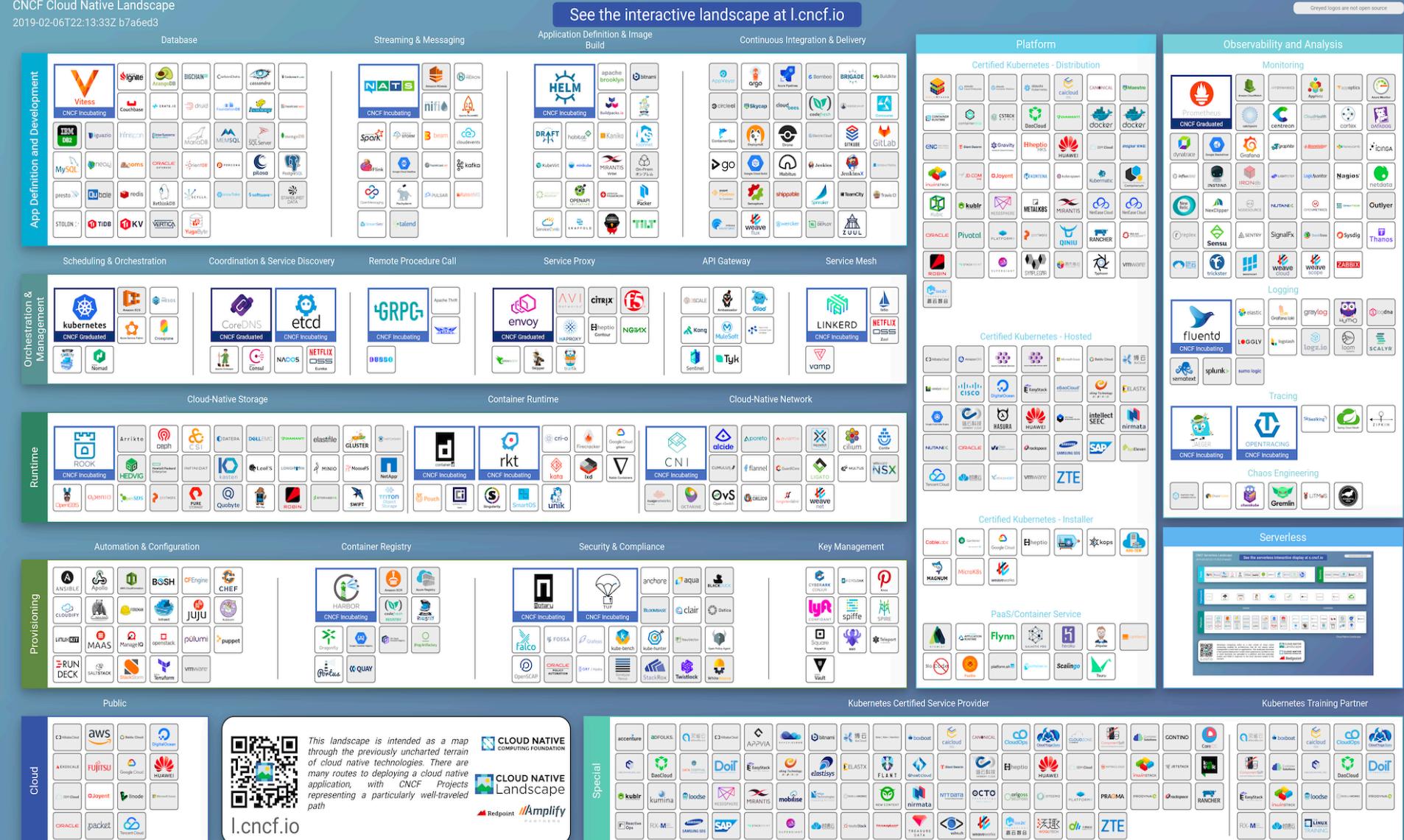
- to build sustainable ecosystems and foster a community around a constellation of ***high-quality projects*** that orchestrate ***containers*** as part of a ***microservices architecture***.



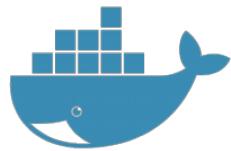
How complex is the Cloud Native space?

CNCF Cloud Native Landscape
2019-02-06T22:13:33Z b7a6ed3

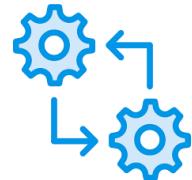
Greyed logos are not open source



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



CONTAINERS



Batches



EVENT-DRIVEN
FUNCTIONS



MICROSERVICES

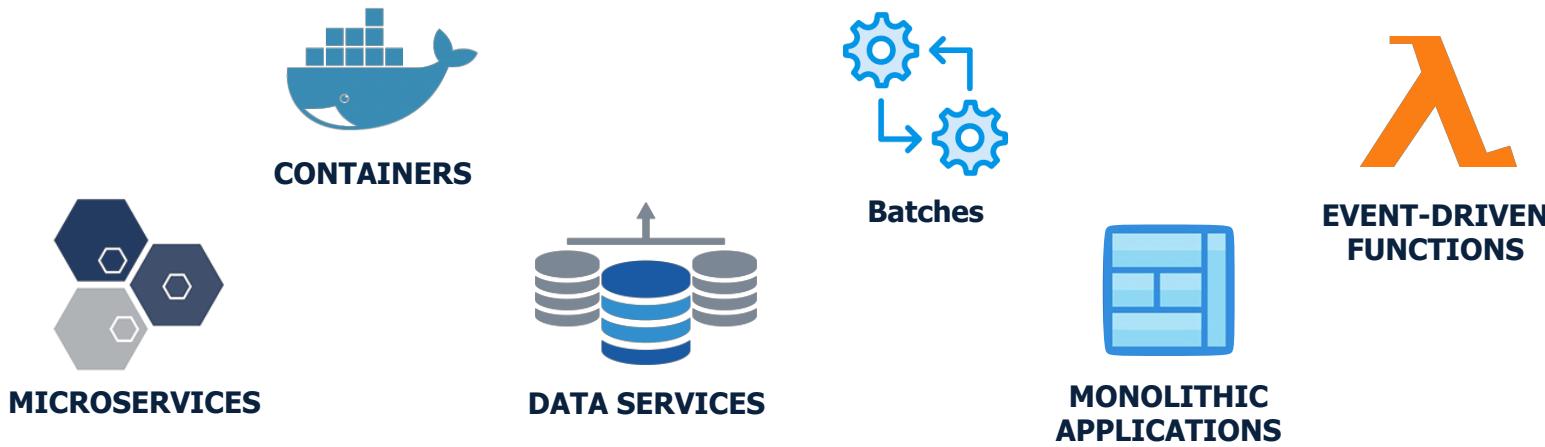


DATA SERVICES

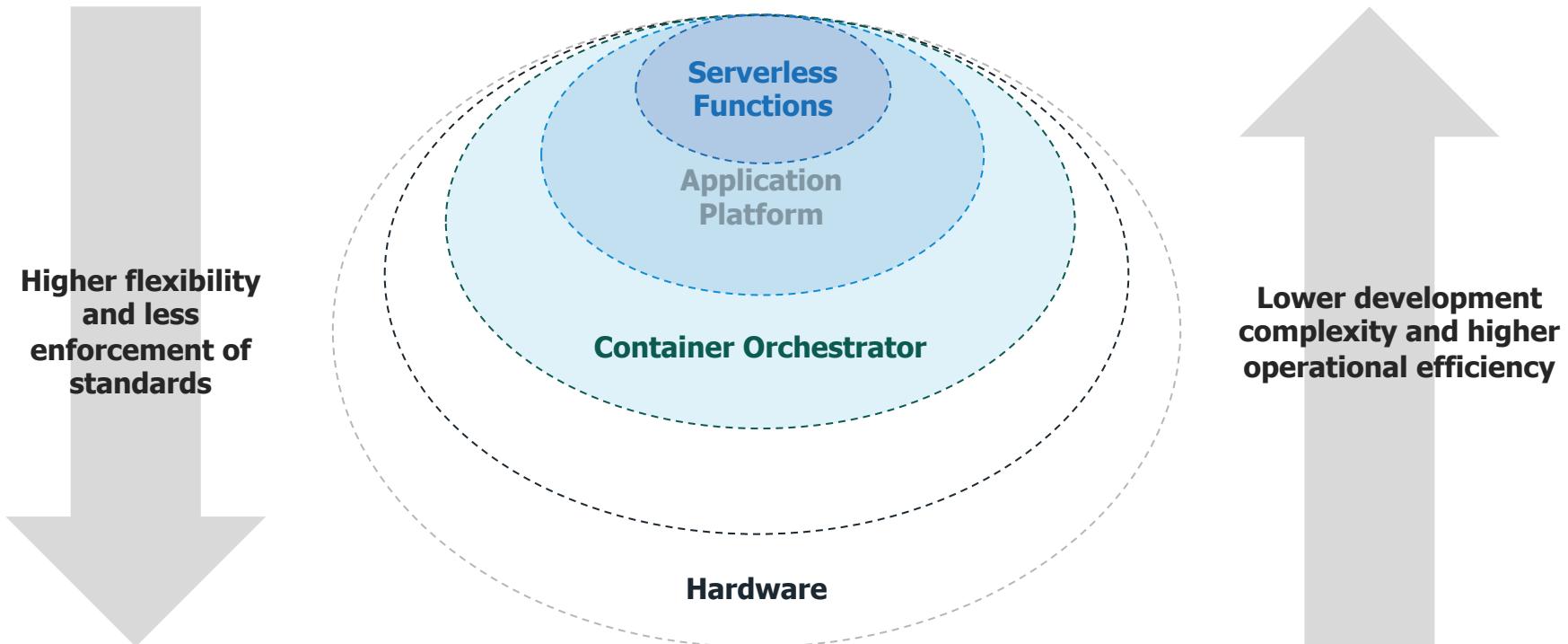


MONOLITHIC
APPLICATIONS

Their goal: pick the right runtime for each workload

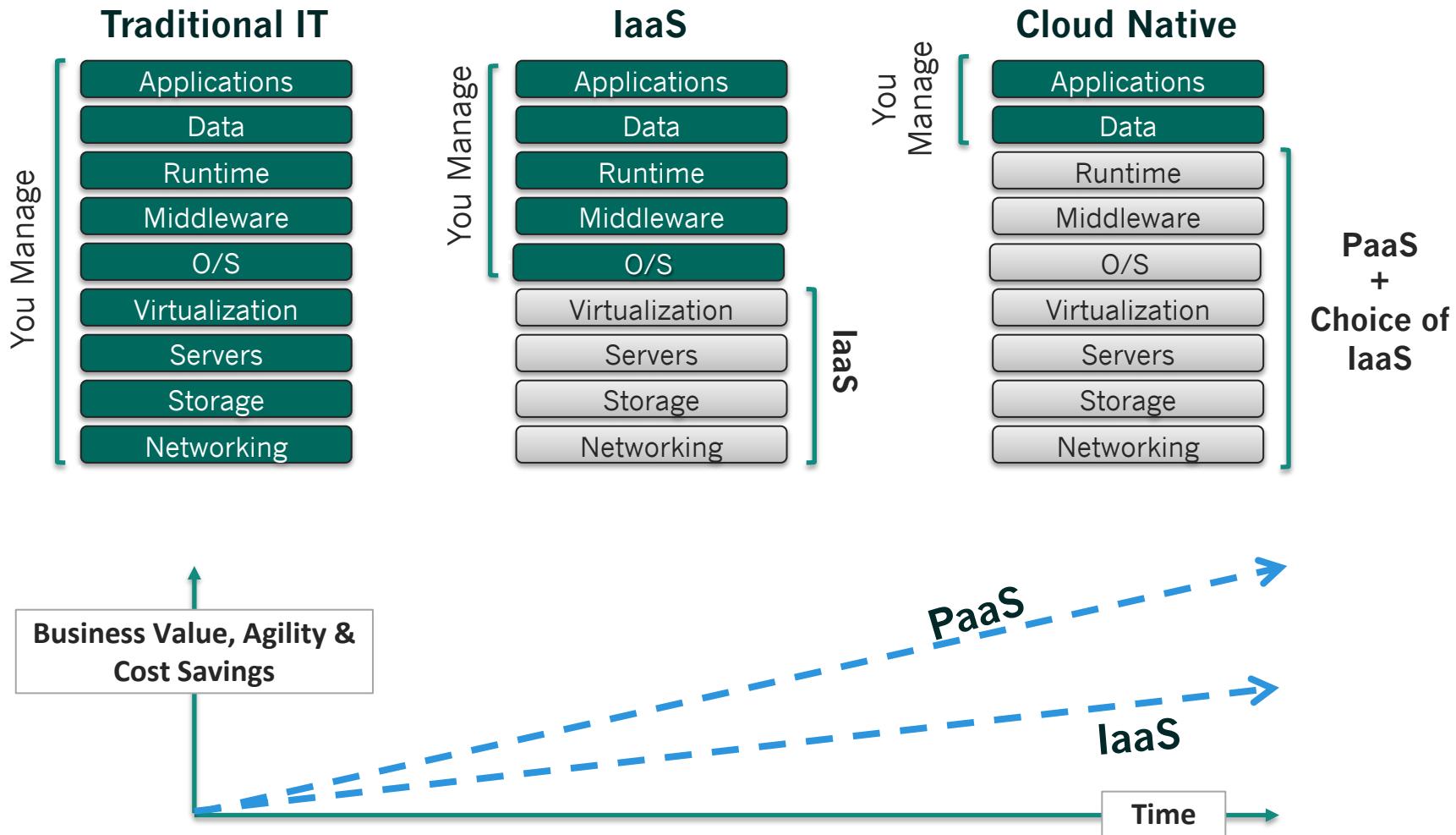


Use the highest abstraction possible



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



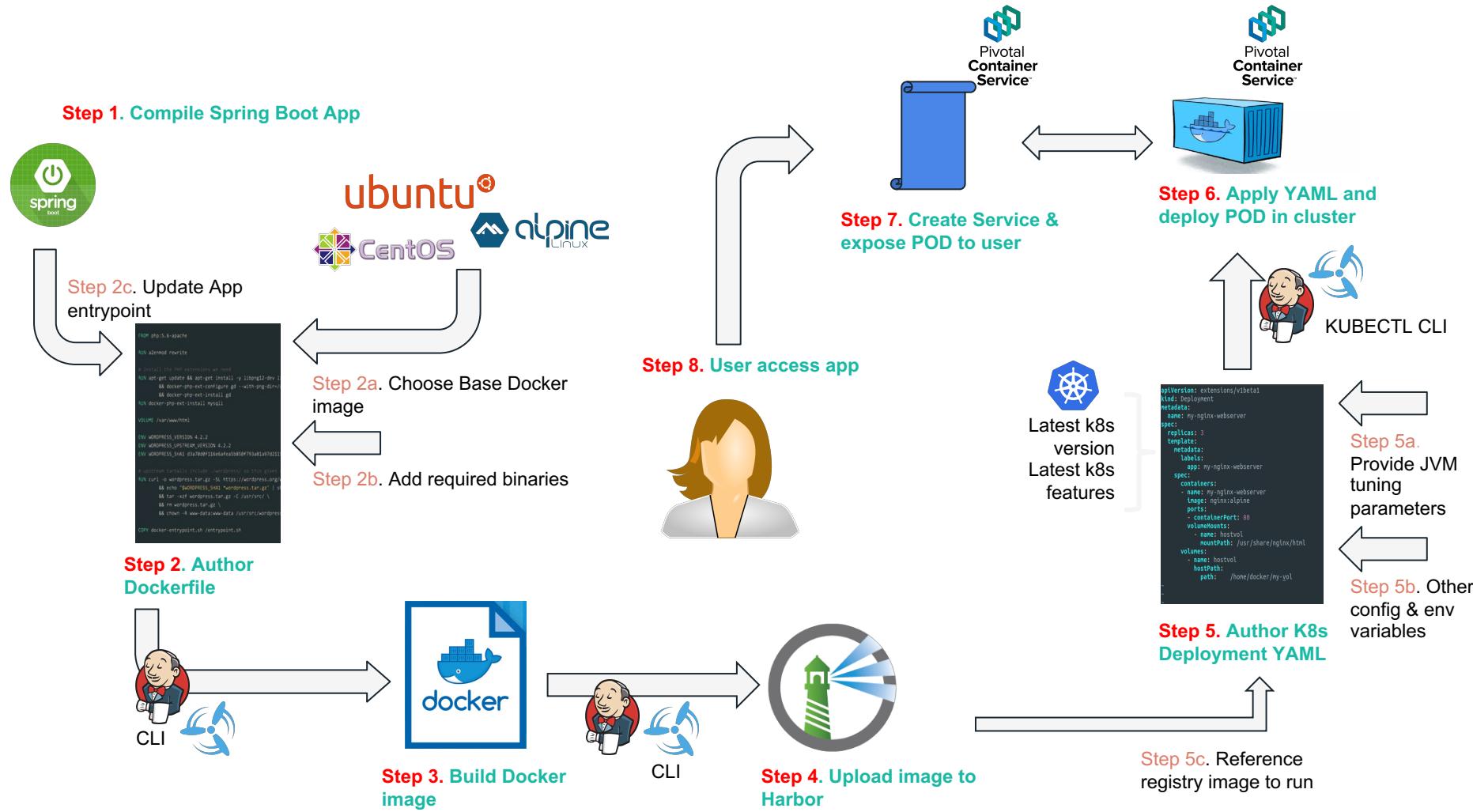
Key Insight about Cloud Computing

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

Paul Maritz – Chairman, Pivotal

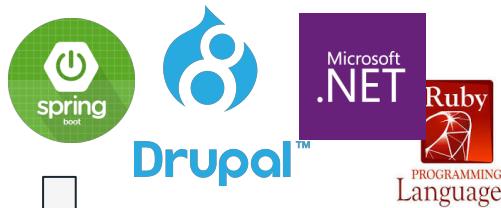


Spring Boot K8s deployment - CaaS



Spring Boot or .NET Core PaaS deployment

Step 1. Compile App

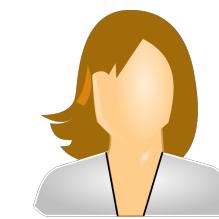
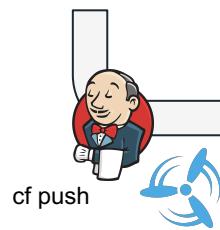


Step 2a. Update App details



```
FROM php:5.6-apache
RUN a2enmod rewrite
# install the PHP extensions we need
RUN apt-get update && apt-get install -y libpng2-dev \
    && docker-php-ext-configure gd --with-png-dir=/usr/include/libpng2 \
    && docker-php-ext-install gd
RUN docker-php-ext-install mysqli
VOLUME /var/www/html
ENV WORDPRESS_VERSION 4.2.2
ENV WORDPRESS_UPDATE_VERSION 4.2.2
ENV WORDPRESS_SHA1 d3a780d116edeaef08bf793a81a97d211
# upstream tarball include --version=0 in this gives
RUN curl -O wordpress.tar.gz -S https://wordpress.org/
    && echo "#WORDPRESS_SHA1 \"wordpress.tar.gz\" | " \
    && tar -xzf wordpress.tar.gz -C /usr/src/ \
    && rm wordpress.tar.gz \
    && chown -R www-data:www-data /usr/src/wordpress
COPY docker-entrypoint.sh /entrypoint.sh
```

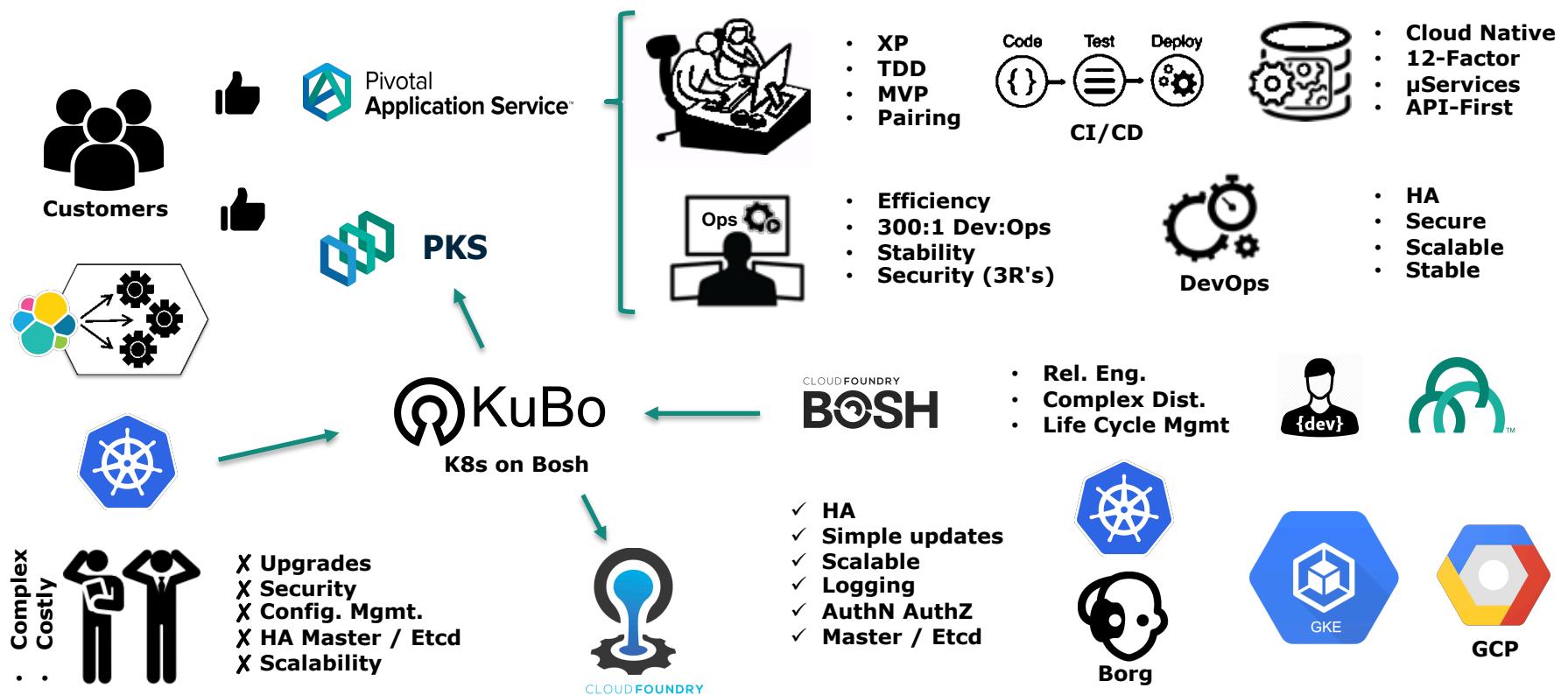
Step 2. Author Manifest file



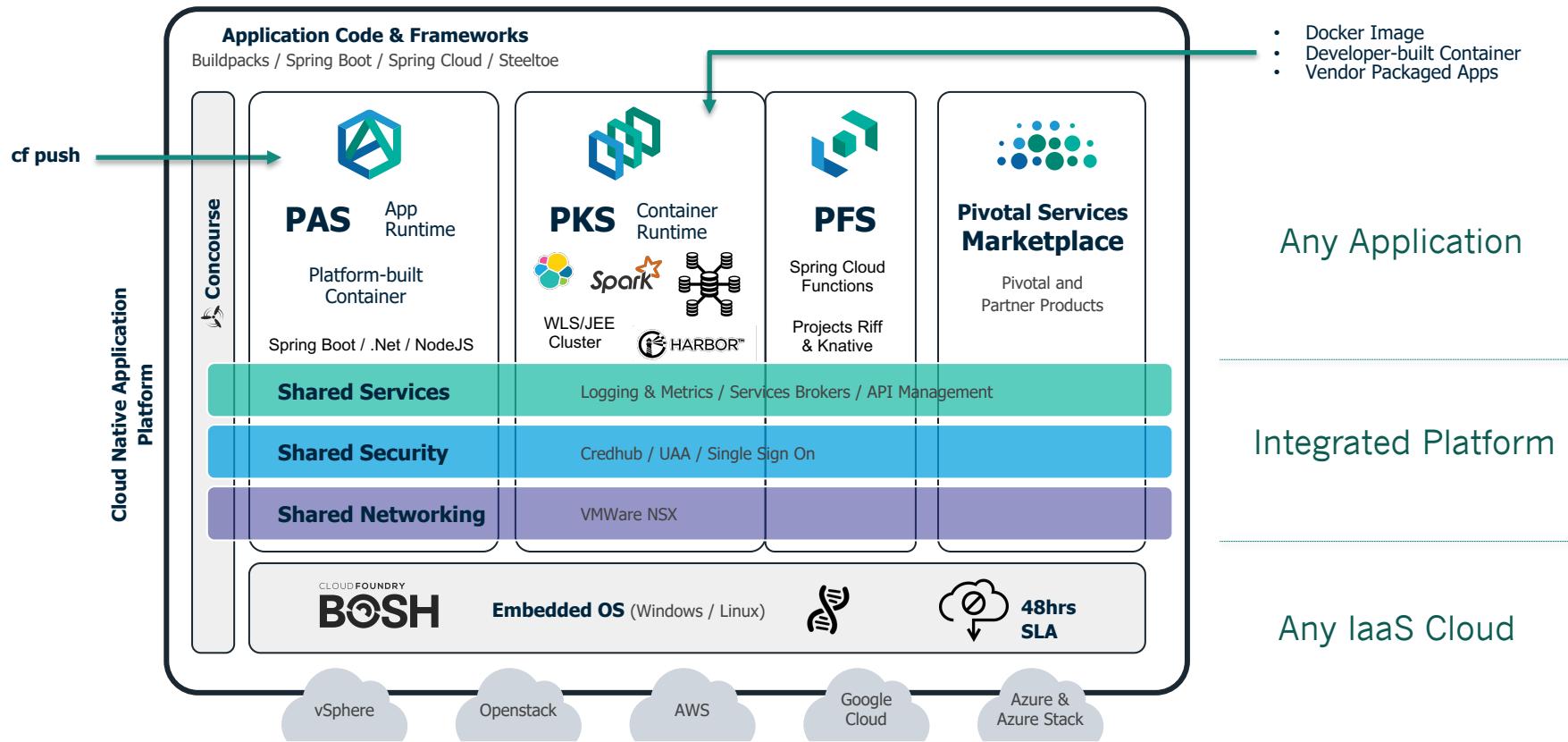
Step 3. User access app



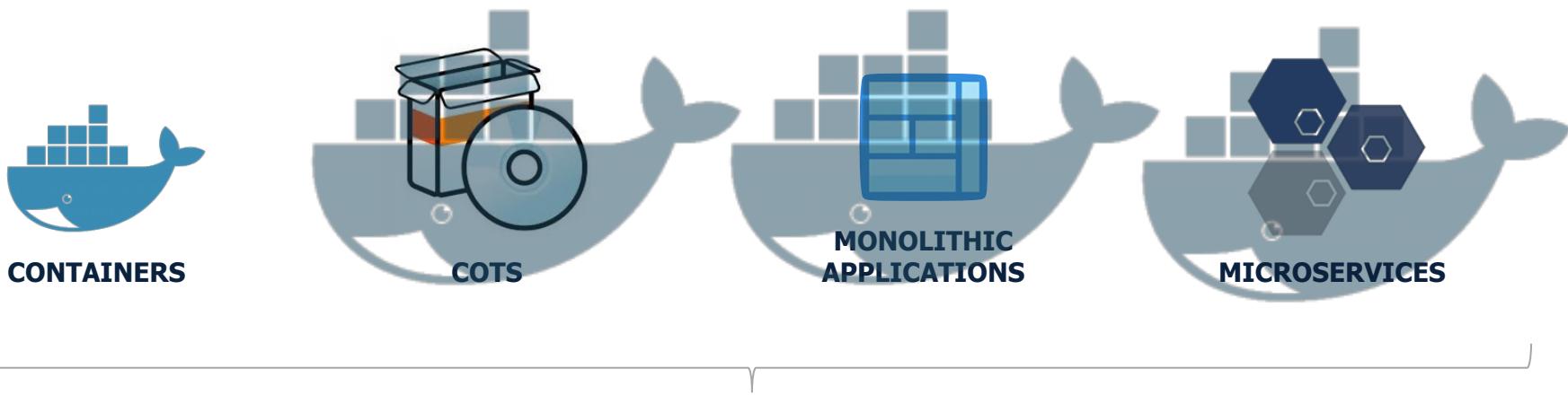
CaaS and PaaS



CaaS, PaaS, FaaS and OSB API Services

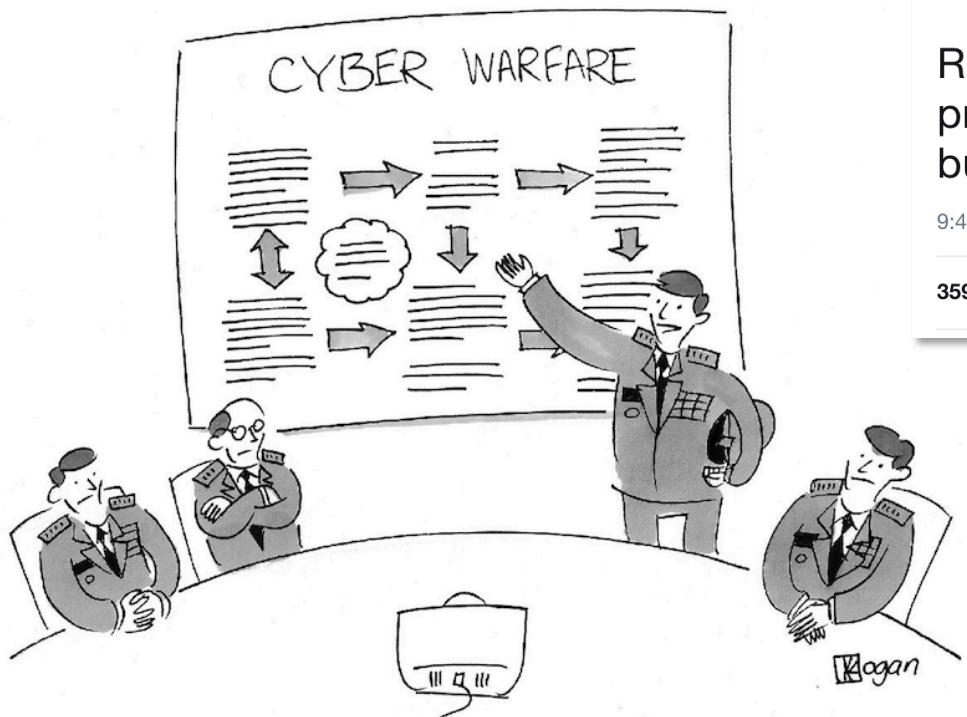


Containerized workloads



Kubernetes

K8s in the Enterprise



"First, we inundate them with quarterly Kubernetes releases."



Kelsey Hightower

@kelseyhightower

Follow

Perception: I'm using pure Kubernetes; I don't need a platform.

Reality: Everything you do above kubectl is proof you need a platform and you're actually building one.

9:43 AM - 24 Feb 2019

359 Retweets 1,157 Likes



From 1.4 to 1.5

12th December 2016

// days

From 1.5 to 1.6

We didn't realize the amount of traffic!
Dev Team: "Can you scale it for us?"

Launch date is next week!

Release Team: "How can we expose our services?"



Just got the latest CVE report in!
Security Team: "Can you patch the environment today?"



From 1.12 to 1.13

From 1.13 to 1.14



Heartbleed

Meltdown

Spectre

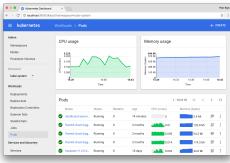
Basic Kubernetes Deployment



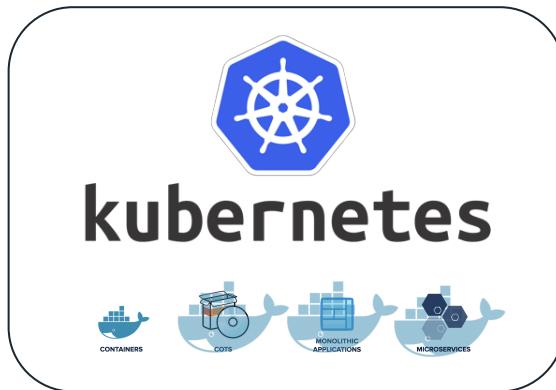
App User



Dev / Apps



Kubernetes Dashboard



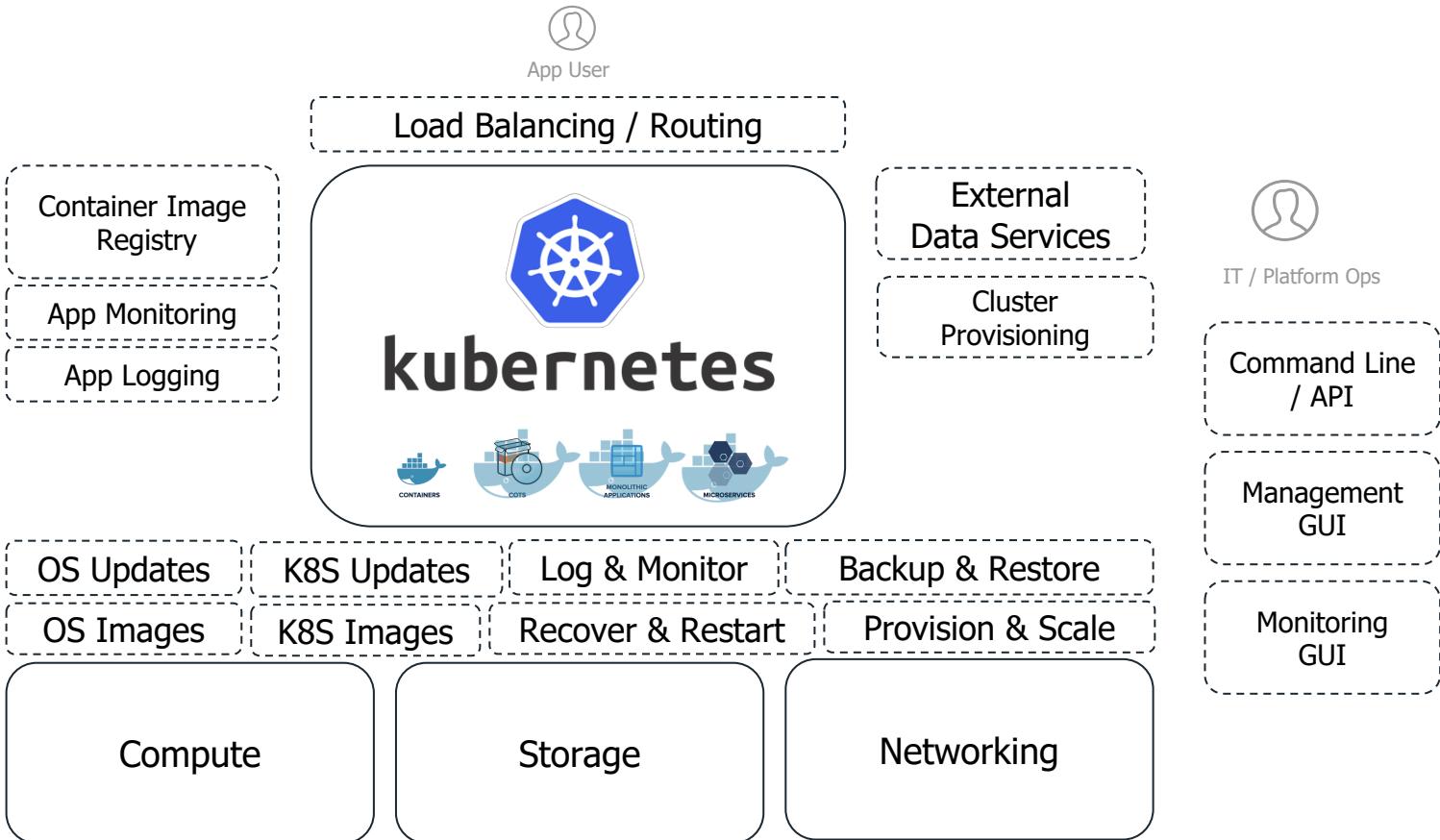
IT / Ops

Compute

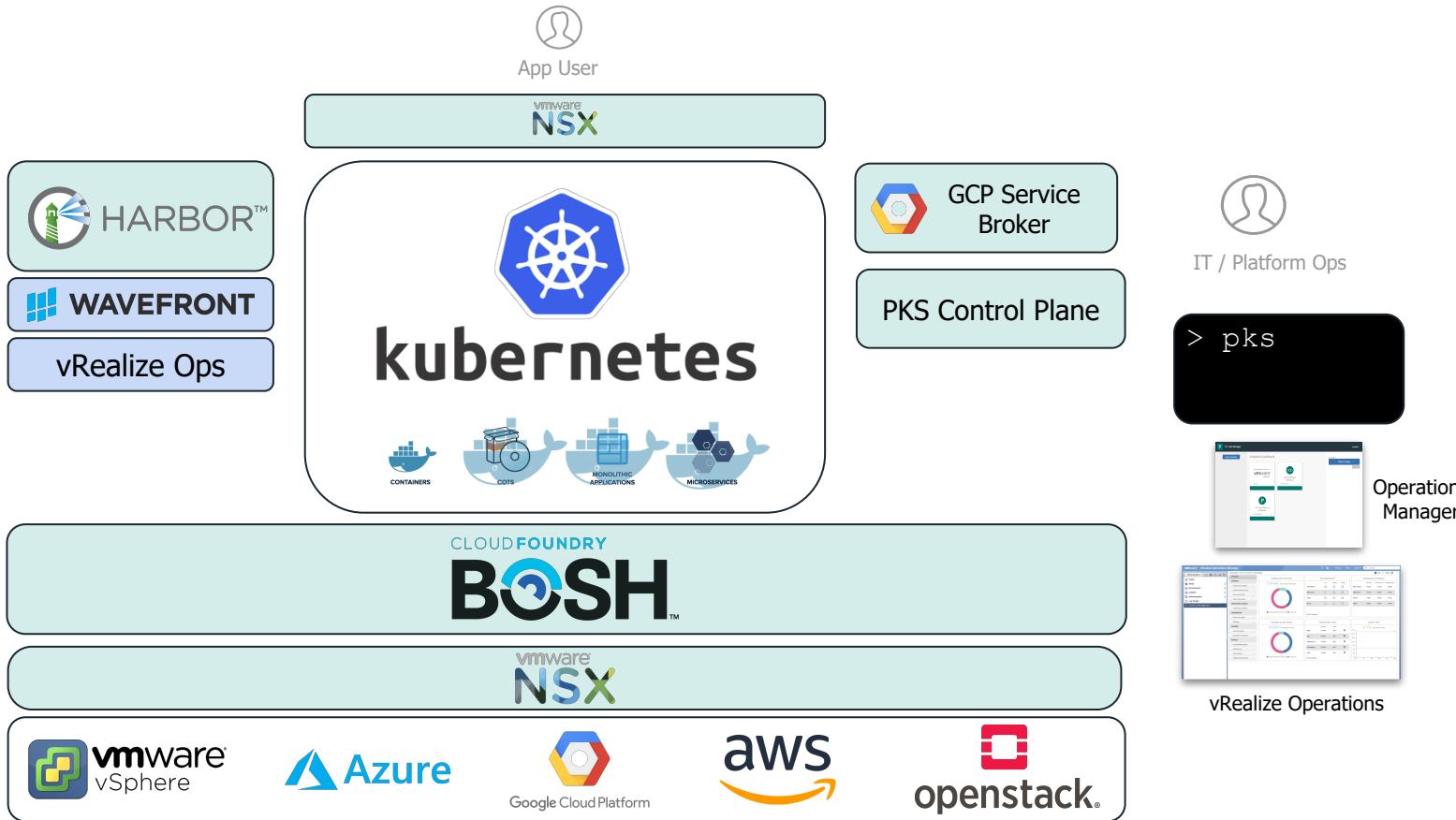
Storage

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 stay online under all circumstances
- Scale up and down, in and out, through automation
- Deploy multi-cloud resilience patterns

