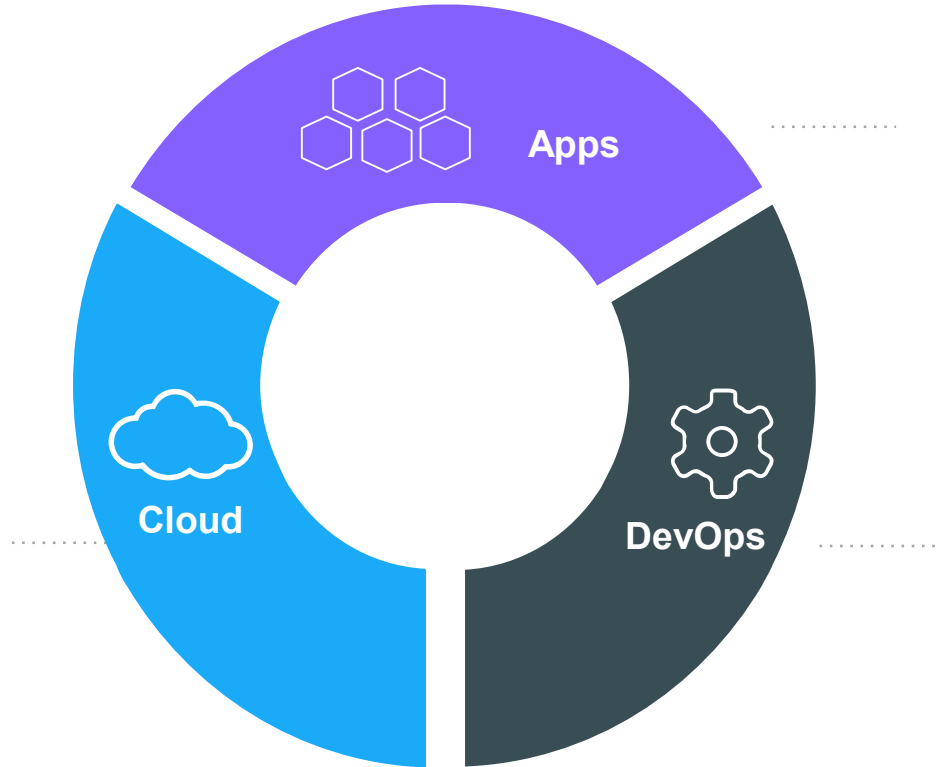


Introduction to Docker



The IT Landscape is Changing



Movement in the cloud

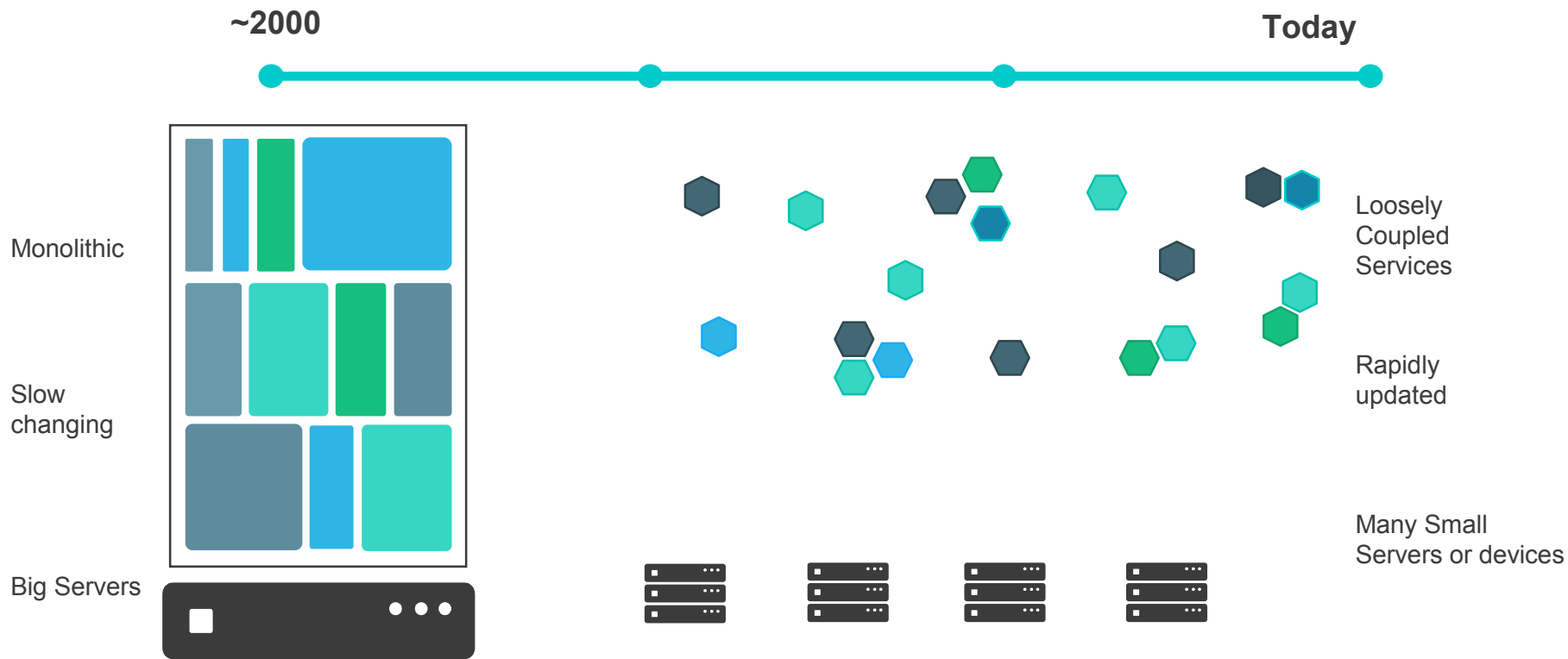


Migrate workloads to cloud

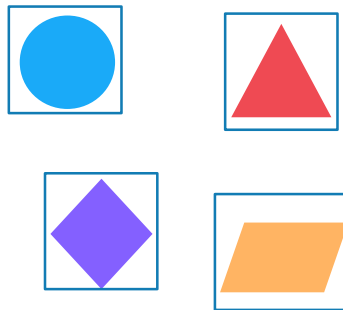
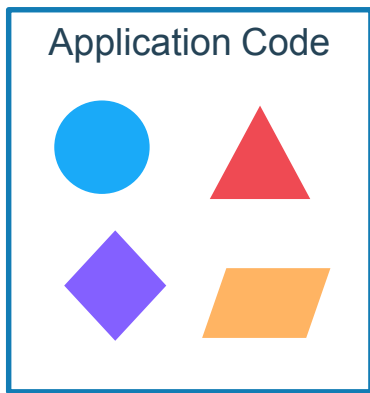
Portability across environments

Want to avoid cloud vendor lock-in

Applications are transforming



Application Modernization



Developer Issues:

- Minor code changes require full re-compile and re-test
- Application becomes single point of failure
- Application is difficult to scale

Microservices: Break application into separate operations

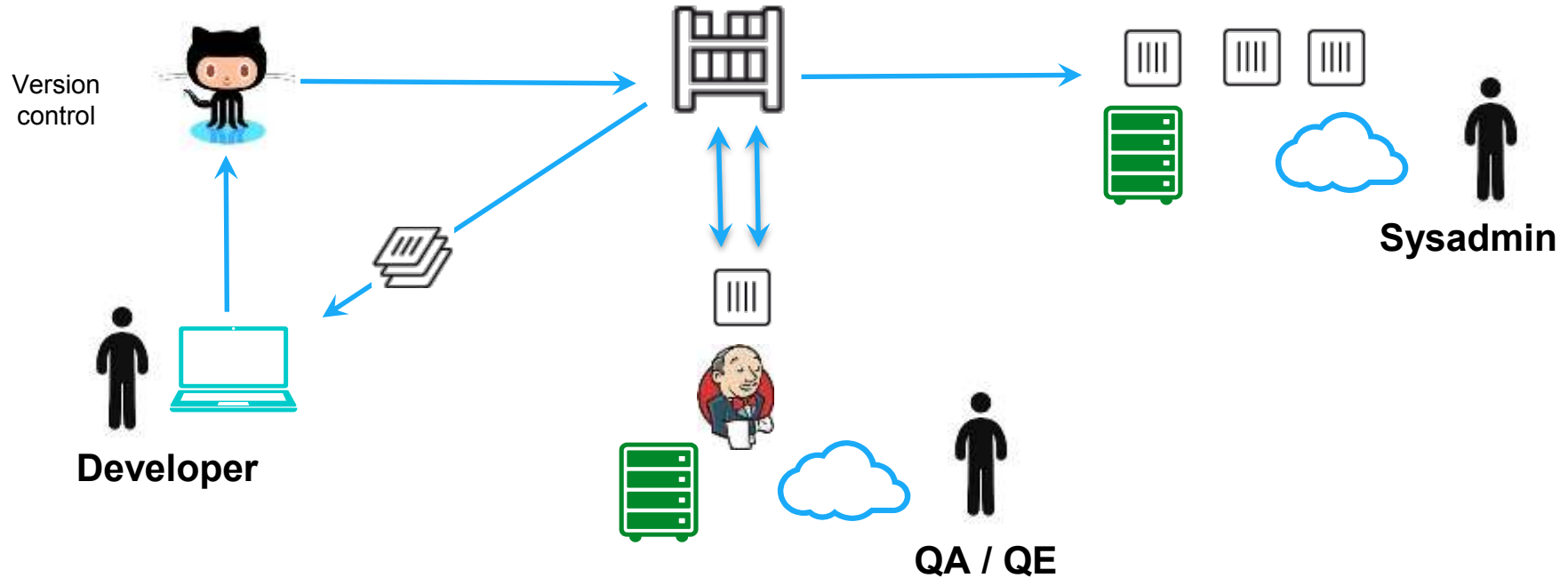
12-Factor Apps: Make the app independently scalable, stateless, highly available by design

Continuous Integration and Delivery

1. Development

2. Test

3. Stage / Production



Tug of War Between Developers and Ops



Developers





- Freedom to create and deploy apps fast
- Define and package application needs







IT Operations

- Quickly and flexibly respond to changing needs
- Standardize, secure, and manage

Organizations Must Deal with Diverse Technology

	Bare Metal
	On Premises
	Linux
	Traditional



	Virtual
	Cloud
	Windows
	Microservices

...and Diverse Organizations



Developers

- Freedom to create and deploy apps fast
- Define and package application needs






IT Operations

- Quickly and flexibly respond to changing needs
- Standardize, secure, and manage

The Myth of Bi-Modal IT

	MICROSERVICES	TRADITIONAL APPS
Cloud or New Infrastructure	You are either here..	
Old Infrastructure		...or here

Enabling a Journey

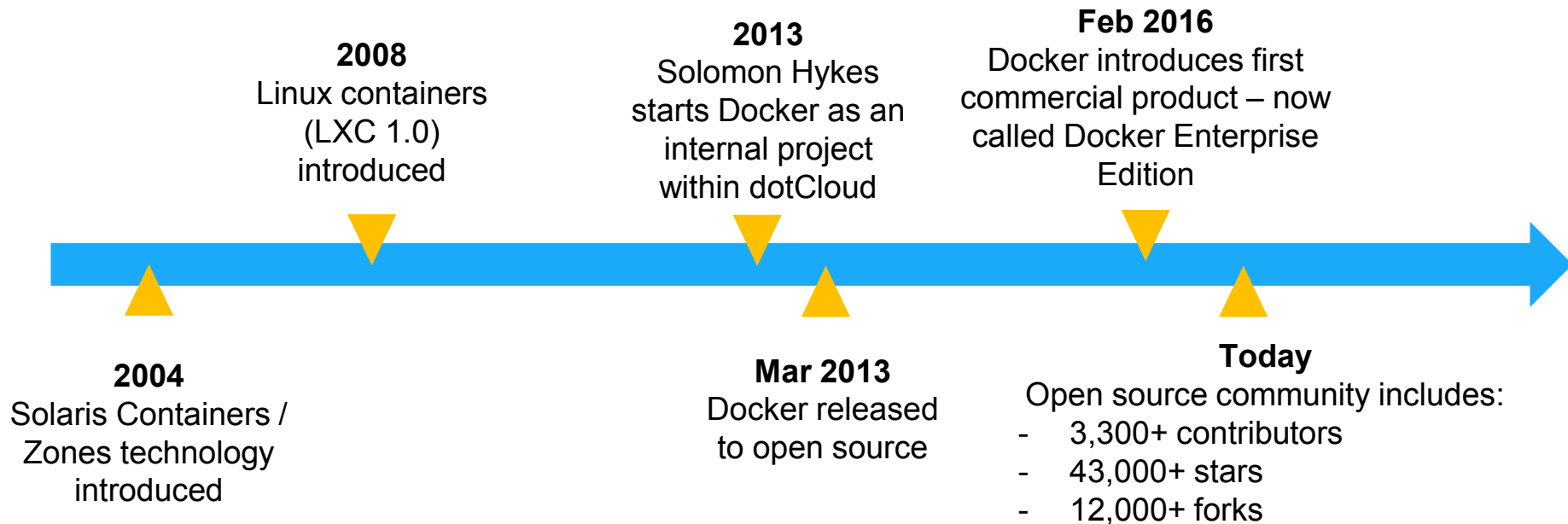
	MICROSERVICES	AGILE TRADITIONAL APPS	TRADITIONAL APPS
Cloud or New Infrastructure			
Old Infrastructure			

...that is past AND future proof

Docker and Container Overview



History of Docker



Incredible adoption in just 4 years



14M

Docker
Hosts



900K

Docker
apps



77K%

Growth in
Docker job
listings



12B

Image pulls
Over 390K%
Growth



3300

Project
Contributors

The Docker Family Tree



Open source **framework** for assembling core components that make a container platform

Intended for:
Open source contributors +
ecosystem developers



Subscription-based, commercially supported **products** for delivering a secure software supply chain

Intended for:
Production deployments +
Enterprise customers



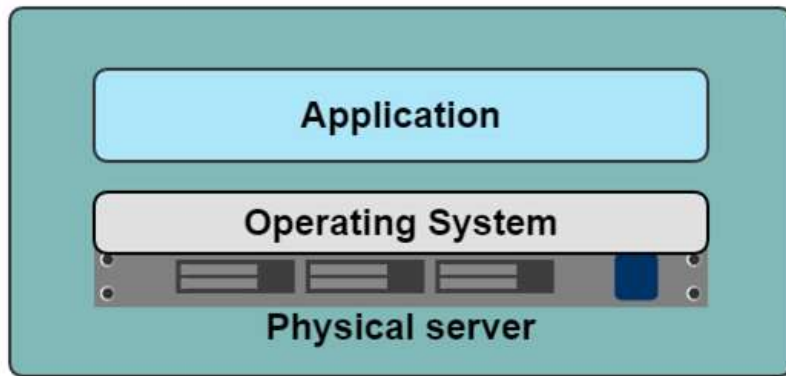
Free, community-supported **product** for delivering a container solution

Intended for:
Software dev & test

A History Lesson

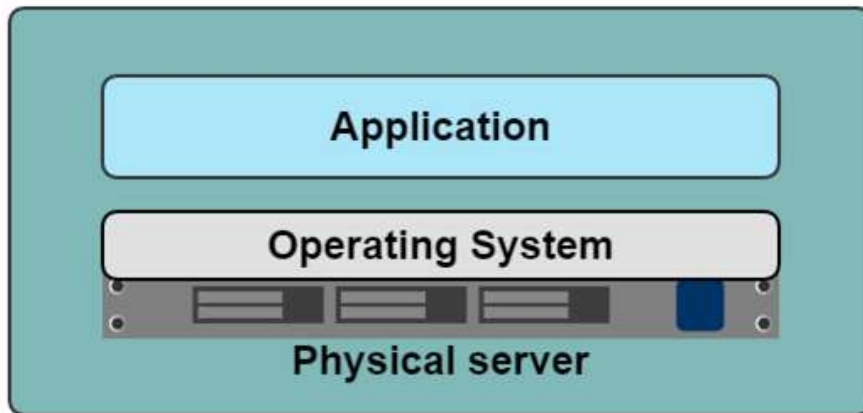
In the Dark Ages

One application on one physical server



Historical limitations of application deployment

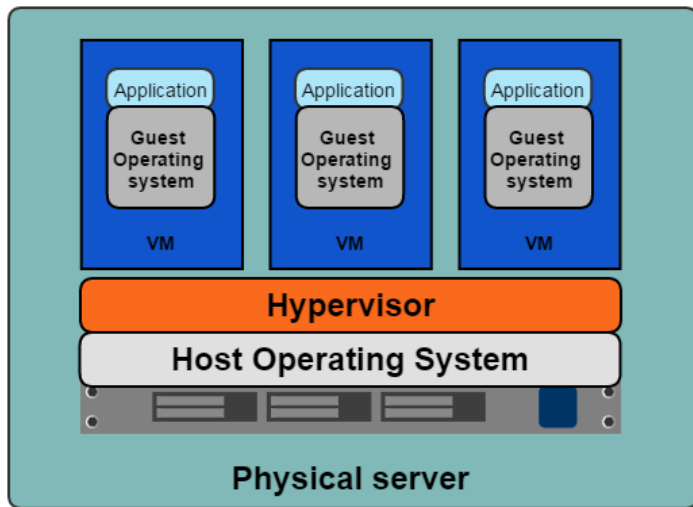
- Slow deployment times
- Huge costs
- Wasted resources
- Difficult to scale
- Difficult to migrate
- Vendor lock in



A History Lesson

Hypervisor-based Virtualization

- One physical server can contain multiple applications
- Each application runs in a virtual machine (VM)



Benefits of VMs

- Better resource pooling
 - One physical machine divided into multiple virtual machines
- Easier to scale
- VMs in the cloud
 - Rapid elasticity
 - Pay as you go model

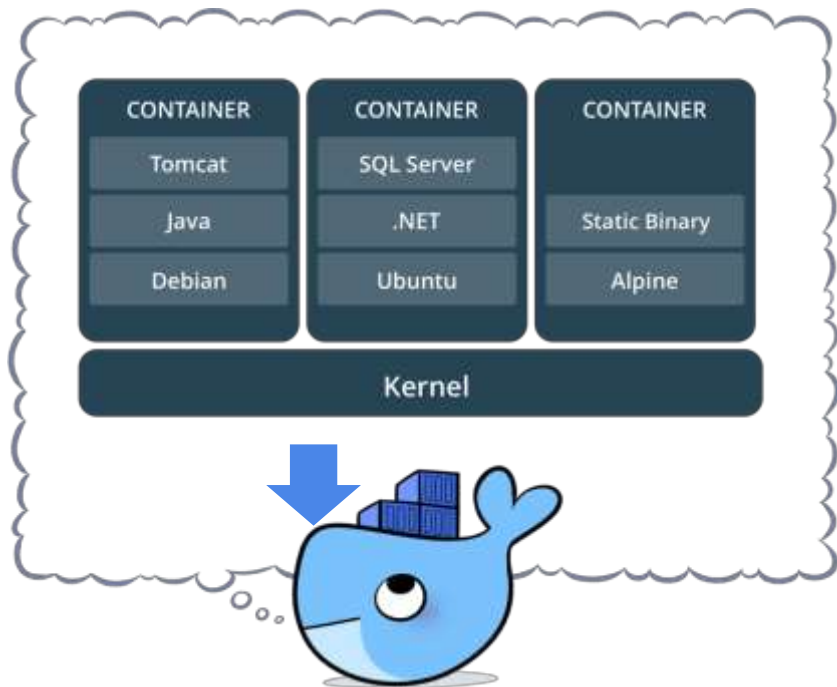


Limitations of VMs

- Each VM stills requires
 - CPU allocation
 - Storage
 - RAM
 - An entire guest operating system
- The more VMs you run, the more resources you need
- Guest OS means wasted resources
- Application portability not guaranteed



What is a container?

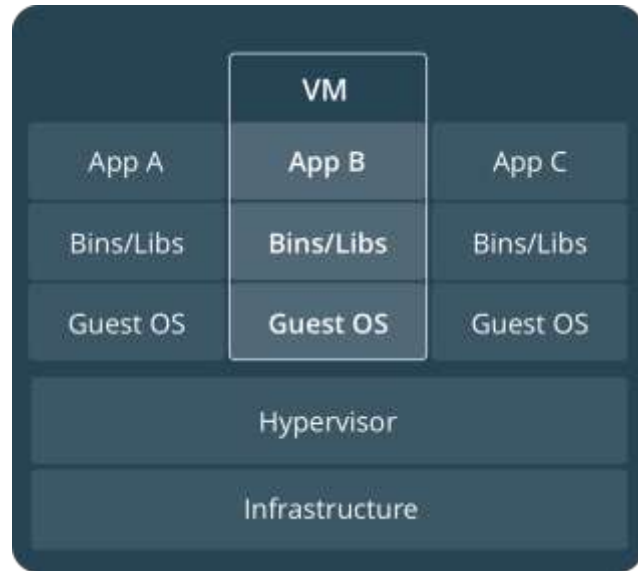


- Standardized packaging for software and dependencies
- Isolate apps from each other
- Share the same OS kernel
- Works with all major Linux and Windows Server

Comparing Containers and VMs

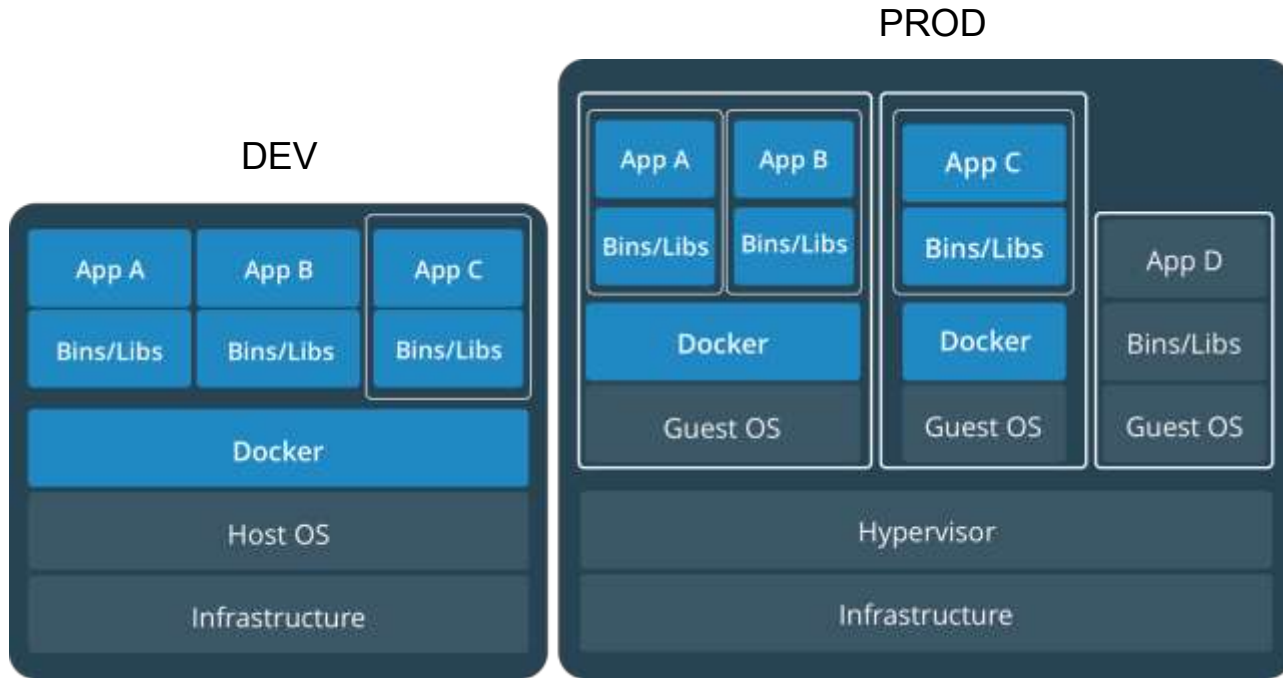


Containers are an app
level construct



VMs are an infrastructure level
construct to turn one machine
into many servers

Containers and VMs together



Containers and VMs together provide a tremendous amount of flexibility for IT to optimally deploy and manage apps.

Key Benefits of Docker Containers

Speed

- No OS to boot = applications online in seconds

Portability

- Less dependencies between process layers = ability to move between infrastructure

Efficiency

- Less OS overhead
- Improved VM density

Container Solutions & Landscape



Docker Basics



Image

The basis of a Docker container. The content at rest.



Container

The image when it is 'running.' The standard unit for app service



Engine

The software that executes commands for containers. Networking and volumes are part of Engine. Can be clustered together.



Registry

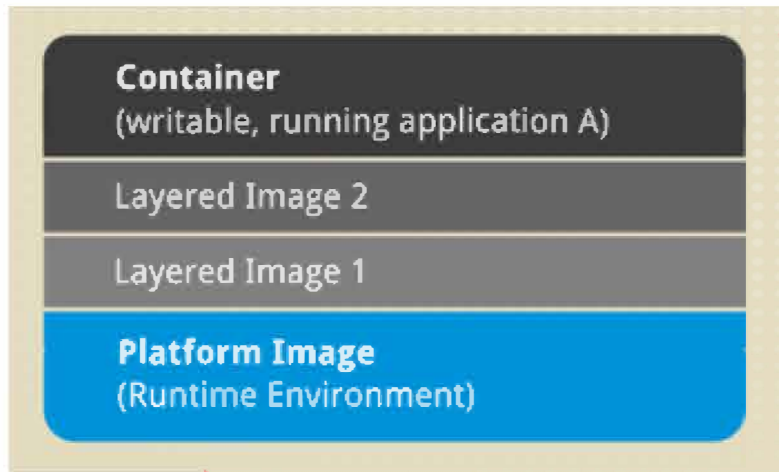
Stores, distributes and manages Docker images



Control Plane

Management plane for container and cluster orchestration

Image Layering



An application sandbox.

- Each container is based on an image that holds necessary config data.
- When you launch a container from an image, a writable layer is added on top of this image

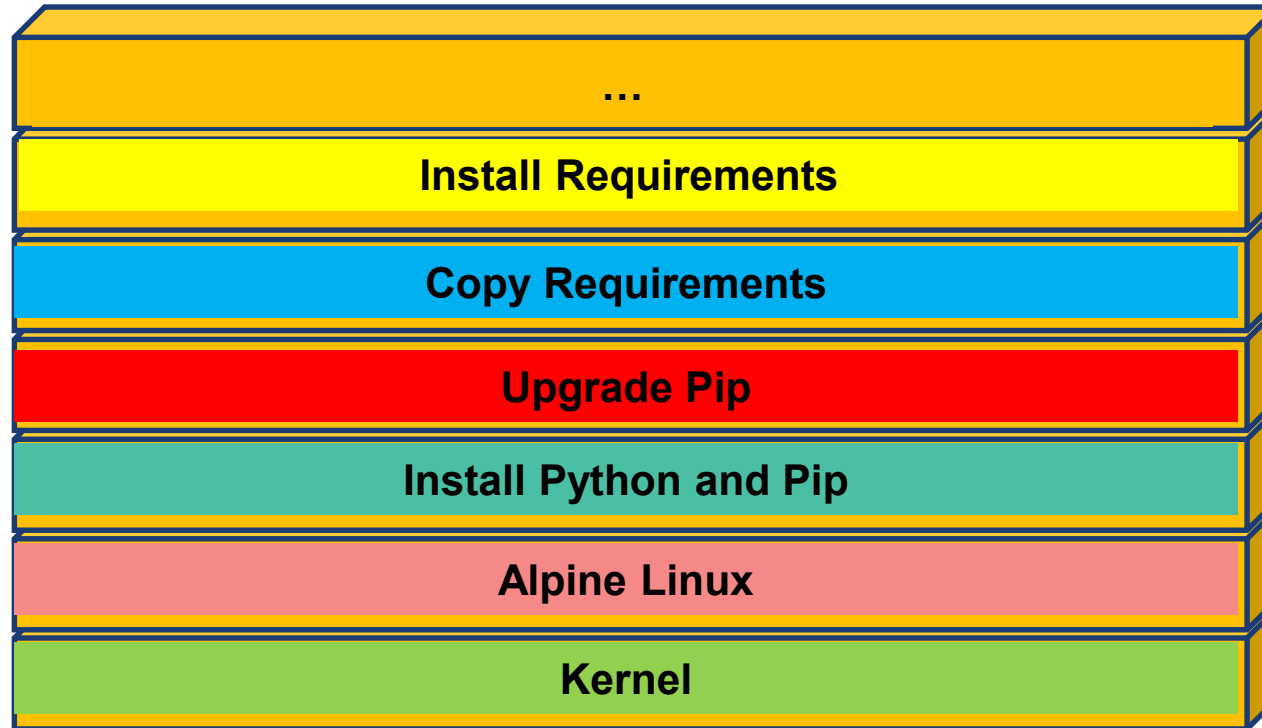
- A static snapshot of the containers' configuration.

- Image is a read-only layer that is never modified, all changes are made in top-most writable layer, and can be saved only by creating a new image.
- Each image depends on one or more parent images

- An image that has no parent.

- Platform images define the runtime environment, packages and utilities necessary for containerized application to run.

Image Layers



Foundation: Docker Engine

Integrated Security

Security	Network	Volumes
Distributed State	Container Runtime	Orchestration



Docker Engine

DEVELOPERS

Microservices



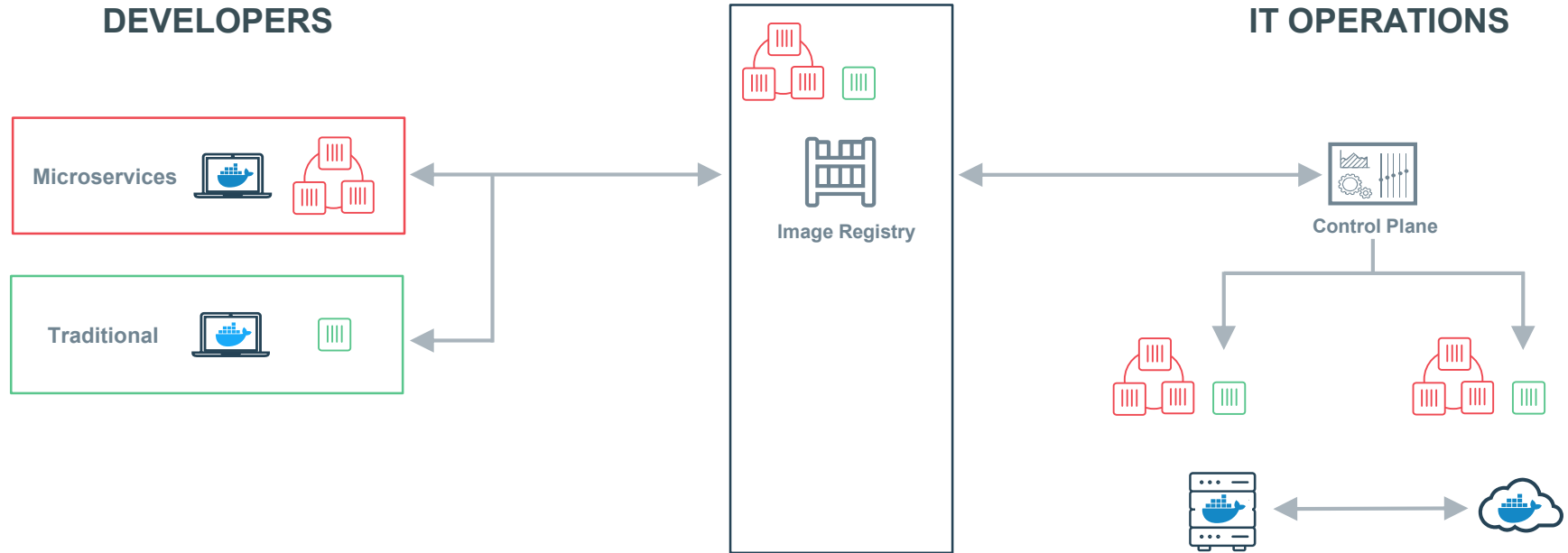
Traditional



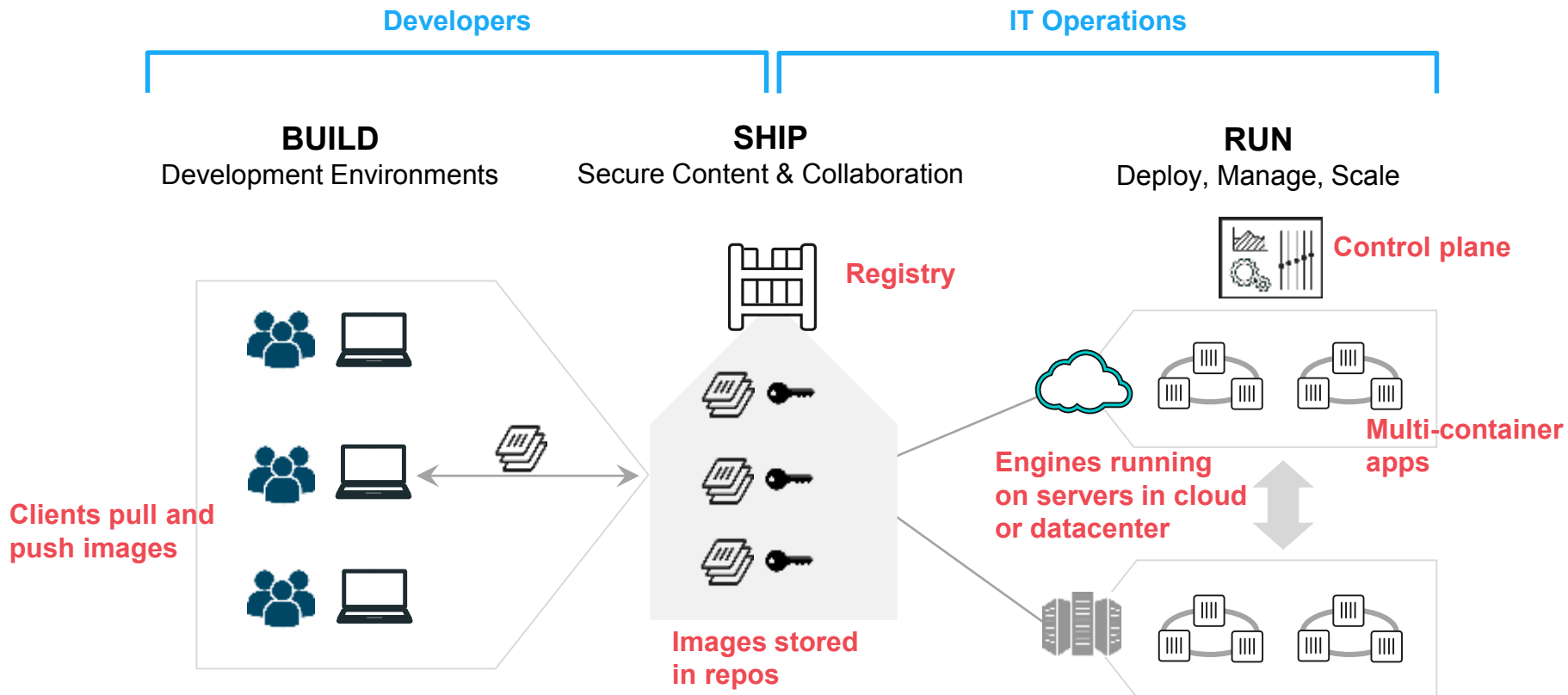
IT OPERATIONS



Building a Software Supply Chain



Containers as a Service



Docker Is in the Enterprise



Service
Provider



Healthcare
& Science



Financial
Services



Tech



Insurance



Public
Sector



Docker delivers agility, security and cost savings



Hardened containers deliver new levels of security to monoliths on the transition to microservices



Transform monoliths to secure and agile DevOps environments



Reduce maintenance costs by 10X for legacy, commercial and new apps

Docker delivers agility, resiliency, portability security and cost savings for all applications

Commercial Off
The Shelf Apps

Homegrown
Traditional Apps

Microservices
Apps

13X

More software releases

65%

Reduction in developer
onboarding time

~47%

Reduction in VMs, OS licensing
and Server costs

Eliminate

“works on my machine”
issues

62%

Report reduction in MTTR

10X

Cost reduction in maintaining
existing applications

One platform and one journey for all applications

1

Traditional apps in containers

Gain portability, efficiency and security



2

Transform to Microservices

Look for shared services to transform



3

Accelerate New Applications

Greenfield innovation



Multiple Stacks, Multiple Stages = Complexity

	Static website	?	?	?	?	?	?	?
	Web frontend	?	?	?	?	?	?	?
	Background workers	?	?	?	?	?	?	?
	User DB	?	?	?	?	?	?	?
	Analytics DB	?	?	?	?	?	?	?
	Queue	?	?	?	?	?	?	?
		Development VM	QA Server	Single Prod Server	Onsite Cluster	Public Cloud	Contributor's laptop	Customer Servers

Solving the deployment matrix

