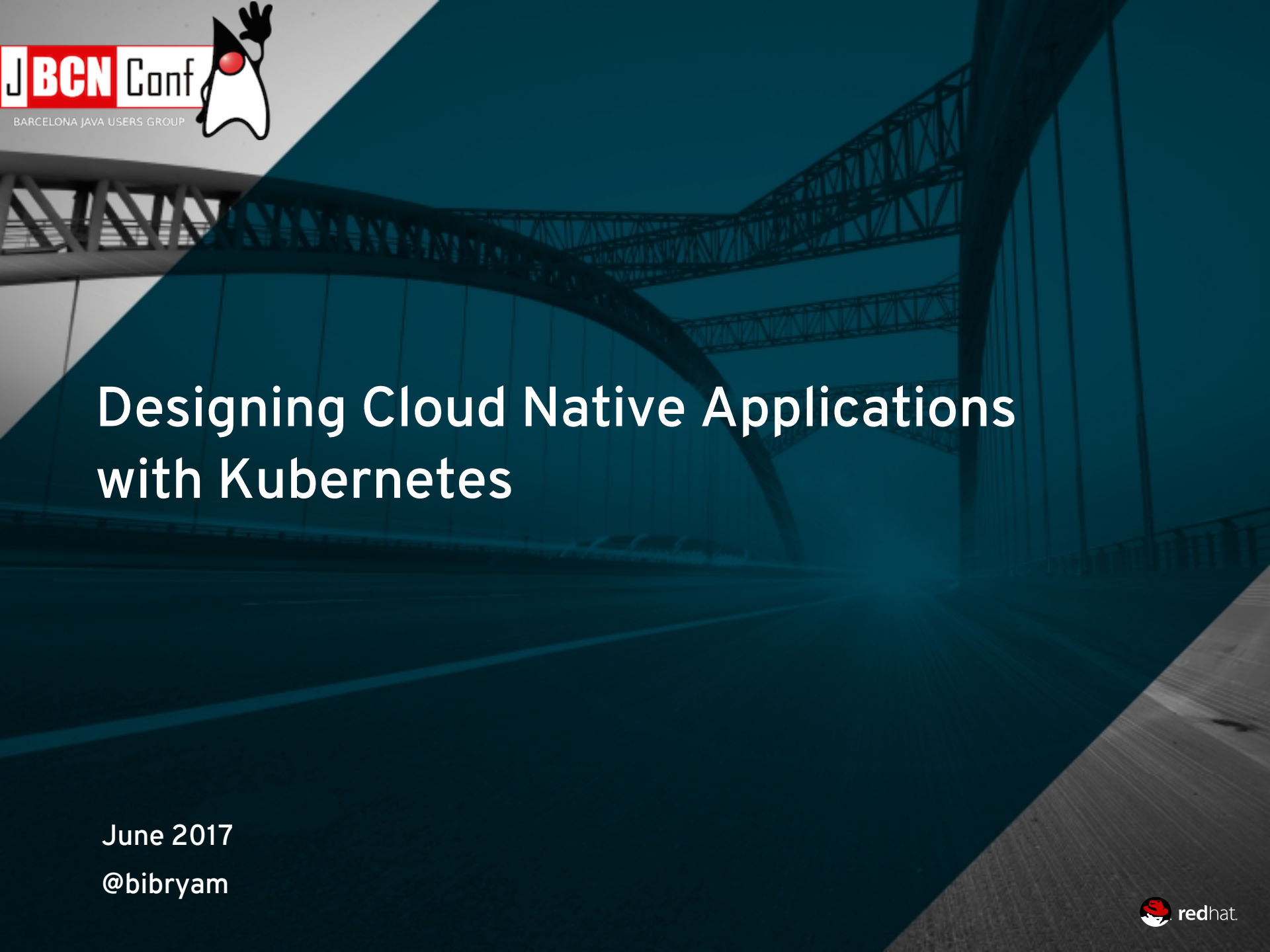


Designing Cloud Native Applications with Kubernetes

June 2017

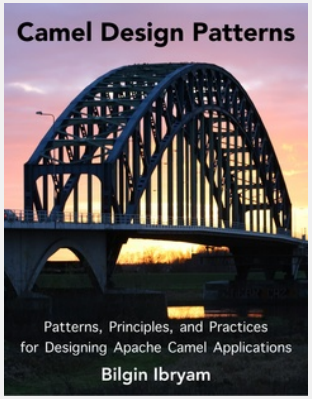
@bibryam

redhat.

Bilgin Ibryam

- Twitter: @bibryam
- Email: bibryam@gmail.com
- Blog: http://ofbizian.com
- Github: https://github.com/bibryam

- Architect at Red Hat
- ASF Member
- Committer for Isis,Camel,OFBiz
- Microservices & Cloud Native



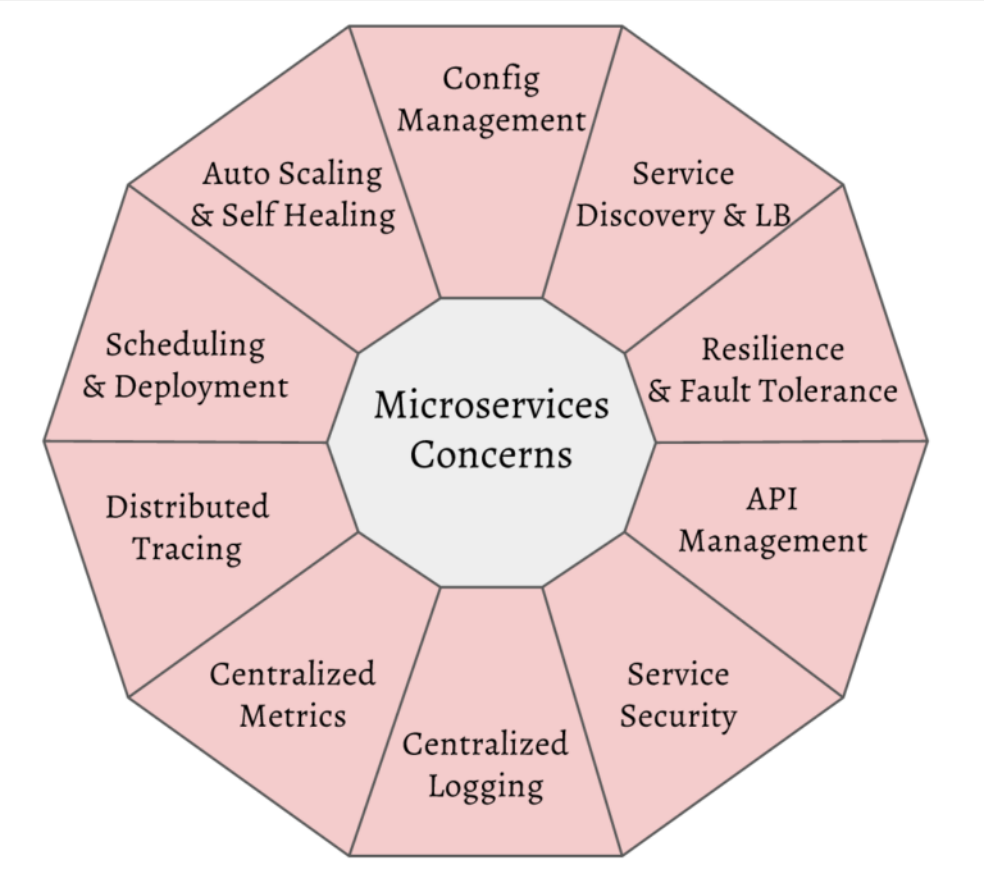
redhat.

Agenda

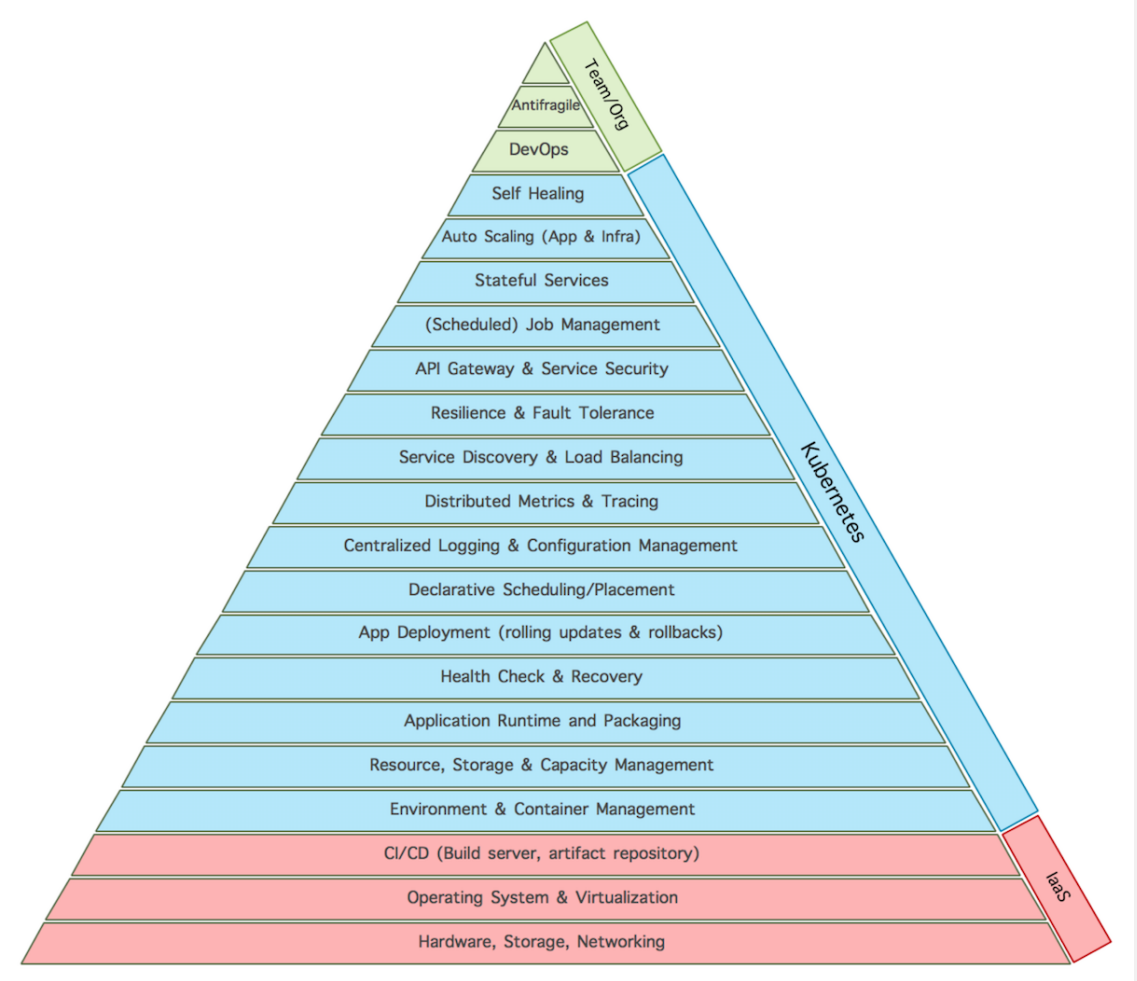
- Cloud Native Ecosystems
- Kubernetes Abstractions & Primitives
- Container Design Principles
- Kubernetes Design Patterns
- Benefits of using Kubernetes

Microservices Architecture

...trades code complexity for operational complexity



Cloud Native Reach



A Good Definition

Cloud Native is structuring **teams**, **culture** and **technology** to utilize **automation** and **architectures** to manage **complexity** and unlock **velocity**.

Joe Beda, Heptio

Common Characteristics

Applications adopting the principles of

Microservices packaged as

Containers orchestrated by

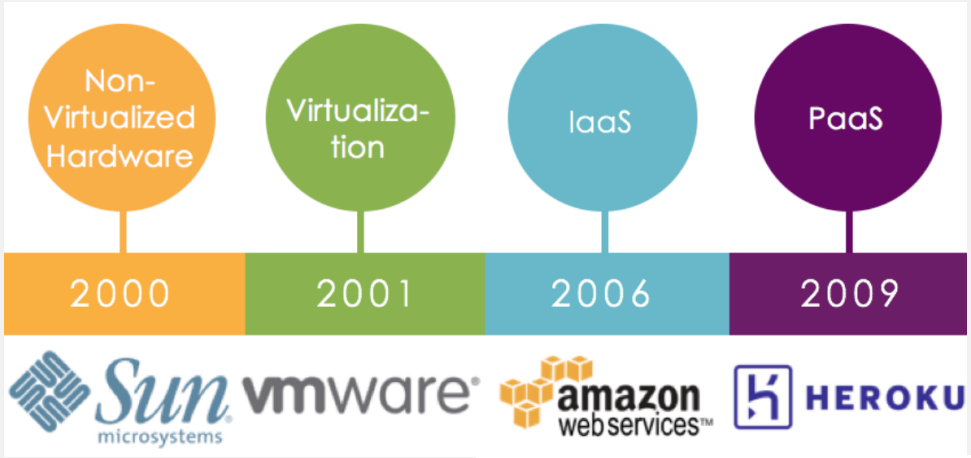
Platforms running on top of

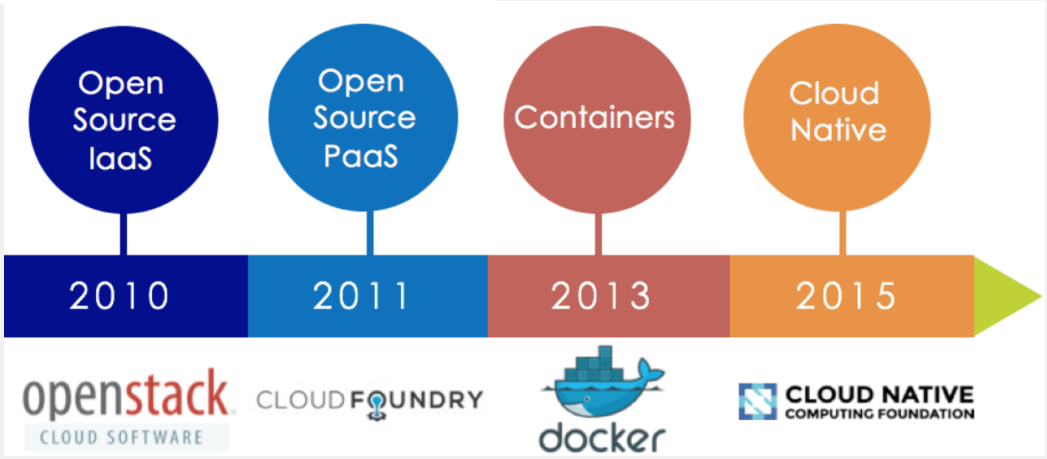
Cloud infrastructure,

developed using practices such as

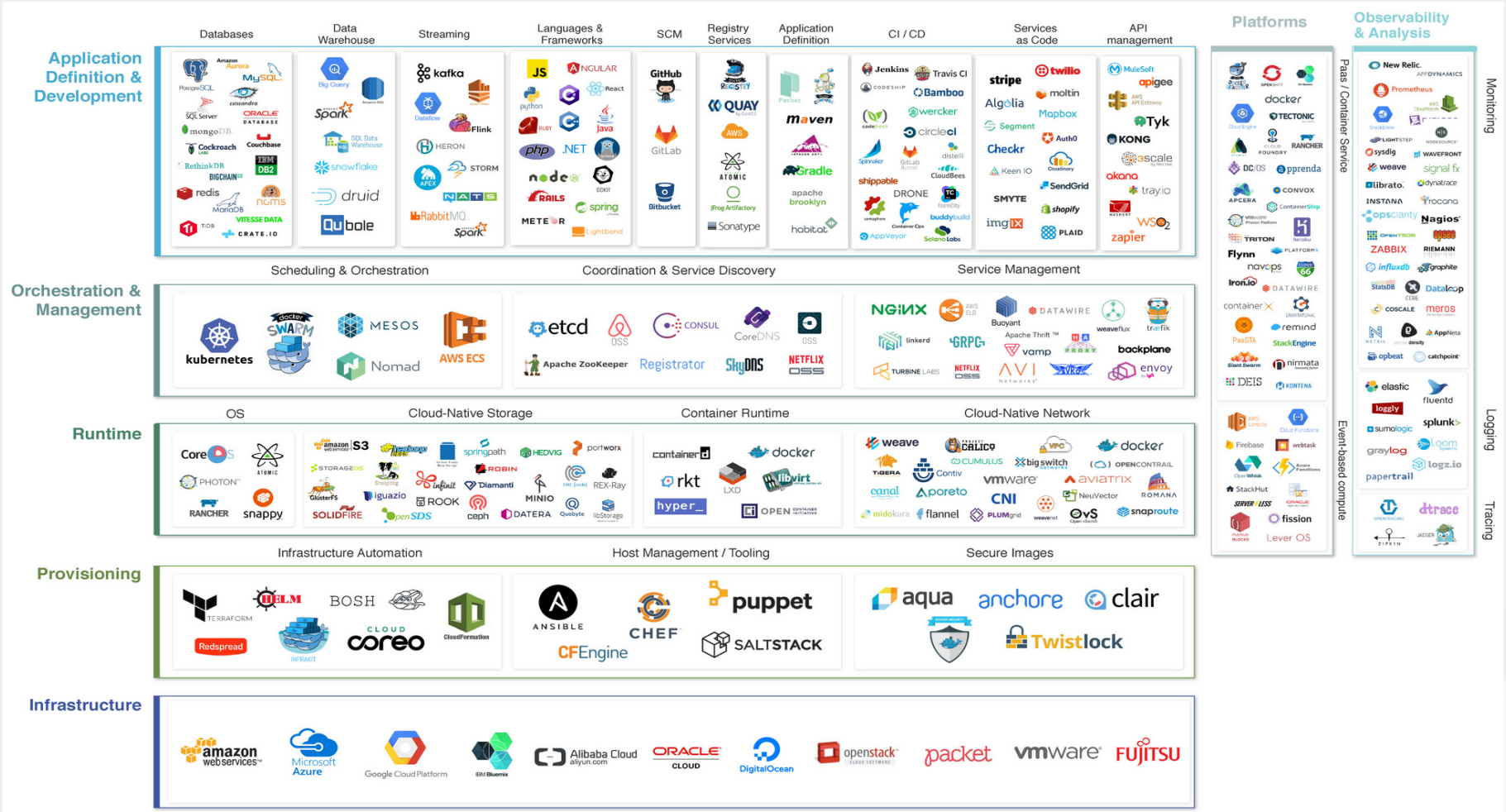
Continuous Delivery and **DevOps**.

Brief Cloud (Native) History





Cloud Native Landscape



@bibryam

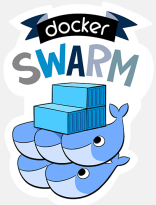
redhat.

Cloud Native Ecosystems

kubernetes

OPENSHIFT

AWS ECS



CLOUDFOUNDRY

Ecosystem Concerns

- Governance model
- Supporting organizations
- User community
- Culture, ways of working
- Technology (maturity * potential)
- Complementary tools
- Documentation, conferences, books
- Coolness factor and developer happiness

CNCF Members

CNCF Projects

CoreDNS
Service Discovery

Prometheus
Monitoring

OpenTracing
Tracing

CNI
Networking

gRPC
Remote Procedure Call

containerd
Container Runtime

Linkerd
Service Mesh

Kubernetes
Orchestration

rkt
Container Runtime

Fluentd
Logging

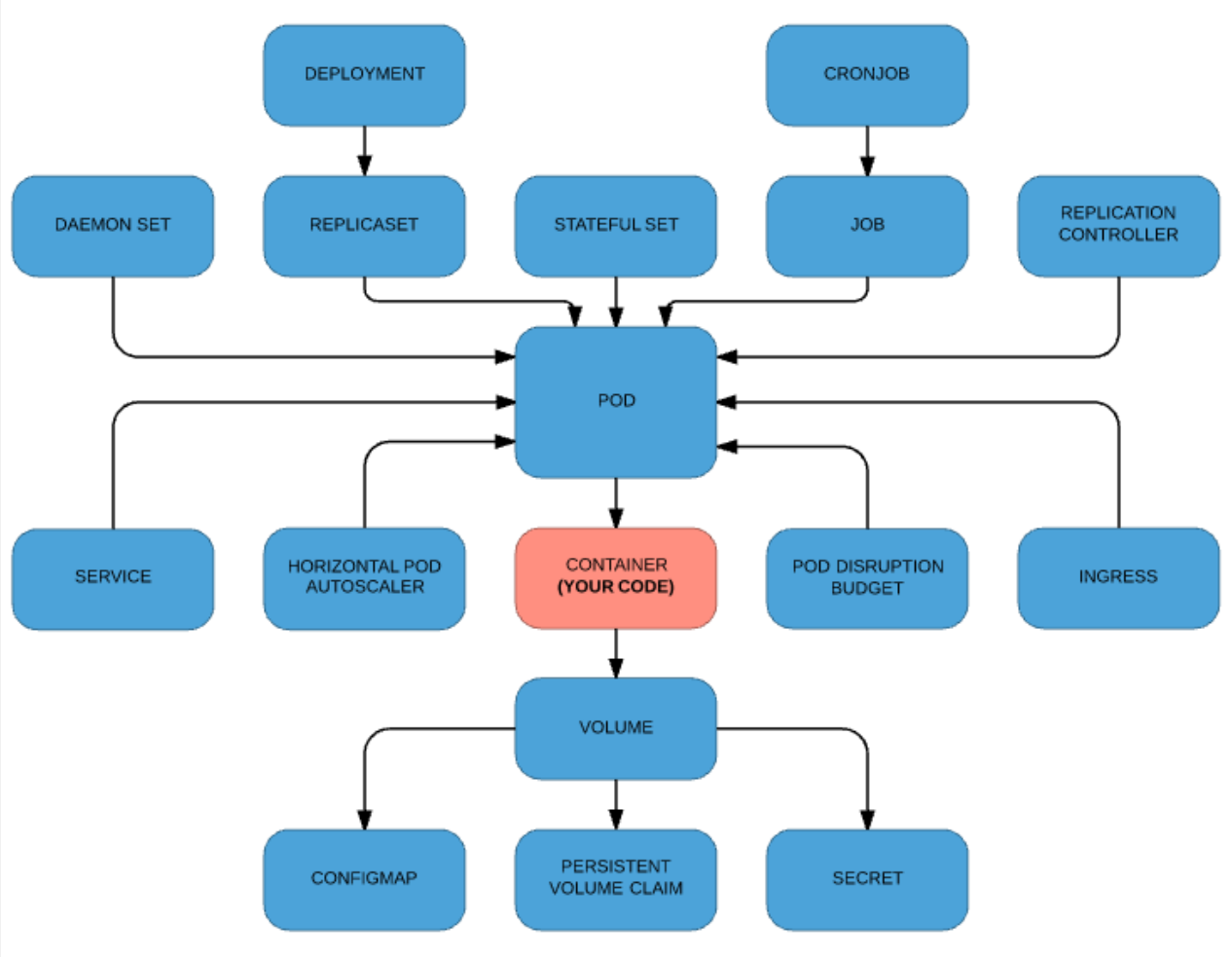
Potential Future Projects

- **Tracing**: OpenTracing
- **Messaging**: NATS
- **Stream Processing**: Heron
- **Networking**: Flannel, Calico, Weave
- **Configuration**: etcd
- **Database**: CockroachDB
- **Storage**: Minio

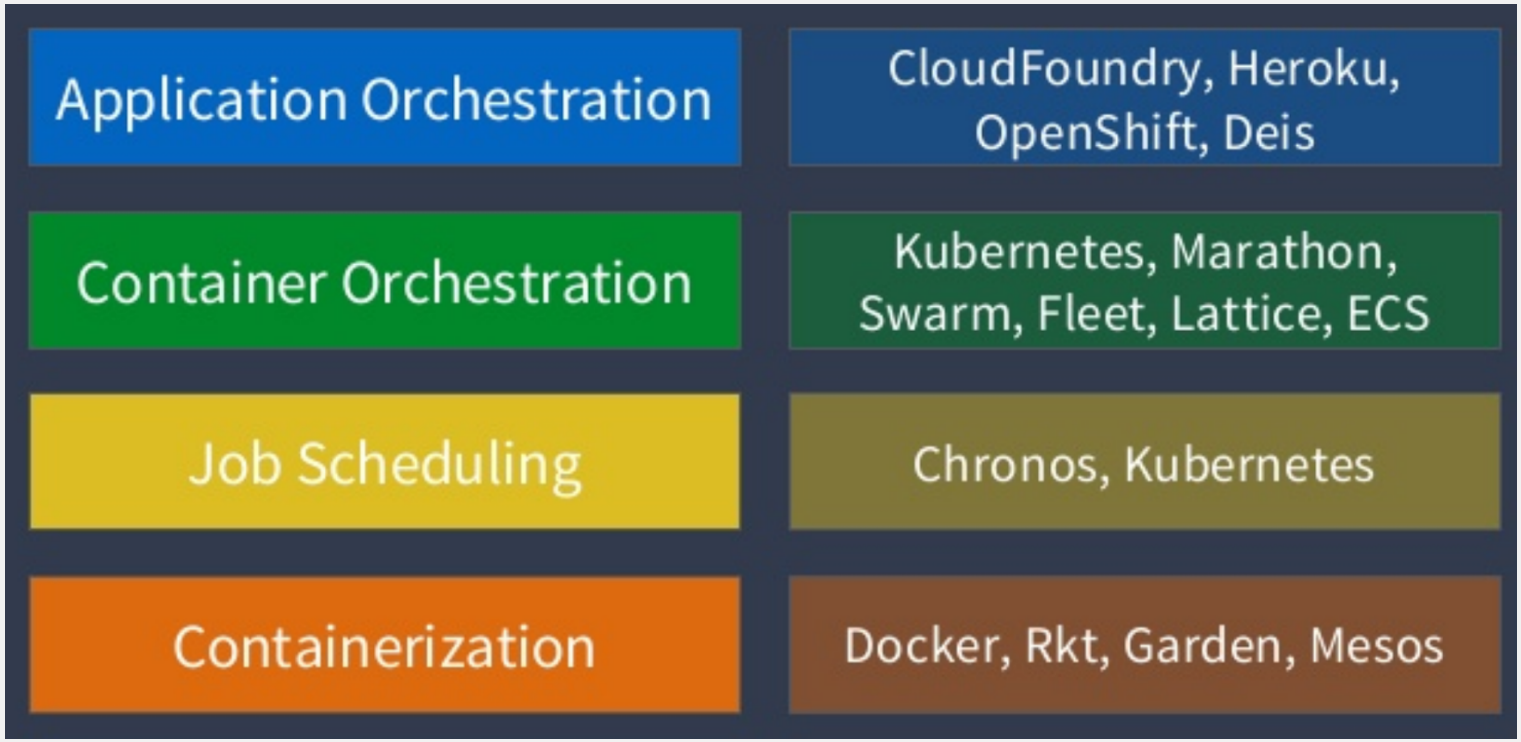
Agenda

- Cloud Native Ecosystems
- **Kubernetes Abstractions & Primitives**
- Container Design Principles
- Kubernetes Design Patterns
- Benefits of using Kubernetes

A Kubernetes Microservice



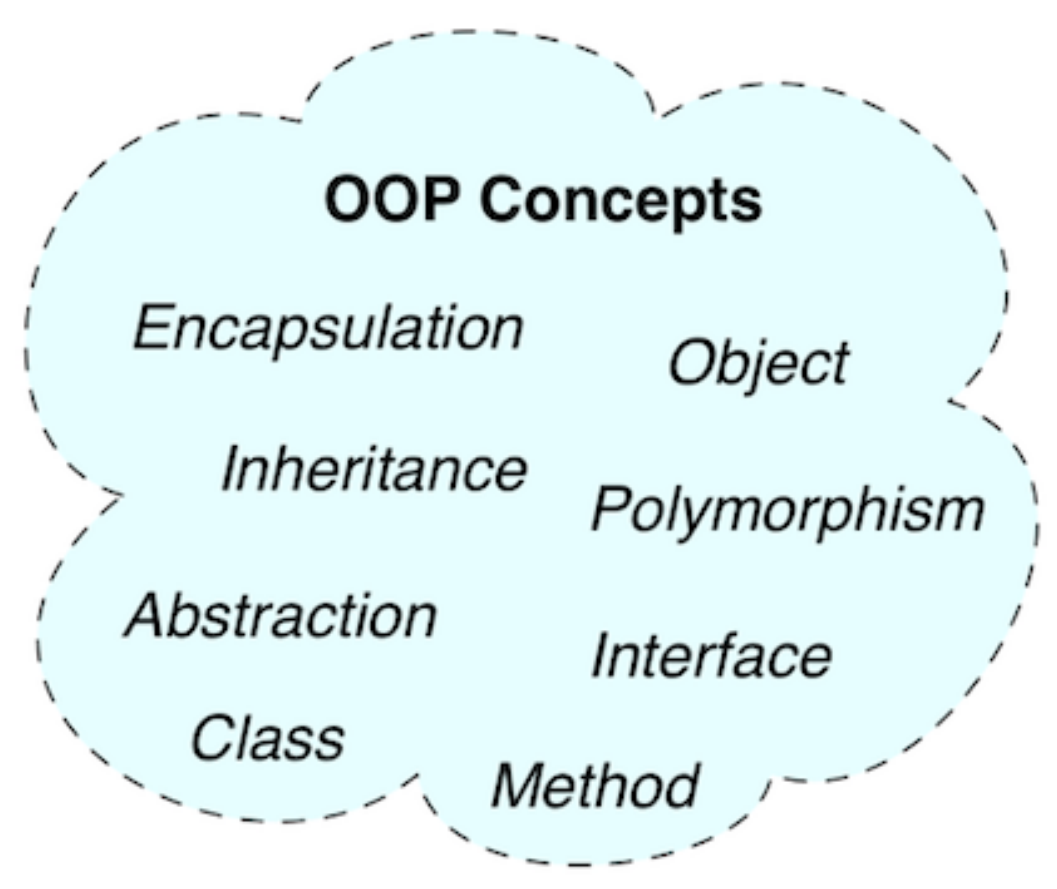
Common Capabilities

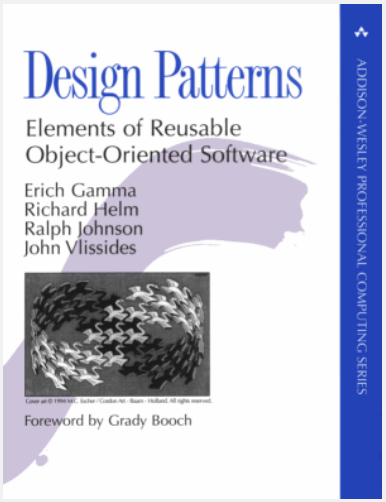


Common Abstractions & Primitives

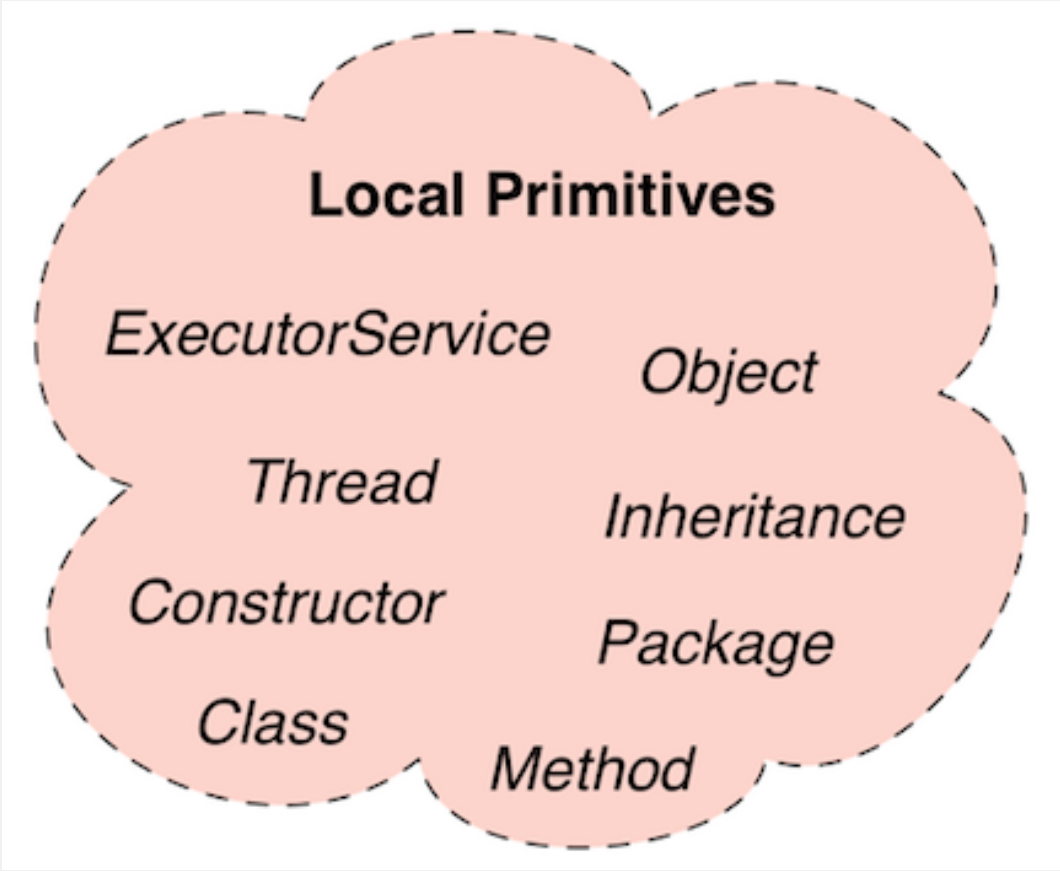
- Application packaging (**Container**)
- Deployment unit (**Pod**)
- Recurring execution (**CronJob**)
- Service discovery & load balancing (**Service**)
- Application placement (**Scheduler**)
- Artifact grouping (**Label**)
- Resources isolation (**Container/Namespace**)

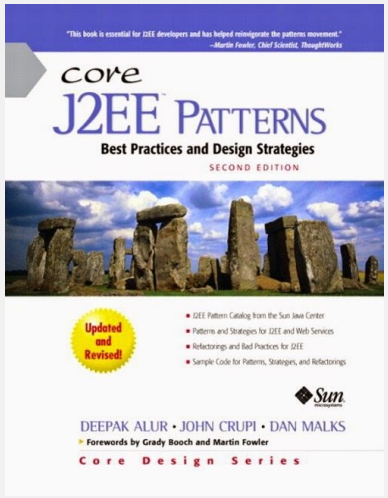
OOP Abstractions & Primitives



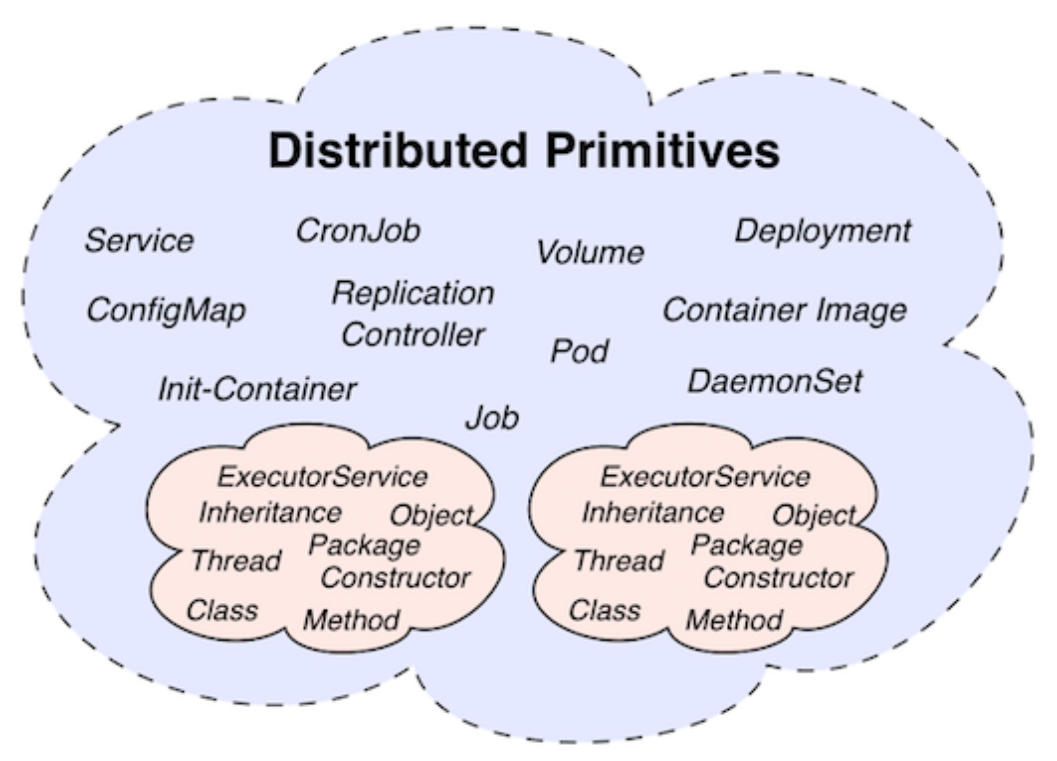


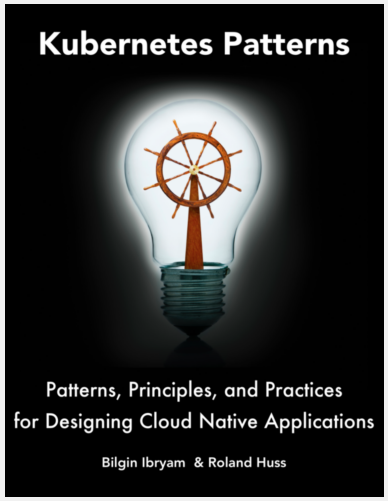
Java Abstractions & Primitives





K8S Abstractions & Primitives





Local & Distributed Primitives

Concept	JVM	Kubernetes
Behaviour encapsulation	Class	Container Image
Behaviour instance	Object	Container
Initialization logic	Constructor	Init-container
Cleanup logic	finalize()/ShutdownHook	Defer-container*
Parallel execution	Thread	Job
Background task	Daemon Thread	DaemonSet
Periodic execution	ExecutorService	CronJob
Buildtime/Runtime isolation	Package/Jigsaw	Namespace
Configuration management	java.util.Properties	ConfigMap
Deployment unit	.jar/.war/.ear	Pod

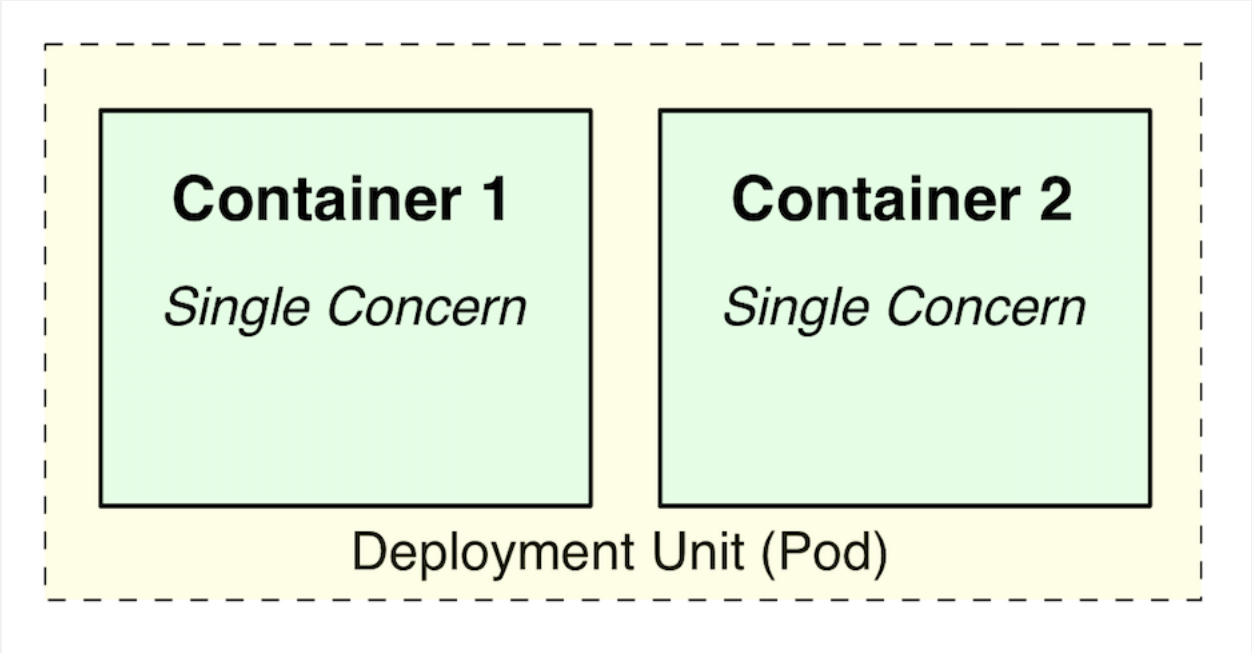
Agenda

- Cloud Native Ecosystems
- Kubernetes Abstractions & Primitives
- **Container Design Principles**
- Kubernetes Design Patterns
- Benefits of using Kubernetes

(SOLID) Principles of Container Design

- Single Concern Principle (SCP)
- Self-Containment Principle (S-CP)
- Image Immutability Principle (IIP)
- High Observability Principle (HOP)
- Lifecycle Conformance Principle (LCP)
- Process Disposability Principle (PDP)
- Runtime Confinement Principle (RCP)

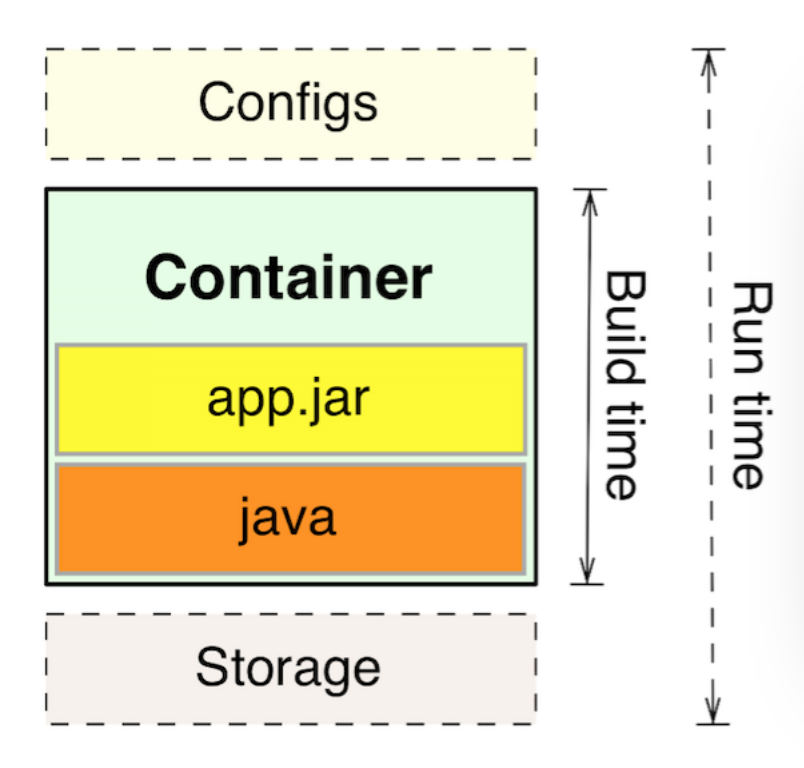
Single Concern Principle



Patterns

- Sidecar
- Ambassador
- Adapter
- Init-container
- Defer-container

Self-Containment Principle



AntiPattern

- Locomotive

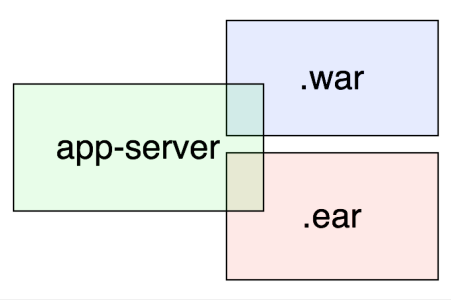
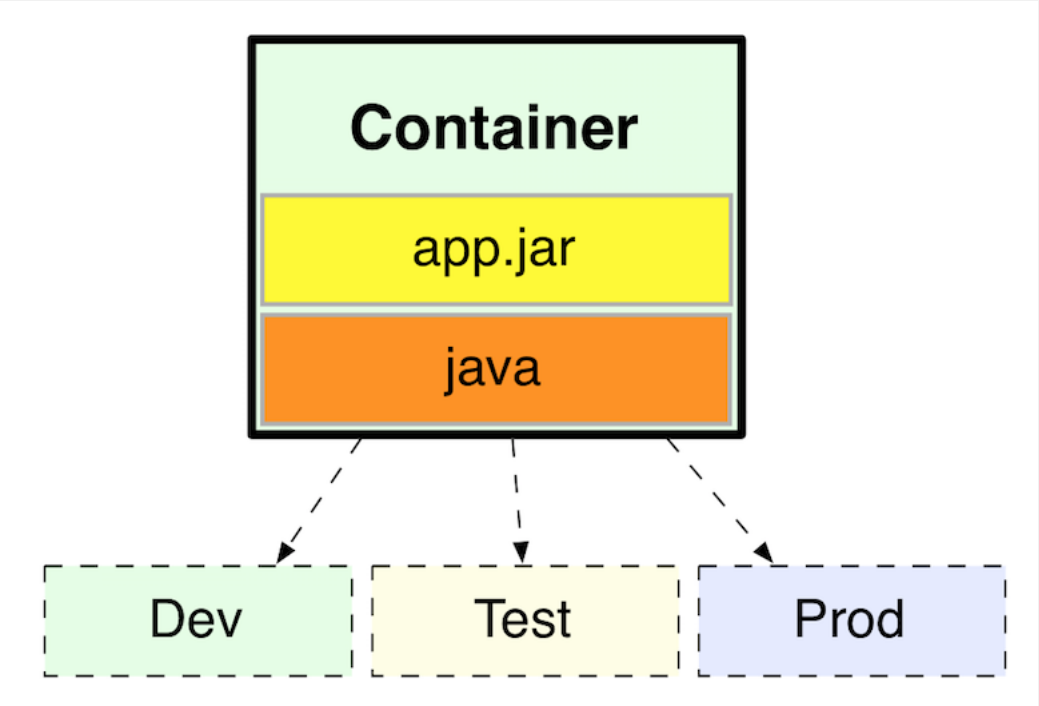
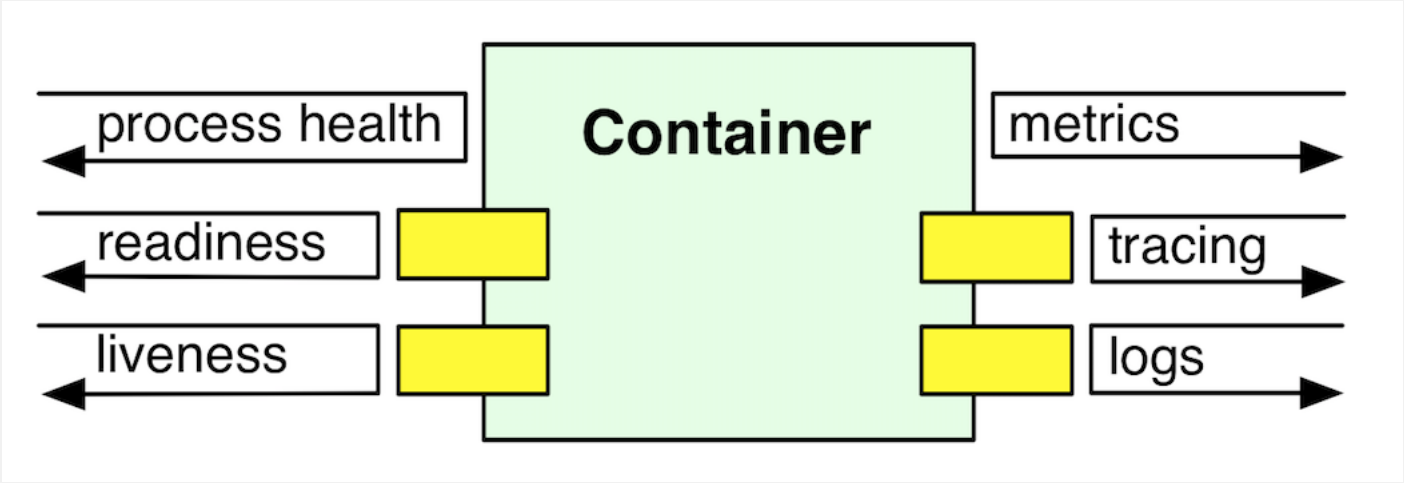


Image Immutability Principle



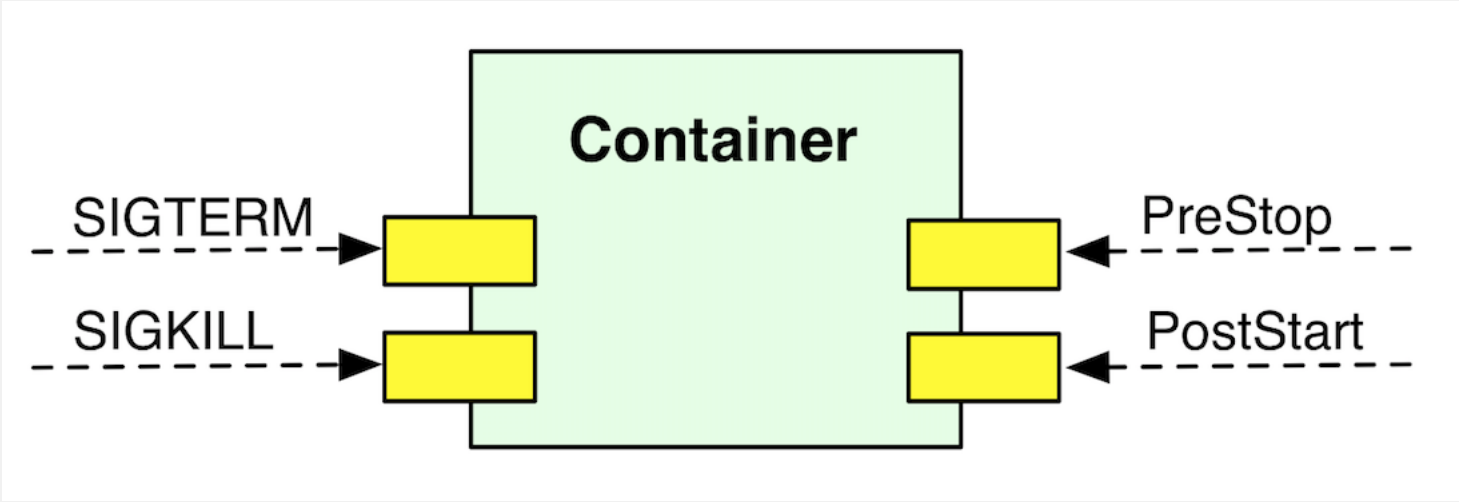
- Dev/prod parity

High Observability Principle



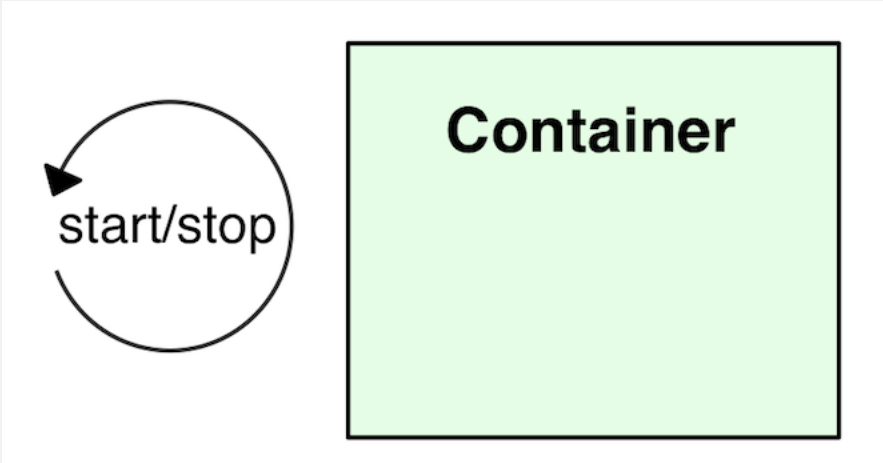
- Spring Boot Actuator
- Dropwizard Metrics
- WildFly Swarm Monitor
- MicroProfile Healthchecks

Lifecycle Conformance Principle



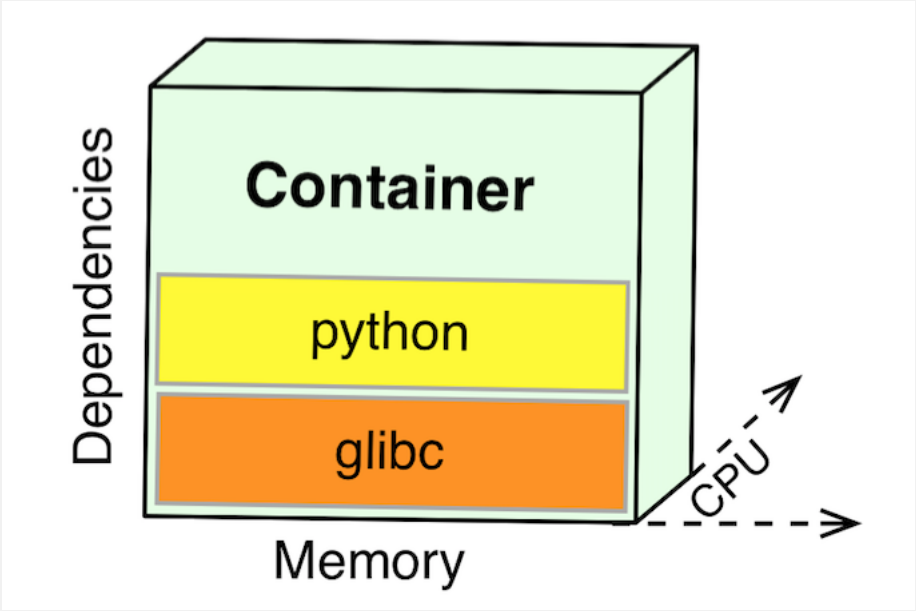
- SIGTERM
- SIGKILL
- PreStop
- PostStart

Process Disposability Principle



- Fast startup
- Graceful shutdown
- Don’t rely on a particular instance
- Be aware of shots at your cattle
- Be robust against sudden death

Runtime Confinement Principle



- resources.limits.cpu
- resources.limits.memory
- resources.requests.cpu
- resources.requests.memory

More Container Best Practices

- Small images
- Arbitrary user IDs
- Port declarations
- Volume declarations
- Image metadata
- Host and image sync

https://www.slideshare.net/luebken/container-patterns

https://docs.docker.com/engine/userguide/eng-image/dockerfile_best-practices

http://docs.projectatomic.io/container-best-practices

https://docs.openshift.com/enterprise/3.0/creating_images/guidelines.html

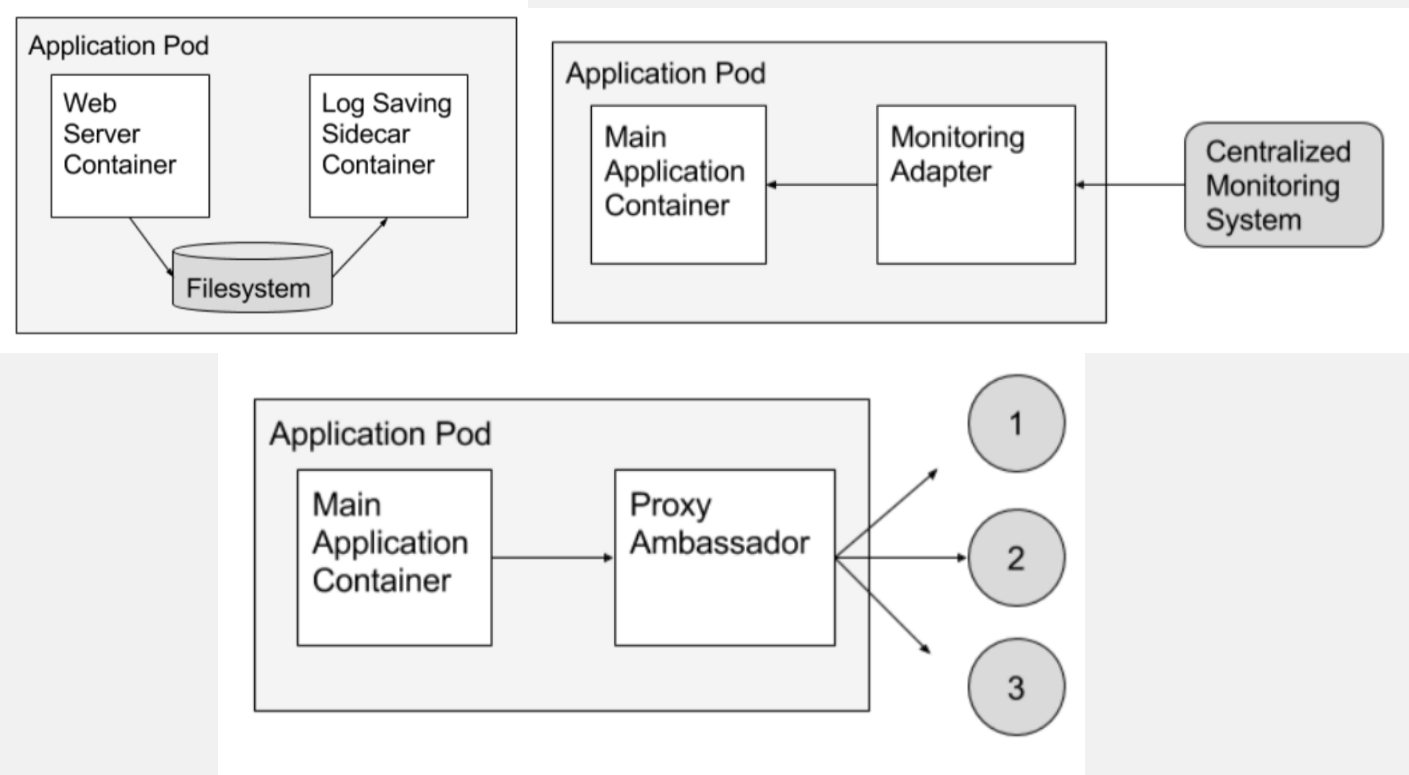
http://blog.arungupta.me/docker-container-anti-patterns/

https://12factor.net/

Agenda

- Cloud Native Ecosystems
- Kubernetes Abstractions & Primitives
- Container Design Principles
- **Kubernetes Design Patterns**
- Benefits of using Kubernetes

Container Oriented Design Patterns



https://www.usenix.org/system/files/conference/hotcloud16/hotcloud16_burns.pdf

More Kubernetes Patterns

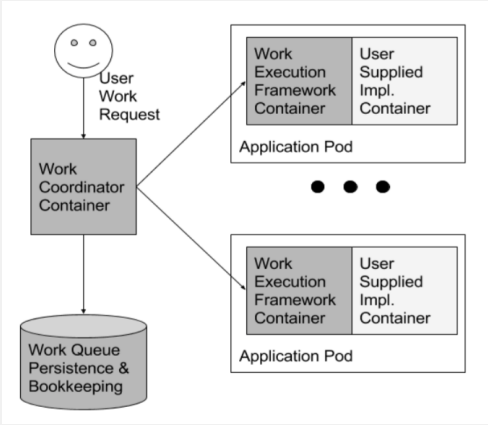
Configuration Management –> **ConfigMaps**

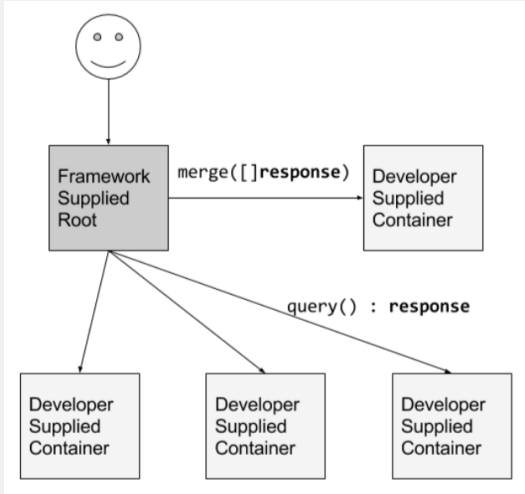
Singleton Services -> **replicas: 1**

Initializers -> **Init Container**

Daemon Services –> **Daemon Sets**

Stateful Services –> **StatefulSet**





https://leanpub.com/k8spatterns/

Agenda

- Cloud Native Ecosystems
- Kubernetes Abstractions & Primitives
- Container Design Principles
- Kubernetes Design Patterns
- **Benefits of using Kubernetes**

Benefits of using Kubernetes

1) Self Service Environments

2)Dynamically Placed Applications

3)Declarative Service Deployments

4)Blue-Green and Canary Releases

5)Application Resilience & Antifragility

6)Service Discovery & Load Balancing

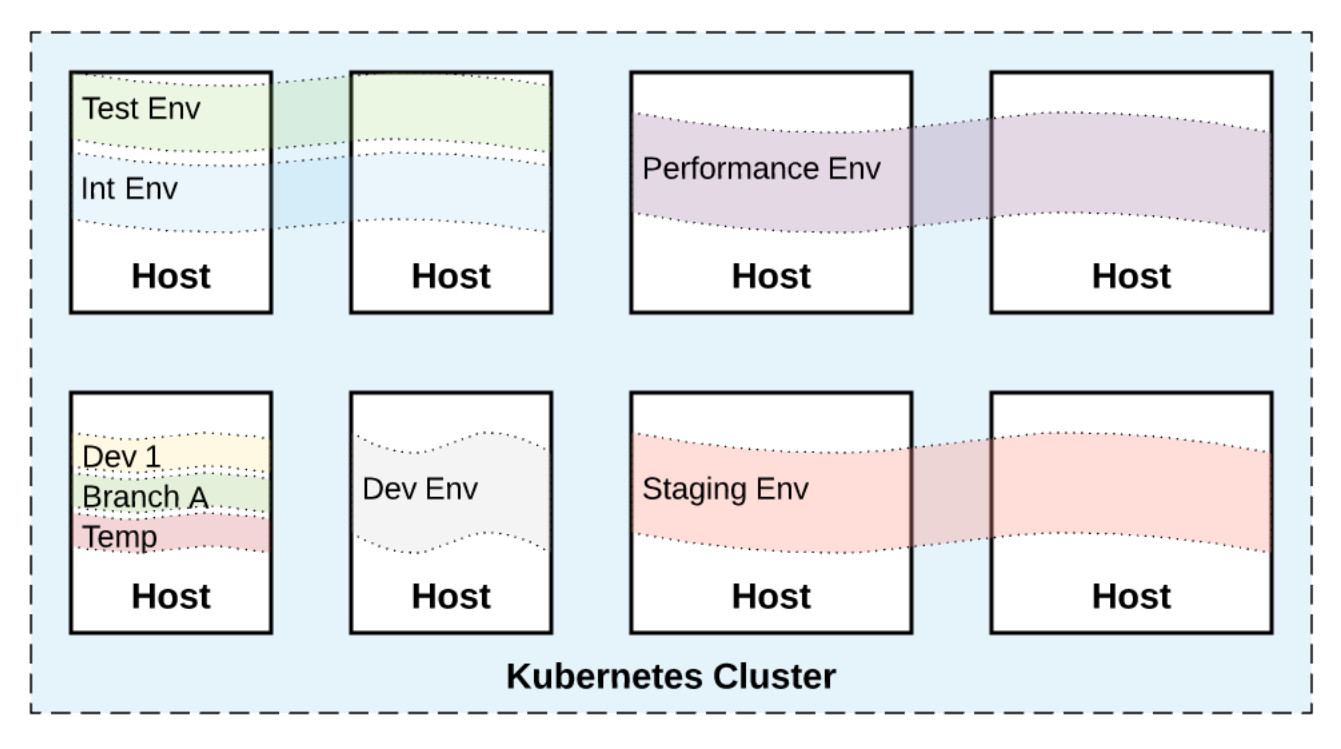
7)Service Clustering

8)(Scheduled) Batch Jobs

9)Immutable Infrastructure

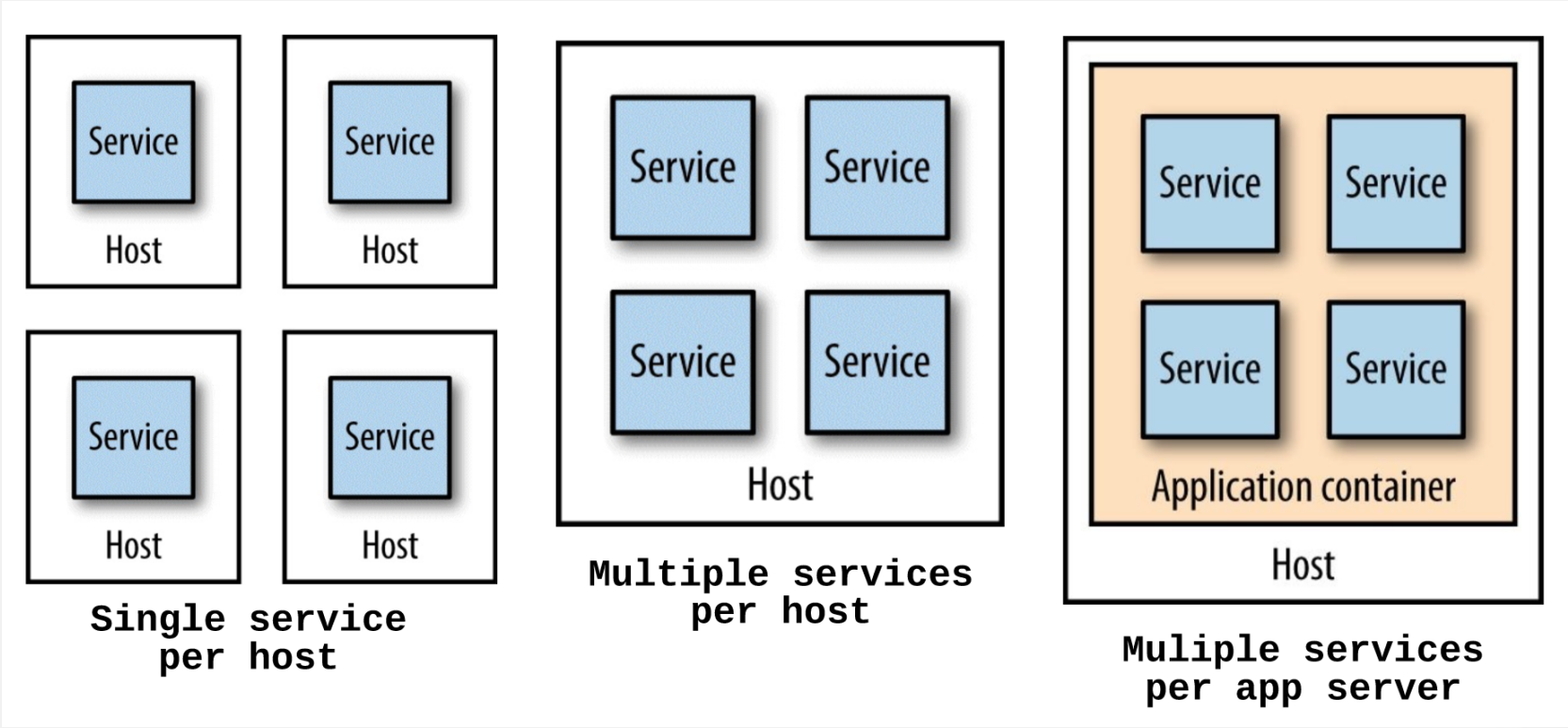
Self Service Environments

An Environment is not a VM any longer



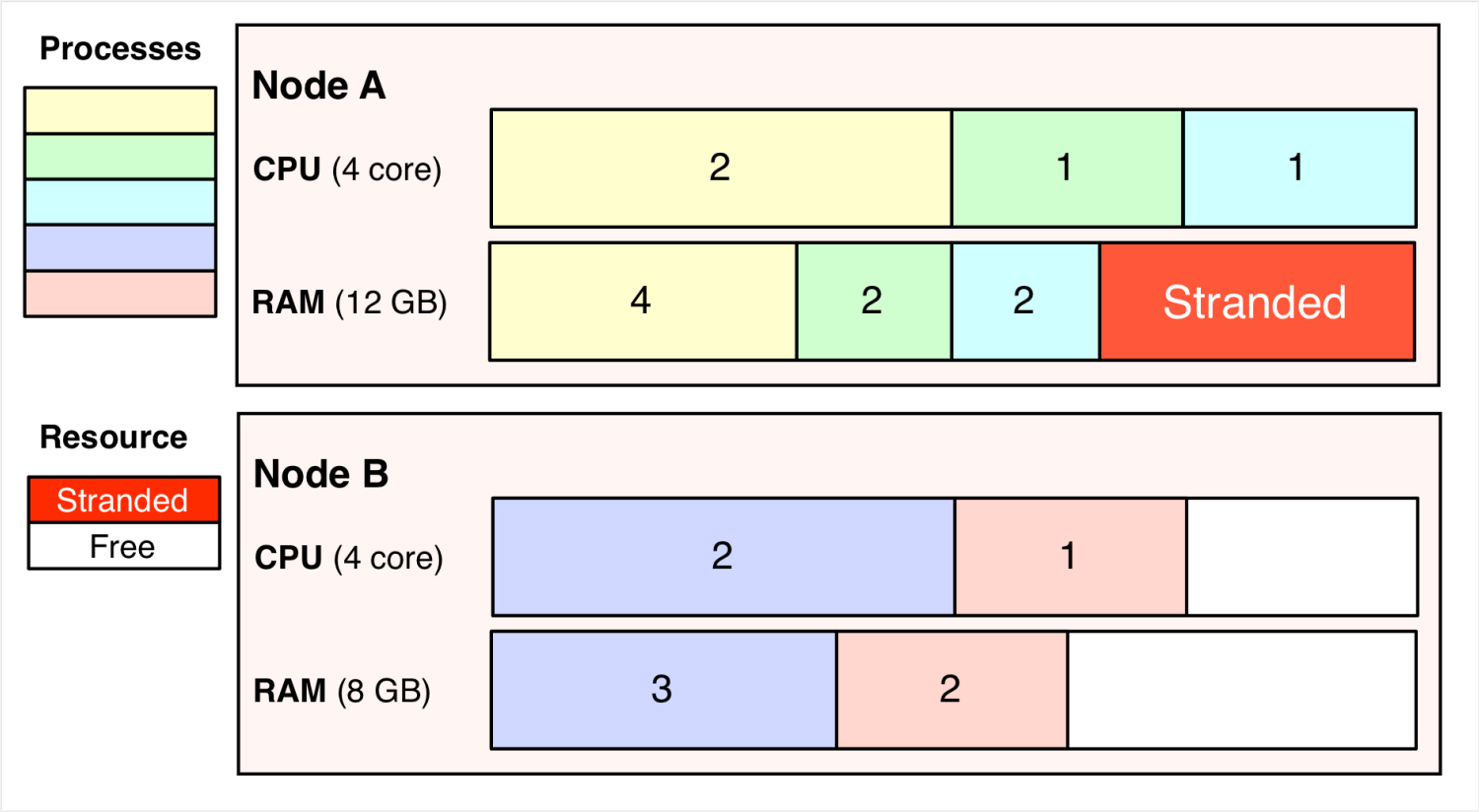
Dynamically Placed Applications

Forget about these manual placement strategies and...



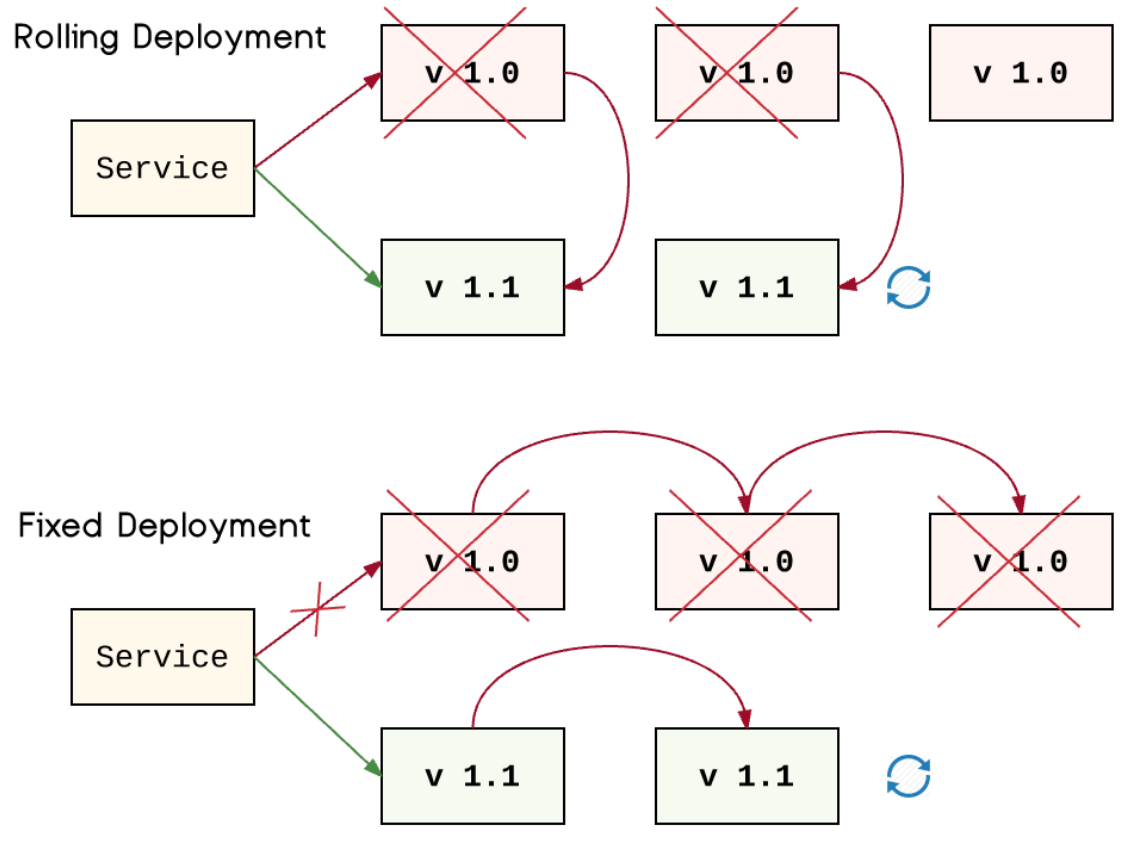
Dynamically Placed Applications

... and trust the Scheduler



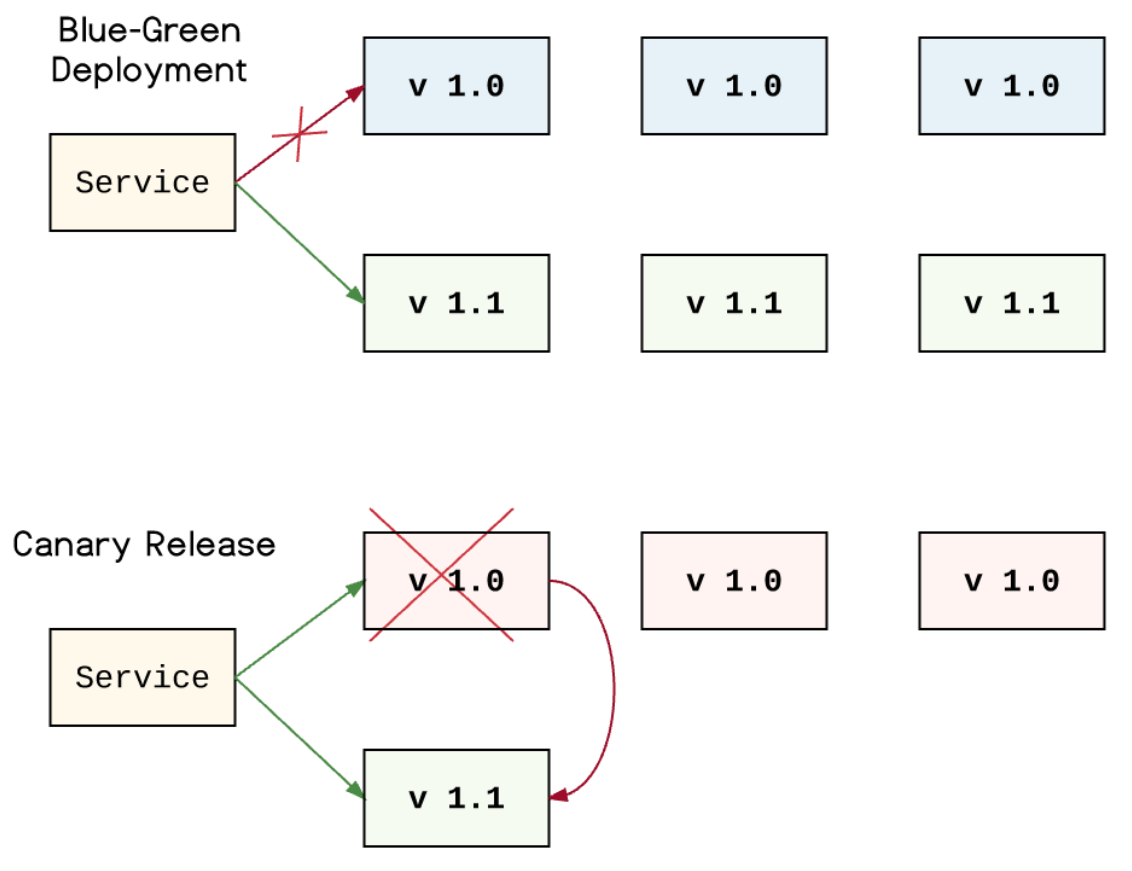
Declarative Service Deployments

Deployment config based options



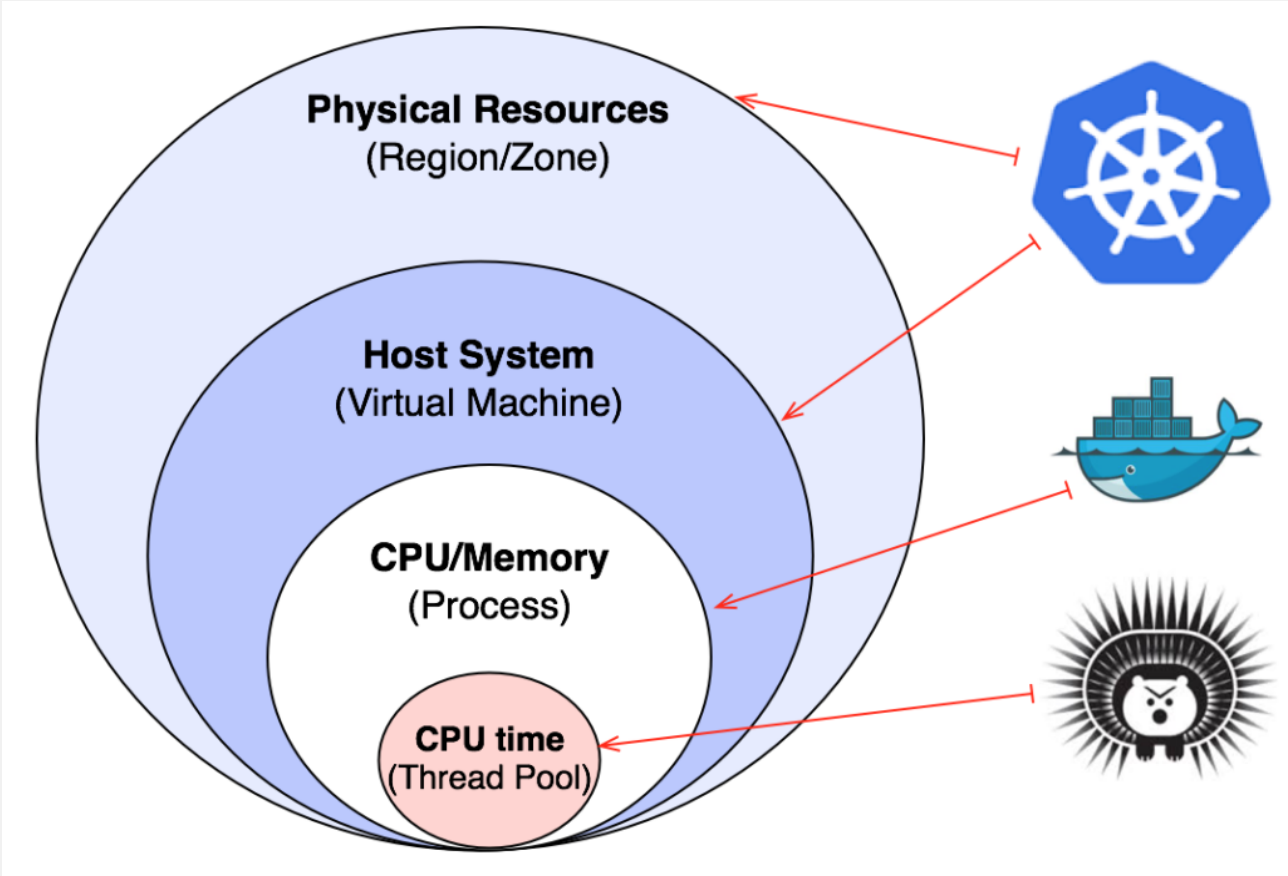
Blue-Green and Canary Releases

Routing config based options



Application Resilience

For true resilience you must go outside of the JVM!

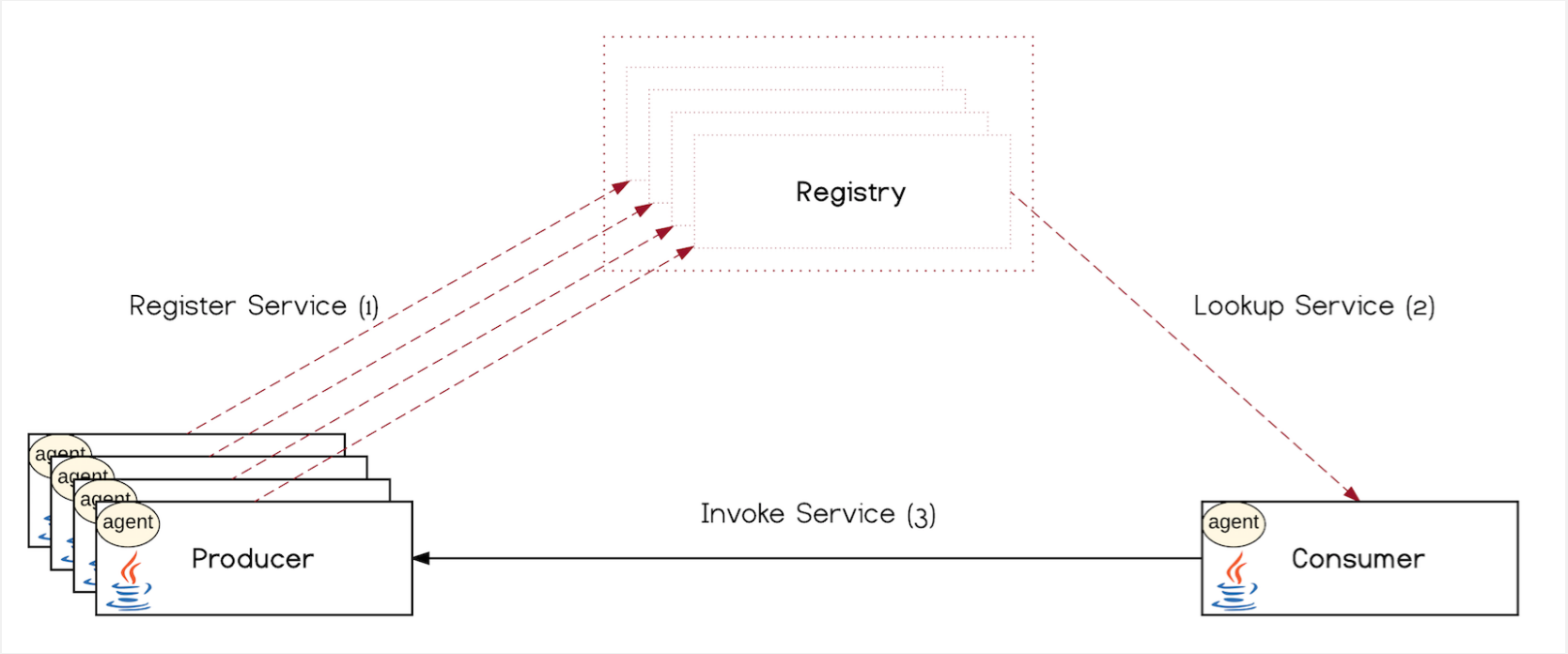


Application Resilience & Antifragility

- Infinite loops: CPU shares and quotas
- Memory leaks: OOM yourself
- Disk hogs: Quotas
- Fork bombs: Process limits
- Circuit Breaker, Timeout, Retry as SideCar
- Failover and Service Discovery as SideCar
- Process Bulkheading with Containers
- Hardware Bulkheading through Scheduler
- Auto-scaling & Self-healing

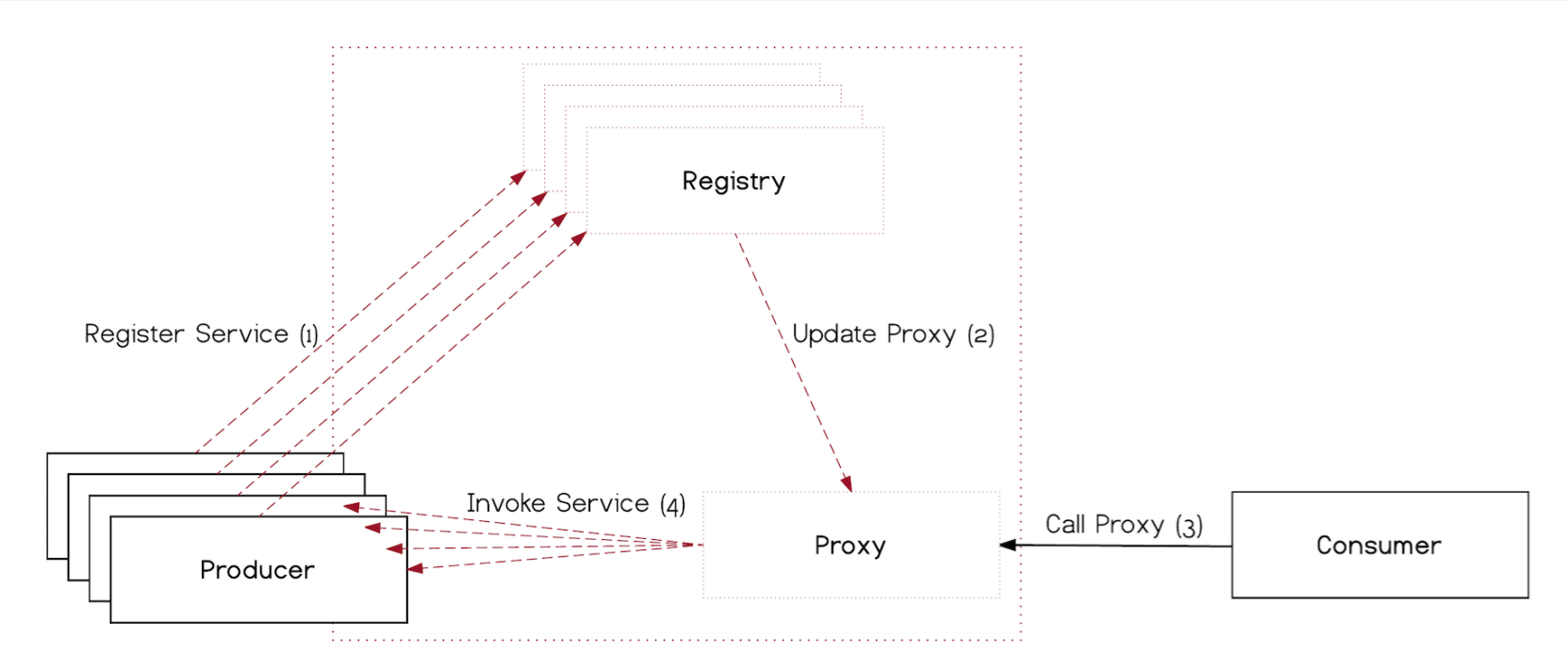
Service Discovery & Load Balancing

Client side – on the JVM



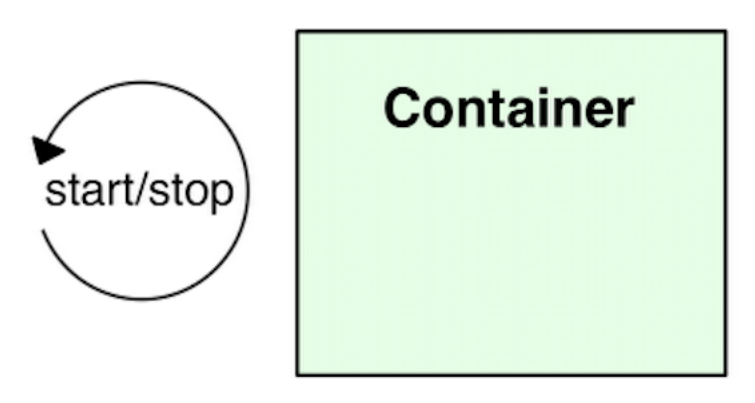
Service Discovery & Load Balancing

Provided by the platform



(Scheduled) Batch Jobs

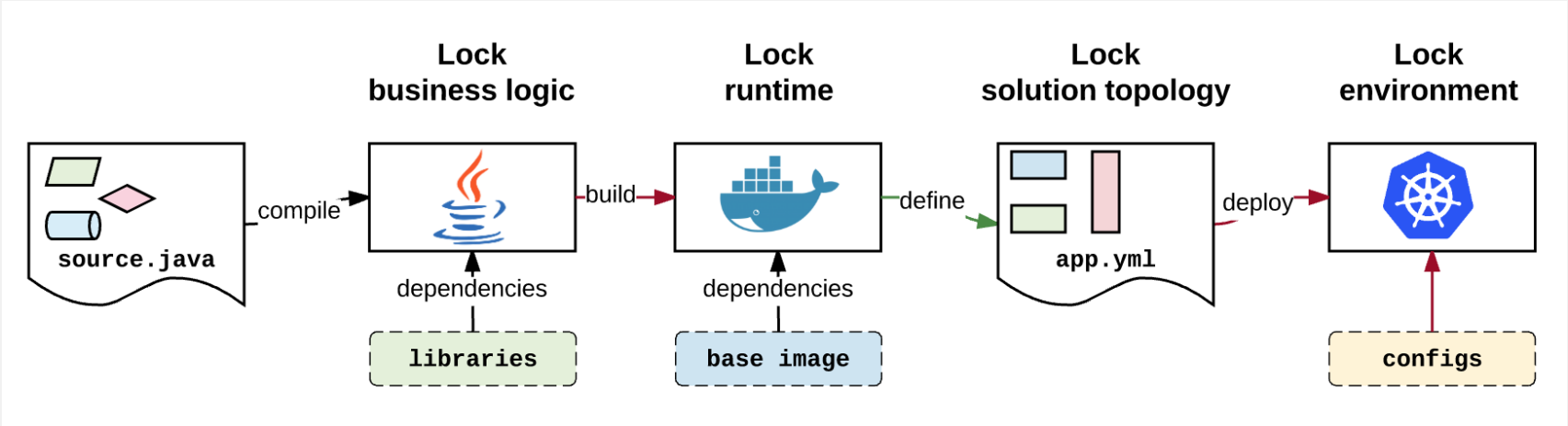
Provided by the platform



Example batch jobs in Java

- JDK Timer
- ScheduledExecutorService
- Quartz Scheduler
- Spring Batch

Immutable Infrastructure



Application binaries → Container
Deployment unit → Pod
Resource demands → request/limit/PVC
Configurations → ConfigMap/Secret
Update/rollback mechanism → Deployment

Executable
app.yml

Key Takeaways

1. Automate routine tasks through a cloud native platform.

(placement, updates, healthchecks, self-healing, scaling)

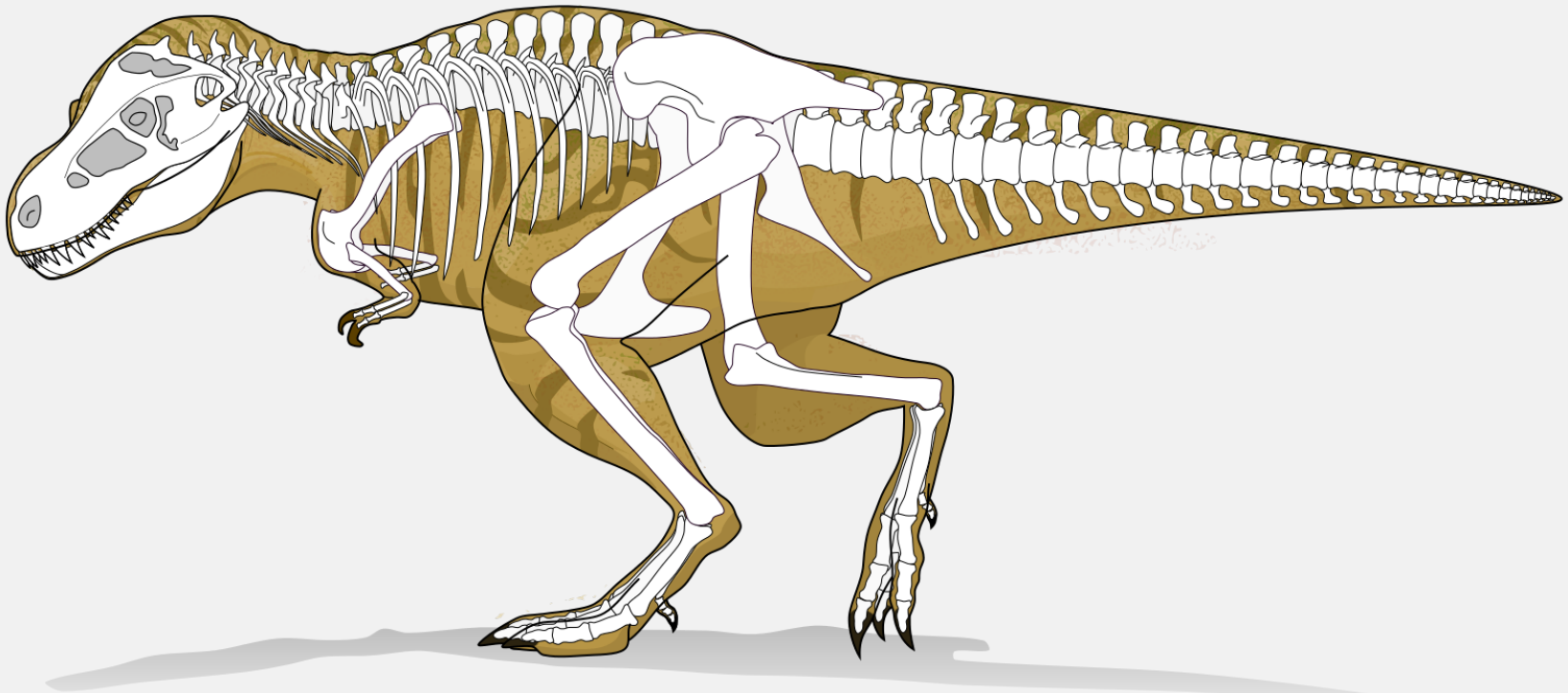
2. Move XFR/NFR from application to platform layer.

(service discovery, circuit breaker, jobs, configurations, logs)

3. Ecosystem matters

(Bet on a complete ecosystem, not a single platform)

Cloud Native is not Mandatory



It is not necessary to change. Survival is not mandatory.

W. Edwards Deming

Q & A

Kubernetes Patterns http://leanpub.com/k8spatterns

More on this topic: https://twitter.com/bibryam

@bibryam

redhat.