KubeCon | CloudNativeCon

Europe 2024



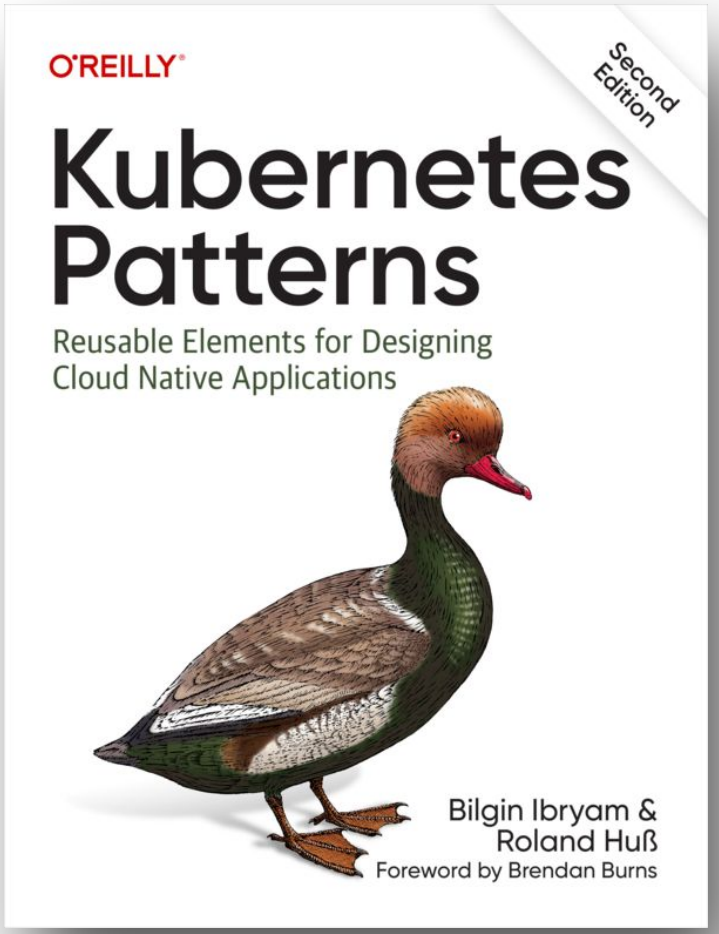
10 Years of Kubernetes Patterns Evolution

Bilgin Ibryam, Diagrid, **@bibryam**

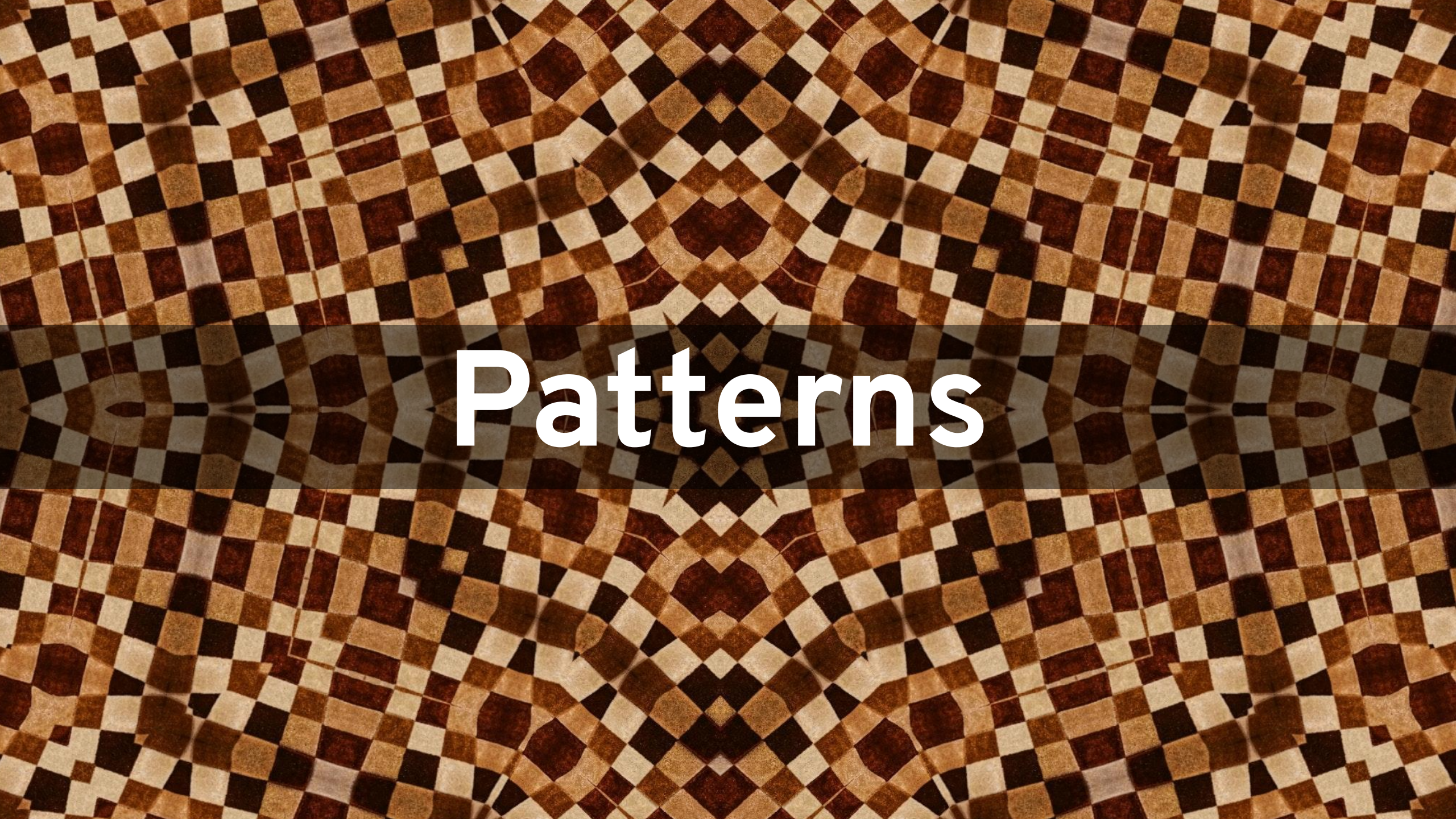
Roland Huß, Red Hat,
@ro14nd@hachyderm.io

k8spatterns.com

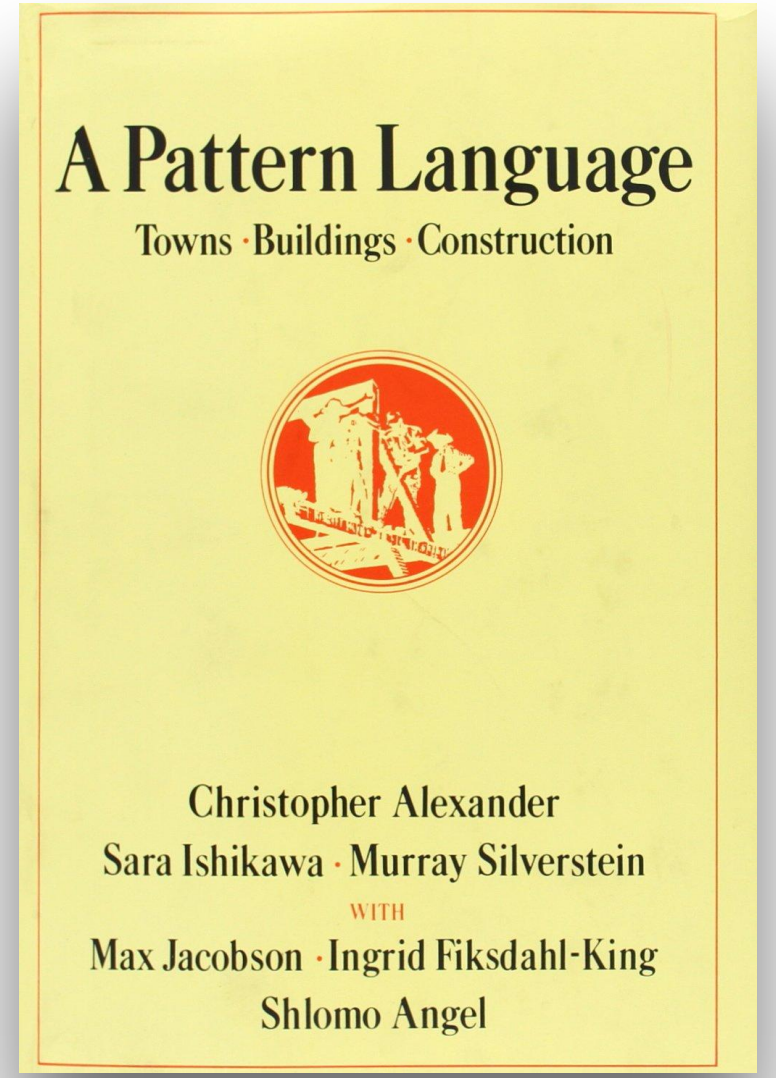
Kubernetes Patterns



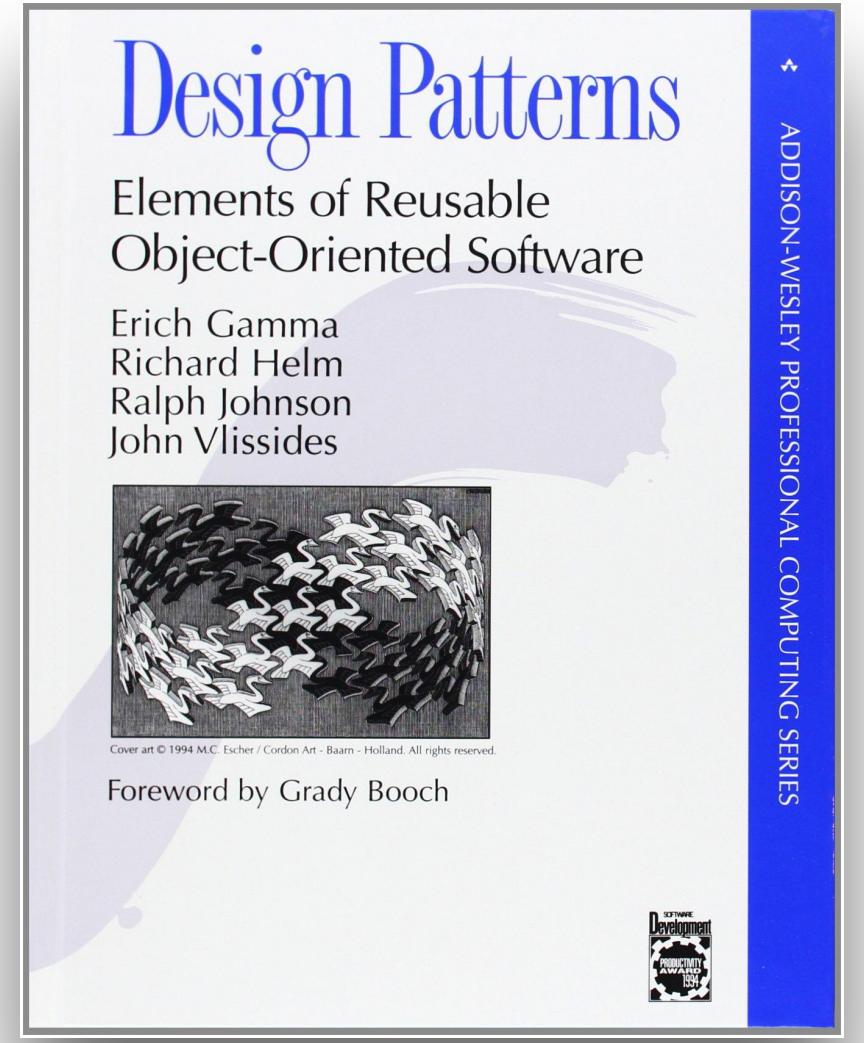
https://k8spatterns.com


Patterns

Patterns


A Pattern Language
Towns · Buildings · Construction
Christopher Alexander
Sara Ishikawa · Murray Silverstein
WITH
Max Jacobson · Ingrid Fiksdahl-King
Shlomo Angel

Patterns


Design Patterns
Elements of Reusable Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Cover art © 1994 M.C. Escher / Cordon Art - Baarn - Holland. All rights reserved.
Foreword by Grady Booch
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

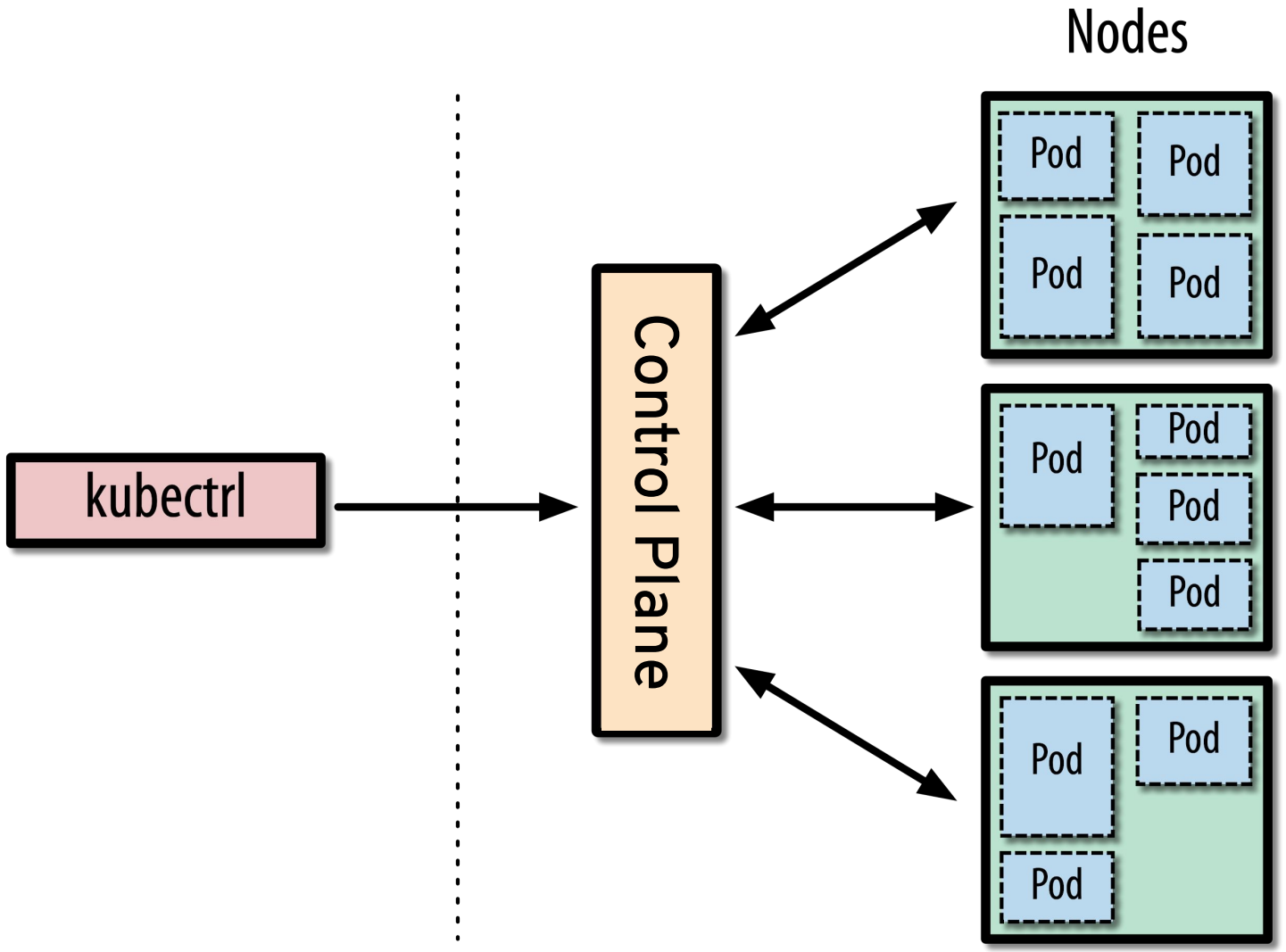

Kubernetes

Kubernetes

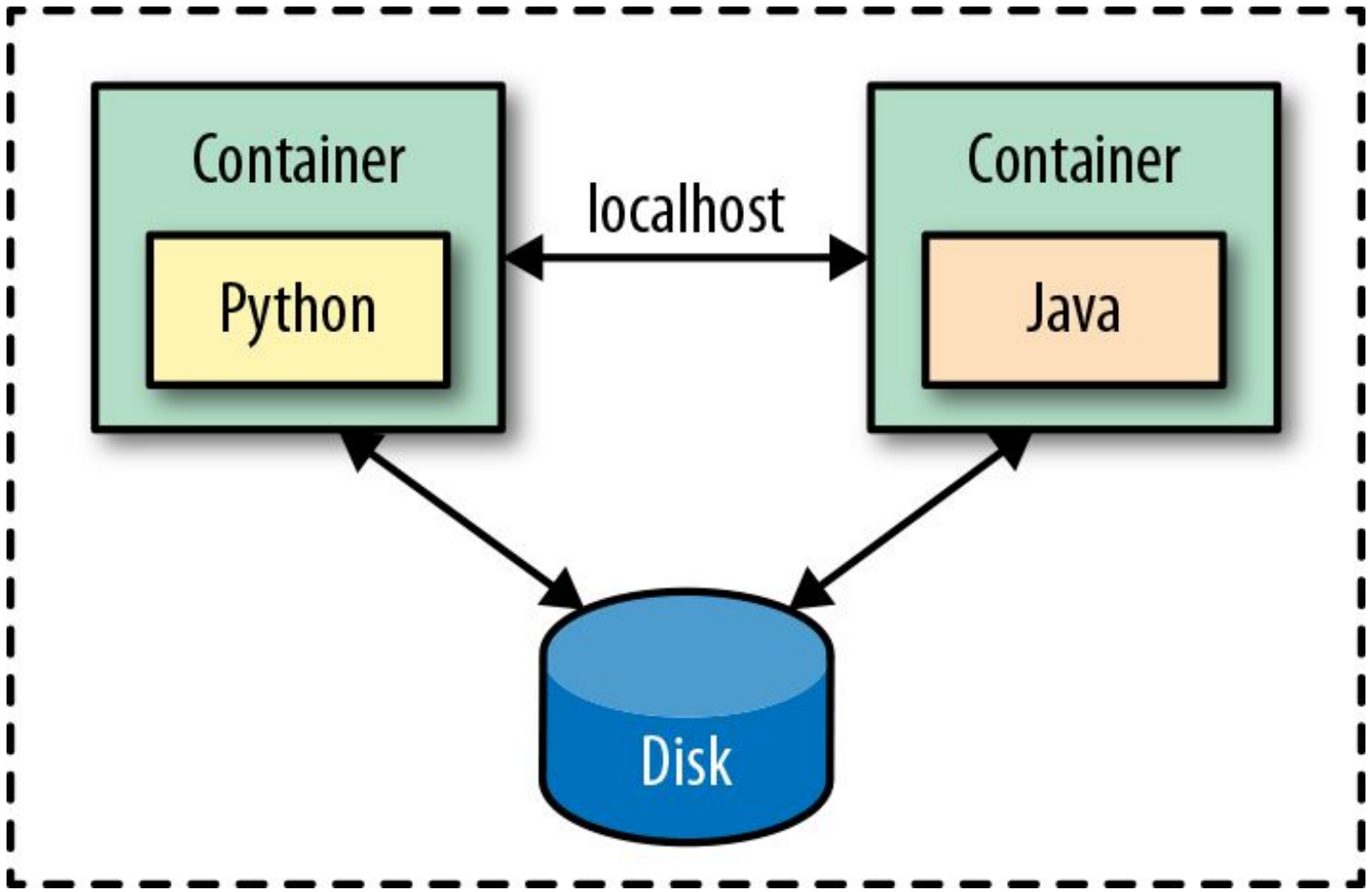


- Open Source container orchestration system started by Google in 2014
  - Scheduling
  - Self-healing
  - Horizontal and vertical scaling
  - Service discovery
  - Application Deployments
- Declarative resource-centric REST API

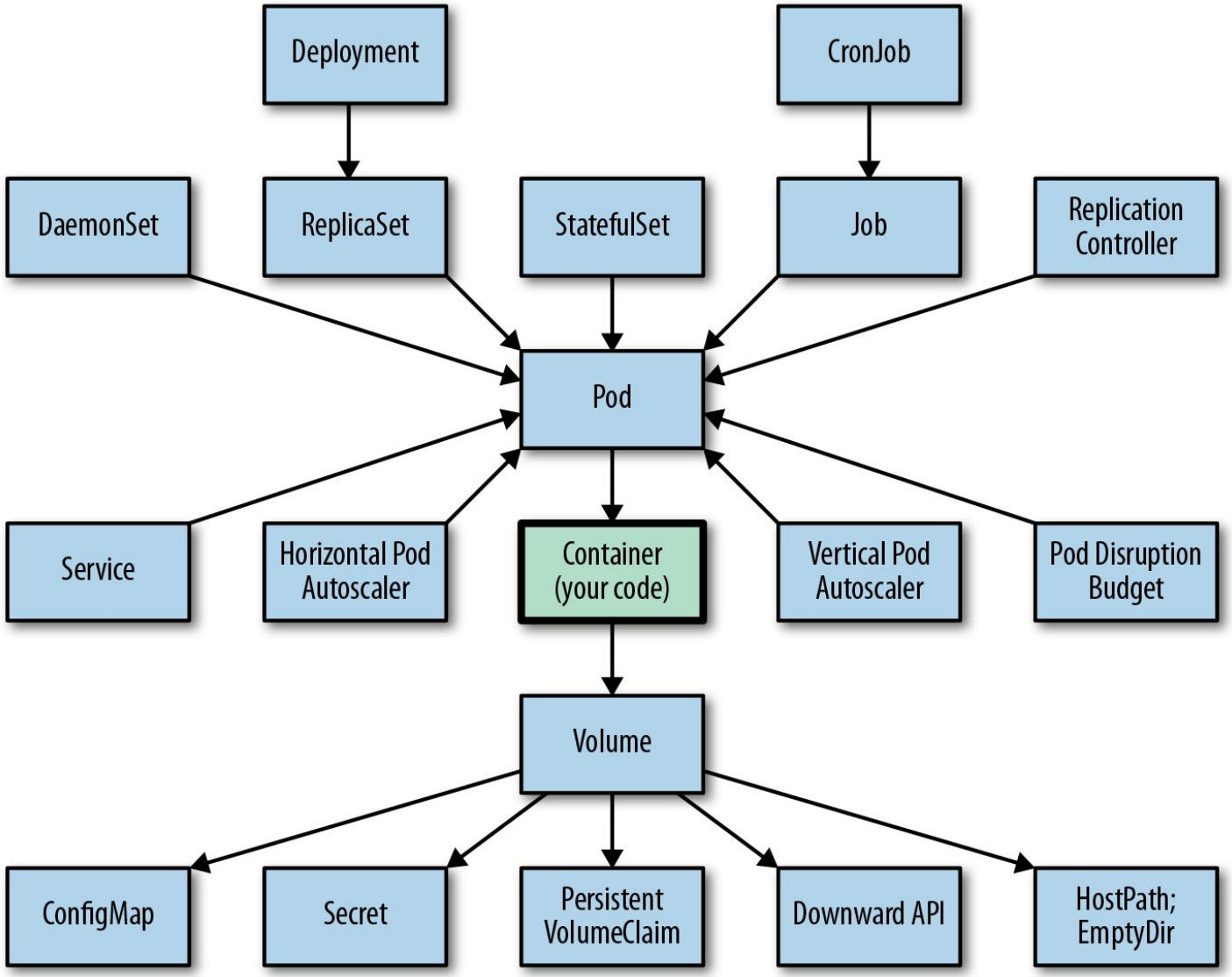
Architecture



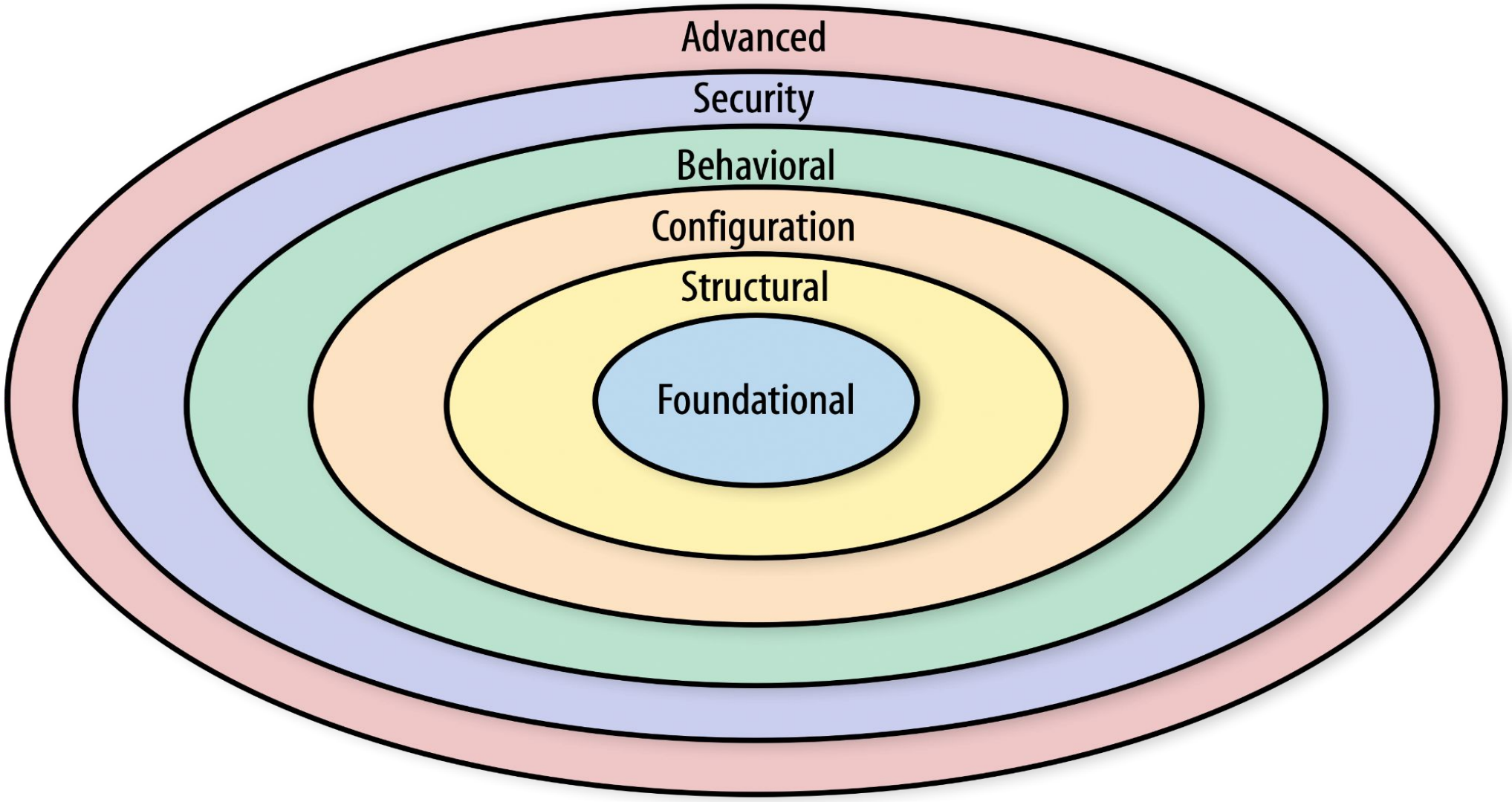
Pod



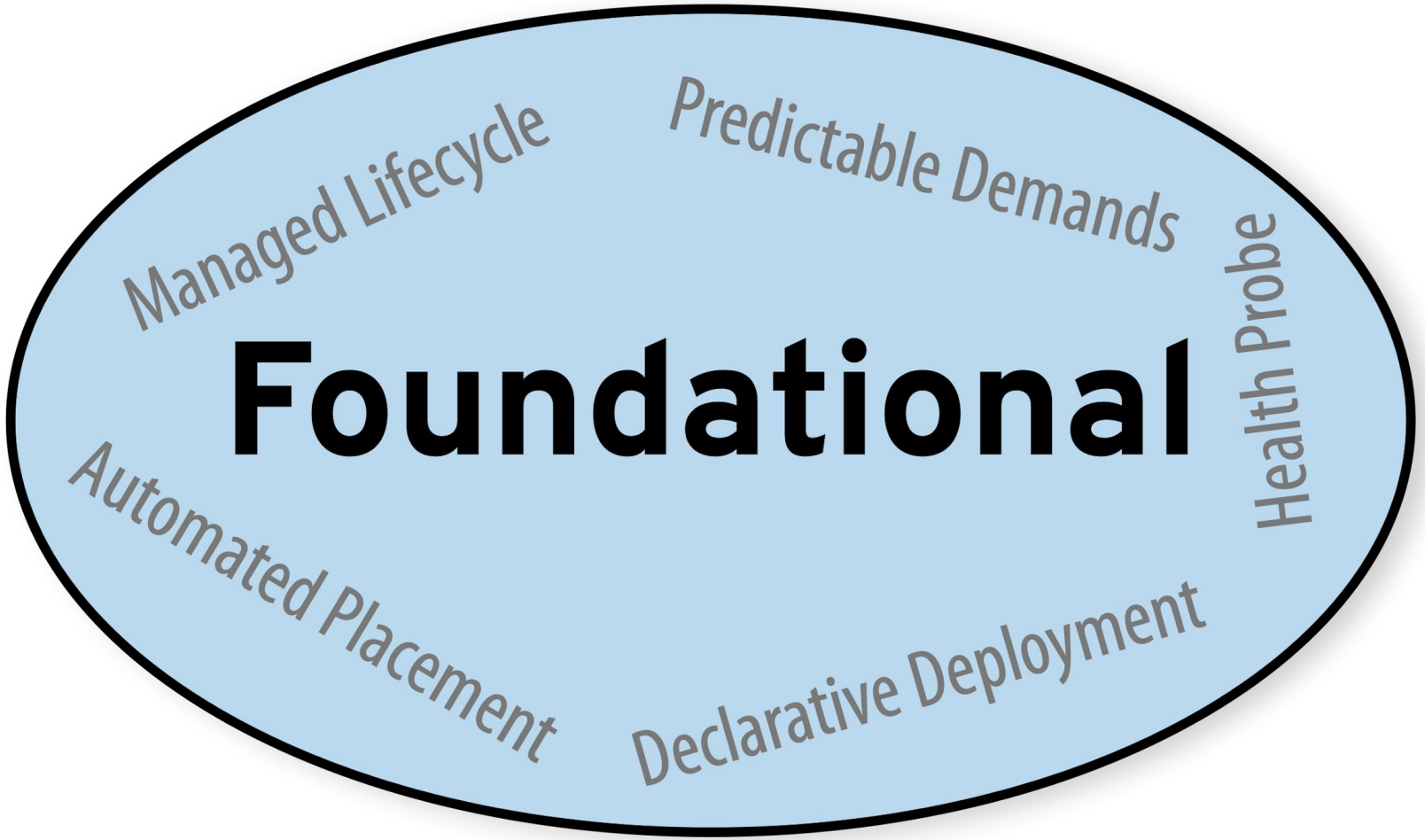
Resources


Deployment
CronJob
DaemonSet
ReplicaSet
StatefulSet
Job
Replication Controller
Pod
Service
Horizontal Pod Autoscaler
Container (your code)
Vertical Pod Autoscaler
Pod Disruption Budget
Volume
ConfigMap
Secret
Persistent VolumeClaim
Downward API
HostPath; EmptyDir

Pattern Categories



Foundational Patterns

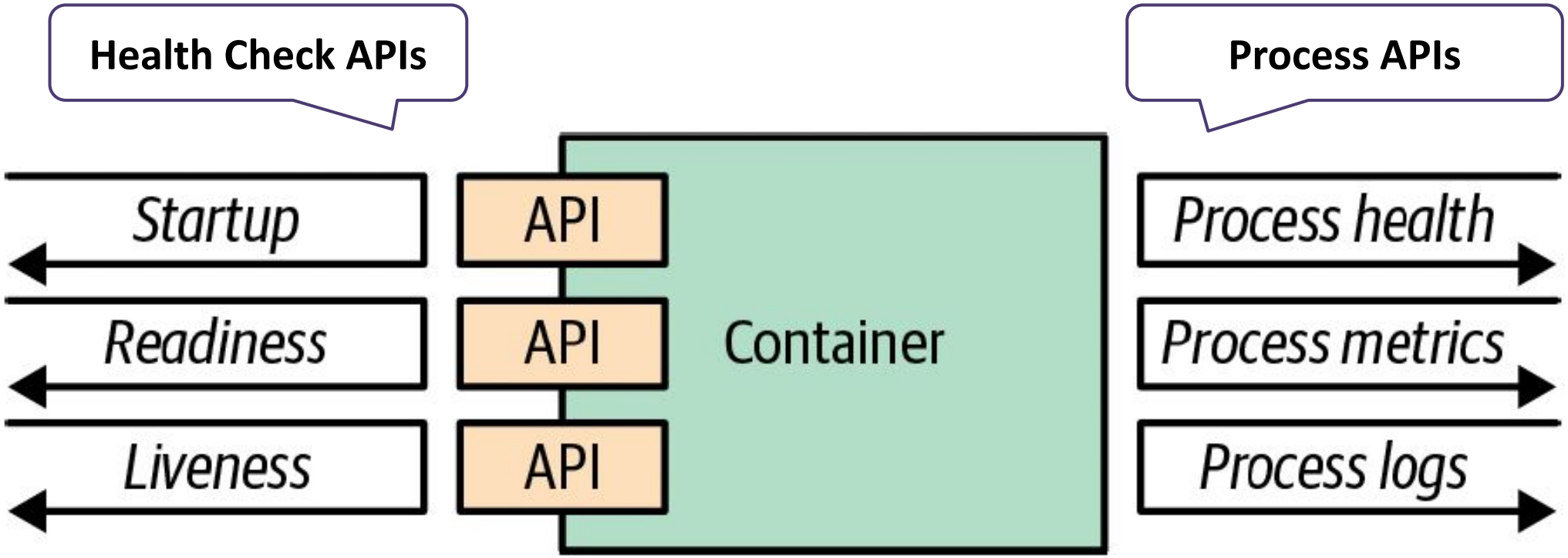



Health
Probe

Health Probe

How to communicate an application's health state to Kubernetes

Container Observability Options



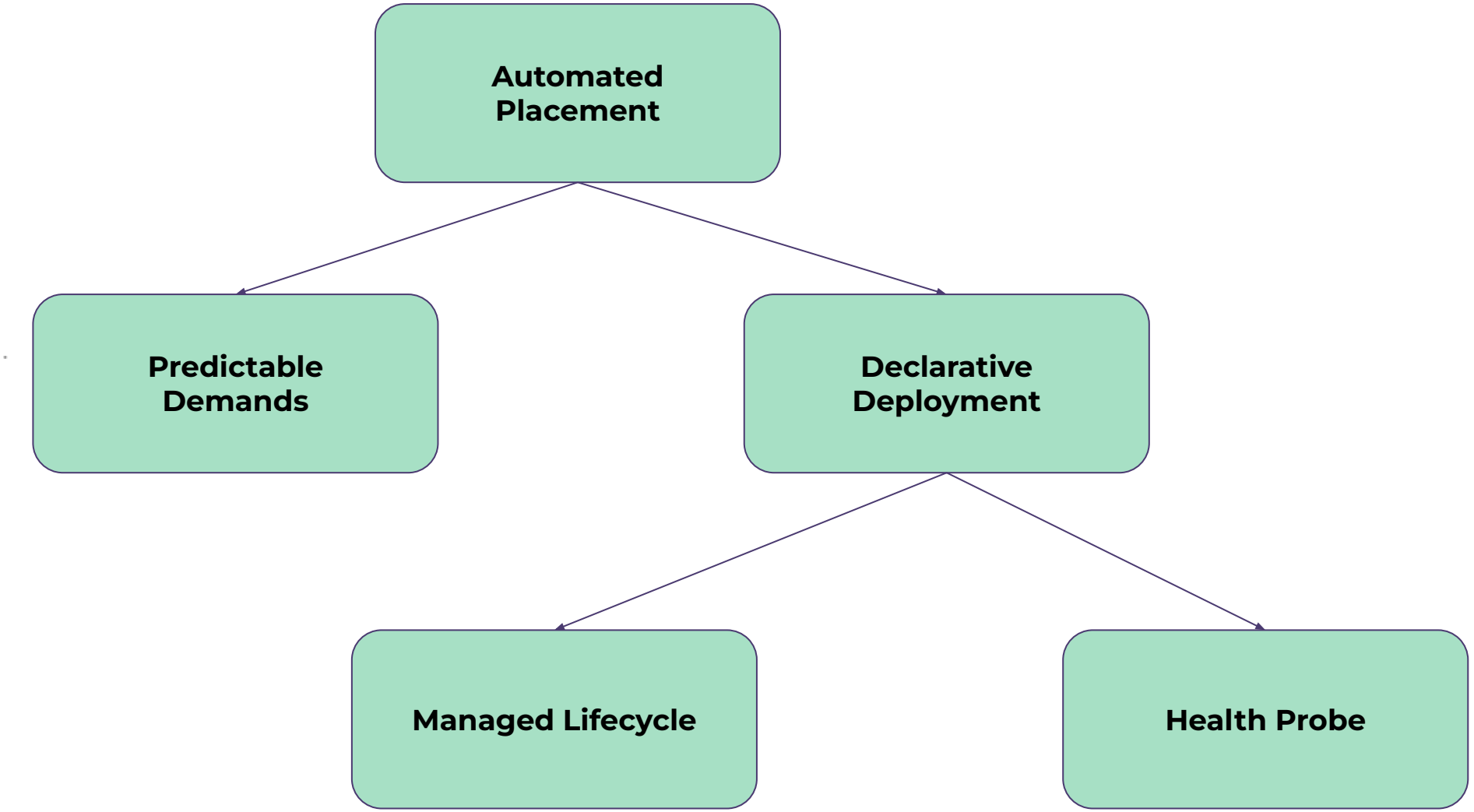
Liveness & Readiness

- Liveness Probe
  - Restarting containers if liveness probes fails
- Readiness Probe
  - Removing from service endpoint if readiness probe fails
- Startup Probe
  - Restarting containers if startup probes fails
- Probe methods
  - HTTP endpoint
  - gRPC endpoint
  - TCP socket endpoint
  - Unix command return value

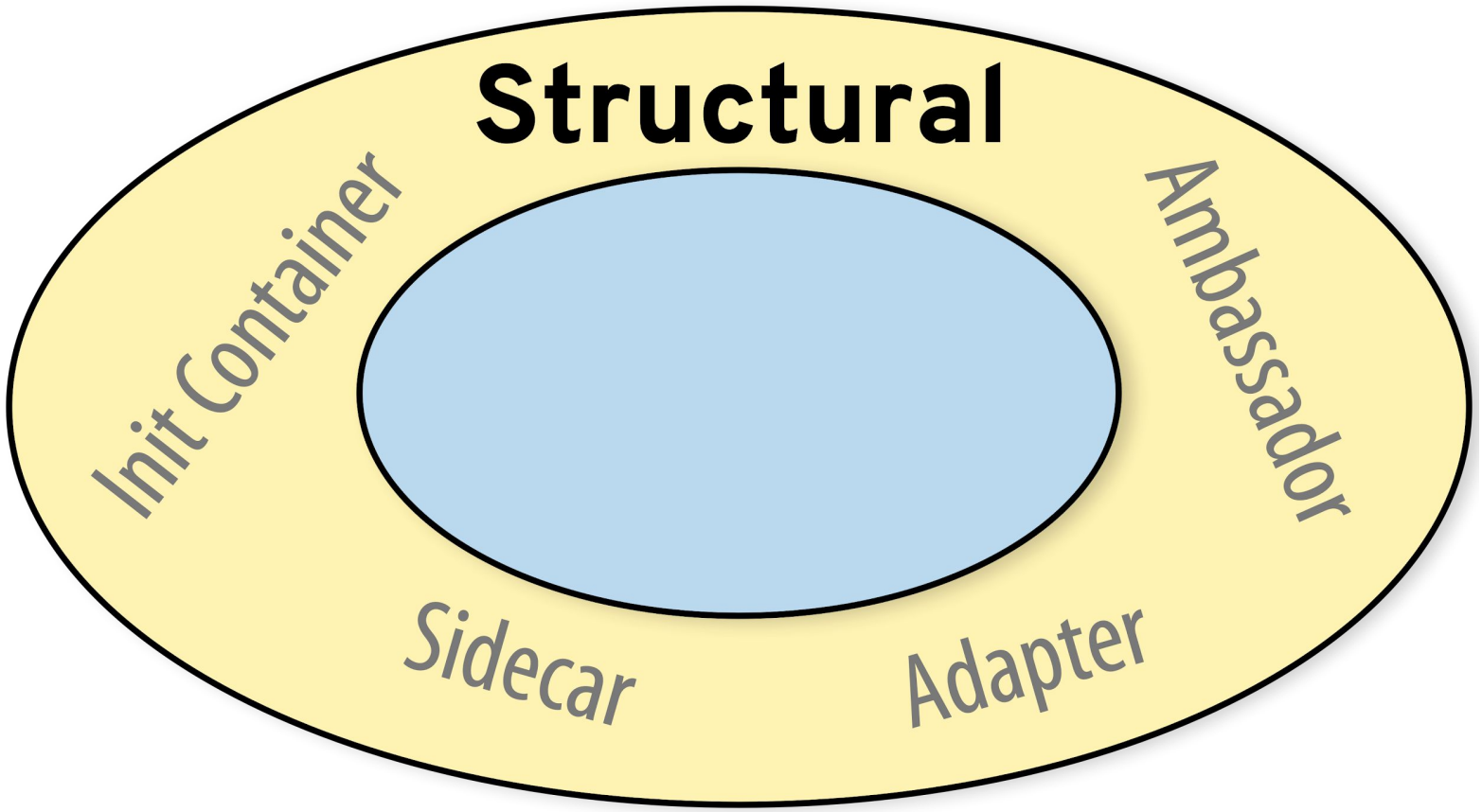
Health Probe Example


```
containers:
- image: k8spatterns/random-generator:1.0
  name: random-generator
  livenessProbe:
    httpGet:
      path: /actuator/health
      port: 8080
  readinessProbe:
    grpc:
      port: 5000
  startupProbe:
    exec:
      command: [ "stat", "/opt/jboss/wildfly/tmp/startup-marker"]
    initialDelaySeconds: 30
    periodSeconds: 60
    failureThreshold: 15
```

Foundational Patterns Dependencies



Structural Patterns





Sidecar

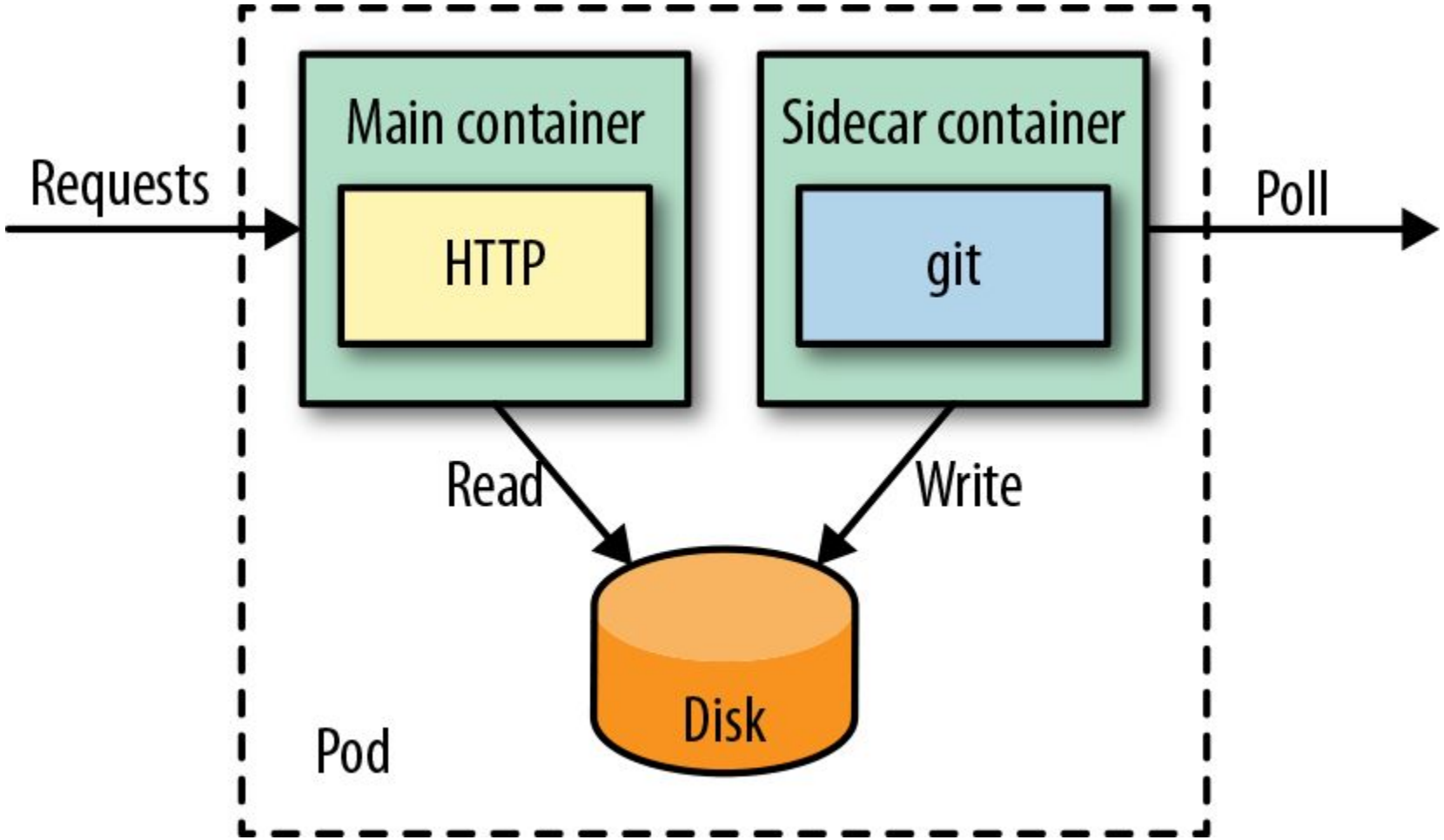
Sidecar

How to enhance the functionality of an application without changing it

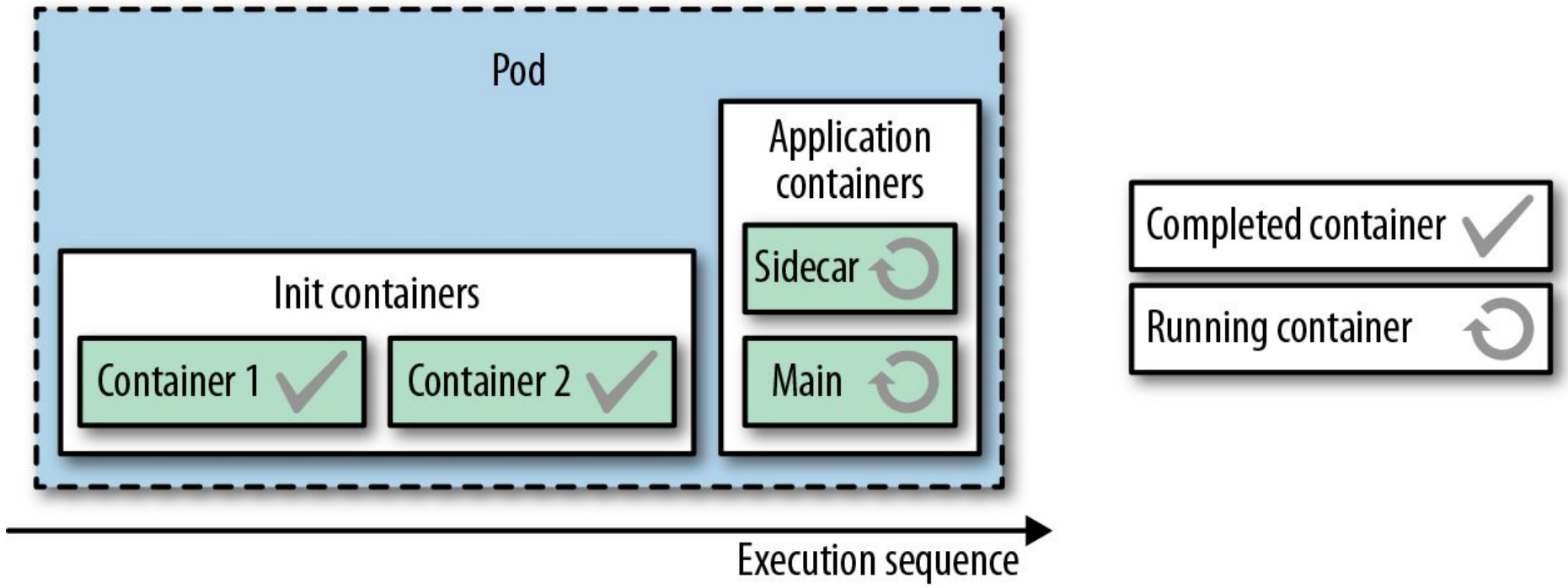
Sidecar

- Runtime collaboration of containers
- Connected via shared resources:
  - Network
  - Volumes
- Similar what AOP is for programming
- Separation of concerns

Sidecar

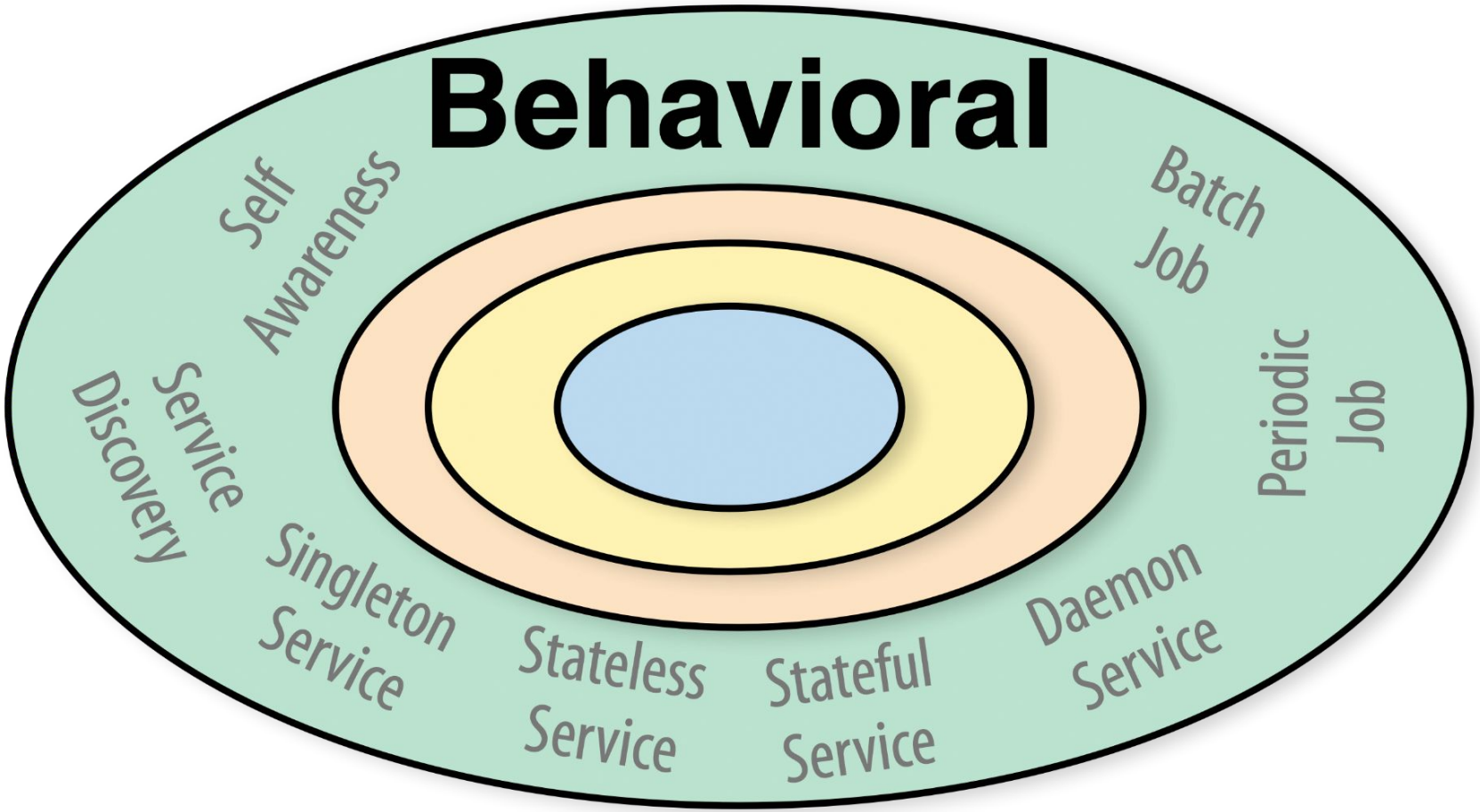

Requests
Main container
HTTP
Sidecar container
git
Poll
Read
Write
Disk
Pod

Startup Order



- Dedicated sidecar containers (since 1.29, beta):
  - `initContainers[].restartPolicy == Always`
  - Continues to run after init containers have finished
  - Supports probes

Behavioral Patterns

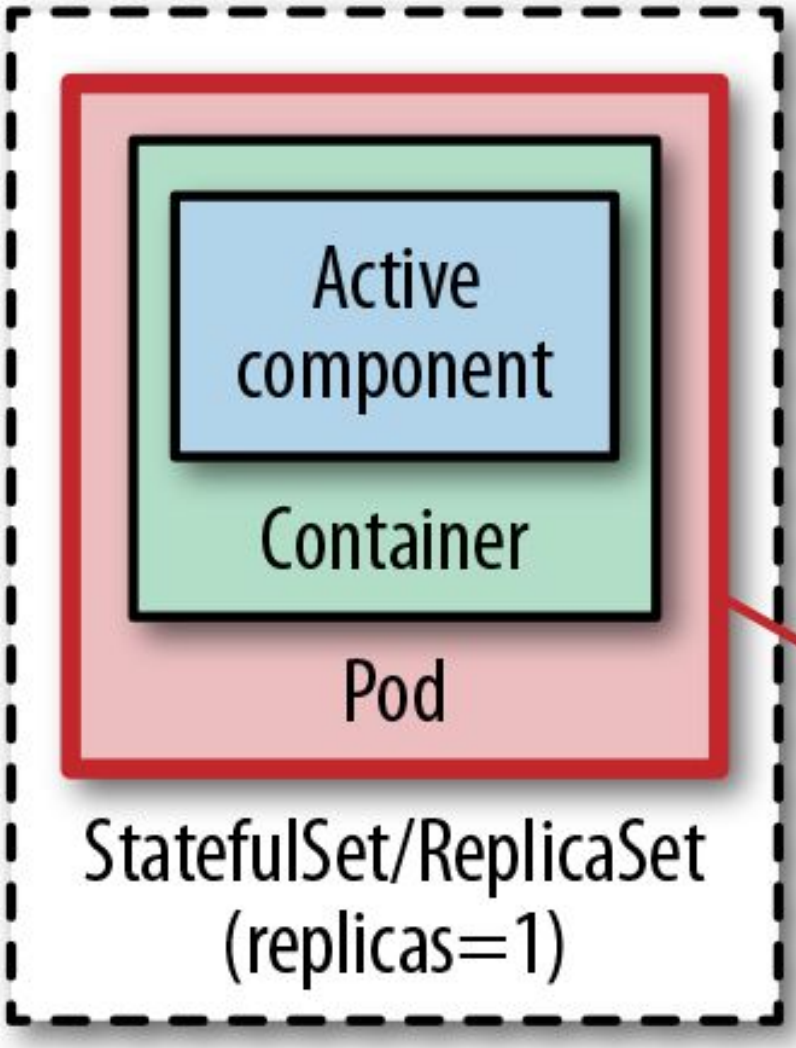


Singleton Service



How to ensure that only one application instance is active

Out-of-Application Locking




```
kind: ReplicaSet
metadata:
  name: file-poller
spec:
  serviceName: file-poller
  replicas: 1
```

Singleton

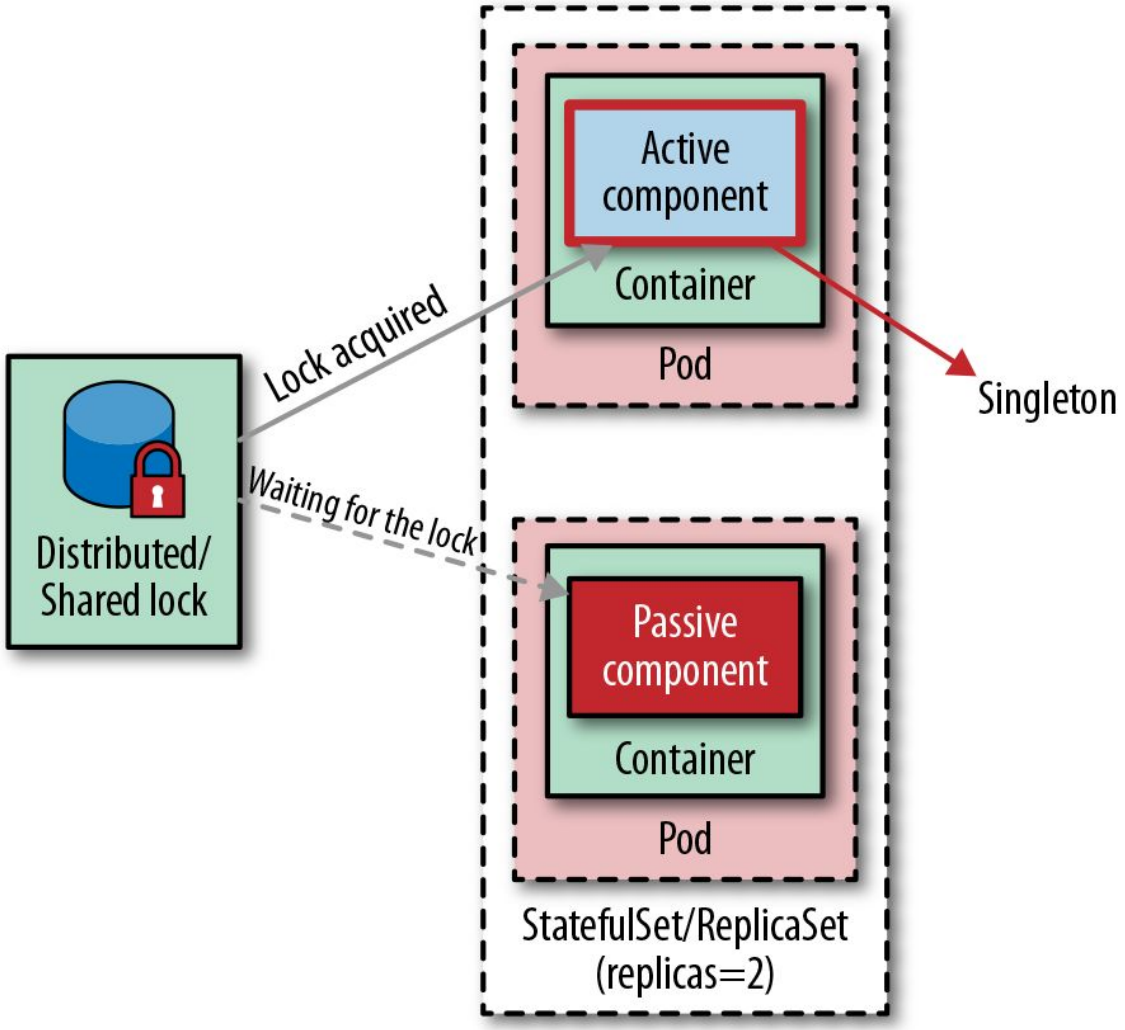
Out-of-Application Locking

- ReplicaSet with **(at least)** 1 replica
  - Highly available Pod which is monitored and restarted in case of failures
  - Favors availability over consistency

    → more than one Pod can exists temporarily
- StatefulSet with **(at most)** 1 replica
  - Favors consistency over availability
  - Favors availability over consistency

    → less than one Pod can exists temporarily

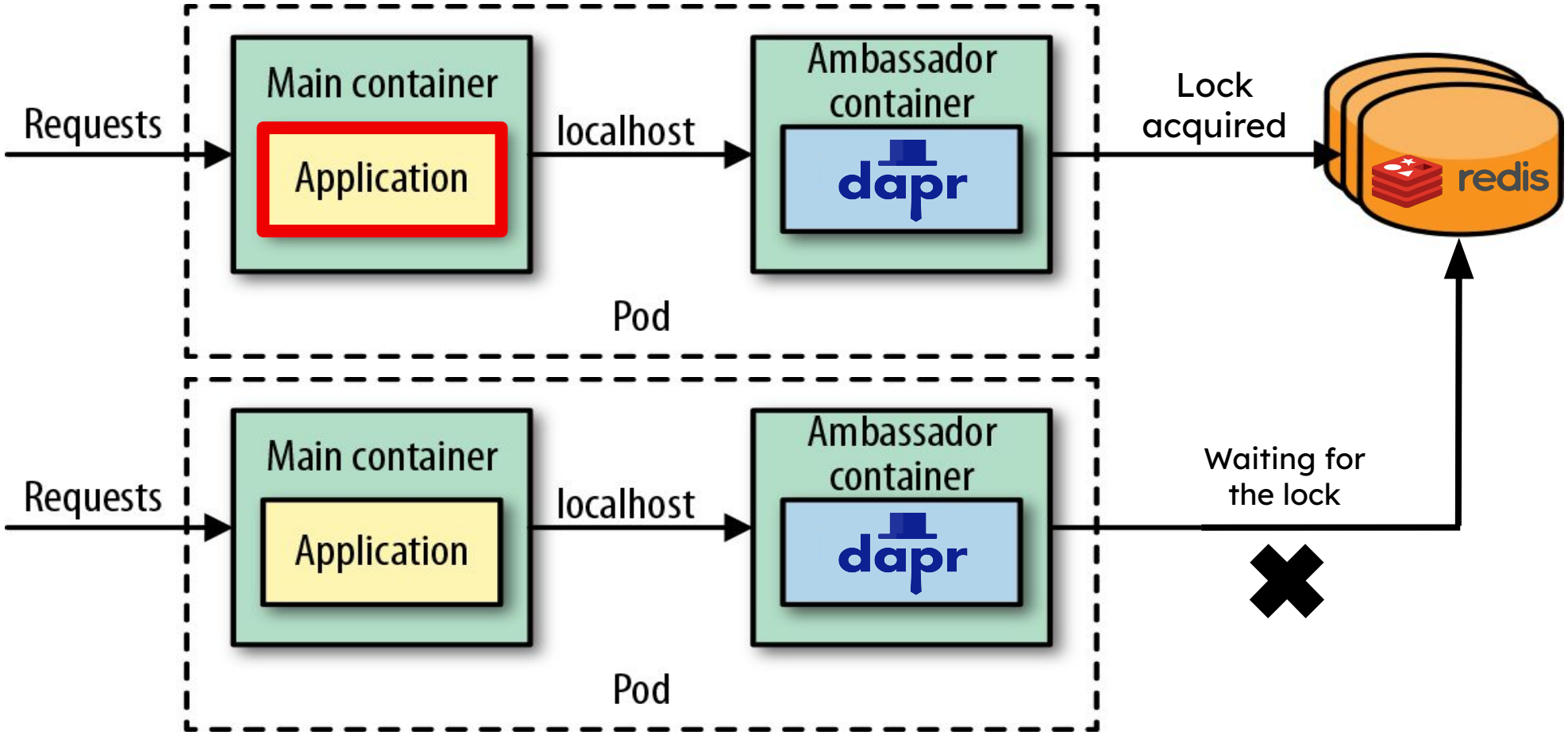
In-Application Locking



In-Application Locking

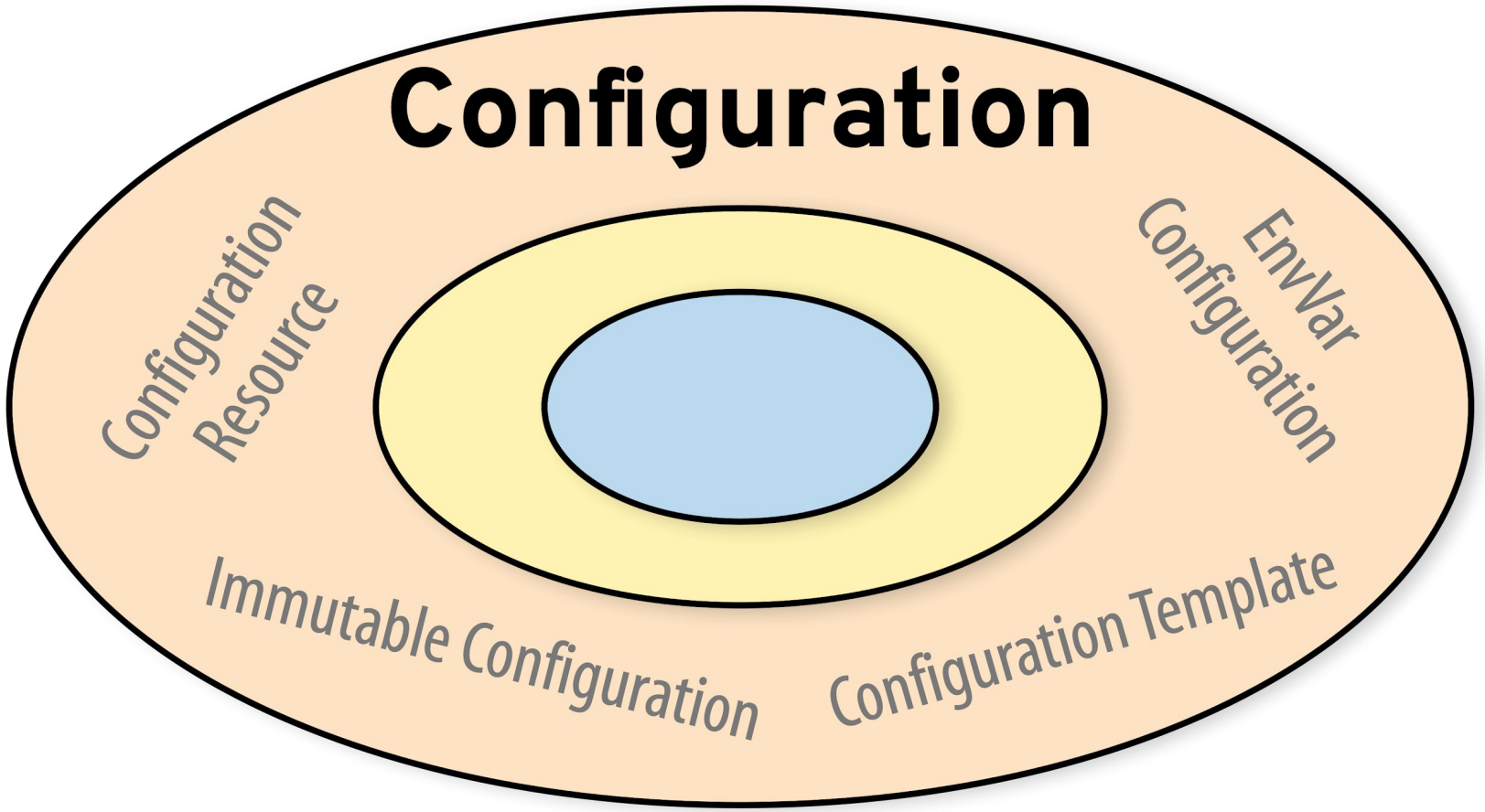
- Distributed lock shared by simultaneously running applications
- *Active-Passive* topology
- Distributed lock implementations e.g.
  - Zookeeper
  - Consul
  - Redis
  - etcd,
  - Kubernetes ConfigMaps, Lease resource

Distributed lock API




```
curl -X POST http://localhost:3500/v1.0-alpha1/lock/lockstore -H 'Content-Type: application/json'
-d ' {"resourceId":"my_file_name", "lockOwner":"abc123", "expiryInSeconds": 60}'
```

Configuration Patterns



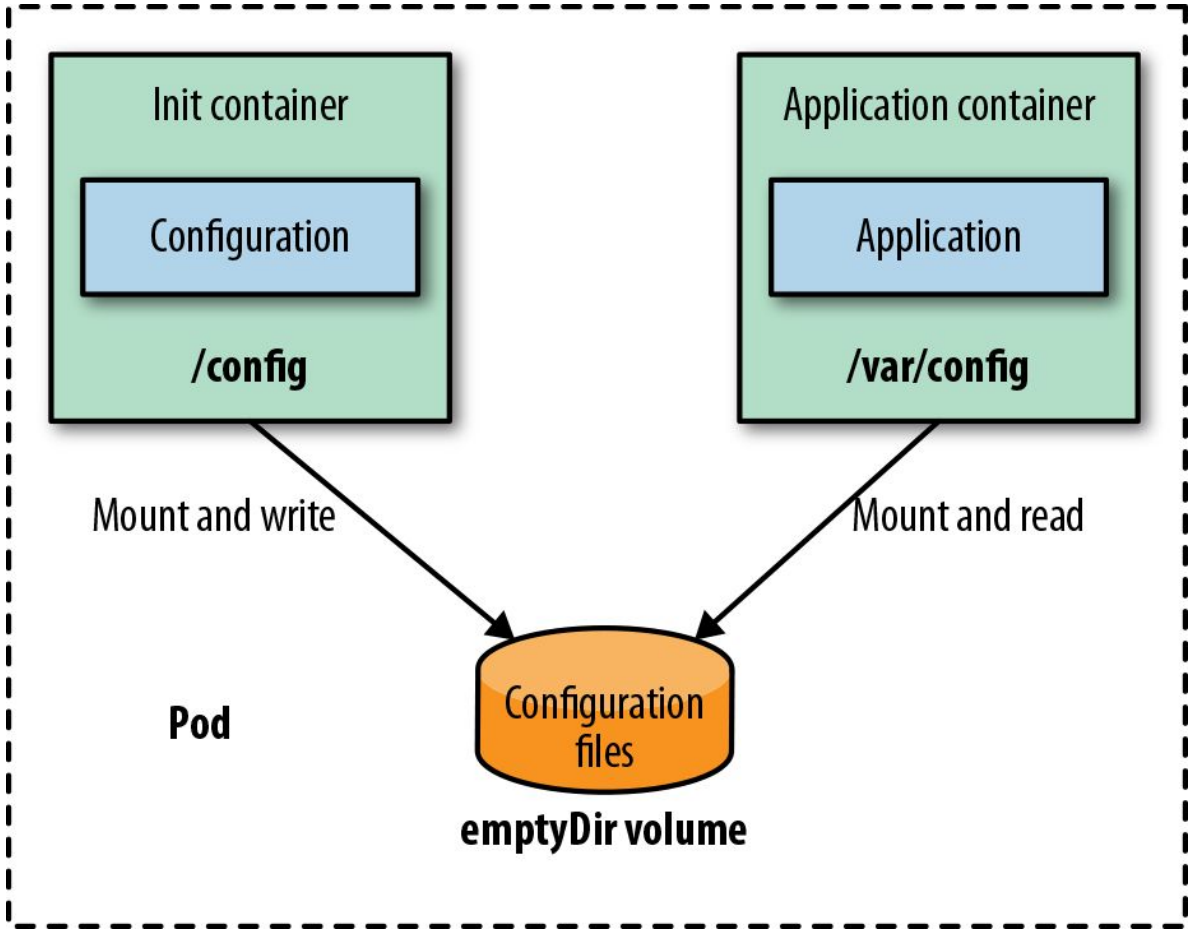

Immutable
Configuration

How to configure your application with immutable container images

Immutable Configuration

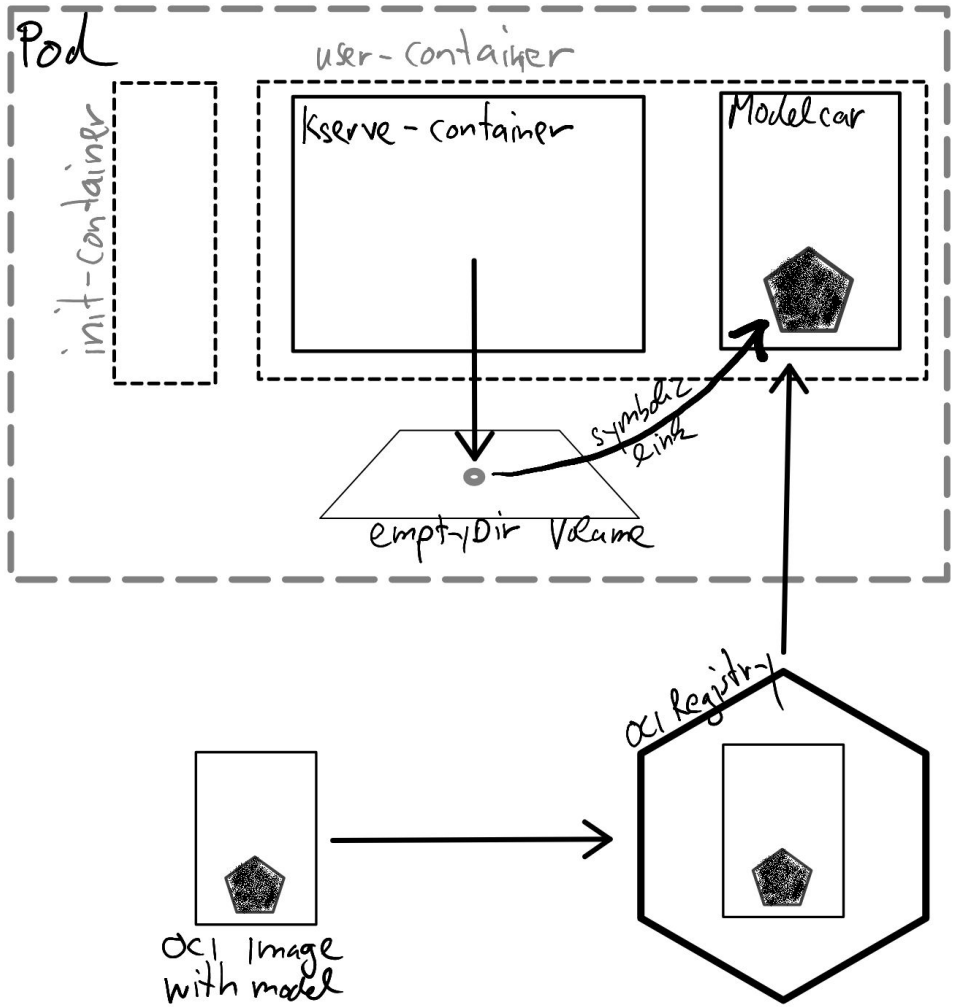
- Configuration that can **not be changed**
- Different sets of configuration for **different environments** (dev/prod)
- **Versioning** and **auditing** for configuration
- **Solution**: Configuration stored in **container images**
  - Versioned via tags and digests
  - Distributed via image registry

Configuration provided by Init Container

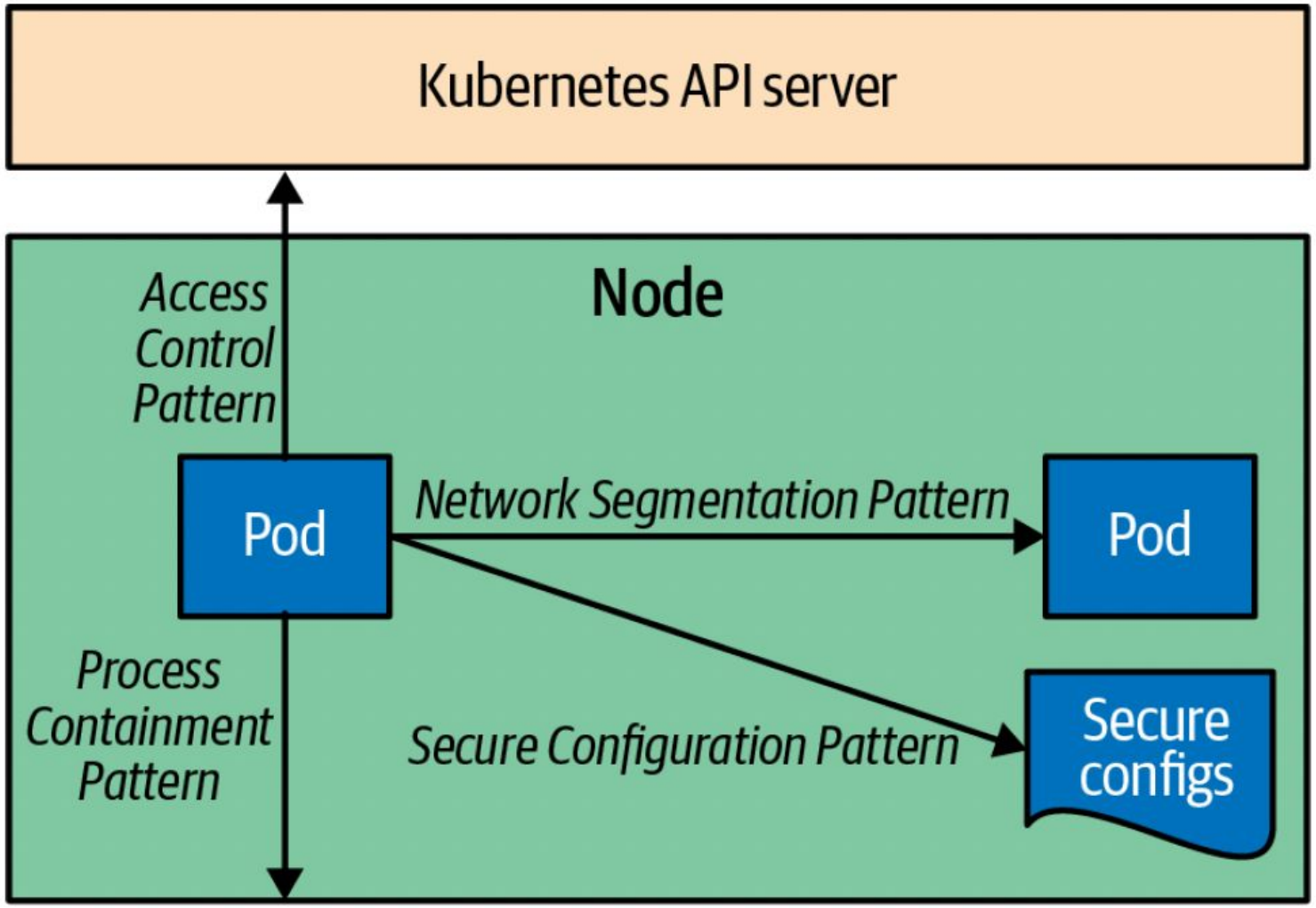


Directly sharing the container filesystem

- Sharing the process namespace, including the file system via **`shareProcessNamespace: true`**
- Technique for directly accessing another containers filesystem via **`/proc/<pid>/root`**
- Used in *modelcars* in KServe for accessing large language models



Security Patterns

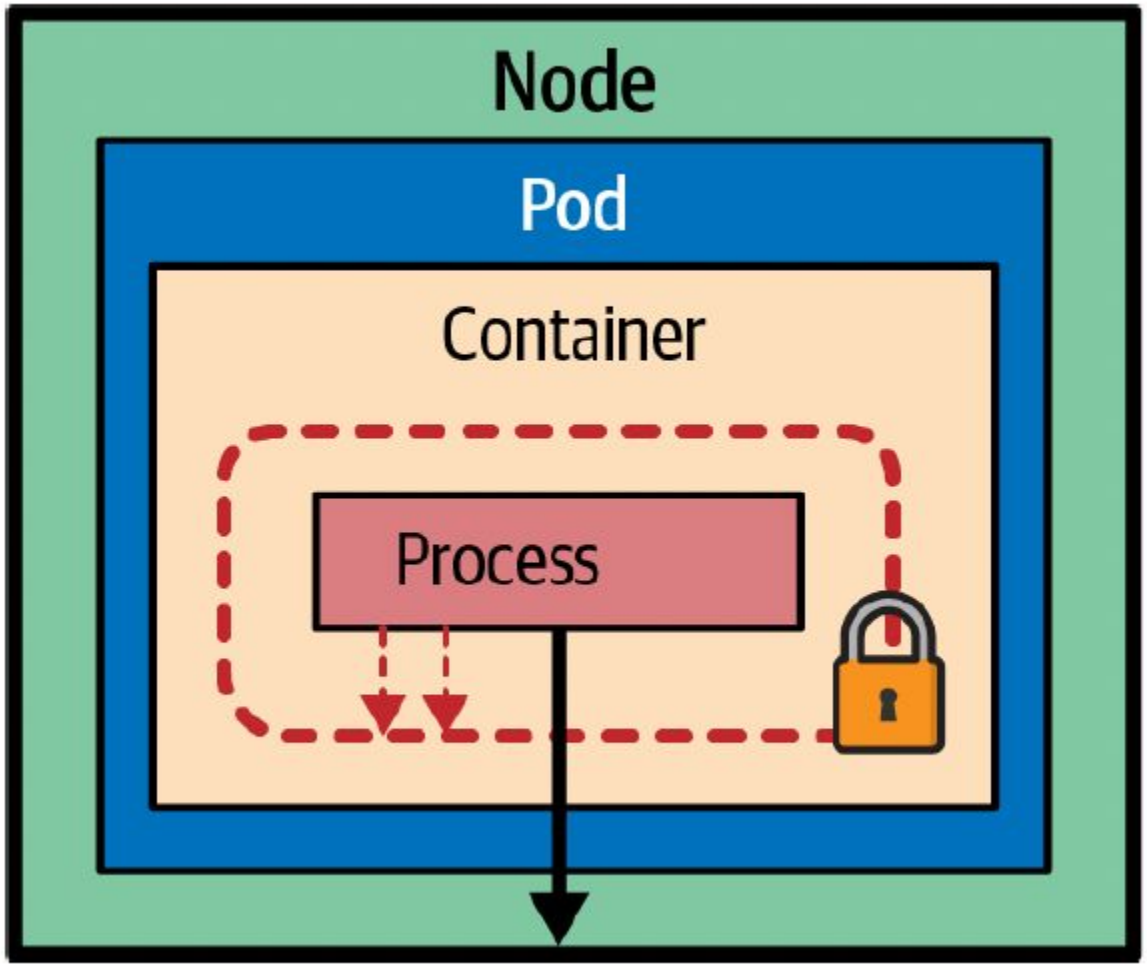




Process Containment

How to protect the platform against deployed code

Security Patterns



Run Containers with a Non-Root User

Forces any container in the Pod to run with user ID 1000 and group ID 2000

```
apiVersion: v1
kind: Pod
metadata:
  name: web-app
spec:
  securityContext:
    runAsUser: 1000
    runAsGroup: 2000
  containers:
  - name: app
```

Prevent any container from starting with a root user—that is, a user with UID 0.

```
apiVersion: v1
kind: Pod
metadata:
  name: web-app
spec:
  securityContext:
    runAsNonRoot: true
  containers:
  - name: app
```

Lock the Capabilities

Prevents privilege escalation

```
apiVersion: v1
kind: Pod
metadata:
  name: web-server
spec:
  containers:
  - name: httpd
    image: httpd
    securityContext:
      allowPrivilegeEscalation: false
```

Reduce capabilities to required ones only

```
apiVersion: v1
kind: Pod
metadata:
  name: web-server
spec:
  securityContext:
    capabilities
      drop: ['ALL']
      add: ['NET_BIND_SERVICE']
  containers:
  - name: httpd
```

Avoiding a Mutable Container Filesystem

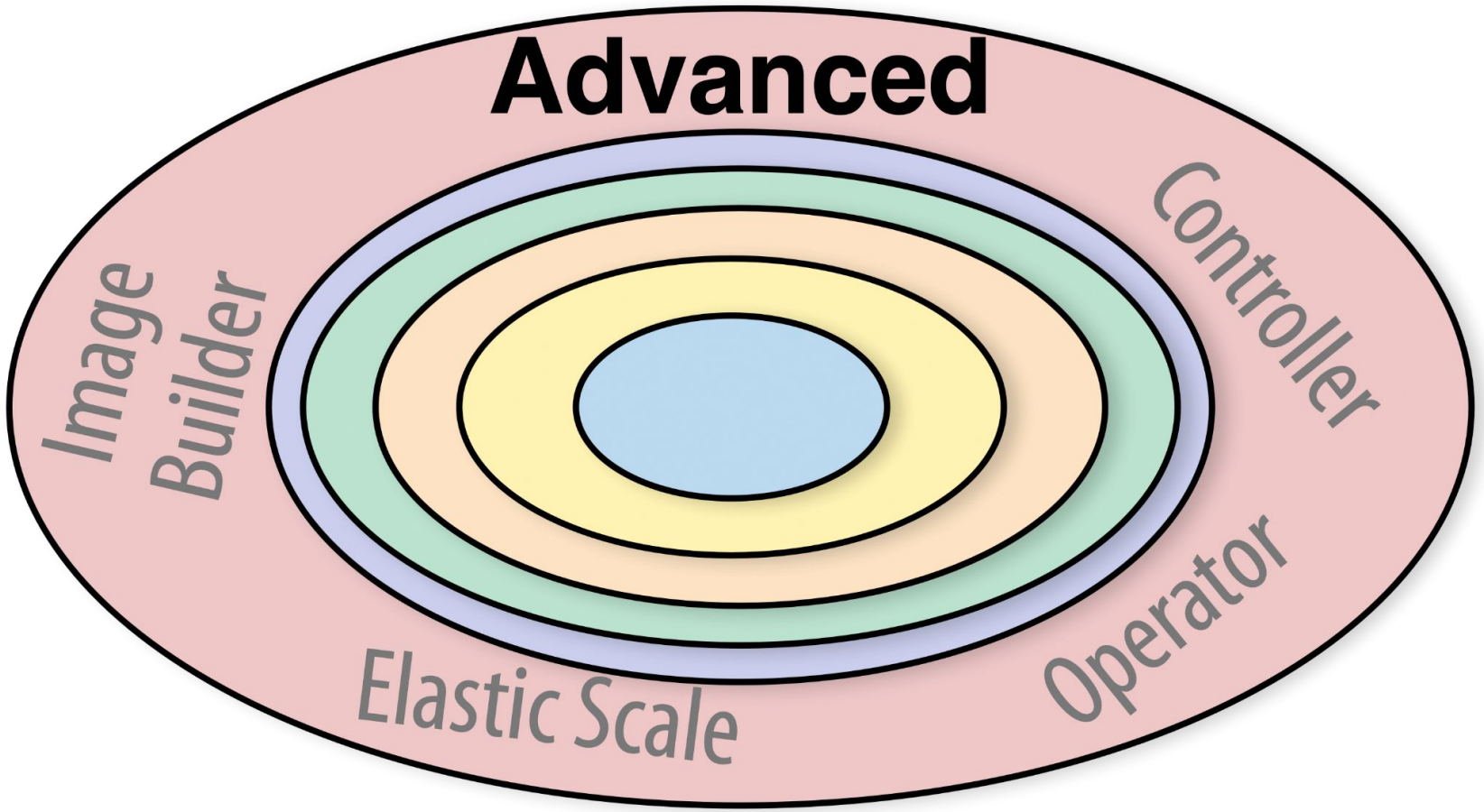
Set read only file system

```
apiVersion: v1
kind: Pod
metadata:
  name: web-server
spec:
  containers:
  - name: httpd
    image: httpd
    securityContext:
      readOnlyRootFilesystem: true
```

Overview of options we covered:

```
apiVersion: v1
kind: Pod
metadata:
  name: web-server
spec:
  containers:
  - name: httpd
    image: httpd
    securityContext:
      runAsNonRoot: true
      allowPrivilegeEscalation: false
      readOnlyRootFilesystem: true
      capabilities:
        drop: - ALL
        add: ['NET_BIND_SERVICE']
```

Advanced Patterns



Controller



How to get from the current state to a declared target state

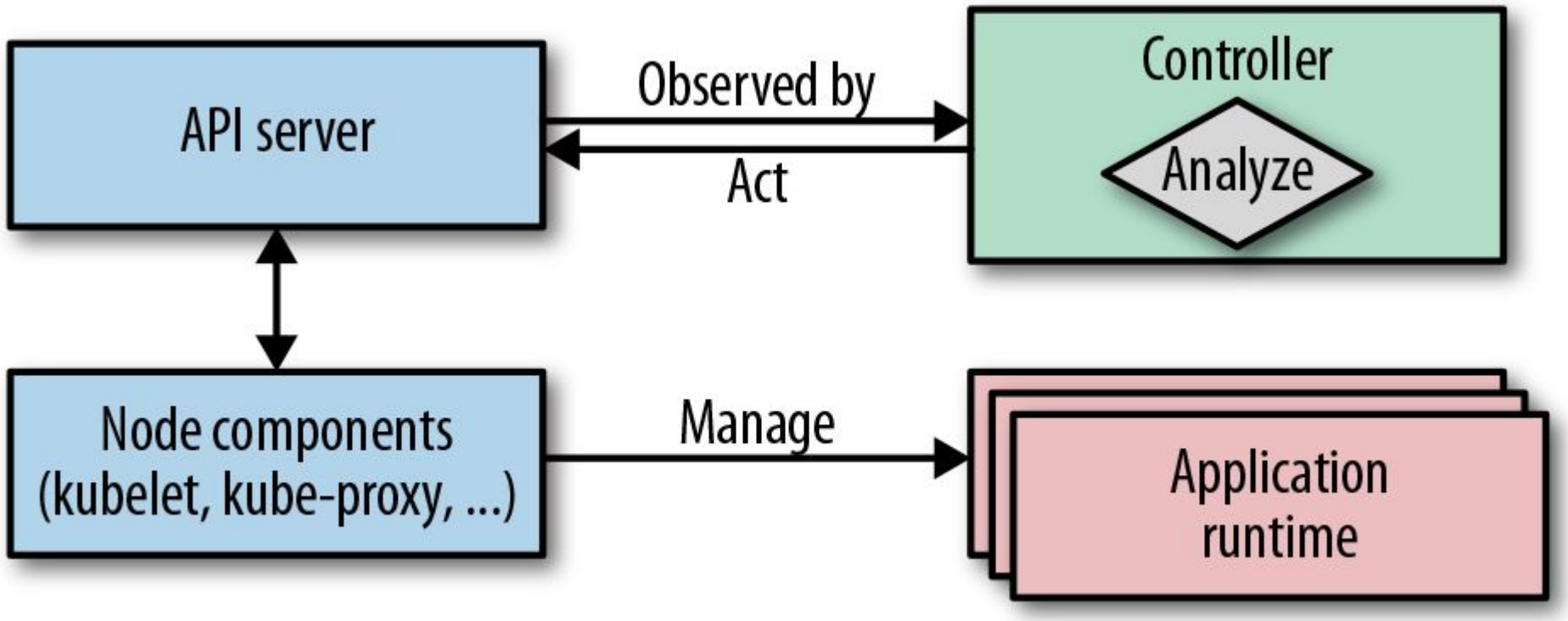
State Reconciliation

- Kubernetes as distributed state manager

- Make the **actual** state more like the declared **target** state.

🔎 ***Observe*** - Discover the actual state
🤔 ***Analyze*** - Determine difference to target state
🔨 ***Act-*** Perform actions to drive the actual to the desired state

Observe - Analyze - Act




Operator

How to encapsulate operational knowledge into executable software

Definition

> An **operator** is a Kubernetes **controller** that understands two domains: Kubernetes and *something else*. By combining knowledge of both areas, it can **automate** tasks that usually require a human operator that understands both domains.
>
> **Jimmy Zelinskie**
> http://bit.ly/2Fjlx1h

Operator = Controller + CustomResourceDefinition

CustomResourceDefinition

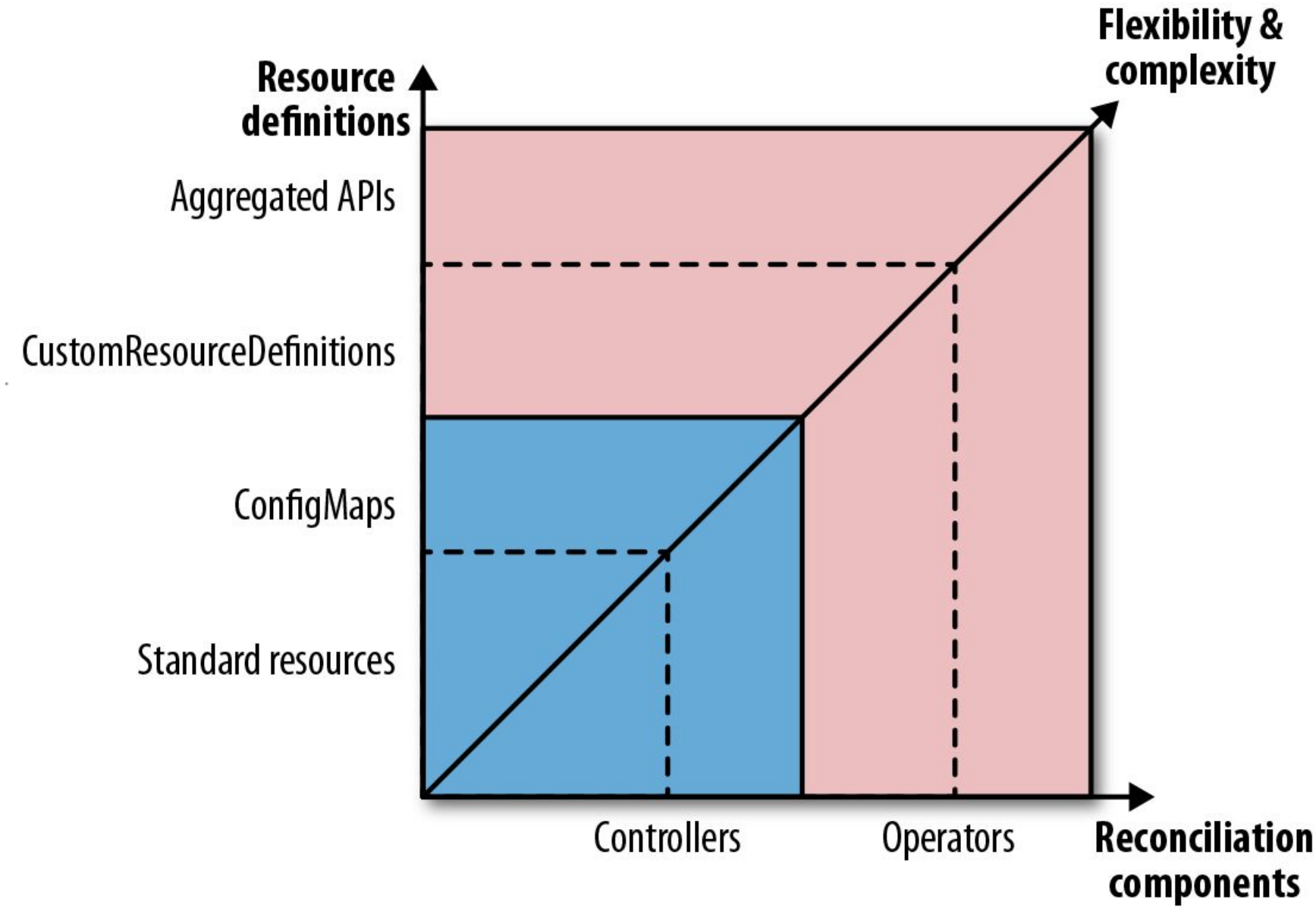
Custom resource is modelling a custom domain and managed through the Kubernetes API

```
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  name: configwatchers.k8spatterns.io
spec:
  scope: Namespaced
  group: k8spatterns.io
  version: v1
  names:
    kind: ConfigWatcher
    plural: configwatchers
  validation:
    openAPIV3Schema:
    ...
```

Custom Resource


```
kind: ConfigWatcher
apiVersion: k8spatterns.io/v1
metadata:
  name: webapp-config-watcher
spec:
  configMap: webapp-config
  podSelector:
    app: webapp
```

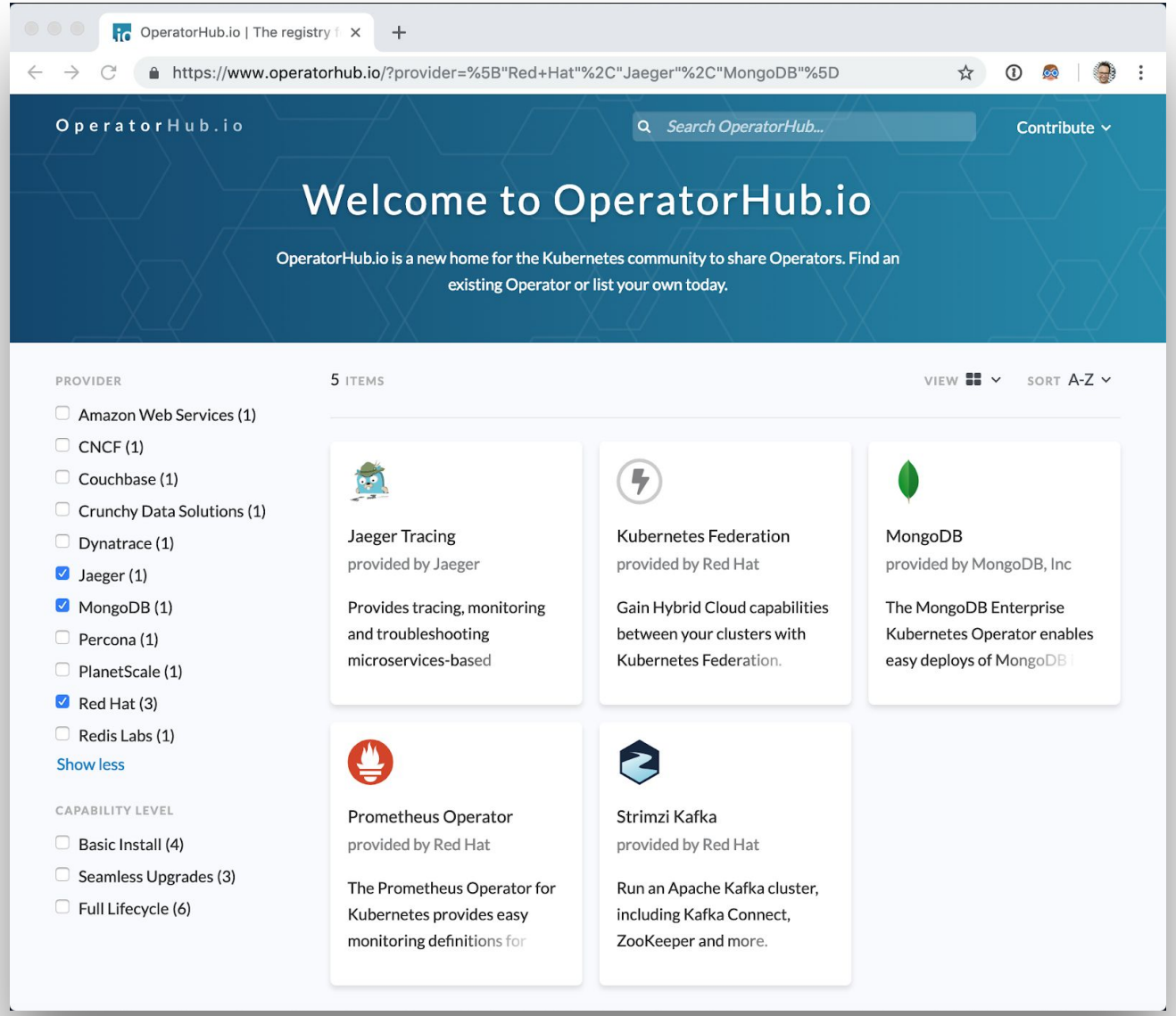
Operator


Flexibility & complexity
Resource definitions
Aggregated APIs
CustomResourceDefinitions
ConfigMaps
Standard resources
Controllers
Operators
Reconciliation components

CRD Classification

- Installation CRDs
  - Installing and operating applications
  - Backup and Restore
  - Monitoring and self-healing
  - Example: Prometheus for installing Prometheus & components
- Application CRDs
  - Application specific domain concepts
  - Example: ServiceMonitor for registering Kubernetes service to be scraped by Prometheus

Operator Hub



May the duck be with you ...



Wednesday, 13:00



Thursday, 12:30

https://k8spatterns.com

Picture Credits

https://www.pexels.com/photo/brown-and-black-pattern-2158386/

https://pixabay.com/photos/ship-helm-sunset-cutter-coast-guard-759954/

https://unsplash.com/photos/yo01Z-9HQAw

https://www.pexels.com/photo/turned-on-light-crane-1117211/

https://www.pexels.com/photo/shallow-focus-photography-of-black-ship-1095814/

https://pixabay.com/photos/containers-storage-rusted-rusty-1209079/

https://pixabay.com/photos/motocross-sidecar-race-motorsport-1045661/

https://www.pexels.com/photo/golden-gate-bridge-san-francisco-california-1141853/

https://unsplash.com/photos/M_I-crkDO-k

https://unsplash.com/photos/FqnkBeq1wGg

https://www.pexels.com/de-de/foto/brown-brick-roof-auf-luftaufnahme-3637890/

https://unsplash.com/de/fotos/yfq6c6j-5WI

https://unsplash.com/de/fotos/3wPJxh-piRw

https://unsplash.com/de/fotos/0ji5tjZQ2I4

https://www.pexels.com/de-de/foto/weisse-tasse-auf-dem-tisch-53405/

https://www.pexels.com/photo/reflection-playstation-pad-gaming-18174/

https://unsplash.com/photos/UJP_QpCKj7M

https://www.pexels.com/photo/grayscale-photo-of-person-holding-chess-piece-1498958/

https://pixabay.com/photos/cans-manufacturing-business-2888650/

https://unsplash.com/photos/lRoX0shwjUQ

https://www.freeimages.com/photo/poppy-in-wheat-1344010

https://pixabay.com/photos/telescope-field-glass-spyglass-1966366/

https://unsplash.com/photos/UHDx3BHIFvY

https://pixabay.com/photos/bake-advent-christmas-cookie-1786926/

https://pixabay.com/photos/balloons-colors-party-celebration-1869790/

https://unsplash.com/photos/ymf4_9Y9S_A

https://unsplash.com/photos/ZUabNmumOcA

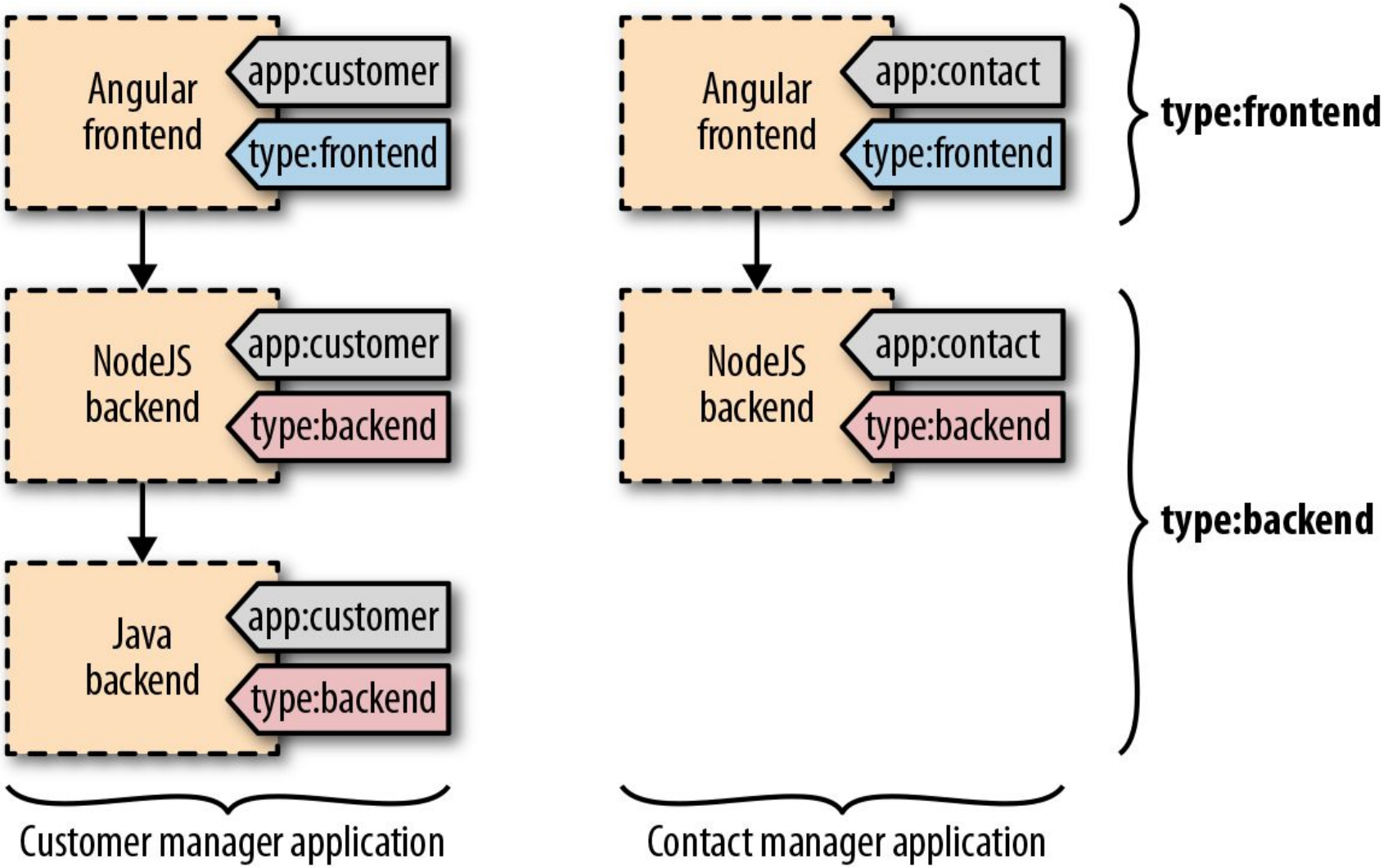
https://pixabay.com/photos/lost-places-machines-old-factory-3991951/

https://pixabay.com/images/id-407101/

https://www.pexels.com/photo/backlit-beach-dawn-dusk-588561/

Red Hat

Labels


Angular frontend
app:customer
type:frontend
NodeJS backend
app:customer
type:backend
Java backend
app:customer
type:backend
Customer manager application
Angular frontend
app:contact
type:frontend
NodeJS backend
app:contact
type:backend
Contact manager application
type:frontend
type:backend

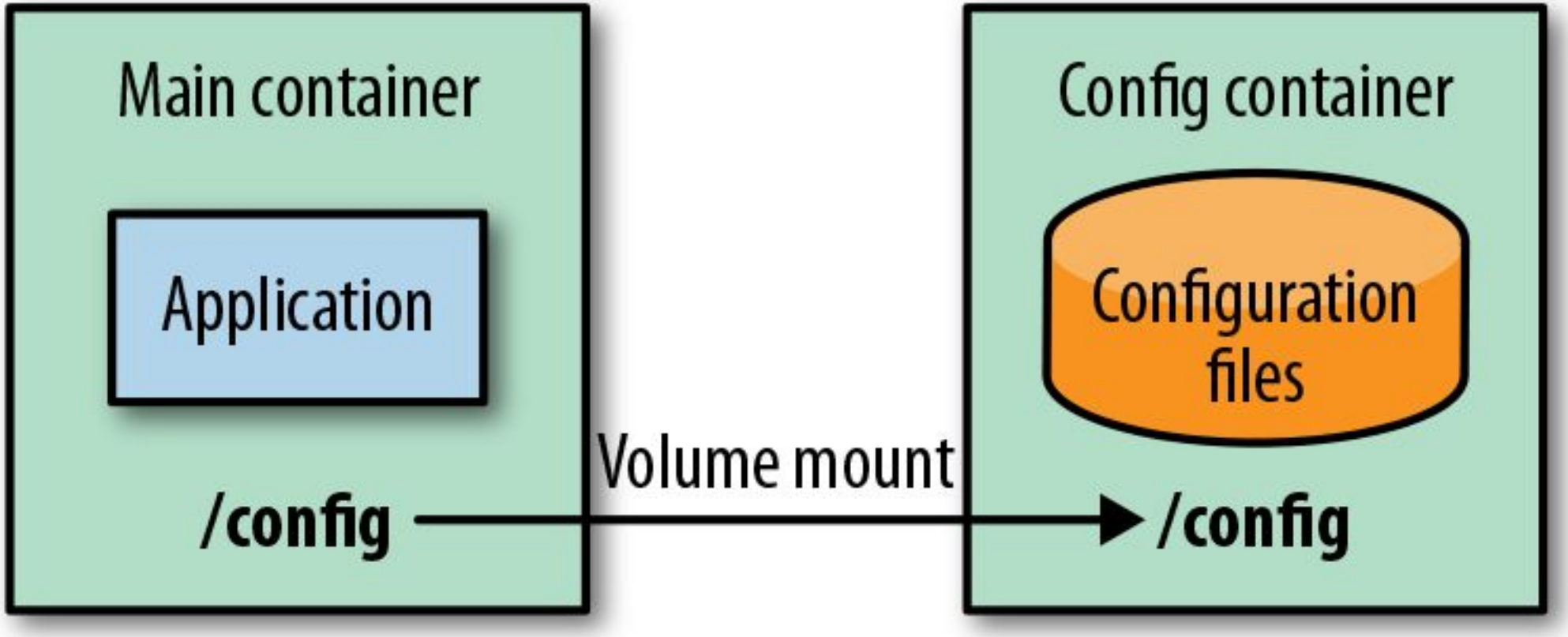
Local vs. Distributed Primitives

Concept	Local primitive	Distributed primitive
Behavior encapsulation	Class	Container image
Behavior instance	Object	Container
Unit of reuse	*.jar*	Container image
Composition	Class A contains Class B	Sidecar pattern
Inheritance	Class A extends Class B	A container's `FROM` parent image
Deployment unit	*.jar*/*.war*/*.ear*	Pod
Buildtime/Runtime isolation	Module, package, class	Namespace, Pod, container
Initialization preconditions	Constructor	Init container
Postinitialization trigger	Init-method	`postStart`
Predestroy trigger	Destroy-method	`preStop`
Cleanup procedure	`finalize()`, shutdown hook	-
Asynchronous and parallel execution	`ThreadPoolExecutor`, `ForkJoinPool`	Job
Periodic task	`Timer`, `ScheduledExecutorService`	CronJob
Background task	Daemon thread	DaemonSet
Configuration management	`System.getenv()`, `Properties`	ConfigMap, Secret

Which Implementation to Choose?

- Out-of-Application Locking
  - If you have good control at runtime
- In-Application Locking
  - If you need advanced features and customization
- Distributed lock API
  - Universal lock API for any language

Docker Volume Mount



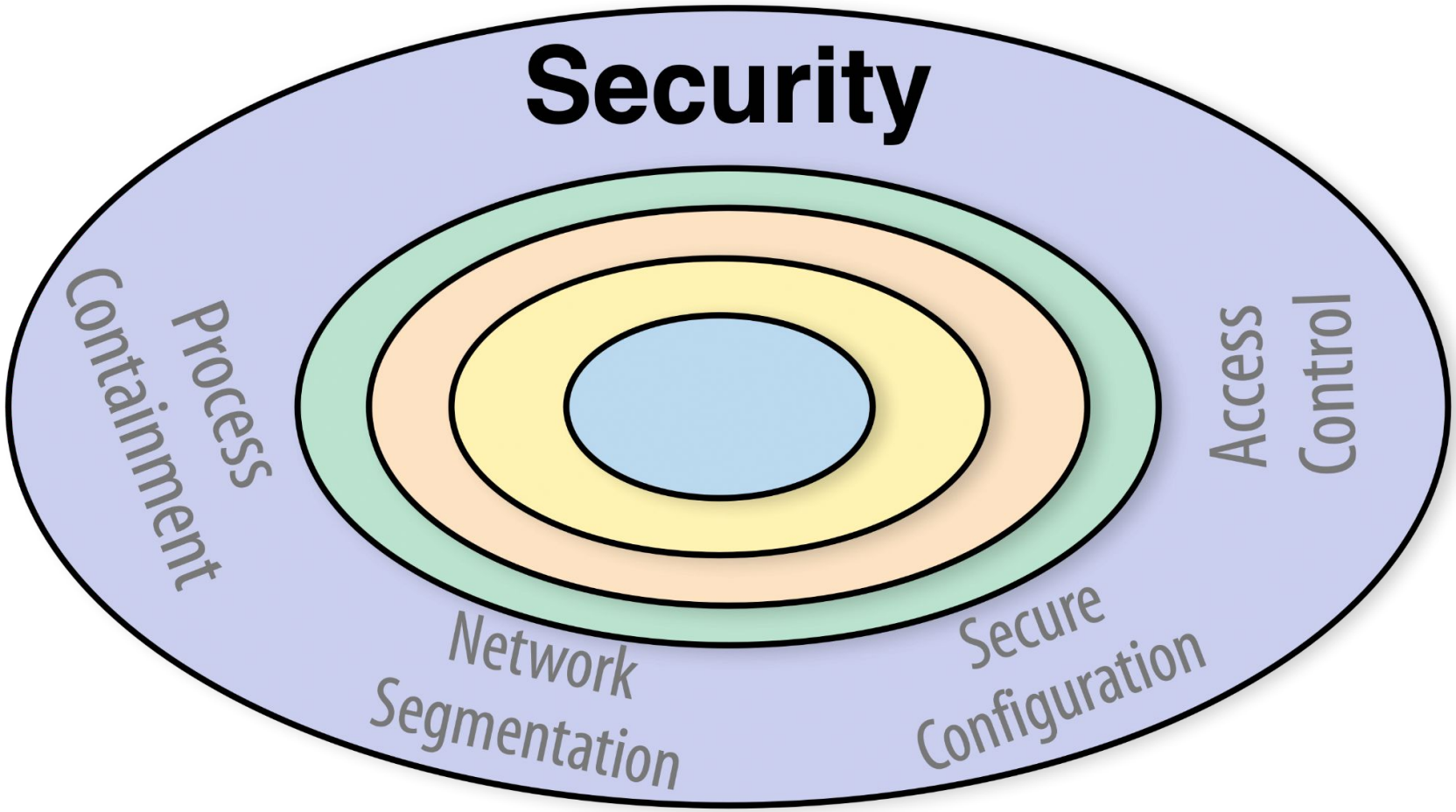
Example


```
initContainers:
- image: k8spatterns/config-dev:1
  name: init
  args:
  - "/config"
  volumeMounts:
  - mountPath: "/config"
    name: config-directory
containers:
- image: k8spatterns/demo:1
  name: demo
  ports:
  - containerPort: 8080
name: http
    protocol: TCP
  volumeMounts:
  - mountPath: "/config"
    name: config-directory
volumes:
  - name: config-directory
    emptyDir: {}
```



```
FROM busybox
ADD dev.properties /config-src/demo.properties
ENTRYPOINT \
   [ "sh", "-c", "cp /config-src/* $1", "--" ]
```

Security Patterns



Init Containers to the rescue

- Kubernetes: **No container image volume mounts**
- Instead: **Copy over configuration** from image to a shared volume in an ***Init Container***
- **Templating** of deployments for adapting to different environments
  - OpenShift Templates, Helm Charts, Operator with CRD …
- Good for **large configuration data**

Common Triggers

- Labels
  - Indexed by backend
  - Suitable for selector-like functionality
  - Limitation on character set for names and values
- Annotations
  - No syntax restrictions
  - Not indexed
- ConfigMaps
  - Good for complex structured state declarations
  - Simple alternative to CustomResourceDefinitions

Operator Hub

- Operator can be implemented in any language
- Frameworks:
  - Operator Framework (Golang, Helm, Ansible, Java)
    https://github.com/operator-framework
  - Kubebuilder (Golang)
    https://github.com/kubernetes-sigs/kubebuilder
  - Metacontroller (Language agnostic)
    https://metacontroller.app/

ConfigMap Watch Controller


```
namespace=${WATCH_NAMESPACE:-default}
base=http://localhost:8001
ns=namespaces/$namespace

curl -N -s $base/api/v1/${ns}/configmaps?watch=true | \
while read -r event
do
  type=$(echo "$event" | jq -r '.type')
  config_map=$(echo "$event" | jq -r '.object.metadata.name')
  annotations=$(echo "$event" | jq -r '.object.metadata.annotations')

  if [ $type = "MODIFIED" ]; then
    # Restart Pods using this ConfigMap
    # ...
  fi
done
```

Notes

- 4 R Intro (Patterns, Kubernetes) - 5 min
- 5 B HealthProbe
- 4 R Sidecar
- 5 B Singleton
- 5 R Immutable Configuration
- 4 B Process Containment
- 7 R Controller / Operator
- 1 B Summary