INNOQ Technology Night Zürich

Application Container Design mit Kubernetes

INNOQ

https://innoq.com/

Berlin · Düsseldorf · Frankfurt · München · Zürich

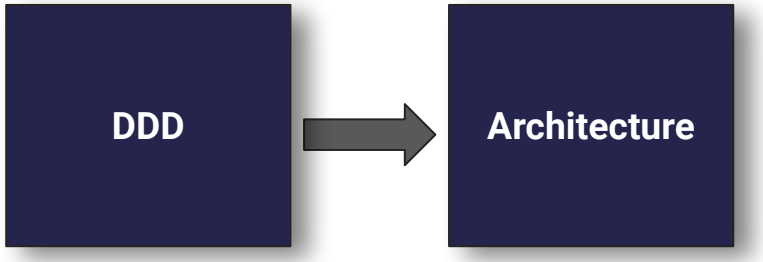
Christopher Schmidt
Senior Consultant
@fakod

- Kubernetes
- Container
- Architecture
- Training
- Consulting

INNOQ

Berlin · Düsseldorf · Frankfurt · München · Zürich

From DDD to Production



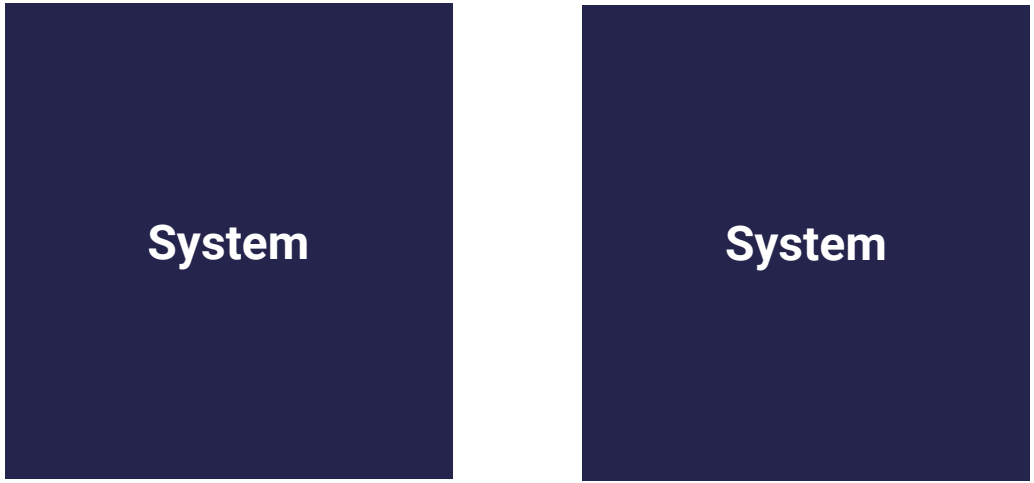
- ❏ Contexts
- ❏ Ctx Mapping

- ❏ SCS
- ❏ µServices

Recap Microservices

- ❏ easy to replace
- ❏ one team, one µService
- ❏ small in size
- ❏ messaging enabled
- ❏ bounded by contexts
- ❏ independently deployable
- ❏ implemented using different technologies

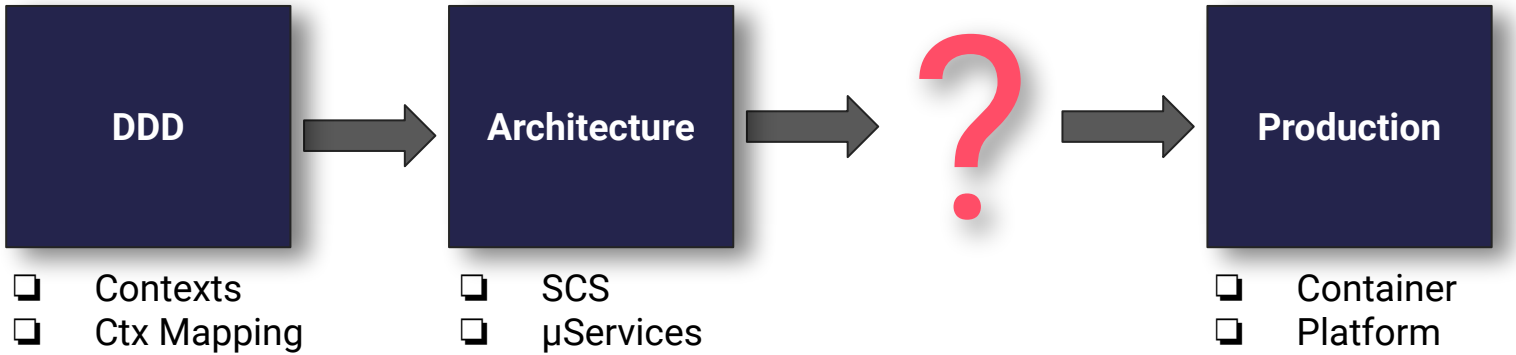
Recap SCS



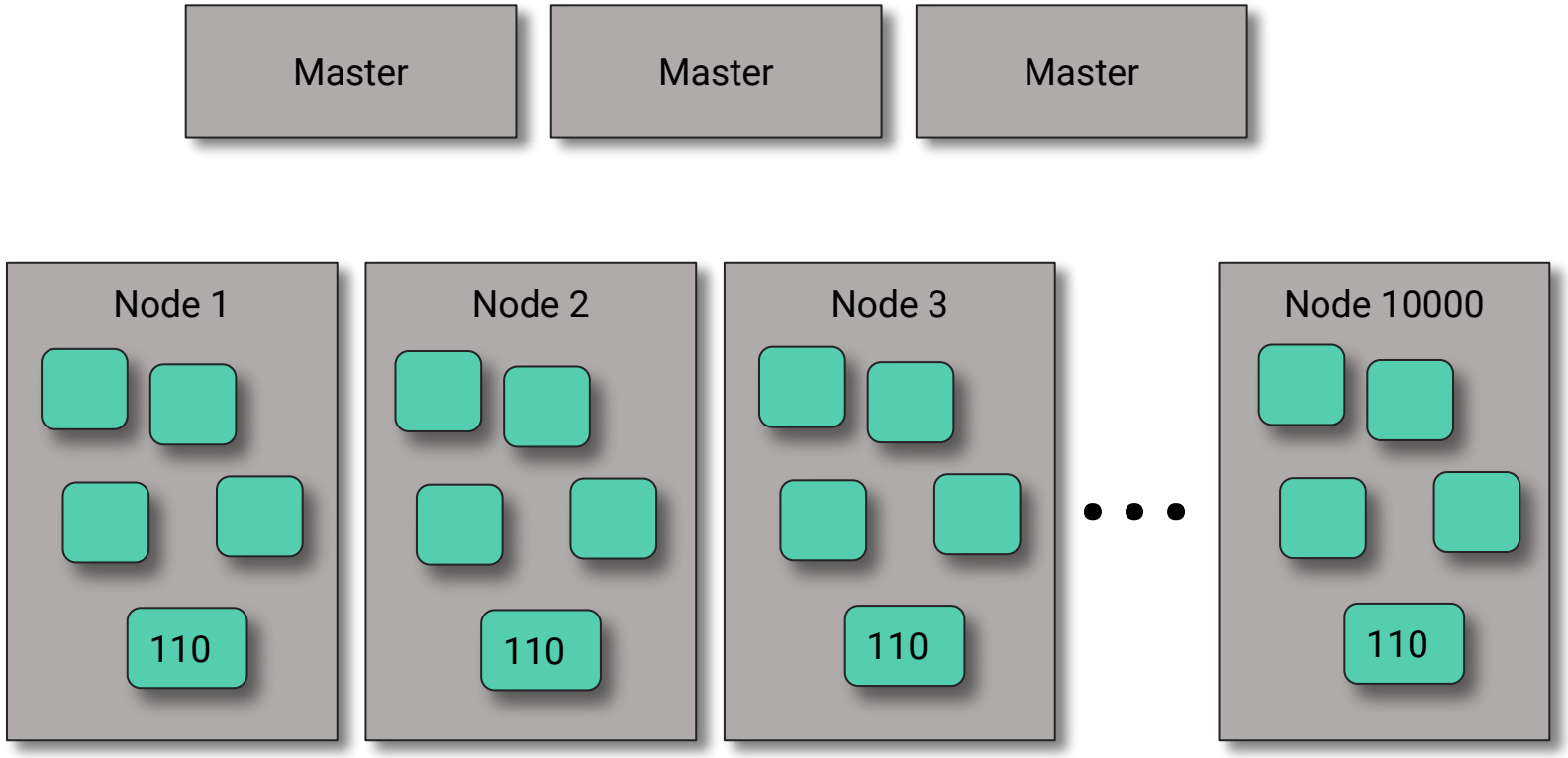
Systems as complete units of DB, Logic, UI

Isolated, independent, autonomous

From DDD to Production

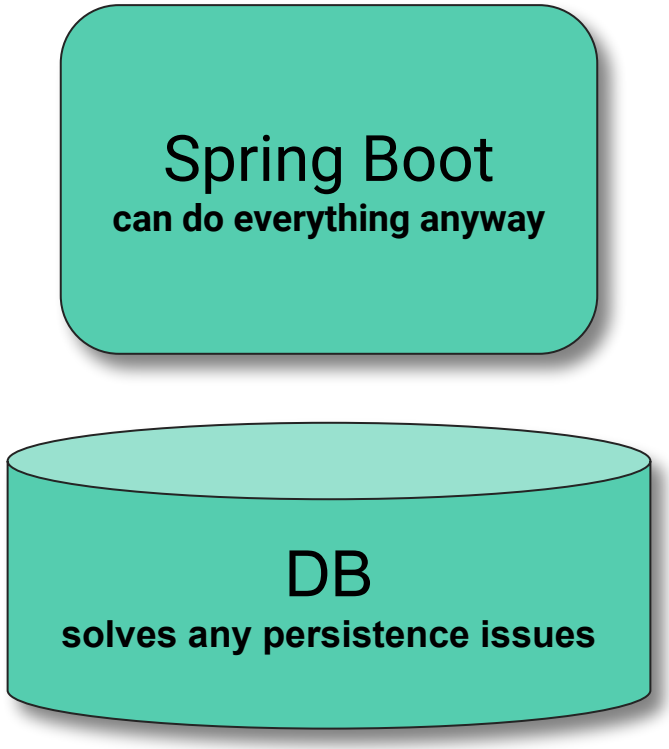


Kubernetes Capabilities

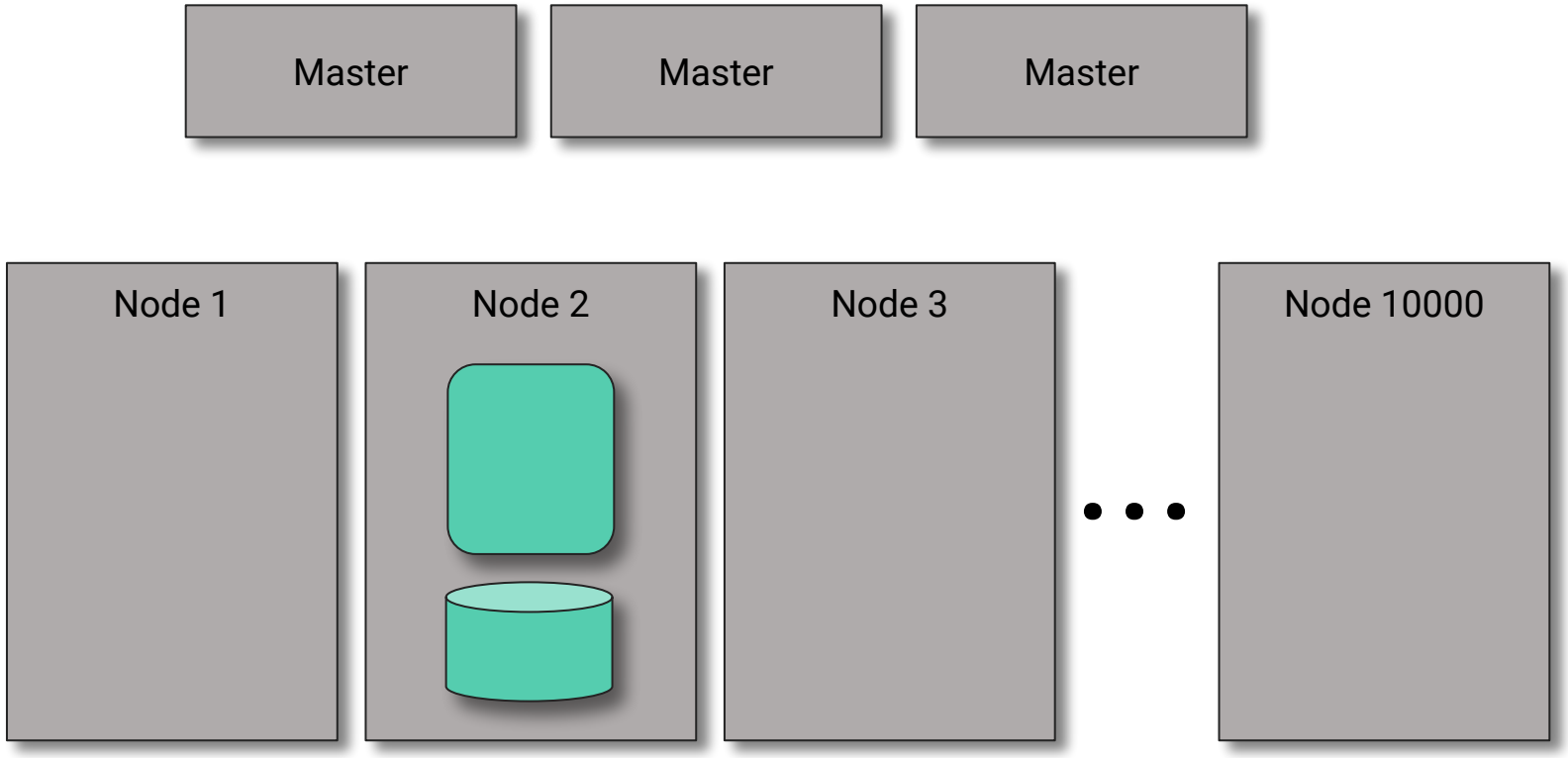


My Application

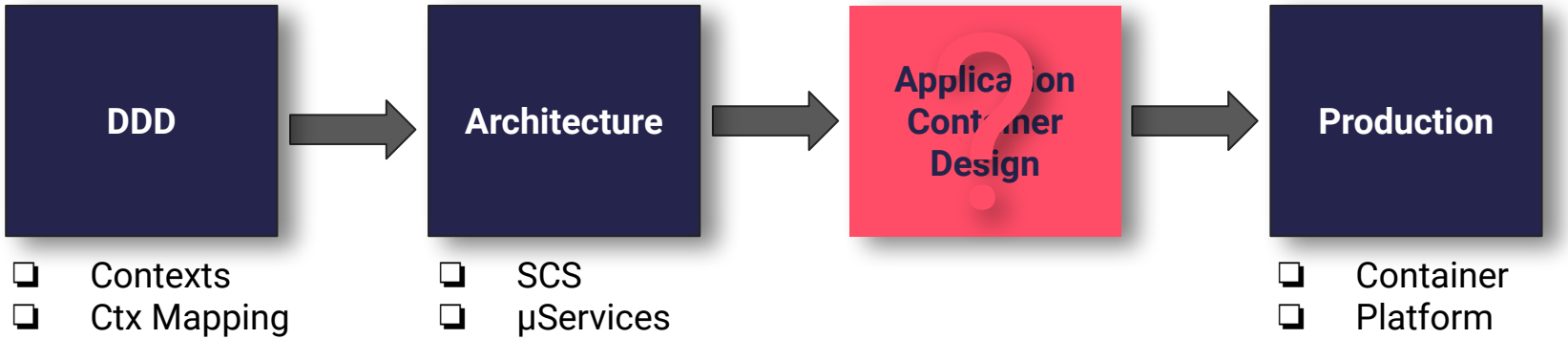
Finally: My Application



Kubernetes and Application

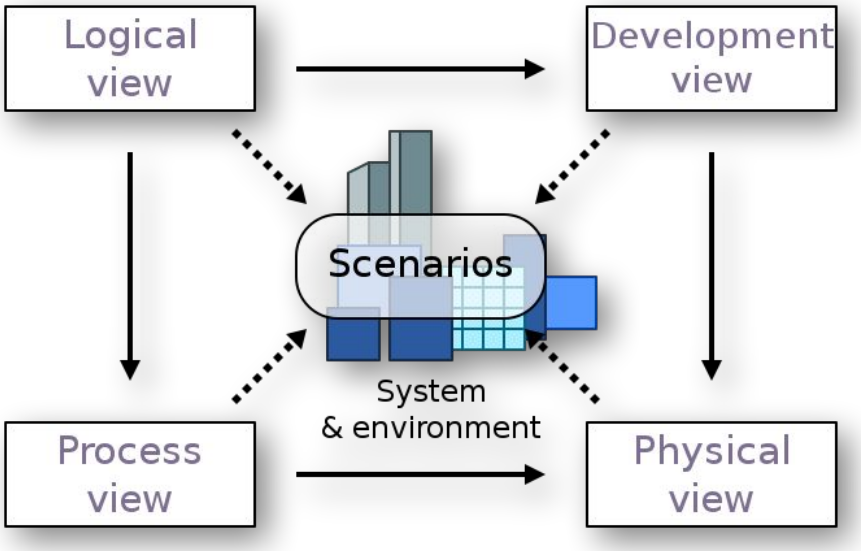


Application Container Design



there is no software architecture without production view

4+1 architectural view model



Logical view:
- functionality provided to end-users
- UML Diagrams

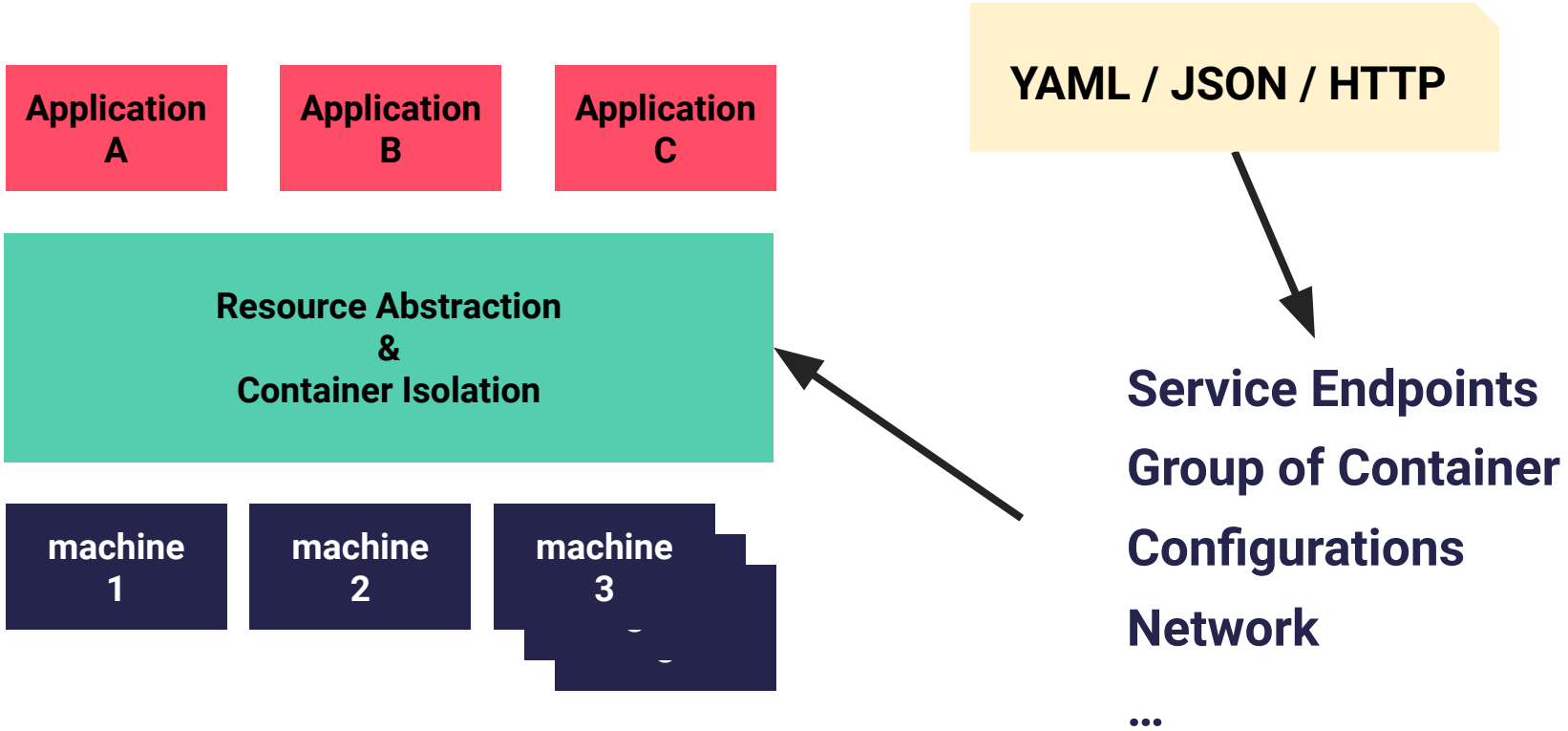
Process view:
- dynamic aspects
- processes and how they communicate

Physical view:
- **topology of software components on the physical layer**
- **network connections**

Development view:
- programmer's perspective
- software management

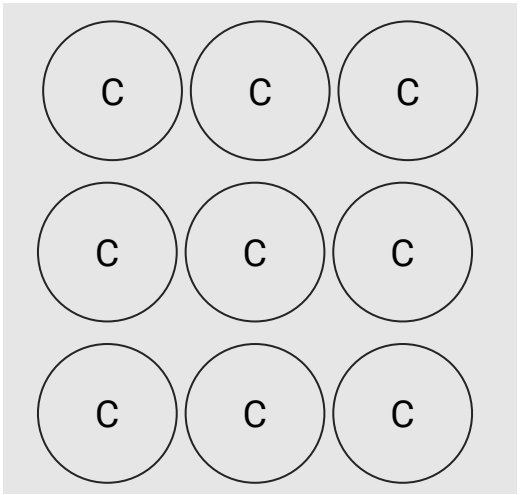
Kubernetes Revolution?

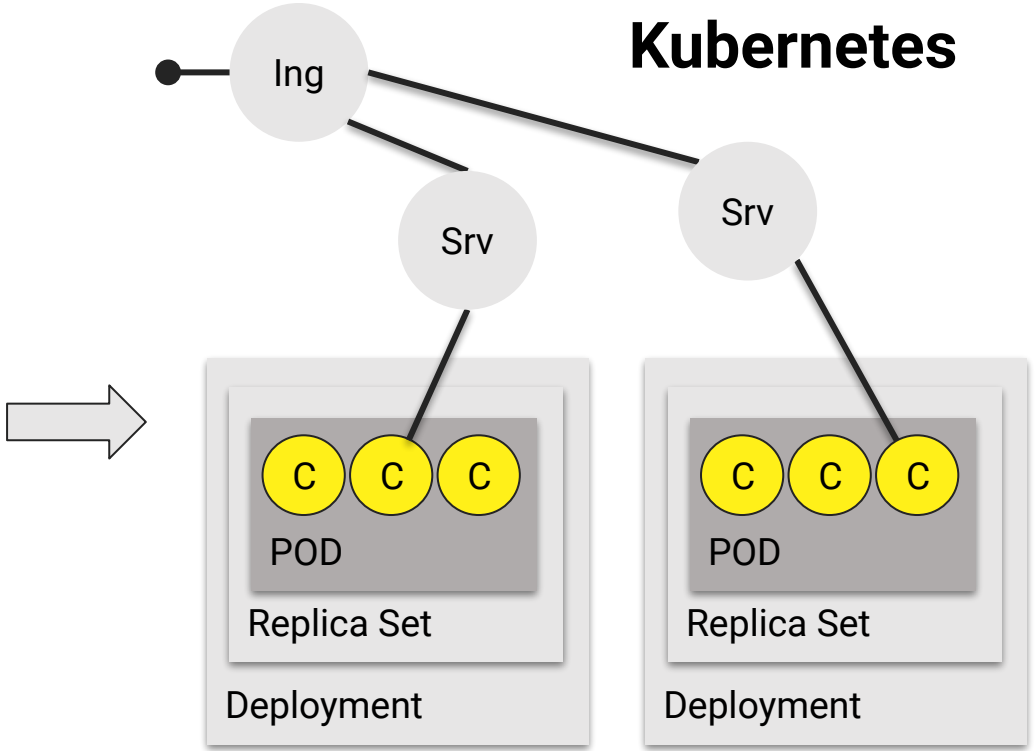
Ressourcen Abstraction



Resource Abstraction

Flat Container level management

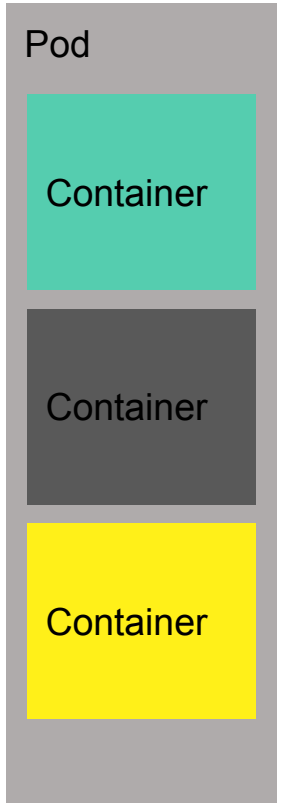




The Pod Ressource?

“Logical Host”

- containers are tightly coupled
- co-located and co-scheduled
- run in a shared context
- shared storage (volumes)
- has its own IP



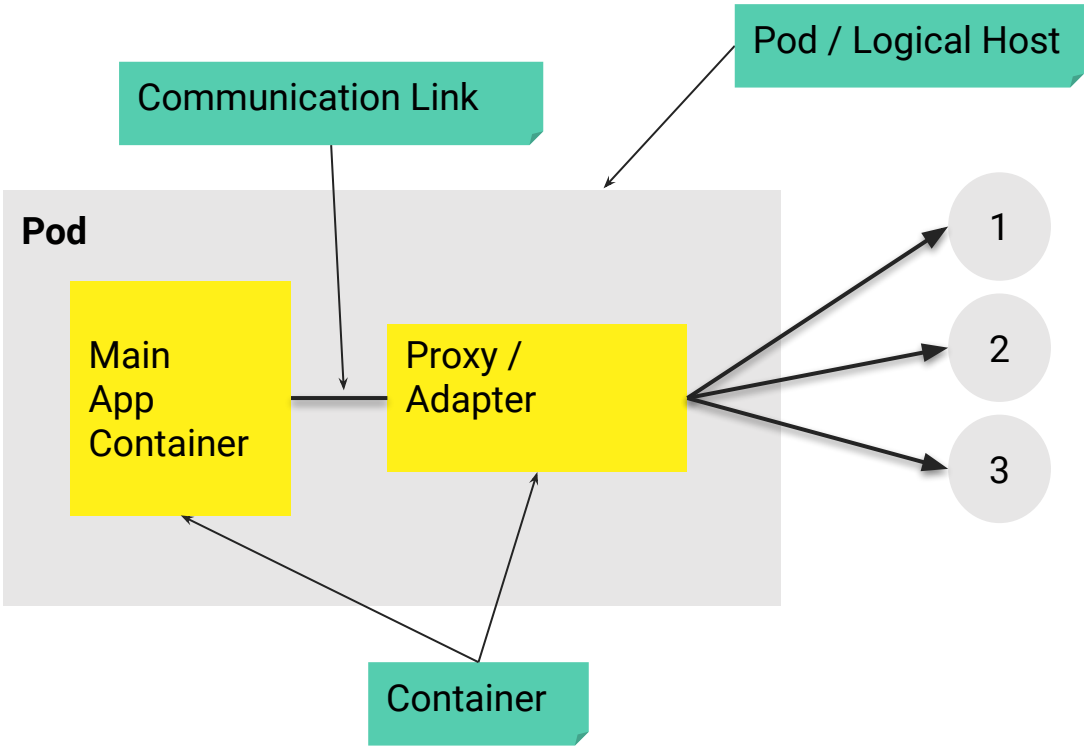
Are there Pattern for Container?

What are Container Pattern?

- Is valid for container as it is for OOP
- Provide general reusable solutions to a common problem
  - Simplifies reuse of images
- Can help to modularize on container level
  - Separation of Concerns
  - Isolation

Single Node, Container Pattern

Sidekick / Sidecar

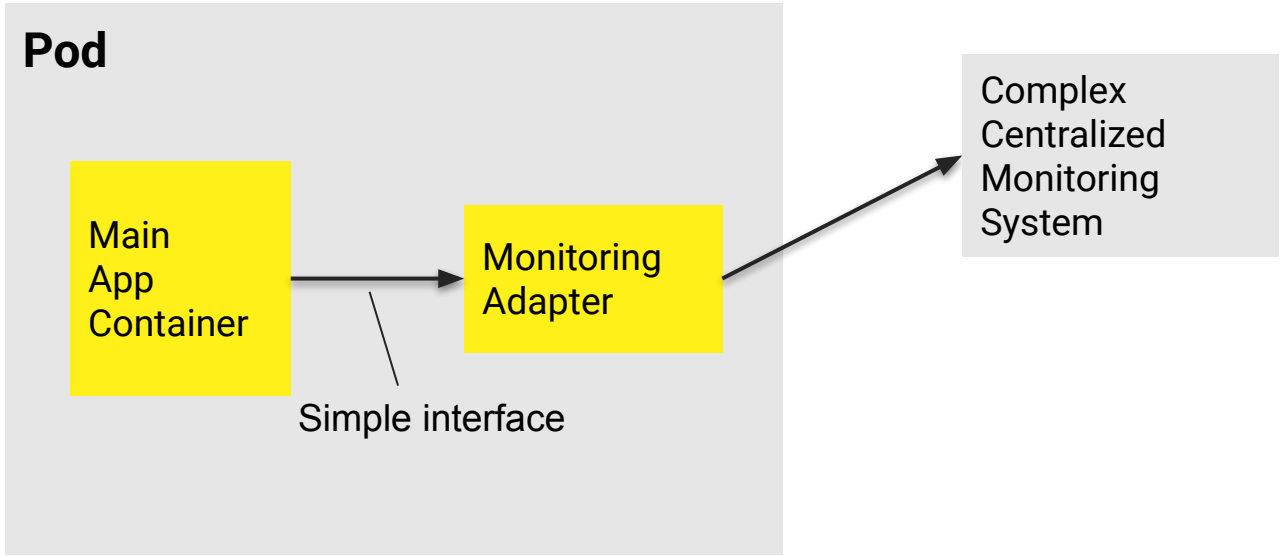


Sidekick / Sidecar

Can be used to abstract communication or infrastructure details

- Protocol switch
- Authorization
- Encryption
- Circuit Breaker Pattern
- Traffic limiting
- . . .

Adapter Pattern



Adapter Pattern

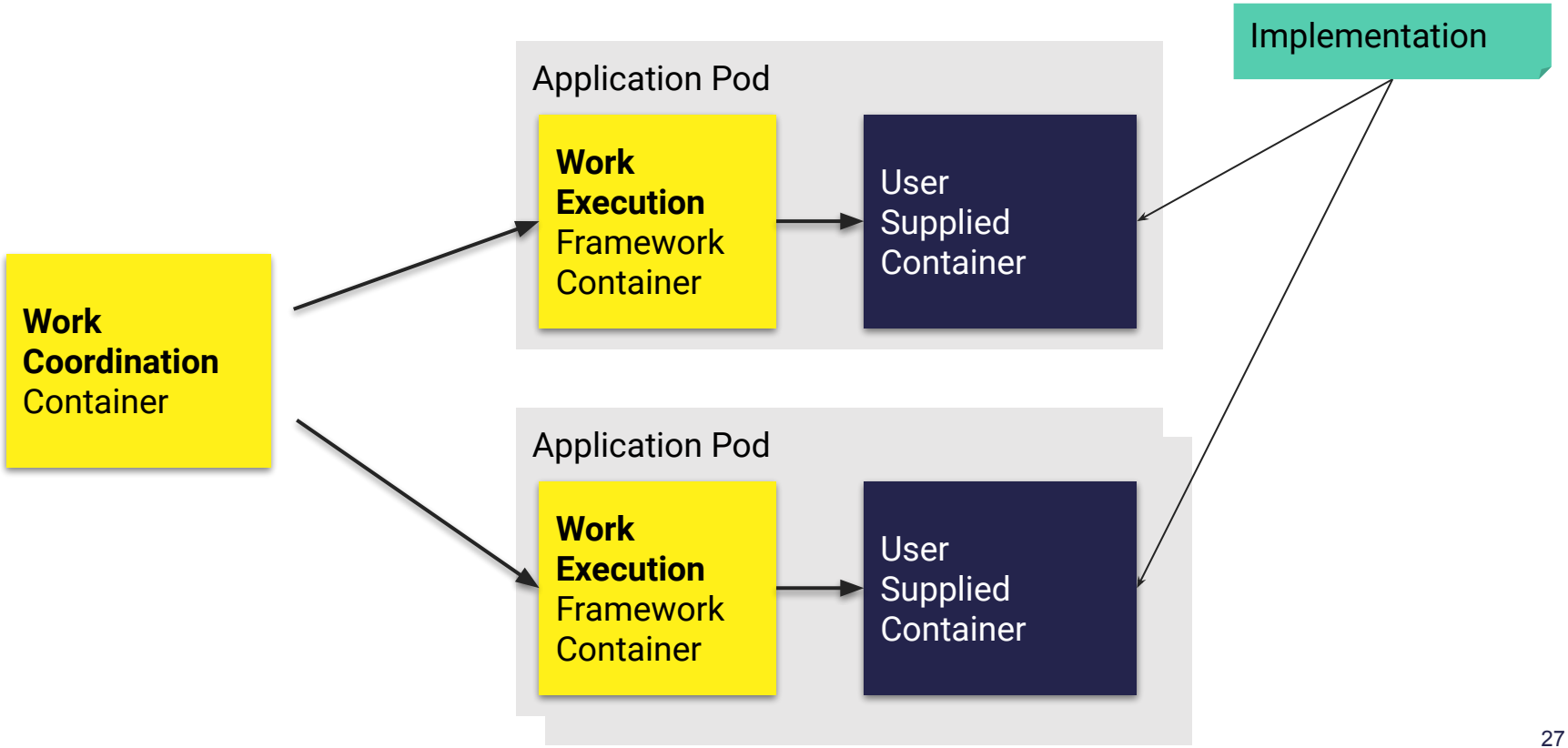
dependency injection

- separation of concerns
- testability
- reuse
- flexibility

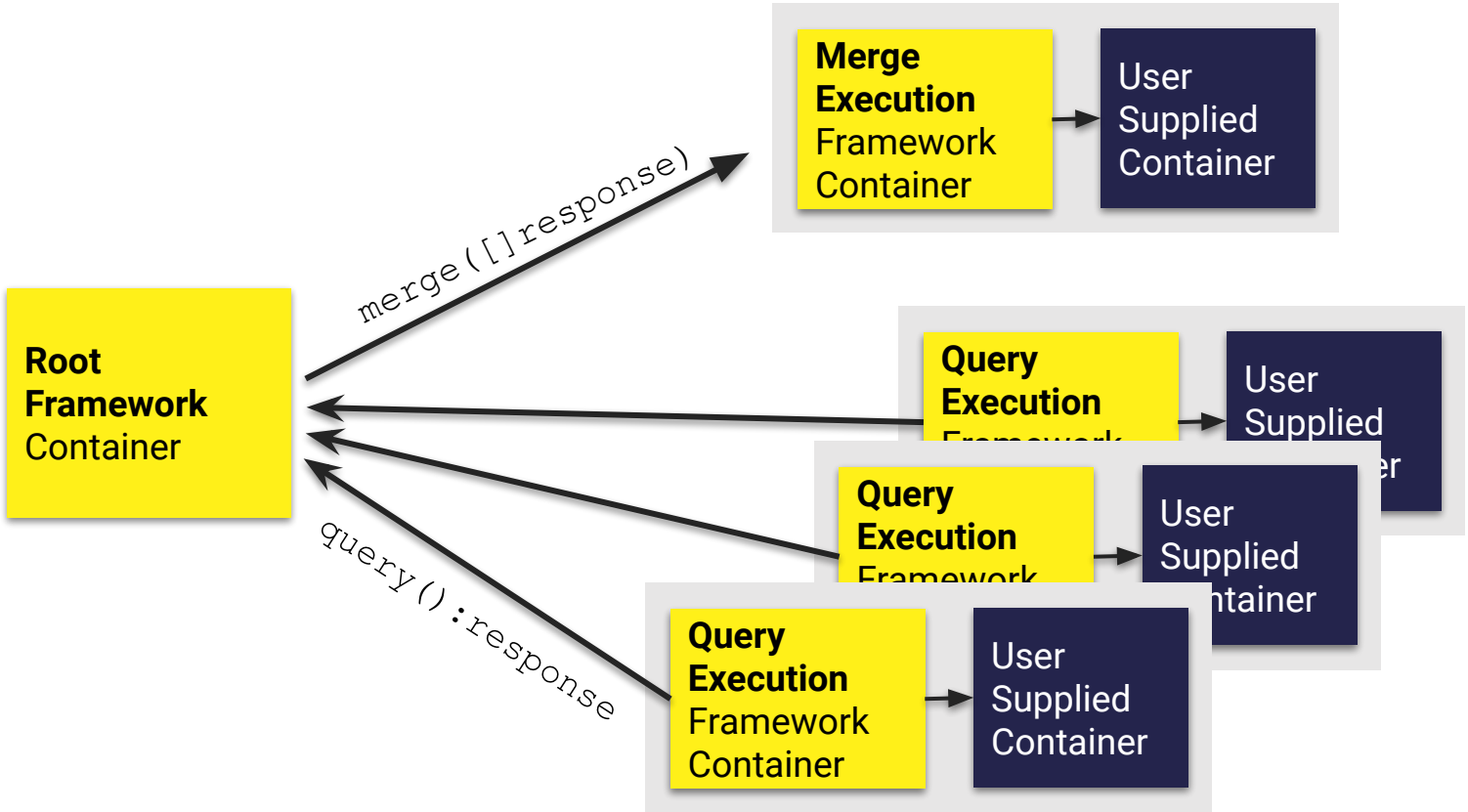
```
apiVersion: v1
kind: Pod
metadata:
 name: the-application
spec:
 containers:
   - name: my-server
     image: innoq/web-server:latest
   - name: mon-adapter
     image: innoq/monitoring-adapter:v1.0.3
```

Multi Node, Container Pattern

Work Queue Pattern



Scatter / Gather (Messaging Pattern)



All this are Pattern, that run distributed on a cluster of nearly infinite machines

Modularize in Container by...

- ➔ Architecture / Business Logic
- ➔ Implements NFRs at runtime
- ➔ Has a Framework characteristics

Further Modularization?

FaaS

Use Functions...

- ➔ Event trigger
- ➔ Business logic that can be decomposed into simple functions
- ➔ Simple tasks
- ➔ Workflow of tasks

Use Functions… advantages:

- ➔ No Build System
- ➔ No Dockerfiles
- ➔ No Deployment Descriptors
- ➔ No Server
- ➔ No Cluster Sizing
- ➔ No Workload Scaling

Kubernetes FaaS Frameworks

- OpenFaaS
- Kubeless
- Fission
- Knative
- . . .

Operator Pattern

Why is is difficult to Operate things?

Day 1 is quite simple (could be done by e.g. HELM)

Day 2 operations:

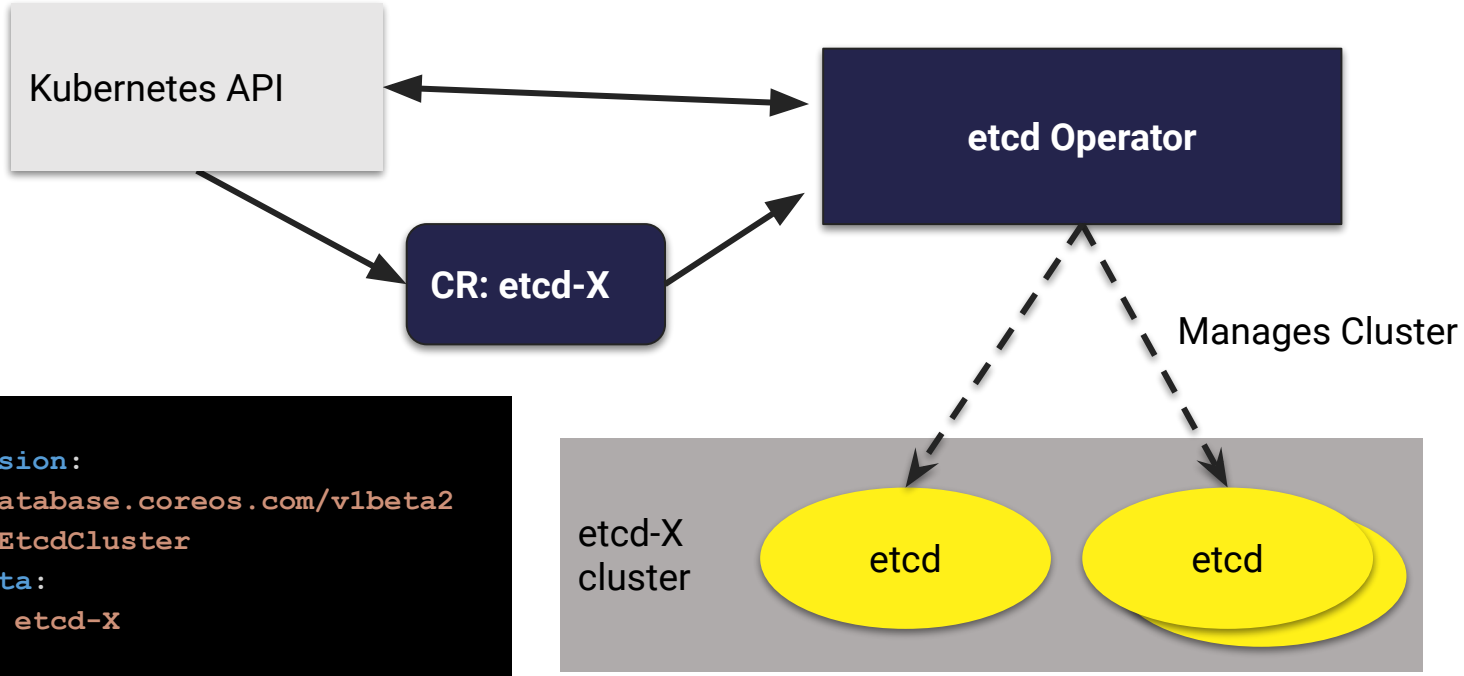
- Up- and downscaling (complex)
- Backup and restore
- Version bump
- Resilience, Encryption, Identities etc.

Operators

Operator

- Operating-a-tool knowledge baked into code
- Is a running Container in K8s (POD)
- Talks to the K8s API as you do it with `kubectl`
- Uses the API extensibility of K8s

Operator example ETCD




```
apiVersion:
etcd.database.coreos.com/v1beta2
kind: EtcdCluster
metadata:
 name: etcd-X
spec:
 size: 3
```

Operator example Kafka (Strimzi)

Creates / deploys

- **Kafka** cluster of broker nodes
- **ZooKeeper** cluster of replicated ZooKeeper instances
- **Kafka Connect** cluster for external data connections
- **Kafka MirrorMaker** cluster to mirror the Kafka cluster in a secondary cluster
- **Kafka Exporter** to extract Kafka metrics data for monitoring
- **Kafka Bridge** to make HTTP-based requests to the Kafka cluster

Operator example Kafka (Strimzi)

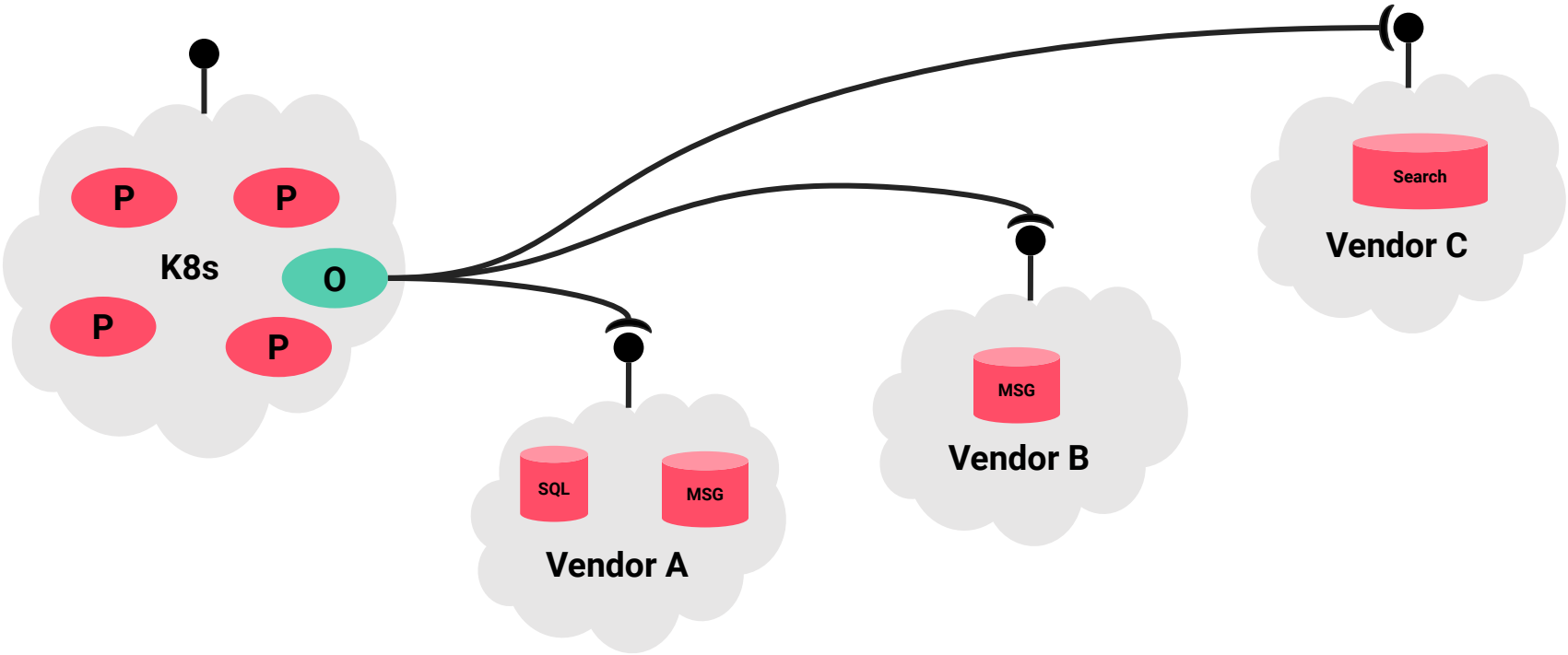
Manages...

- Topics
- Users
- Connections
- Security (Encryption, Authentication etc.)
- . . .

Example Kafka Cluster

```
apiVersion: kafka.strimzi.io/v1beta1
kind: Kafka
metadata:
 name: my-cluster
spec:
 kafka:
   replicas: 3
   version: 0.16.1
   jvmOptions:
     -Xmx: 8192m
   listeners:
     tls:
       authentication:
         type: tls
     external:
       type: route
       authentication:
         type: tls
   authorization:
     type: simple
   config:
    auto.create.topics.enable : "false"
    offsets.topic.replication.factor : 3
    transaction.state.log.replication.factor : 3
    transaction.state.log.min.isr : 2
   storage:
     type: persistent-claim
     size: 10000Gi
```

Operator example vendor API complexity



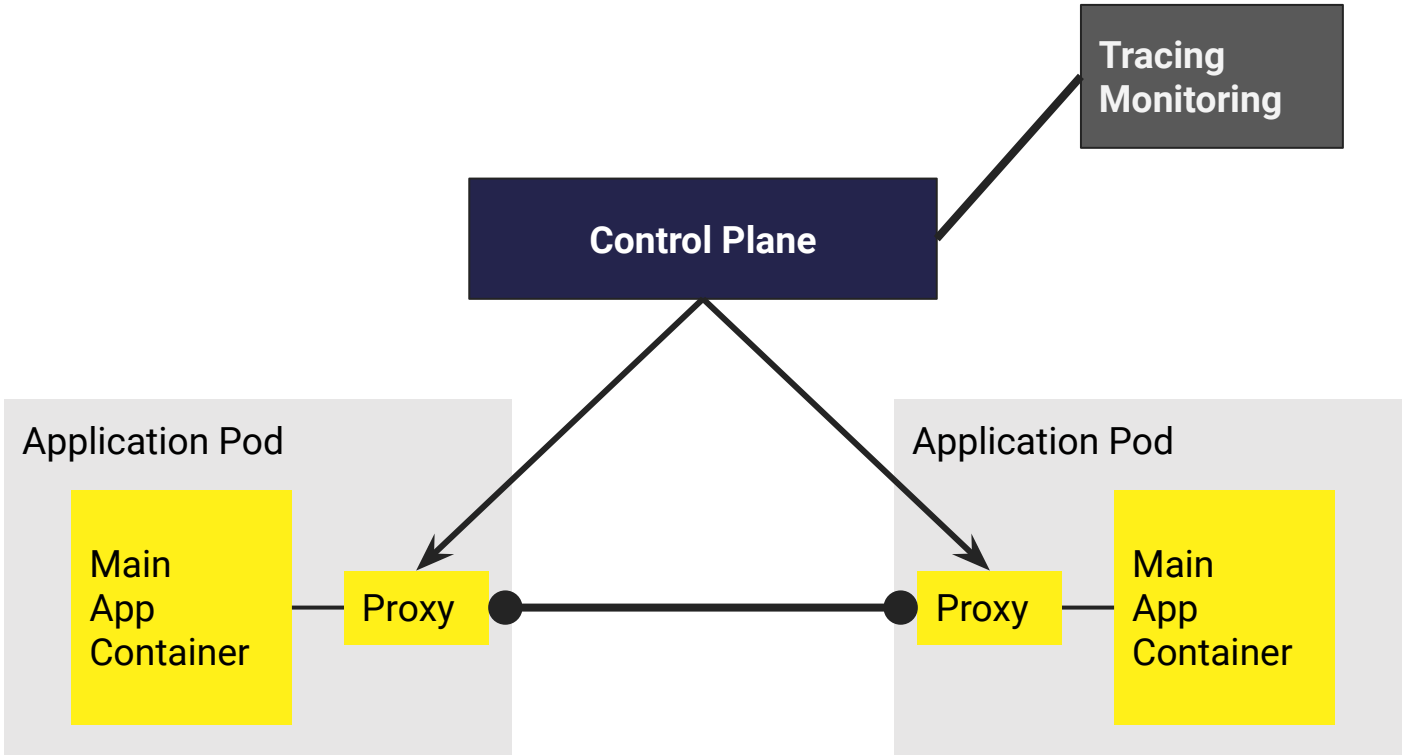
Usual App characteristic and Day 2 issues?

- Distributed Computing
- Shared Nothing Architecture
- Network between Services
- Encryption
- Identity
- Resiliency
- Observability
- Releasing Things

Service Mesh

A Service Mesh is a decentralized infrastructure level in which functions for observability, routing, resilience and secure communication between microservices are implemented.

Basic Pattern



Service Mesh Features

traffic management and shifting

- ❏ separates traffic from replicas
- ❏ fault injection
- ❏ rate limiting

security

- ❏ mTLS between services
- ❏ service identity

policies

visualization

Options for Kubernetes

Istio

- ❏ most buzz around
- ❏ huge

Linkerd 2

- ❏ smarter (not so feature rich as istio)

stock solution with

- ❏ Cilium
- ❏ nginx ingress

Recap

Recap

Container Pattern

- Increases reuse, flexibility and testability
- Better resource usage
- YAMLs can wire functionality together

Operators

- Complex operation made simpler and transferable
- Tool vendors regain control
- Think of K8s as cloud operating system

Recap

Service Mesh

- Use of dynamically configurable proxies with all its associated possibilities
- Security and missing-feature evaluation will eventually lead to a service mesh
- Is a monster

thx