The Operator Pattern

Extending Kubernetes with Java

Gintaras — Platform Engineer

```
kubectl apply -f talk.yaml
```

GM

ABOUT THE SPEAKER

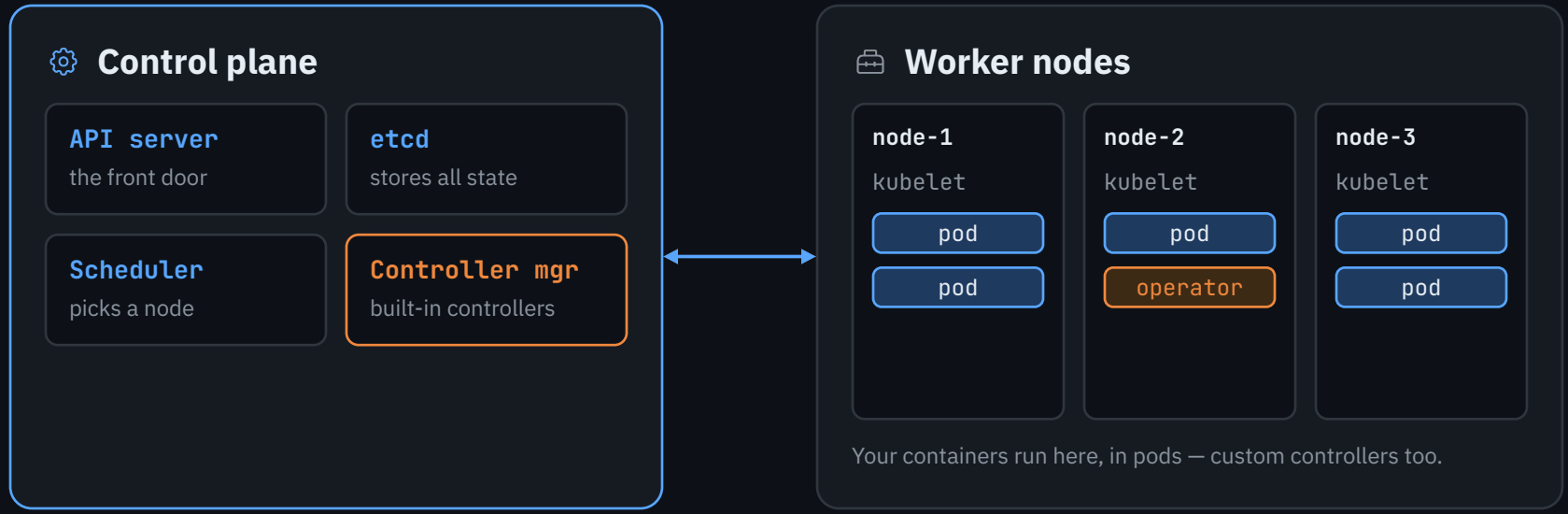
Gintaras Matulas

Member of Technical Staff @ Atoms — Platform

Work in a compute team providing a Kubernetes-based platform for other developers to run their workloads on.

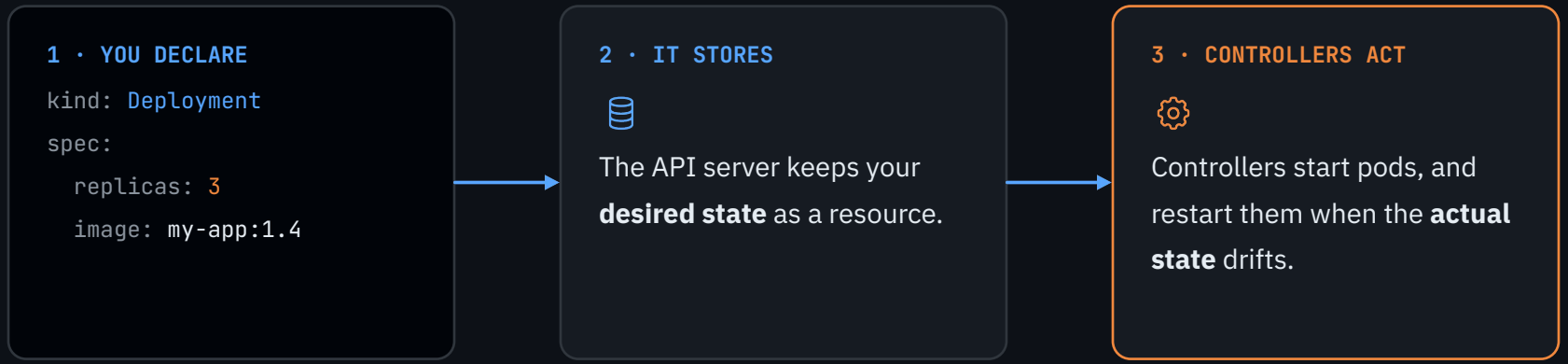
linkedin.com/in/gintaras-matulas-5090b8100

K8s Cluster


Control plane
API server
the front door
etcd
stores all state
Scheduler
picks a node
Controller mgr
built-in controllers
Worker nodes
node-1
kubelet
pod
pod
node-2
kubelet
pod
operator
node-3
kubelet
pod
pod
Your containers run here, in pods — custom controllers too.

KUBERNETES BASICS

Declarative: you describe what you want. Kubernetes makes it so — and keeps it so.



Built-in nouns: Pod · Deployment · Service · ConfigMap · Namespace

THE PROBLEM

Every service has a human who knows exactly how it should run — how to deploy it, and what to do when it breaks.

For a platform team, that knowledge usually lives in a support request, an incident mitigation, or a runbook nobody's kept up to date.

```
Today: a glimpse of how to turn that knowledge into code — in Java.
```

What is an Operator?

Kubernetes itself is a pile of controllers reconciling state — the Deployment controller, the ReplicaSet controller. An Operator is one you write, for your own domain.



Controller vs. Operator

Controller

The generic term: any watch → compare → act loop.

Kubernetes ships full of them — Deployment, ReplicaSet, Node — watching **built-in** objects.

Operator

The same loop, watching a **CRD you defined** — a custom resource, not a built-in one.
It encodes operational knowledge a human would otherwise apply by hand.

```
Every Operator is a controller. Not every controller is an Operator.
```

Operators you've probably already used

JAVA · JOSDK

Flink Kubernetes Operator

`kind: FlinkDeployment`

Deploys Flink jobs, handles upgrades with savepoints, suspends and autoscales them.

JAVA

Strimzi

`kind: Kafka, KafkaTopic`

Runs Kafka clusters, topics and users as Kubernetes resources.

GO

cert-manager

`kind: Certificate`

Issues TLS certificates and renews them before they expire.

GO

Prometheus Operator

`kind: ServiceMonitor`

Turns “scrape this service” into live Prometheus config.

The Flink operator is built on the same Java Operator SDK we'll use today.

CRDs: teaching Kubernetes a new noun

A RESOURCE OF THE NEW KIND

```yaml
apiVersion: org.operator/v1
kind: EphemeralNamespace
metadata:
  name: demo-team
spec:
  teamName: platform
  ttl: 15m
```

- ✓ kubectl treats it like any built-in resource
- ✓ kubectl get ephemeralnamespaces just works
- ✓ your operator is the only thing that gives it meaning

Why Java?

Most operator tooling — kubebuilder, operator-sdk — is **Go-native**, because Kubernetes itself is Go.

The **Java Operator SDK (JOSDK)** sits on the Fabric8 Kubernetes client.

It handles the watch / informer / event-queue plumbing so you mostly implement one method.

```
class EphemeralNamespaceReconciler
    implements Reconciler<EphemeralNamespace> {

  UpdateControl<EphemeralNamespace> reconcile(
    EphemeralNamespace resource,
    Context<EphemeralNamespace> context) {
    // watch → compare → act
  }
}That's really it.
```

Operator vs. Helm

Helm

Renders and applies **once**. Then it's gone.

No runtime component — nothing is watching afterward.

Simpler, smaller blast radius. The right default for “deploy this app.”

Operator

A **continuously running** control loop.

Reacts to drift, external change, or the passage of time — no human re-running anything.

More power. Also a long-running privileged workload you now own.

Reach for Helm first. Reach for an Operator when Helm's answer to “and then what happens if things change?” is “nothing, until someone runs helm upgrade again.”

Operator maturity: the capability model

LEVEL 5	**Auto Pilot**	Automatic scaling, tuning and self-healing
LEVEL 4	**Deep Insights**	Metrics, alerts and log processing
LEVEL 3	**Full Lifecycle**	Backup, restore and failure recovery
LEVEL 2	**Seamless Upgrades**	Version upgrades, including migrations
LEVEL 1	**Basic Install**	Automated deployment and configuration

LIVE DEMO

EphemeralNamespace

Need somewhere to run a demo app? Get a namespace — and have it cleaned up when you're done.

```yaml
apiVersion: org.operator/v1
kind: EphemeralNamespace
metadata:
  name: demo-team
spec:
  teamName: platform
  resourceQuotaCpu: 200m
  ttl: 15m
```

AUTOMATED FOR YOU

01 **Creates the namespace**
with a CPU quota from the spec

02 **Adds RBAC**
the team gets access to its namespace

03 **Cleans up after the TTL**
namespace and everything in it, deleted

What you'll see

01 Apply an EphemeralNamespace → Namespace, quota and RBAC appear

02 Edit the team name, reapply → the RoleBinding updates to the new team

03 TTL keeps ticking in the background — no one touches it

04 TTL expires → the operator deletes the Namespace and everything in it

Best practices

Make reconcile idempotent

Running it twice must be as safe as running it once.

Read state, don’t trust events

Look at what exists now — events can be missed or repeated.

Set owner references

Let Kubernetes garbage-collect the children for you.

Report status on the CR

Conditions tell users what’s happening, right in kubectl.

Reschedule, don’t block

Ask to be called again later instead of sleeping in reconcile.

Least-privilege RBAC

Your operator runs privileged — grant only what it needs.

When to use an Operator (and when not to)

Use one when

- ✓ Stateful apps with domain-specific operations
- ✓ Multi-step procedures where order matters
- ✓ Day-2 work: backups, cert rotation, upgrades
- ✓ A self-service API for your platform

Skip it when

- × A Deployment or StatefulSet is enough
- × A Helm chart already solves it
- × The logic is trivial
- × Nobody has time to maintain it

Rule of thumb: a page-long runbook full of “if X, then do Y” → operator. A runbook that says “run helm upgrade” → not.

Go deeper

- Java Operator SDK — javaoperatorsdk.io
- Fabric8 Kubernetes Client — github.com/fabric8io/kubernetes-client
- operatorhub.io — browse real-world Operators
- CNCF Operator White Paper — tag-app-delivery.cncf.io/whitepapers/operator

Thank you! Questions?

linkedin.com/in/gintaras-matulas-5090b8100