Operate Kubernetes workloads

extend the platform with the operator pattern!

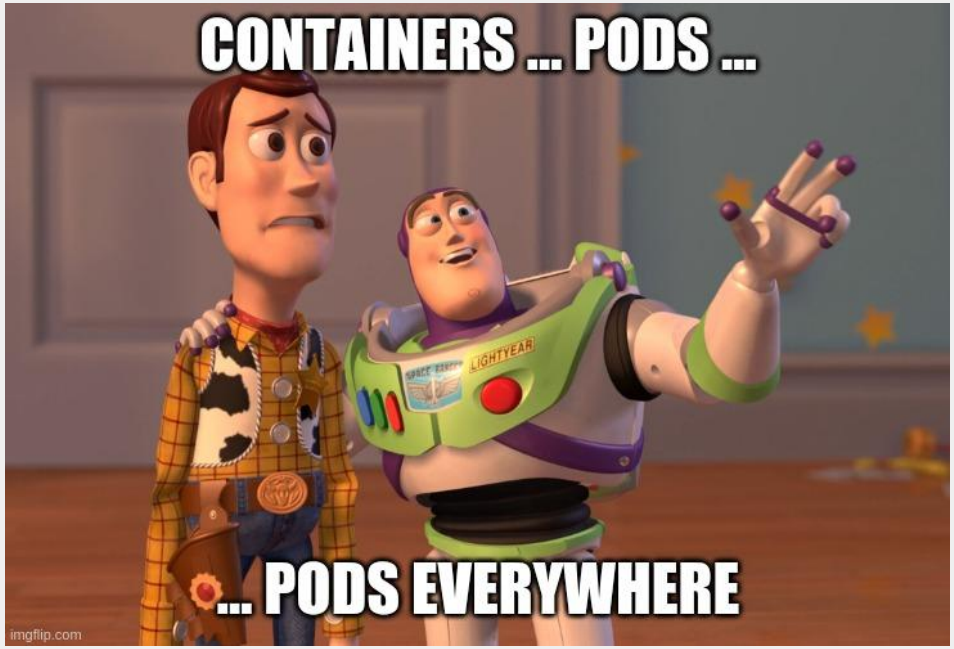
Paolo Patierno
Senior Principal Software Engineer @Red Hat

About me

```
apiVersion: v1
kind: SeniorPrincipalSoftwareEngineer
metadata:
  name: Paolo Patierno
  namespace: Red Hat, Messaging & Data Streaming
  labels:
    cncf/maintainer: Strimzi
    cncf/maintainer: true
    eclipse/committer: Vert.x, Hono & Paho
    microsoft/mvp: Azure
  annotations:
    family: dad of two, husband of one
    sports: running, swimming, motogp, vr46, formula1, ferrari, ssc napoli
    community: cncf napoli, devday
spec:
  replicas: 1
  containers:
  - image: patiernohub.io/paolo:latest
```

@ppatierno

Kubernetes workloads ...



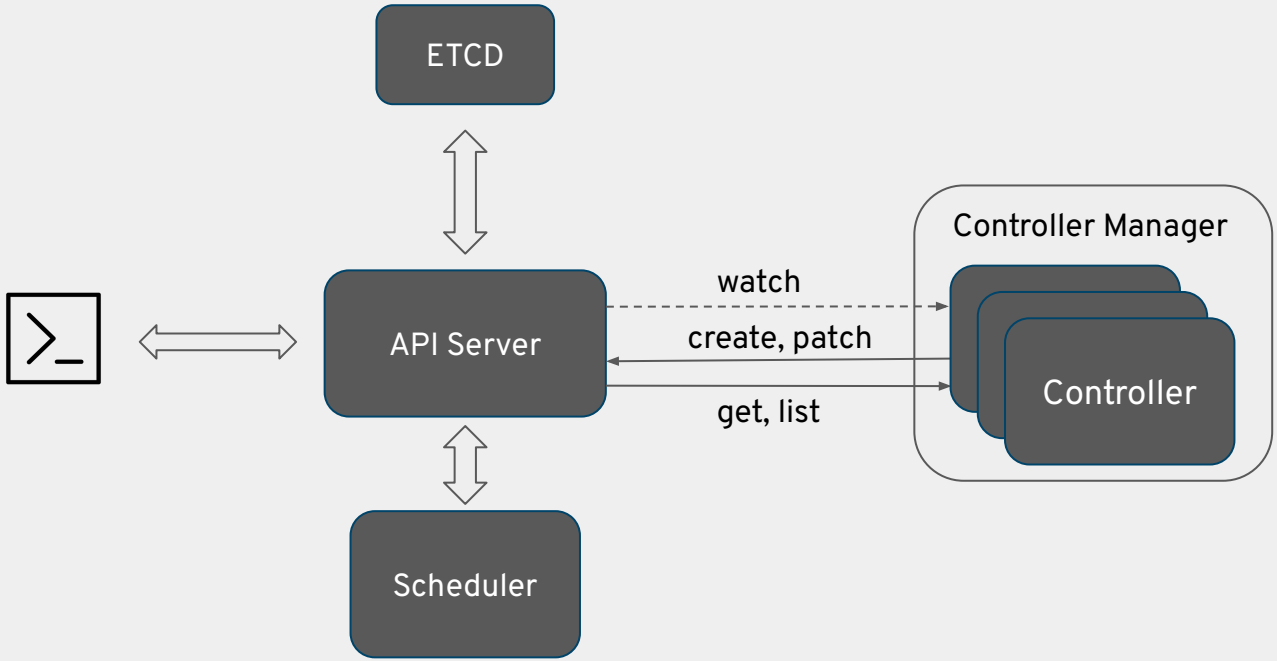
How does Kubernetes handle scaling, rollout, batch execution and so on?

Not just pods …

- Don’t use Pod(s) … let’s use something more sophisticated!
- ReplicaSet
  - Guarantees a specific number of running replicas
  - Spins pods, based on a template, if there are not enough …
  - … or deletes pods if too many
- Deployment
  - It’s based on ReplicaSet (used to run replicas)
  - Adds extra layer for rollout and rollback
- … and more with StatefulSet, Job, DaemonSet …

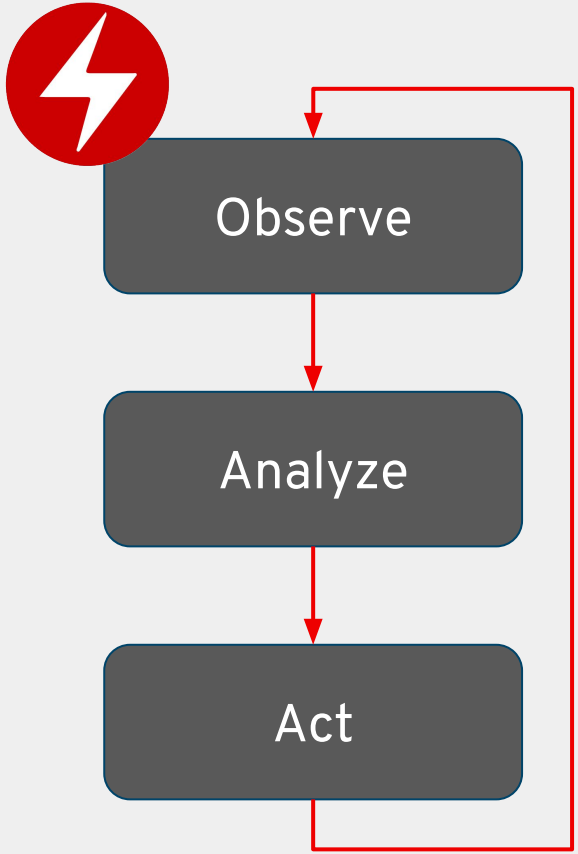
How does it work?
Let’s use a controller!!!
…. But not this one ;-)

Kubernetes Control Plane



Reconcile Loop

- Observe
  - Watch for resource/object creation or changes
- Analyze
  - Check that the resource/object desired state (“spec”) reflects the current state on the cluster
- Act
  - Makes the needed changes



ReplicaSet Controller

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: my-replicaset
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
      - name: my-application
        image: quay.io/ppatierno/my-application:latest
```

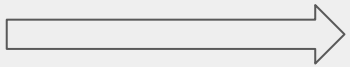
my-replicaset-bf5zv

my-replicaset-1tf5a

my-replicaset-gb65f

ReplicaSet Controller

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: my-replicaset
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
      - name: my-application
        image: quay.io/ppatierno/my-application:latest
```

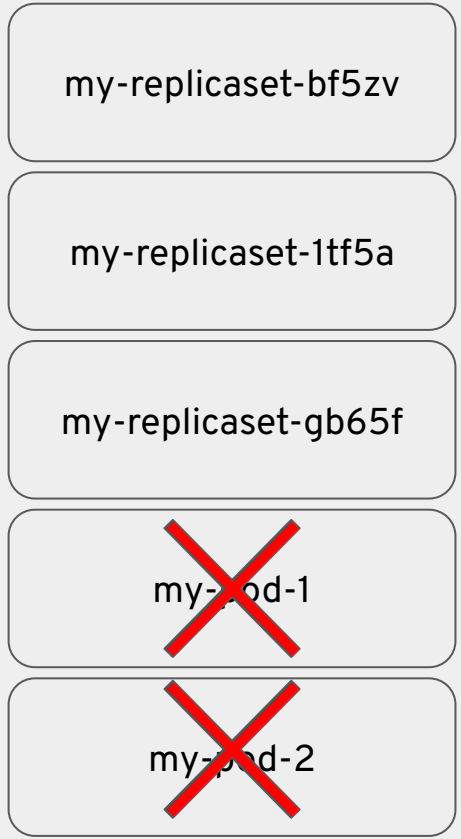


- my-replicaset-bf5zv
- my-replicaset-1tf5a
- my-replicaset-gb65f
- my-replicaset-5tfgs
- my-replicaset-rf43g

ReplicaSet Controller

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-pod-1
  labels:
    app: my-app
spec:
  containers:
  - name: my-app
    image: quay.
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-pod-2
  labels:
    app: my-app
spec:
  containers:
  - name: my-application
    image: quay.io/ppatierno/my-application:latest
```



ReplicaSet Controller

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-pod-1
  labels:
    app: my-app
spec:
  containers:
  - name: my-app
    image: quay.
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-pod-2
  labels:
    app: my-app
spec:
  containers:
  - name: my-application
    image: quay.io/ppatierno/my-application:latest
```

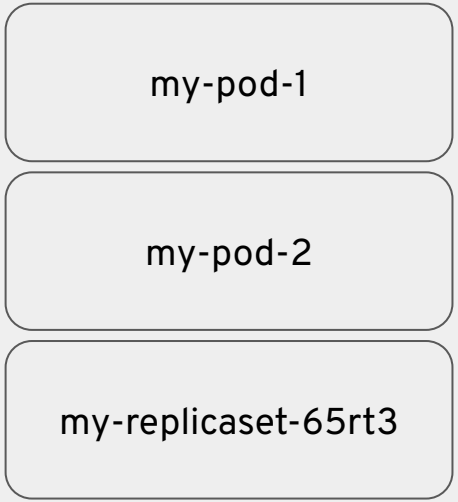
my-pod-1

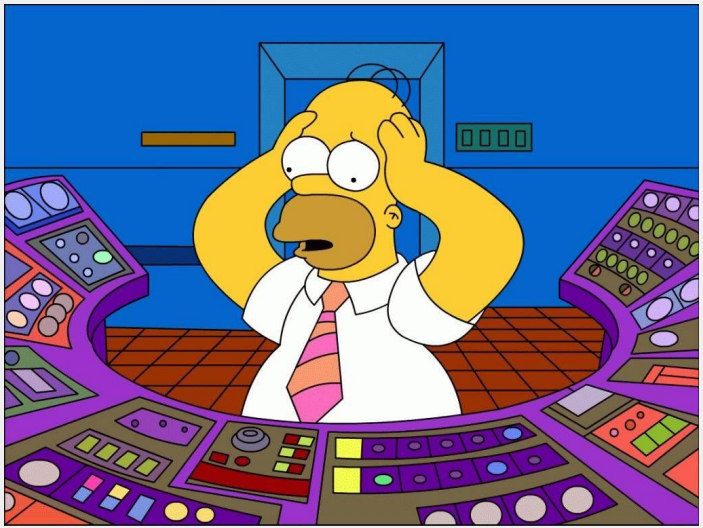
my-pod-2

CLOUD NATIVE
COMPUTING NAPOLI

ReplicaSet Controller

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: my-replicaset
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
      - name: my-application
        image: quay.io/ppatierno/my-application:latest
```

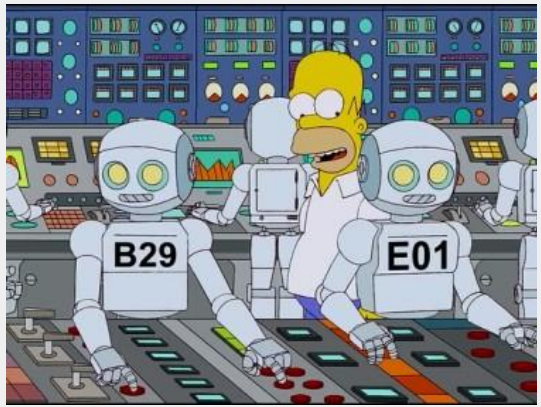




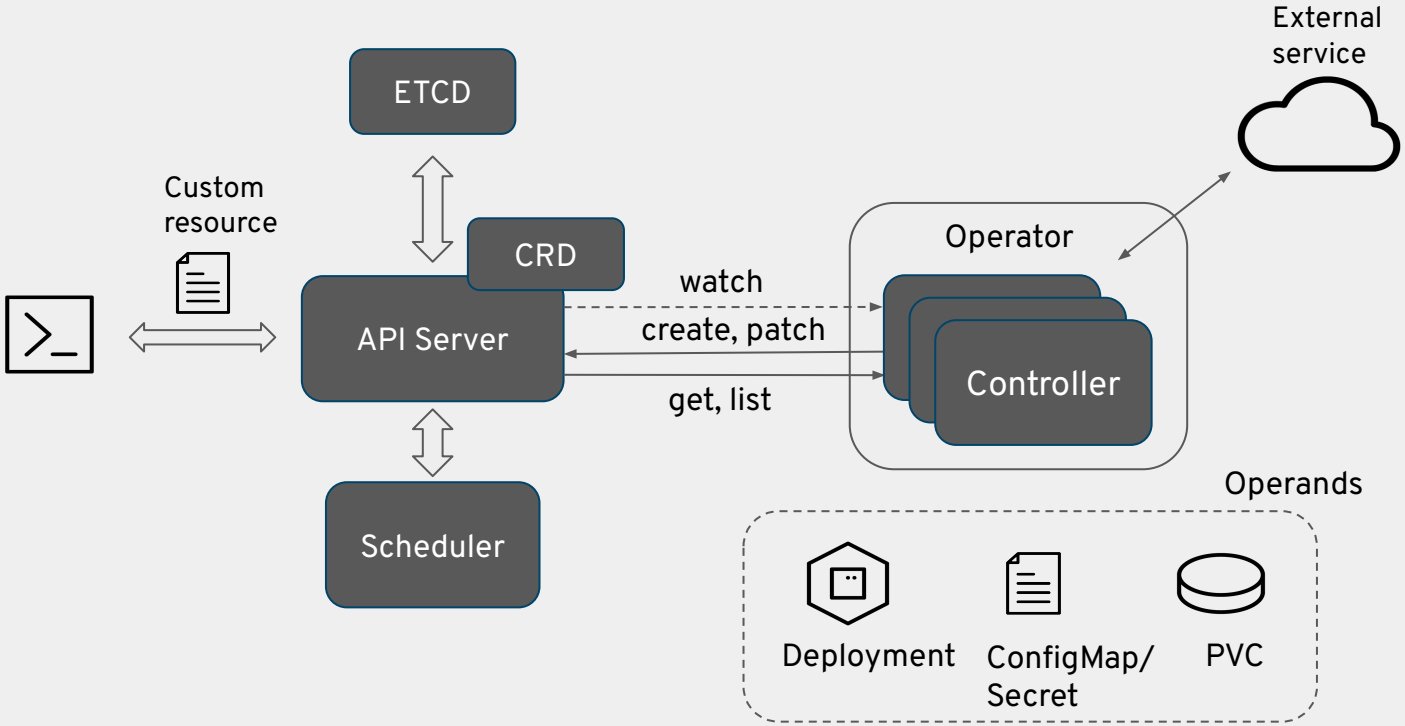
How to automate operating complex applications?
The Operator pattern!

Operator

- It’s yet another containerized application!
- Has the knowledge of a specific business domain
- Manage the application lifecycle
- Leverage CRDs (Custom Resource Definition) to extend API server
- Takes care of one (or more) custom resources/objects
  - By having a controller for each resource
  - Creating native Kubernetes resources … aka “operands”
  - Leveraging built-in controllers via API server interaction



Operator Pattern



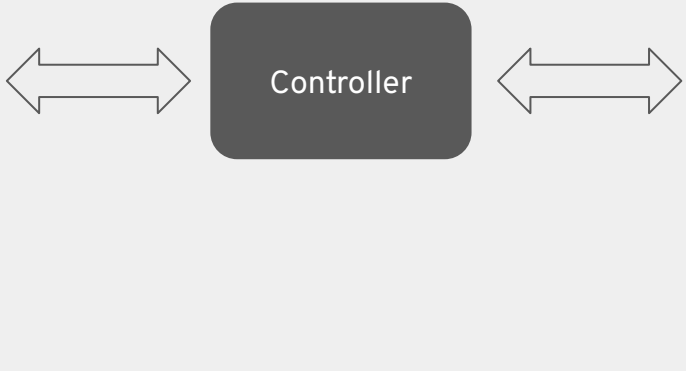
From the Custom Resource Definition ...

- Apache Kafka is a … Kubernetes resource!
- Declare a new Kubernetes “kind”
  - Group
  - Versions
- Define the new “kind” structure using an OpenAPI schema
  - Spec
  - status

```yaml
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
  name: kafkas.kafka.strimzi.io
spec:
  group: kafka.strimzi.io
  names:
    kind: Kafka
    listKind: KafkaList
  ...
  versions:
  - name: v1beta2
    schema:
      openAPIV3Schema:
        type: object
        properties:
          spec:
            # spec definition with for
            # the custom resource
            ...
          status:
            # status definition reported back
            # in the custom resource
```

… to the Custom Resource

```yaml
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
metadata:
  name: my-cluster
spec:
  kafka:
    version: 3.1.0
    replicas: 3
    listeners:
      - name: plain
        port: 9092
        type: internal
        tls: false
      - name: tls
        port: 9093
        type: internal
        tls: true
    config:
      default.replication.factor: 3
      min.insync.replicas: 2
      inter.broker.protocol.version: "3.1"
      ...
    storage:
      type: ephemeral
  zookeeper:
    replicas: 3
    storage:
      type: ephemeral
status:
  ...
  ...
```



VS

Why? What about Helm Charts?

Helm Charts

- “Package manager” for Kubernetes
  - Charts = template + values
- Rely on Kubernetes built-in resources (i.e. Deployment, ConfigMap, …)
- Simplify to write YAMLs with parameters via templating
- Simplify day-1 operation, for deploying applications
- Key problems fixed
  - Deploy same application with different configuration
  - Deploy same application on different environments

Operators

- Control “life cycle” of Kubernetes workloads
  - Operator = CRD(s) + Controller(s)
- Extend the Kubernetes API with CRDs (Custom Resource Definitions)
- Simplify to write one (or a few) “custom resource” related YAMLs
- Acting since day-1 to day-2 operation from deployment to upgrades, through manage
- … deployable via an Helm Charts :-)
  - Helm guarantees CRDs are installed before operator
  - Operator configurable via values or ConfigMap

Thank you!

CLOUD NATIVE
COMPUTING NAPOLI