Managing Kubernetes workloads

Extend the platform with operators

Paolo Patierno

Kate Stanley

Red Hat

Paolo Patierno

- Senior Principal Software Engineer at Red Hat
- CNCF Ambassador
- Strimzi maintainer
- Formula 1 & MotoGP addicted

Kate Stanley

- Principal Software Engineer at Red Hat
- Java Champion
- LinkedIn Learning Presenter
- Author

Kubernetes

Kubernetes

" A system for ..." " ... automating deployment ..."

" ... scaling ..." " ... management ..."

" ... of containerized applications ..."

Red Hat

Kubernetes

" A system for ..." " ... automating deployment ..."

" ... scaling ..." " ... management ..."

" ... of containerized applications ..."

" It's like a Linux kernel ... but for distributed systems"

Red Hat

“ … automating deployment …”

- Container scheduling
- Automated rollout/rollback
- Self healing
- Batch execution

“ … scaling …”

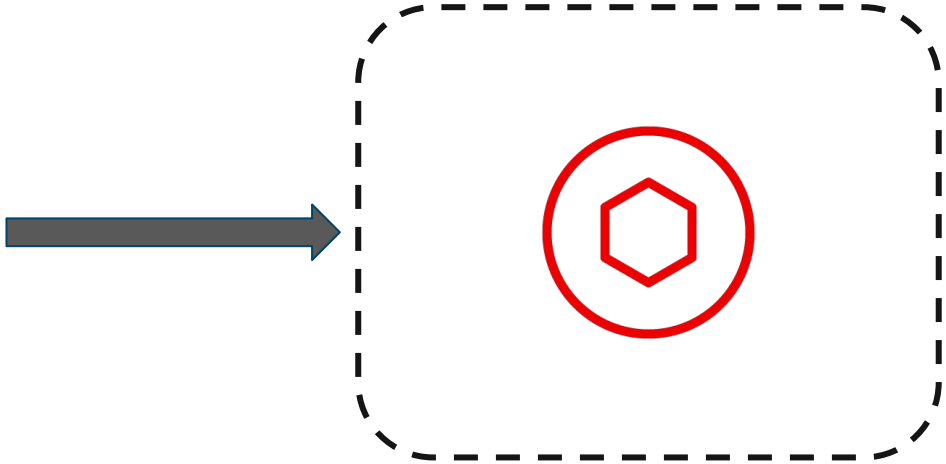
- Horizontal scaling
- Load balancing

“ … management …”

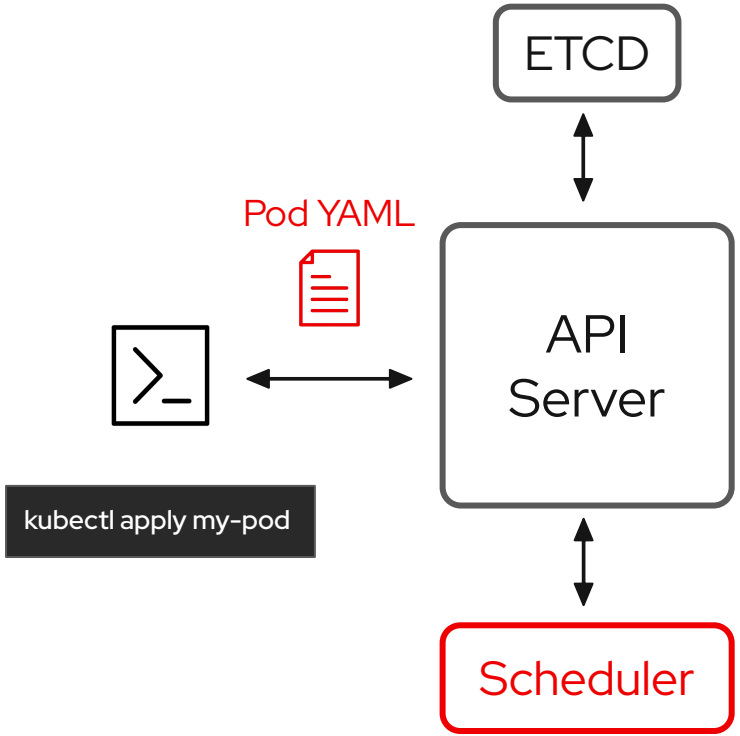
- Service discovery
- Storage orchestration
- Secret & configuration management

It's declarative!

```yaml
apiVersion: v1
kind: Pod
metadata:
 name: nginx
spec:
 containers:
 - name: nginx
   image: nginx:1.14.2
   ports:
   - containerPort: 80
```



Kubernetes cluster



Red Hat

Red Hat

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

```yaml
apiVersion: v1
kind: Pod
# ...
```

```yaml
apiVersion: apps/v1
kind: ReplicaSet
#
```

```yaml
apiVersion: apps/v1
kind: Deployment
#
```

```yaml
apiVersion: apps/v1
kind: StatefulSet
#
```

```yaml
apiVersion: v1
kind: ConfigMap
# ...
```

⋱

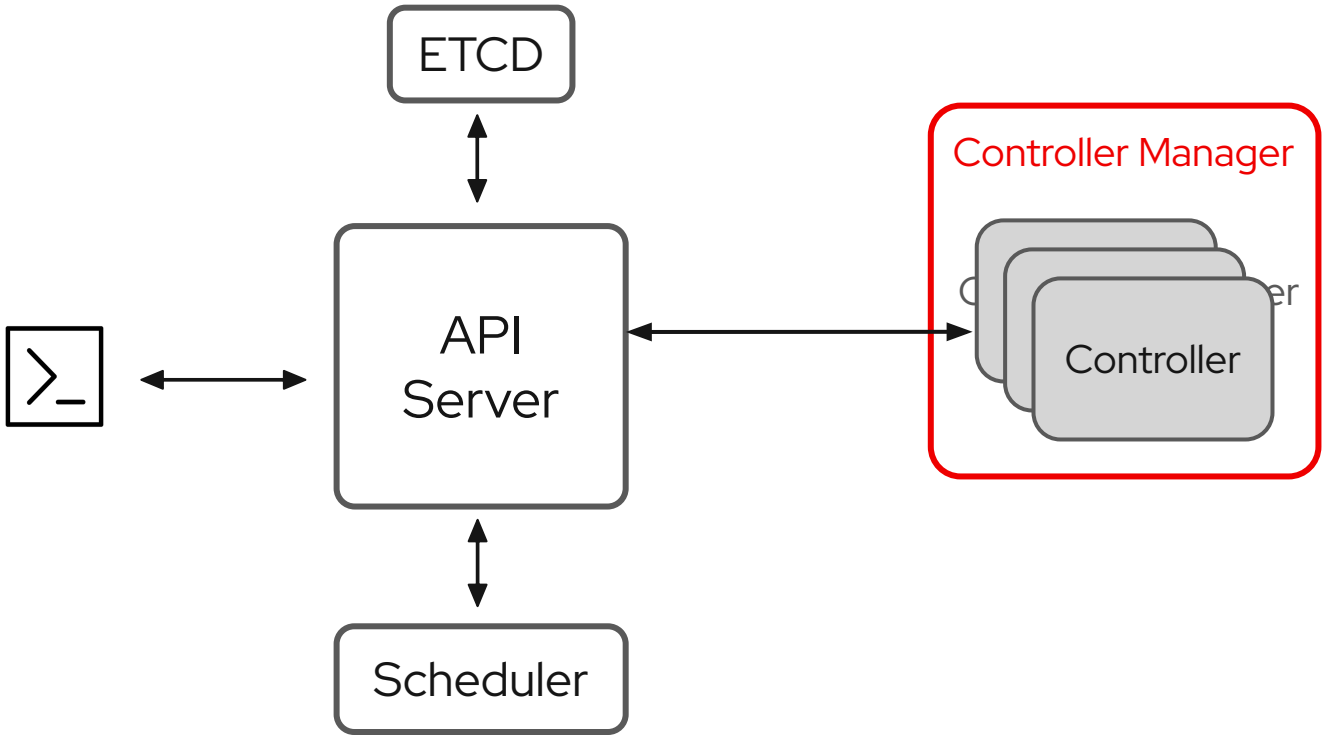
```yaml
apiVersion: v1
kind: Secret
# ...
```

How does it work?

Let’s use a controller!!!

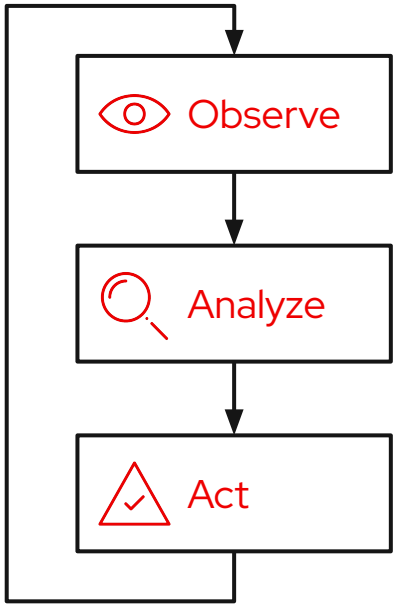
…. But not this one ;-)

Red Hat



Red Hat

Reconcile Loop

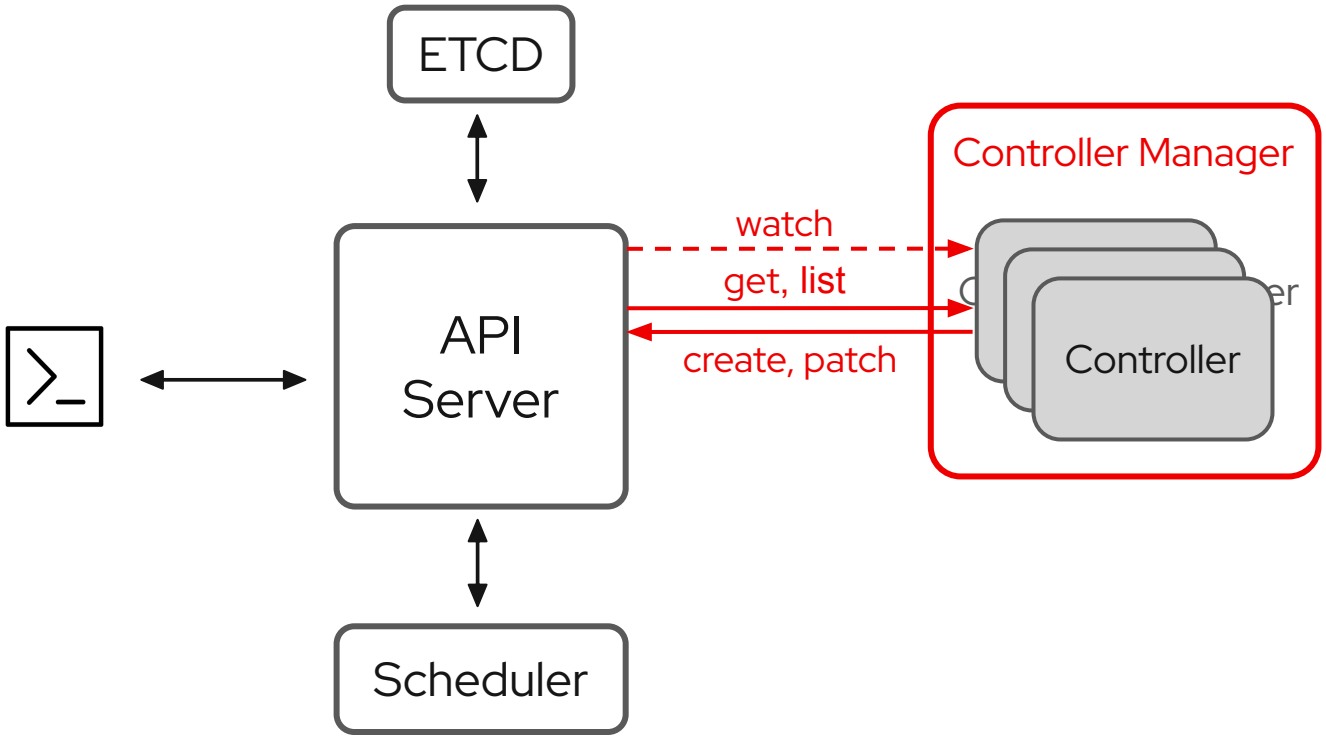


Watch for resource/object creation or changes

Check that the resource/object desired state (“spec”) reflects the current state on the cluster

Makes the needed changes

Red Hat


ETCD
Controller Manager
watch
get, list
create, patch
API
Server
Controller
Scheduler

Red Hat

Observe

ReplicaSet resource created

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
 name: my-replicaset
spec:
 replicas: 2
 selector:
   matchLabels:
     app: my-app
 template:
   metadata:
     labels:
       app: my-app
   spec:
     containers:
     - name: my-application
       image: quay.io/devoxxuk/my-application:latest
```

Analyze

2 replicas, spec

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
 name: my-replicaset
spec:
 replicas: 2
 selector:
   matchLabels:
     app: my-app
 template:
   metadata:
     labels:
       app: my-app
   spec:
     containers:
     - name: my-application
       image: quay.io/devoxxuk/my-application:latest
```

Act

Create new pods

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
 name: my-replicaset
spec:
 replicas: 2
 selector:
   matchLabels:
     app: my-app
 template:
   metadata:
     labels:
       app: my-app
   spec:
     containers:
     - name: my-application
       image: quay.io/devoxxuk/my-application:latest
```

```yaml
apiVersion: v1
kind: Pod
metadata:
 name: my-replicaset-bf5zv
 labels:
   app: my-app
spec:
 containers:
 - name: my-application
   image: quay.io/devoxxuk…
```

```yaml
apiVersion: v1
kind: Pod
metadata:
 name: my-replicast-1tf5a
 labels:
   app: my-app
spec:
 # ...
```


Red Hat

What happens if the spec changes?

Red Hat

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
 name: my-replicaset
spec:
 replicas: 2
 selector:
   matchLabels:
     app: my-app
 template:
   metadata:
     labels:
       app: my-app
   spec:
     containers:
     - name: my-application
       image: quay.io/devoxxuk/my-application:latest
```

Observe

ReplicaSet resource updated

```yaml
apiVersion: v1
kind: Pod
metadata:
 name: my-replicaset-bf5zv
 labels:
   app: my-app
spec:
 # ...
```

```yaml
apiVersion: v1
kind: Pod
metadata:
 name: my-replicaset-1tf5a
 labels:
   app: my-app
spec:
 # ...
```


Red Hat

```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/devoxxuk/my-application:latest
```

Observe

ReplicaSet resource updated

```yaml
apiVersion: v1
kind: Pod
metadata:
 name: my-replicaset-bf5zv
 labels:
   app: my-app
spec:
 # ...
```

```yaml
apiVersion: v1
kind: Pod
metadata:
 name: my-replicaset-1tf5a
 labels:
   app: my-app
spec:
 # ...
```


Red Hat

```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/devoxxuk/my-application:latest
```

Analyze

Search for pods matching resources

```yaml
apiVersion: v1
kind: Pod
metadata:
 name: my-replicaset-bf5zv
 labels:
   app: my-app
spec:
 # ...
```

```yaml
apiVersion: v1
kind: Pod
metadata:
 name: my-replicaset-1tf5a
 labels:
   app: my-app
spec:
 # ...
```


Red Hat

```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/devoxxuk/my-application:latest
```

Act

Create new pod

```yaml
apiVersion: v1
kind: Pod
metadata:
 name: my-replicaset-bf5zv
 labels:
   app: my-app
spec:
 # ...
```

```yaml
apiVersion: v1
kind: Pod
metadata:
 name: my-replicaset-1tf5a
 labels:
   app: my-app
spec:
 # ...
```

```yaml
apiVersion: v1
kind: Pod
metadata:
 name: my-replicaset-gb65f
 labels:
   app: my-app
spec:
 # ...
```


Red Hat

What happens if there are resources already created?

Red Hat

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

```yaml
apiVersion: v1
kind: Pod
metadata:
 name: my-pod-2
 labels:
   app: my-app
spec:
 # ...
```


Red Hat

```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/devoxxuk/my-application:latest
```

Observe

ReplicaSet resource created

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

```yaml
apiVersion: v1
kind: Pod
metadata:
 name: my-pod-2
 labels:
   app: my-app
spec:
 # ...
```


Red Hat

```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/devoxxuk/my-application:latest
```

Analyze

Search for pods matching resources

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

```yaml
apiVersion: v1
kind: Pod
metadata:
 name: my-pod-2
 labels:
   app: my-app
spec:
 # ...
```


Red Hat

```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/devoxxuk/my-application:latest
```

Act

Take ownership of existing pods

Create new pod

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

```yaml
apiVersion: v1
kind: Pod
metadata:
 name: my-pod-2
 labels:
   app: my-app
spec:
 # ...
```

```yaml
apiVersion: v1
kind: Pod
metadata:
 name: my-replicaset-gb65f
 labels:
   app: my-app
spec:
 # ...
```


Red Hat

Is it really that simple?

Red Hat

```yaml
apiVersion:
apps/v1
kind: ReplicaSet
# ...
spec:
 replicas: 2
 # ...
```

```yaml
apiVersion: v1
kind: Pod
metadata:
 name: nginx
spec:
 # ...
```

```yaml
apiVersion: v1
kind: Pod
metadata:
 name: nginx
spec:
 # ...
```

```yaml
apiVersion: apps/v1
kind: Deployment
# ...
spec:
 replicas: 2
 strategy:
  type: RollingUpdate
 # ...
```

```yaml
apiVersion: apps/v1
kind: Deployment
# ...
spec:
 replicas: 2
 strategy:
  type: RollingUpdate
 # ...
```

v1

```yaml
apiVersion: apps/v1
kind: ReplicaSet
# ...
spec:
 replicas: 2
 # ...
```

v1

```yaml
apiVersion: v1
kind: Pod
metadata:
 name: nginx
spec:
 # ...
```

v1

```yaml
apiVersion: v1
kind: Pod
metadata:
 name: nginx
spec:
 # ...
```

v1

```yaml
apiVersion: apps/v1
kind: Deployment
# ...
spec:
 replicas: 2
 strategy:
  type: RollingUpdate
 # ...
```

v2

```yaml
apiVersion: apps/v1
kind: ReplicaSet
# ...
spec:
 replicas: 2
 # ...
```

v1

```yaml
apiVersion: apps/v1
kind: ReplicaSet
# ...
spec:
 replicas: 1
 # ...
```

v2

```yaml
apiVersion: v1
kind: Pod
metadata:
 name: nginx
spec:
 # ...
```

v1

```yaml
apiVersion: v1
kind: Pod
metadata:
 name: nginx
spec:
 # ...
```

v1

```yaml
apiVersion: v1
kind: Pod
metadata:
 name: nginx
spec:
 # ...
```

v2

```yaml
apiVersion: apps/v1
kind: Deployment
# ...
spec:
 replicas: 2
 strategy:
  type: RollingUpdate
 # ...
```

v2

```yaml
apiVersion: apps/v1
kind: ReplicaSet
# ...
spec:
 replicas: 1
 # ...
```

v1

```yaml
apiVersion: apps/v1
kind: ReplicaSet
# ...
spec:
 replicas: 1
 # ...
```

v2

```yaml
apiVersion: v1
kind: Pod
metadata:
 name: nginx
spec:
 # ...
```

v1

```yaml
apiVersion: v1
kind: Pod
metadata:
 name: nginx
spec:
 # ...
```

v2

```yaml
apiVersion: apps/v1
kind: Deployment
# ...
spec:
 replicas: 2
 strategy:
  type: RollingUpdate
 # ...
```

v2

```yaml
apiVersion: apps/v1
kind: ReplicaSet
# ...
spec:
 replicas: 1
 # ...
```

v1

```yaml
apiVersion: apps/v1
kind: ReplicaSet
# ...
spec:
 replicas: 2
 # ...
```

v2

```yaml
apiVersion: v1
kind: Pod
metadata:
 name: nginx
spec:
 # ...
```

v1

```yaml
apiVersion: v1
kind: Pod
metadata:
 name: nginx
spec:
 # ...
```

v2

```yaml
apiVersion: v1
kind: Pod
metadata:
 name: nginx
spec:
 # ...
```

v2

```yaml
apiVersion: apps/v1
kind: Deployment
# ...
spec:
 replicas: 2
 strategy:
  type: RollingUpdate
 # ...
```

v2

```yaml
apiVersion: apps/v1
kind: ReplicaSet
# ...
spec:
 replicas: 2
 # ...
```

v2

```yaml
apiVersion: v1
kind: Pod
metadata:
 name: nginx
spec:
 # ...
```

v2

```yaml
apiVersion: v1
kind: Pod
metadata:
 name: nginx
spec:
 # ...
```

v2

What have we learnt?

- User manages highest level YAML

- Controllers perform reconcile loop
  - Observe
  - Analyze
  - Act

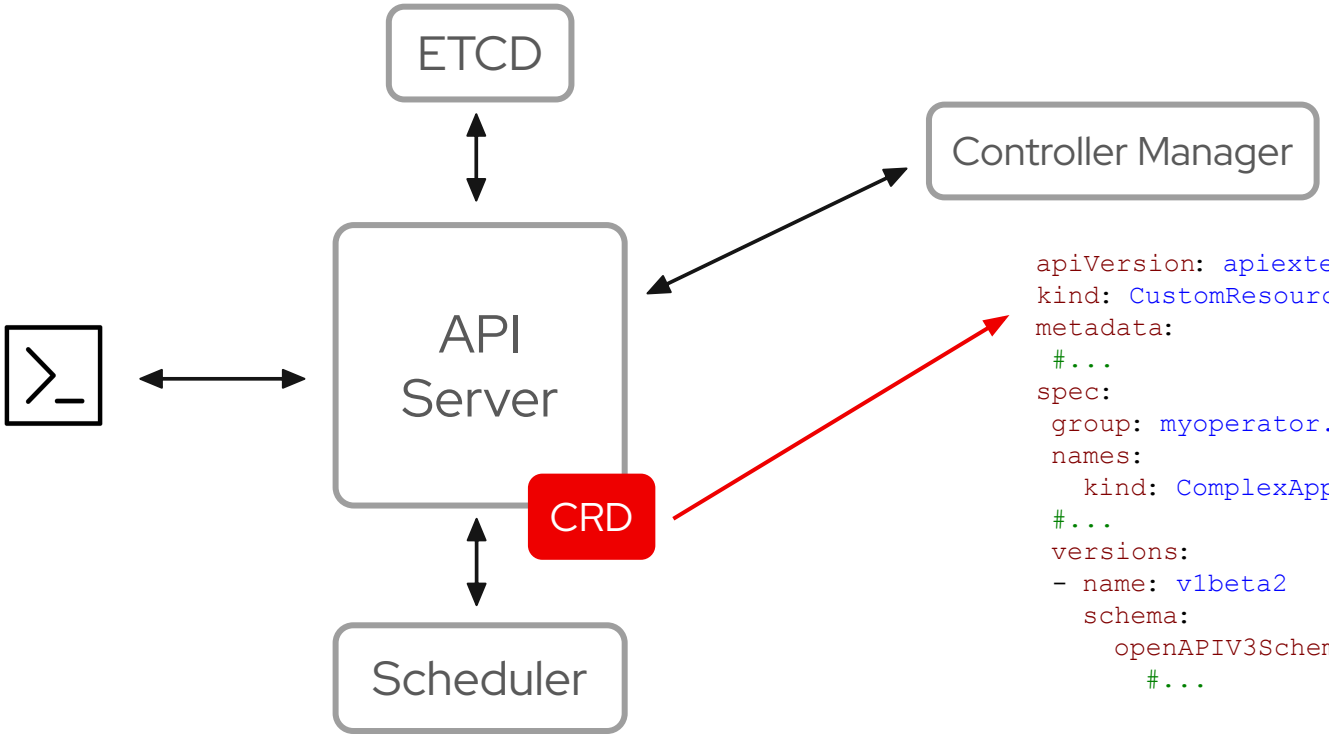
Red Hat

How to automate operating complex “containerized” applications?

Red Hat

The Operator pattern!

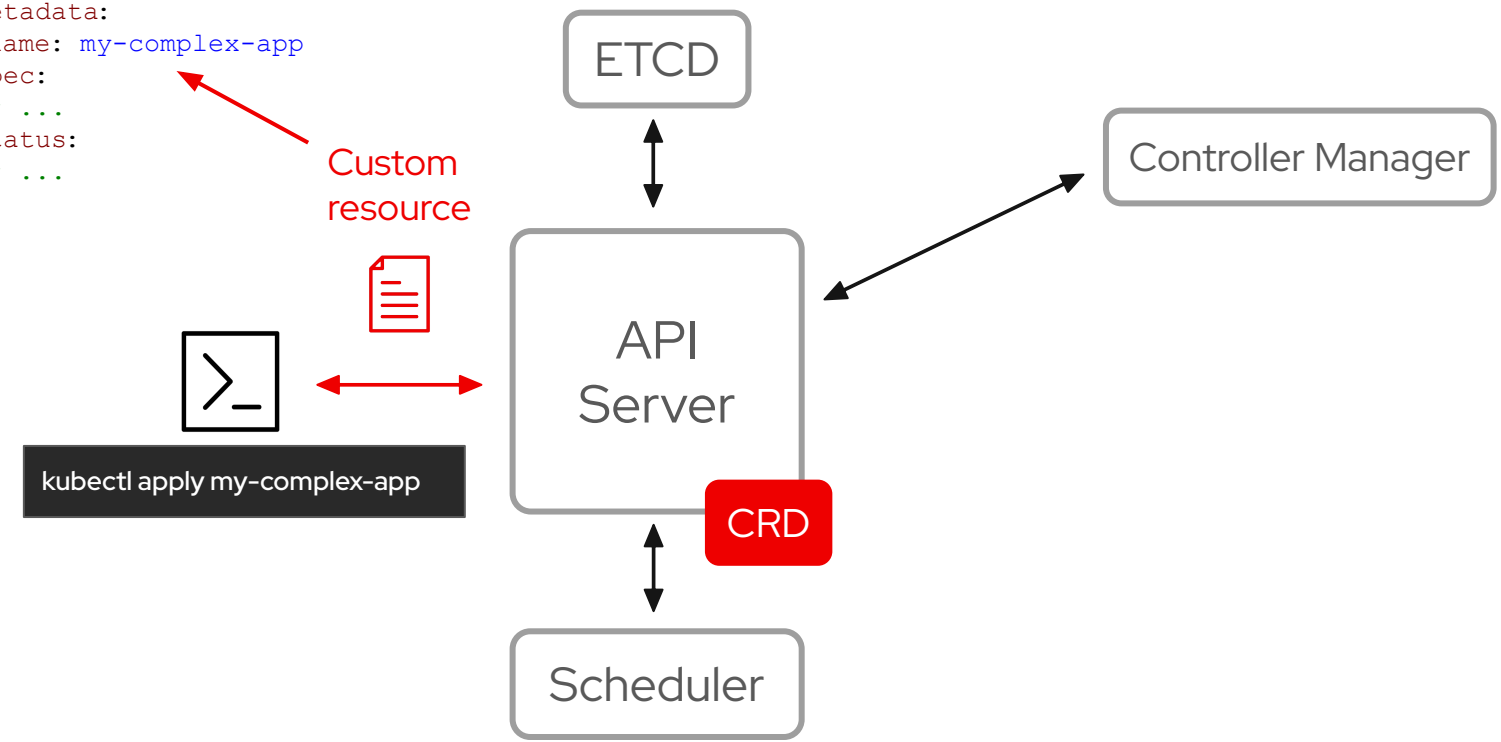
Red Hat



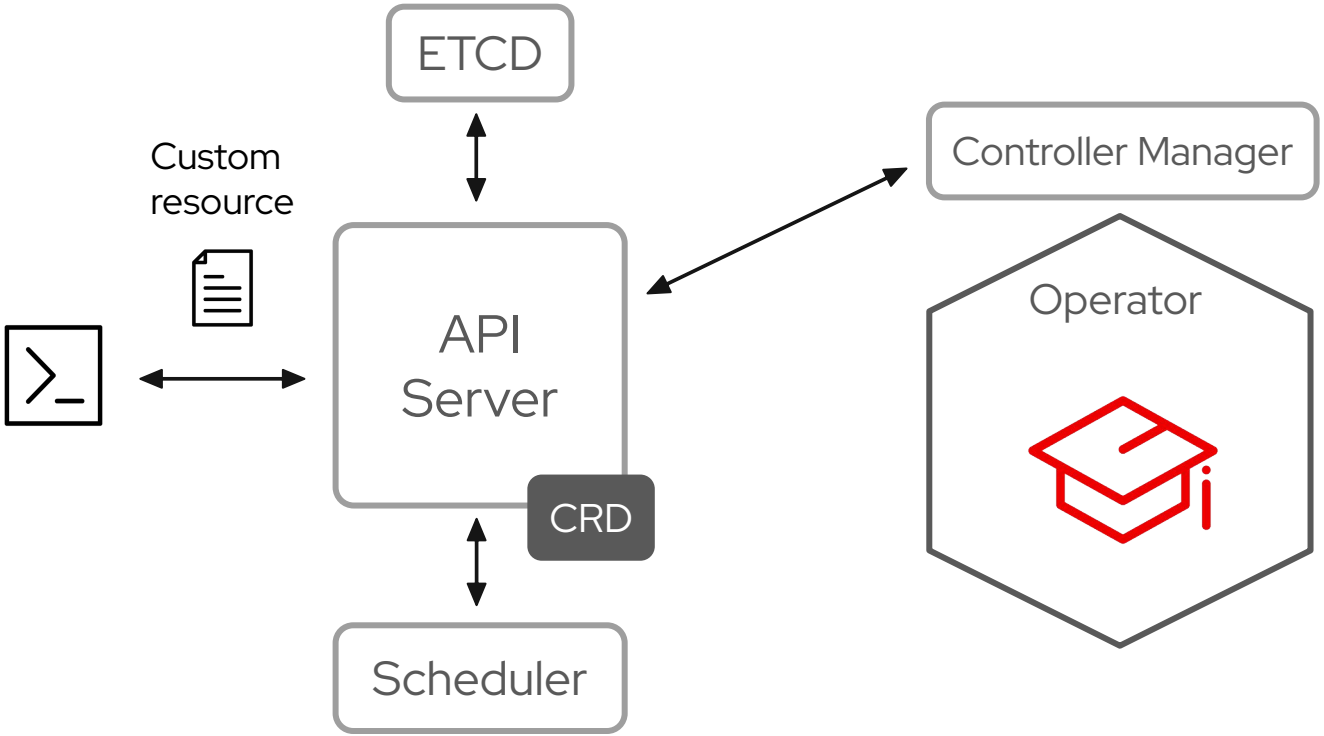

```yaml
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
 #...
spec:
 group: myoperator.io
 names:
   kind: ComplexApp
 #...
 versions:
 - name: v1beta2
   schema:
     openAPIV3Schema:
       #...
```


Red Hat

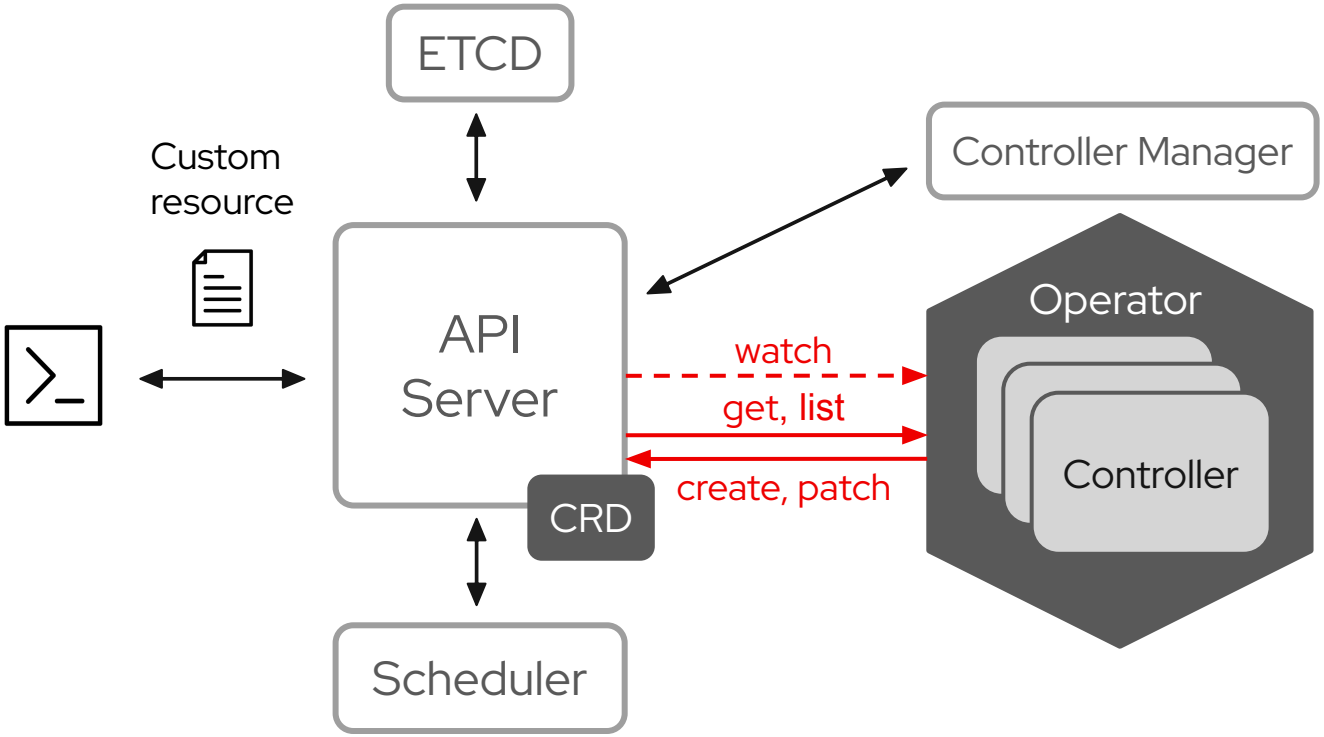
```yaml
apiVersion: myoperator.io/v1
kind: ComplexApp
metadata:
 name: my-complex-app
spec:
 # ...
status:
 # ...
```



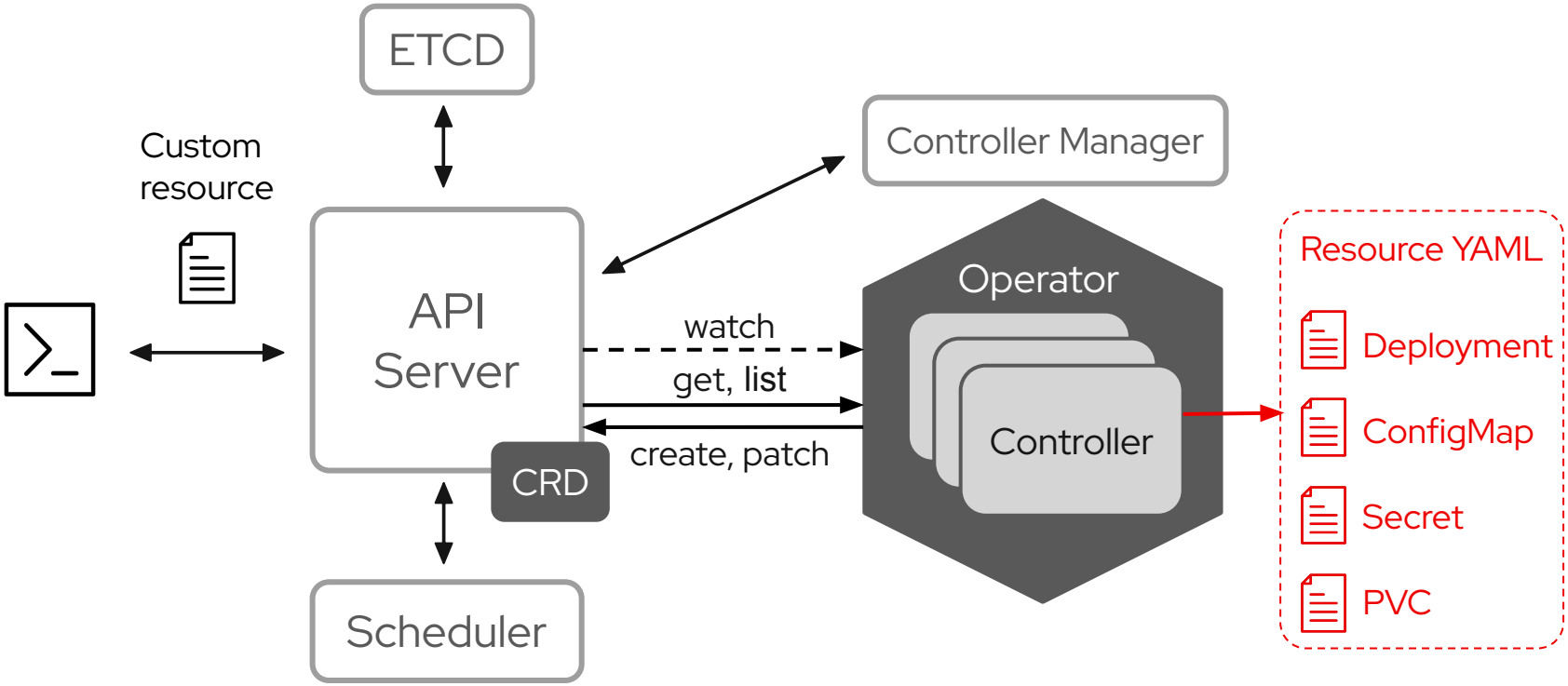
Red Hat


ETCD
Custom resource
API Server
CRD
Scheduler
Controller Manager
Operator

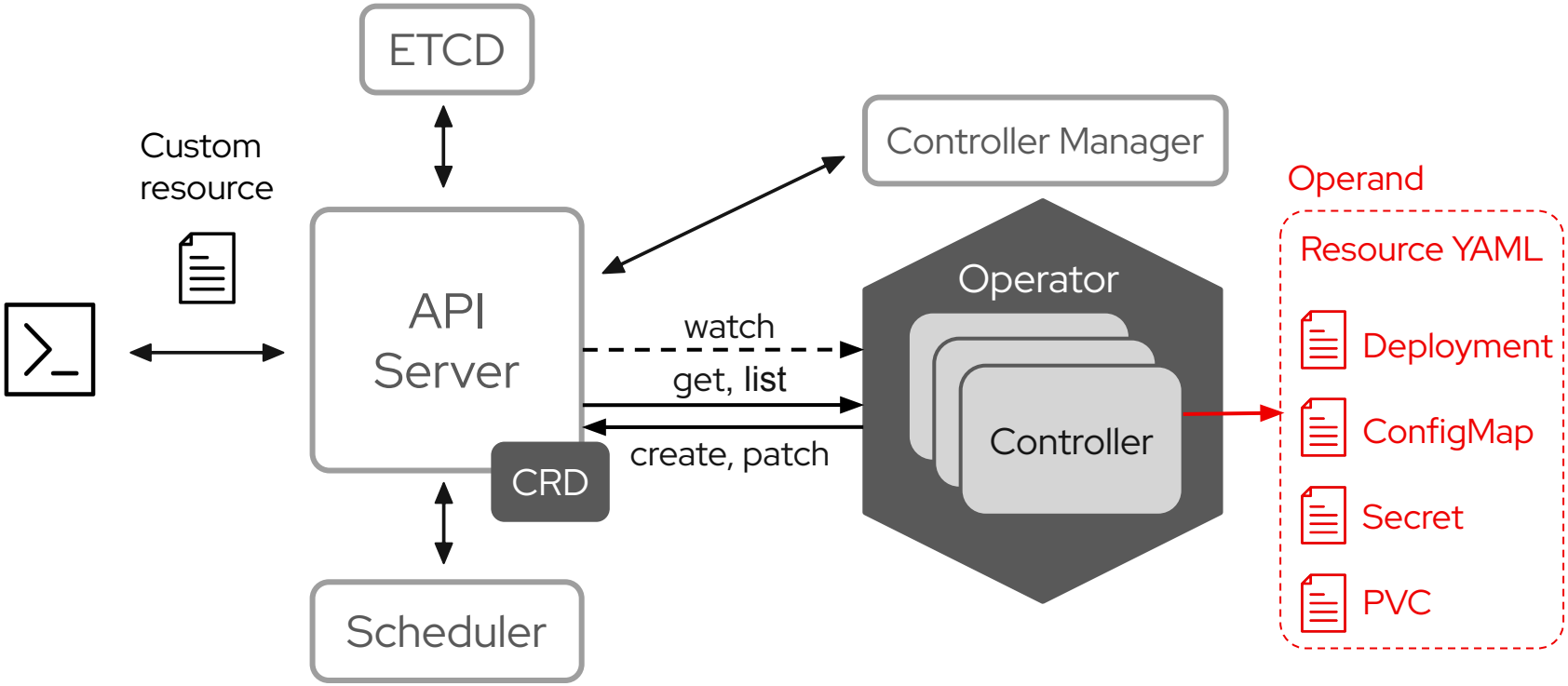
Red Hat


ETCD
Custom
resource
Controller Manager
API
Server
Operator
watch
get, list
create, patch
Controller
CRD
Scheduler

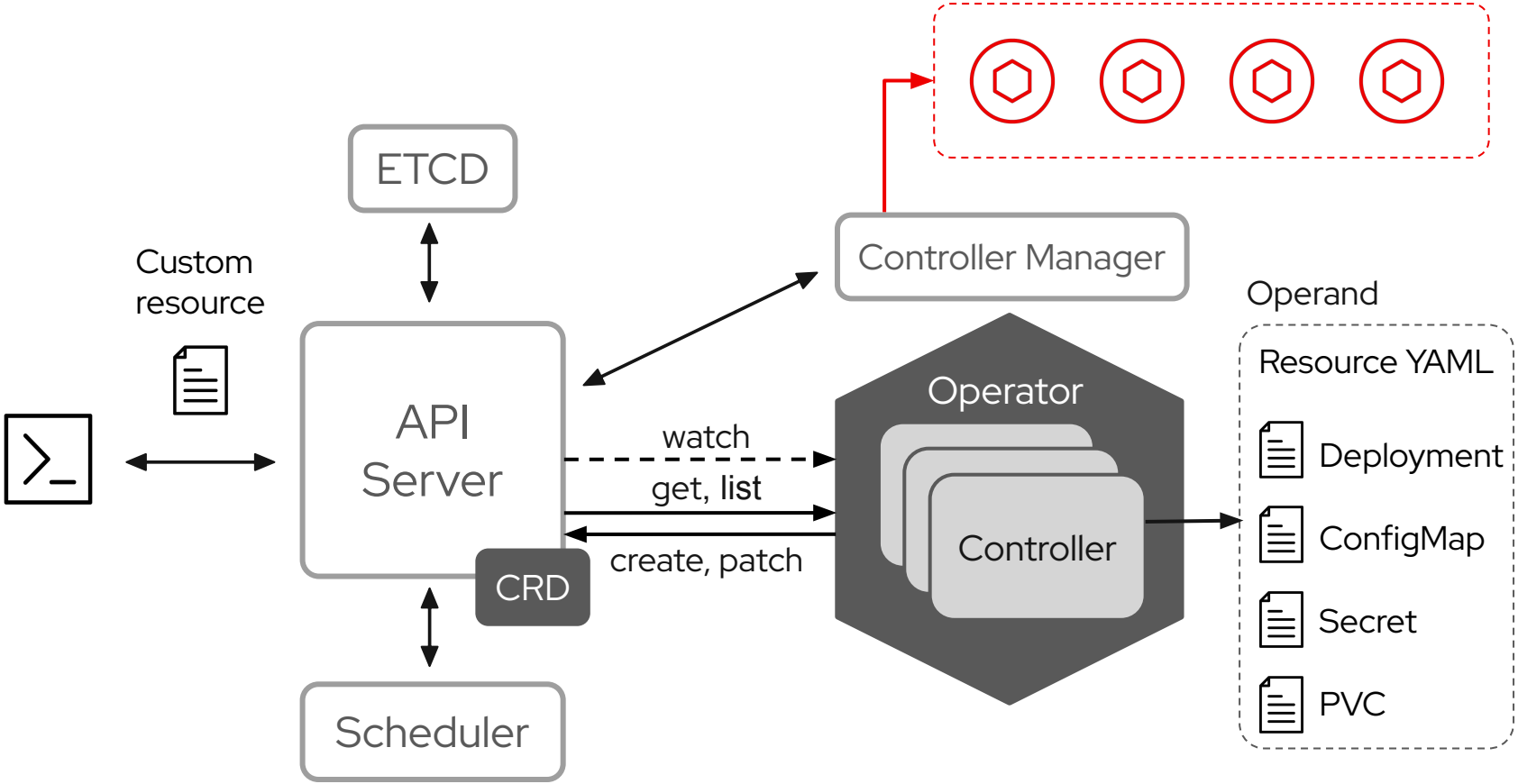
Red Hat


ETCD
Custom
resource
API
Server
CRD
Scheduler
Controller Manager
watch
get, list
create, patch
Operator
Controller
Resource YAML
Deployment
ConfigMap
Secret
PVC

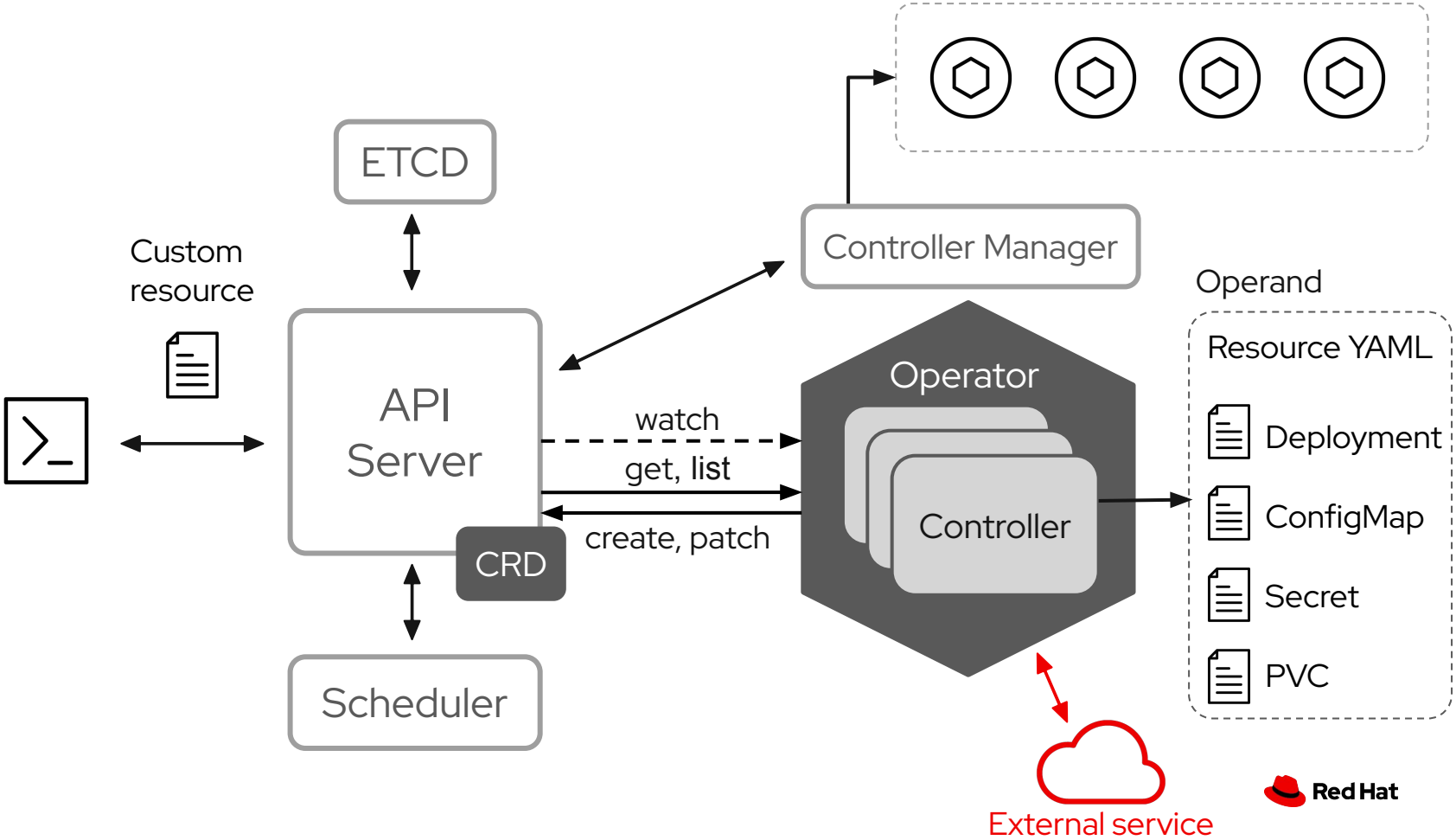
Red Hat


ETCD
Custom resource
API Server
CRD
Scheduler
Controller Manager
watch
get, list
create, patch
Operator
Controller
Operand
Resource YAML
Deployment
ConfigMap
Secret
PVC

Red Hat


ETCD
Custom resource
API Server
CRD
Scheduler
Controller Manager
watch
get, list
create, patch
Operator
Controller
Operand
Resource YAML
Deployment
ConfigMap
Secret
PVC

Red Hat


ETCD
Custom resource
API Server
CRD
Scheduler
Controller Manager
watch
get, list
create, patch
Operator
Controller
Operand
Resource YAML
Deployment
ConfigMap
Secret
PVC
External service

Red Hat

What if I want to deploy Apache Kafka on Kubernetes?

Apache Kafka

"A publish/subscribe messaging system ..."

"A data & event streaming platform ..."

Red Hat

Apache Kafka

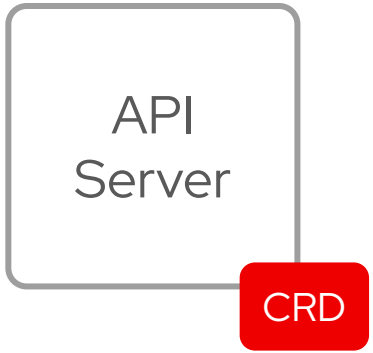
"A publish/subscribe messaging system ..."

"A data & event streaming platform ..."

with its own complexity to ...

- deploy and run
- apply configuration changes
- upgrade to a new release
- secure

```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 for the custom resource
           kafka:
             #...
         status:
           # status definition reported back
           # in the custom resource
```



GET /apis/kafka.strimzi.io/v1beta2/kafkas/

kubectl get kafka

Red Hat

```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 for the custom resource
           kafka:
             #...
         status:
           # status definition reported back
           # in the custom resource
```

```yaml
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
metadata:
 name: my-cluster
spec:
 kafka:
   version: 3.7.0
   replicas: 3
   listeners:
     - name: plain
       port: 9092
       type: internal
       tls: false
   # ...
   config:
     default.replication.factor: 3
     min.insync.replicas: 2
     # ...
   storage:
     type: ephemeral
  # ...
status:
 clusterId: UUq-xVw5TdW8GVBXappe4g
 conditions:
 - lastTransitionTime: "2024-04-09T12:51:02"
   status: "True"
   type: Ready
 kafkaMetadataState: Kraft
 kafkaMetadataVersion: 3.7-IV4
 # ...
```


Red Hat

CLOUD NATIVE
COMPUTING FOUNDATION

Strimzi

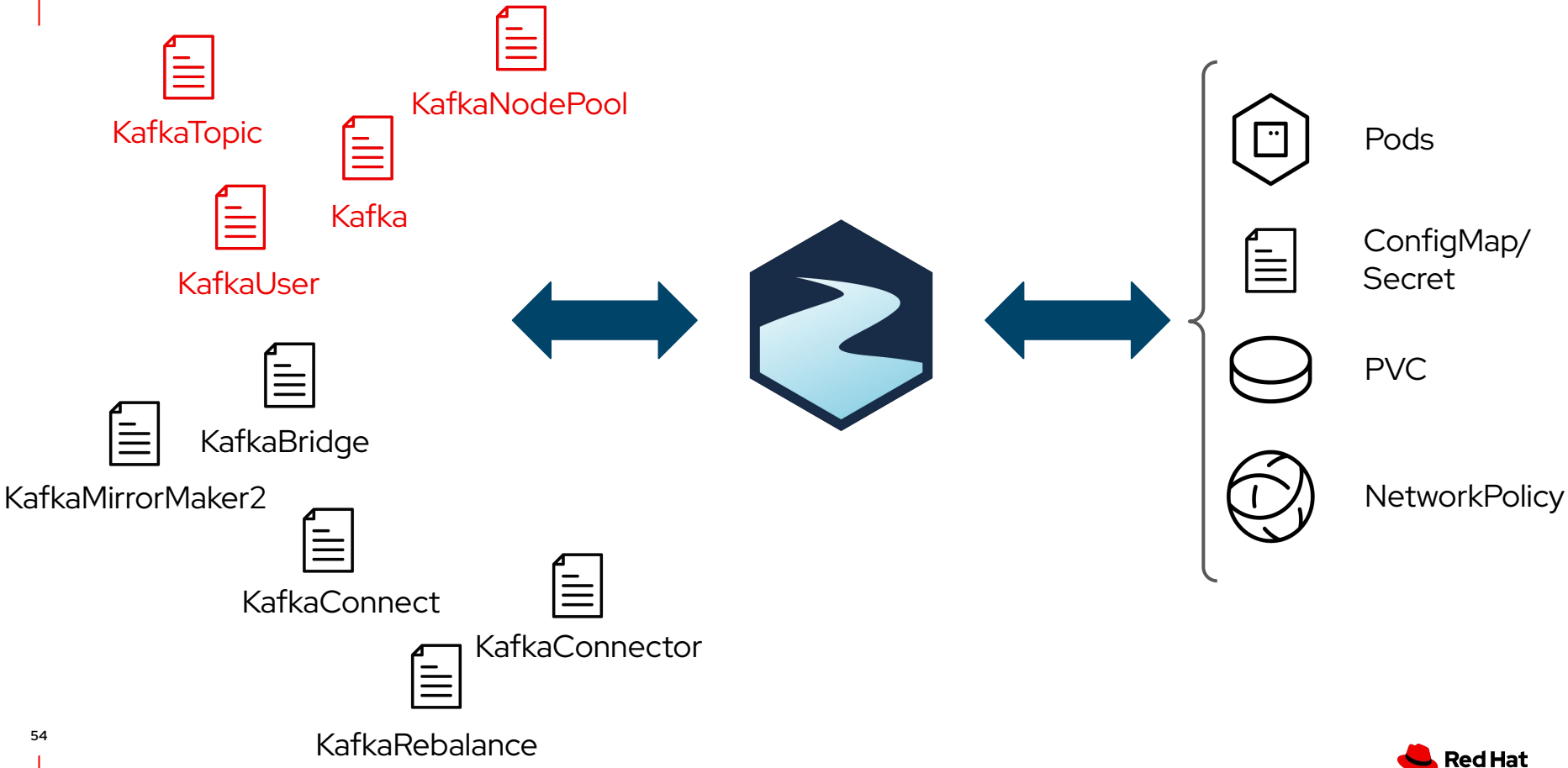
strimzi.io

Open Source project (Apache License 2.0)

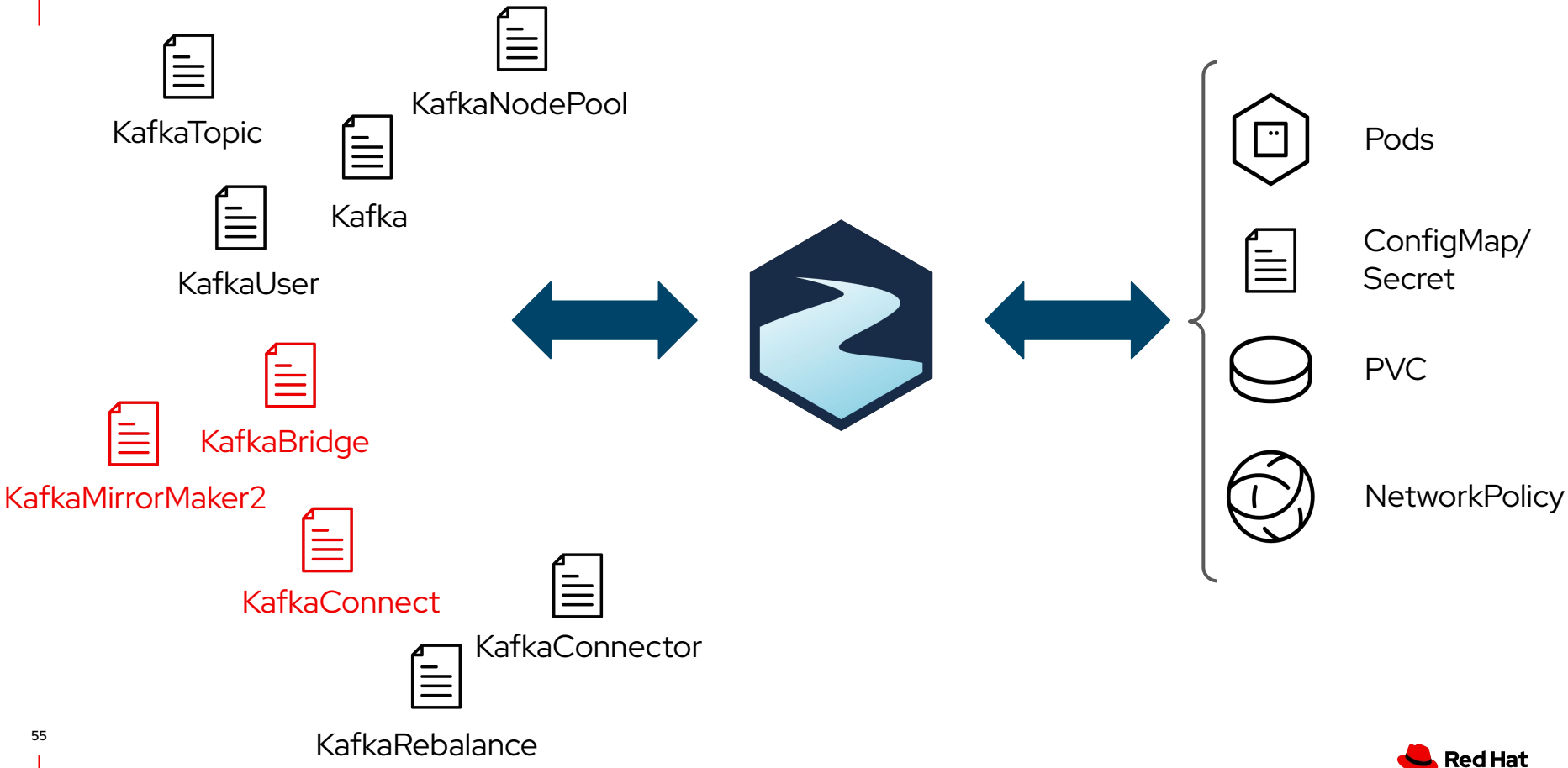
Focuses on Apache Kafka on Kubernetes

CNCF Incubating Project

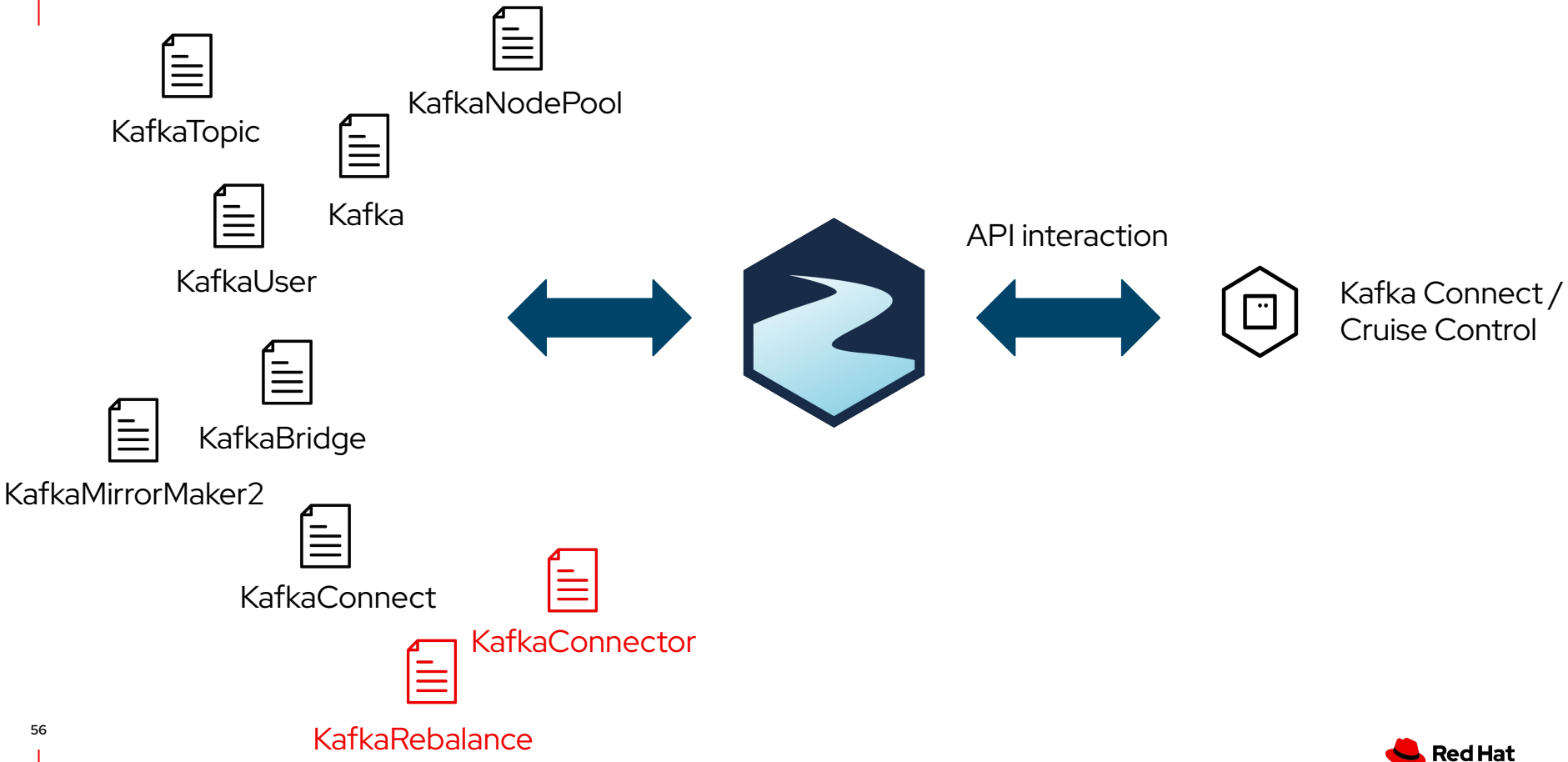
Red Hat


KafkaNodePool
KafkaTopic
Kafka
KafkaUser
KafkaBridge
KafkaMirrorMaker2
KafkaConnect
KafkaConnector
KafkaRebalance
Pods
ConfigMap/
Secret
PVC
NetworkPolicy

Red Hat


KafkaTopic
KafkaNodePool
Kafka
KafkaUser
KafkaBridge
KafkaMirrorMaker2
KafkaConnect
KafkaConnector
KafkaRebalance
Pods
ConfigMap/
Secret
PVC
NetworkPolicy

Red Hat


KafkaNodePool
KafkaTopic
Kafka
KafkaUser
API interaction
Kafka Connect /
Cruise Control
KafkaBridge
KafkaMirrorMaker2
KafkaConnect
KafkaConnector
KafkaRebalance

Red Hat

Demo time!

Red Hat

What are the alternatives?

CLOUD NATIVE
COMPUTING FOUNDATION

Helm

helm.sh

Open Source project (Apache License 2.0)

Package manager for Kubernetes

CNCF Graduated Project

Red Hat

Helm vs Operator

Helm

- Relies on Kubernetes built-in resources

Operator

- Extends the Kubernetes API with CRDs

Helm vs Operator

Helm

- Relies on Kubernetes built-in resources
- Many YAMLs with customization via templating

Operator

- Extends the Kubernetes API with CRDs
- One (or a few) “custom resource” YAMLs

Helm vs Operator

Helm

- Relies on Kubernetes built-in resources
- Many YAMLs with customization via templating
- Ideal for day-1 operation (deploying)

Operator

- Extends the Kubernetes API with CRDs
- One (or a few) “custom resource” YAMLs
- Useful for day-1 and day-2 operations (upgrading, scaling)

Helm vs Operator

Helm

- Relies on Kubernetes built-in resources
- Many YAMLs with customization via templating
- Ideal for day-1 operation (deploying)

Operator

- Extends the Kubernetes API with CRDs
- One (or a few) “custom resource” YAMLs
- Useful for day-1 and day-2 operations (upgrading, scaling)
- Deployable via Helm charts!

You’ve convinced me!

How do I start?

Red Hat

CLOUD NATIVE
COMPUTING FOUNDATION

Operator Framework

OPERATOR
FRAMEWORK

operatorframework.io

Open Source project (Apache License 2.0)

Toolkit to manage Kubernetes operators

- Build - Operator SDK
- Manage - OLM
- Discover - OperatorHub.io

CNCF Incubating Project

Red Hat

OperatorHub.io

OperatorHub.io

Search OperatorHub...

Contribute

Welcome to OperatorHub.io

OperatorHub.io is a new home for the Kubernetes community to share Operators. Find an existing Operator or list your own today.

CATEGORIES

- AI/Machine Learning
- Application Runtime
- Big Data
- Cloud Provider
- Database
- Developer Tools
- Drivers and plugins
- Integration & Delivery
- Logging & Tracing
- Modernization & Migration

352 ITEMS

VIEW SORT A-Z

Aerospike Kubernetes Operator
provided by Aerospike
The Aerospike Kubernetes Operator automates the

Airflow Helm Operator
provided by opdev
An experimental operator that installs Apache Airflow.

Aiven Operator
provided by aiven
Manage your https://aiven.io resources with Kubernetes.

Akka Cluster Operator
provided by Lightbend, Inc.
Run Akka Cluster applications on Kubernetes.

Altinity Operator for ClickHouse
provided by Altinity
ClickHouse Operator manages full lifecycle of ClickHouse

Red Hat

[JAVA OPERATOR SDK]

javaoperatorsdk.io

Java Operator SDK

Open Source project (Apache License 2.0)

Framework for writing Java based operators

Built on top of the Fabric8 Client

Thank you

Operator Pattern - https://kubernetes.io/docs/concepts/extend-kubernetes/operator/

Strimzi - strimzi.io

Apache Kafka at Red Hat - https://developers.redhat.com/topics/kafka-kubernetes

github.com/ppatierno

github.com/katheris

Red Hat