
KubeCon
CloudNativeCon
North America 2023


KubeCon | CloudNativeCon

North America 2023

What's New in Operator Framework?

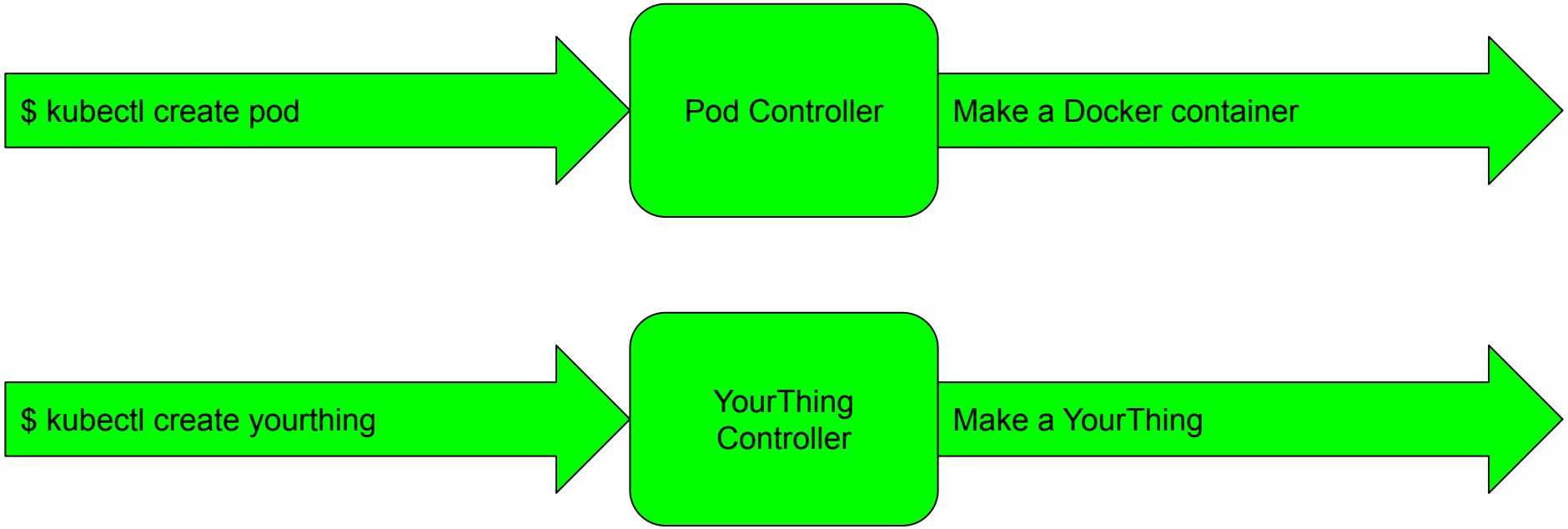
Jonathan Berkhahn, IBM
Attila Mészáros, Red Hat

Who are we?

- Jonathan Berkhahn
  - Open Source Contributor for IBM
  - Member of Operator Framework Steering Committee
  - Previously contributed to Kubernetes, Cloud Foundry
- Attila Mészáros
  - Engineer at Red Hat
  - Maintainer of Java Operator SDK
  - Member of Operator Framework Steering Committee

What is an Operator?

- Operators are a design pattern that contain application specific logic that is usually handled by a human operator, hence the term “operator”.



What is Operator Framework?

An open source toolkit to manage Kubernetes native applications, called Operators, in an effective, automated, and scalable way.

Things we make include:

- Java Operator SDK (!)
- Operator SDK
- Operator Lifecycle Manager (OLM)
- Operator Package Manager (OPM)

V1

- Major refactoring of our internal APIs and the components that comprise them
- OLM and it’s resources (CatalogSources, etc.) going away
- More generic resources that are more human-usable

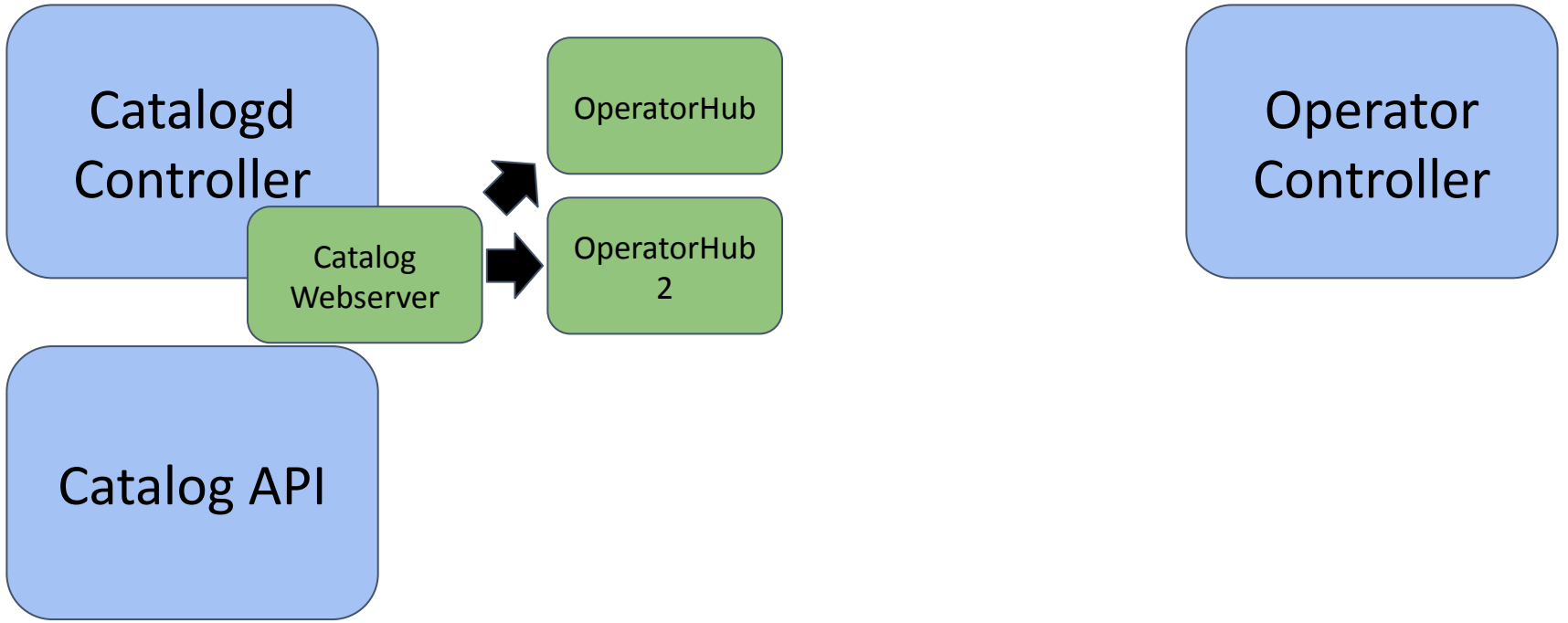
V1

Catalogd Controller

Operator Controller

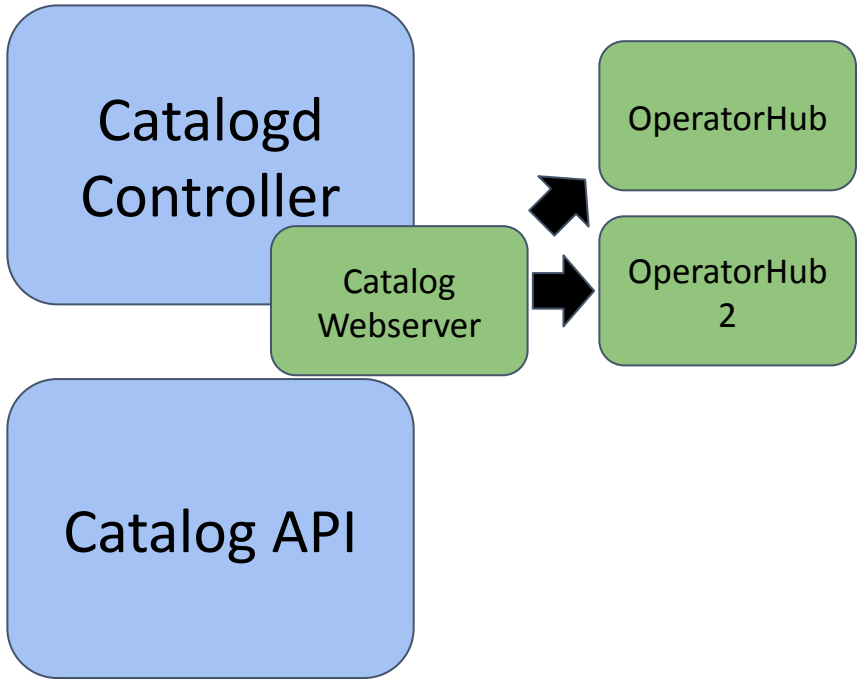
V1

“YUM repo”


Catalogd Controller
Catalog Webserver
OperatorHub
OperatorHub 2
Catalog API
Operator Controller

V1

“YUM repo”



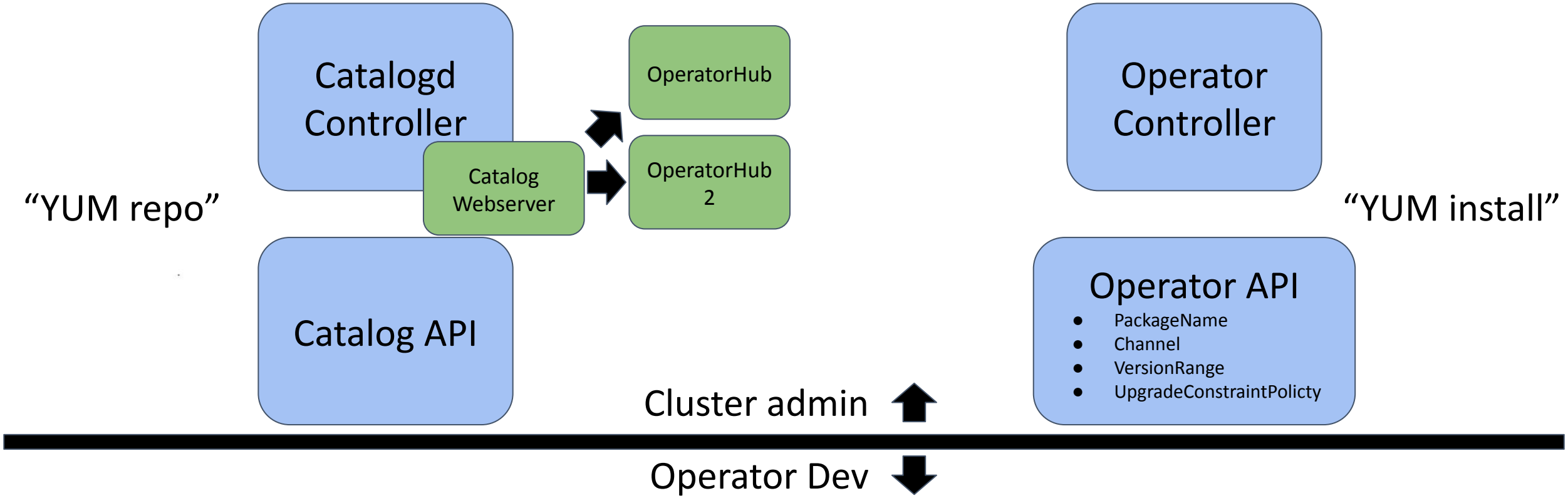
Operator Controller

“YUM install”

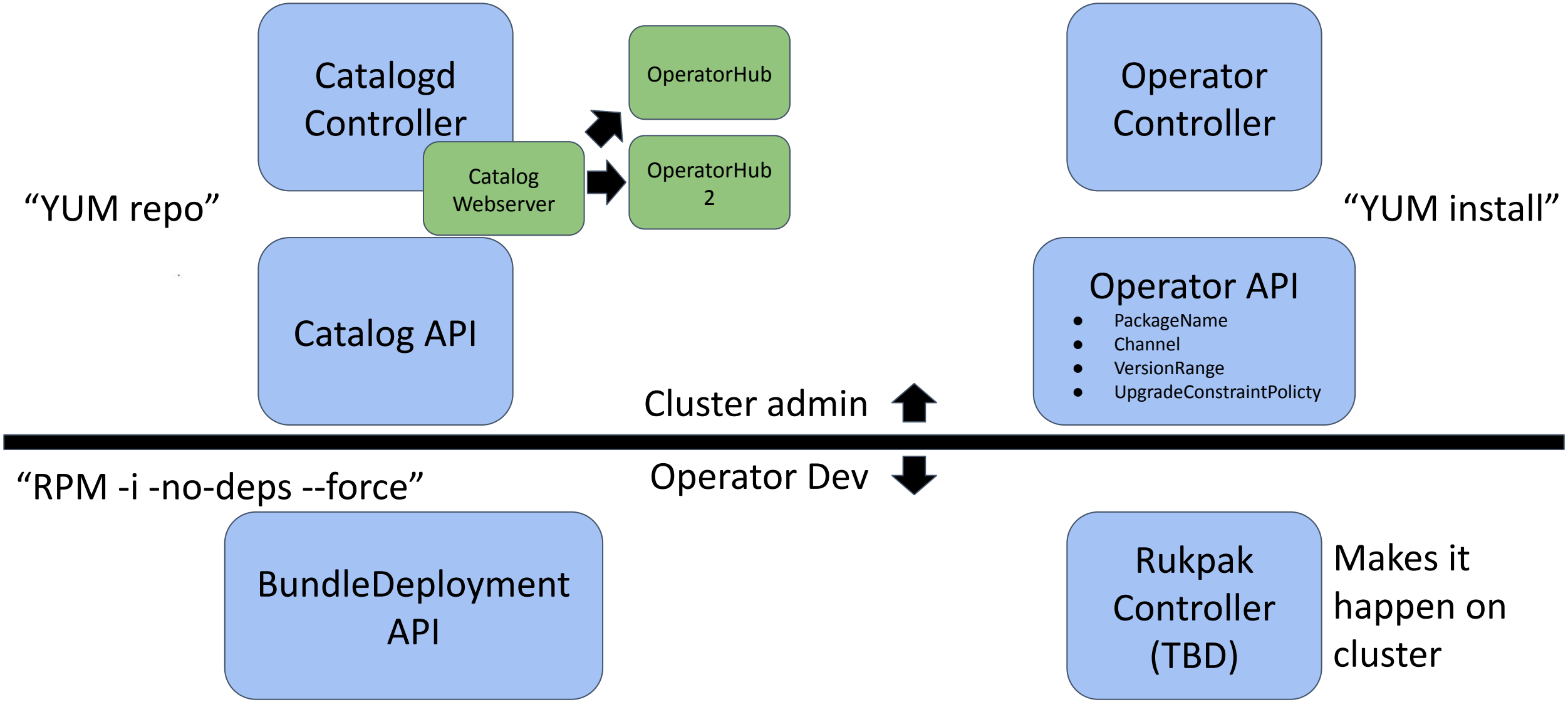
Operator API

- PackageName
- Channel
- VersionRange
- UpgradeConstraintPolicty

V1


Catalogd Controller
Catalog Webserver
OperatorHub
OperatorHub 2
Operator Controller
“YUM repo”
“YUM install”
Catalog API
Operator API
PackageName
Channel
VersionRange
UpgradeConstraintPolicty
Cluster admin
Operator Dev

V1


Catalogd Controller
Catalog Webserver
OperatorHub
OperatorHub 2
Operator Controller
"YUM repo"
"YUM install"
Catalog API
Operator API
PackageName
Channel
VersionRange
UpgradeConstraintPolicty
Cluster admin
Operator Dev
"RPM -i -no-deps --force"
BundleDeployment API
Rukpak Controller (TBD)
Makes it happen on cluster

V1

- API that is declarative, more in-line with Kube API philosophy
- Git ops friendly
- Human friendly
- Non-explody :)


KubeCon | CloudNativeCon

North America 2023

Java Operator SDK in a Nutshell

Attila Mészáros (@csviri)

What is Java Operator SDK?

- https://github.com/operator-framework/java-operator-sdk
- Feature complete and production ready framework to implement K8S Operators in Java
- Core framework (includes controller runtime)
- Additional Components
- Users and/or target audience
  - Keycloak, Flink, Debezium,...

Components

- Core operator framework
- Built on top of **Fabric8 Kubernetes Client**
  - Generator for Java -> CRD & CRD -> Java
  - Informers, Leader Election and all the standard functionalities also in go-client
- Support for integration testing (JUnit5 extension)
- Java Framework Support
  - **Quarkus Extension**
  - Spring Boot Starter
- (Go) Operator SDK Plugin - scaffolding
- Kubernetes Webhooks Framework
  - Conversion Hooks
  - Dynamic Admission Controllers

Quarkus Extension

- https://github.com/quarkiverse/quarkus-operator-sdk
- Builds upon core Java Operator SDK
- Build time optimization for Quarkus
- Native builds
- Nice configuration approach
- Support for Helm, OLM, pure K8S resource generator

More About the Core Framework

- Similar concepts as in go controller-runtime
  - but the “Java way”
- Batteries included
- Bit higher level regarding features like:
  - **Finalizer handling**
  - Declarative Retries
  - Error Handling
  - Rate Limiting
  - …

Implementing a Reconciler


```java
@ControllerConfiguration
public class SampleOperatorReconciler implements Reconciler<SampleCustomResource> {

    public UpdateControl<SampleCustomResource> reconcile(
            SampleCustomResource primary, Context<SampleCustomResource> context) {

        // reconciliation logic

        return UpdateControl.patchStatus(primary);
    }
}
```

Run it!


```java
public class Runner {

    private static final Logger log = LoggerFactory.getLogger(Runner.class);

    public static void main(String[] args) {
        Operator operator = new Operator();
        operator.register(new SampleOperatorReconciler());
        operator.start();
    }
}
```

Support for Finalizers

- So what “higher level” approach means?
- Finalizers are added when explicit cleanup is needed
  - Thus cleanup not covered by K8S Garbage Collector (Owner References)
- Just implement “*Cleaner*” interface

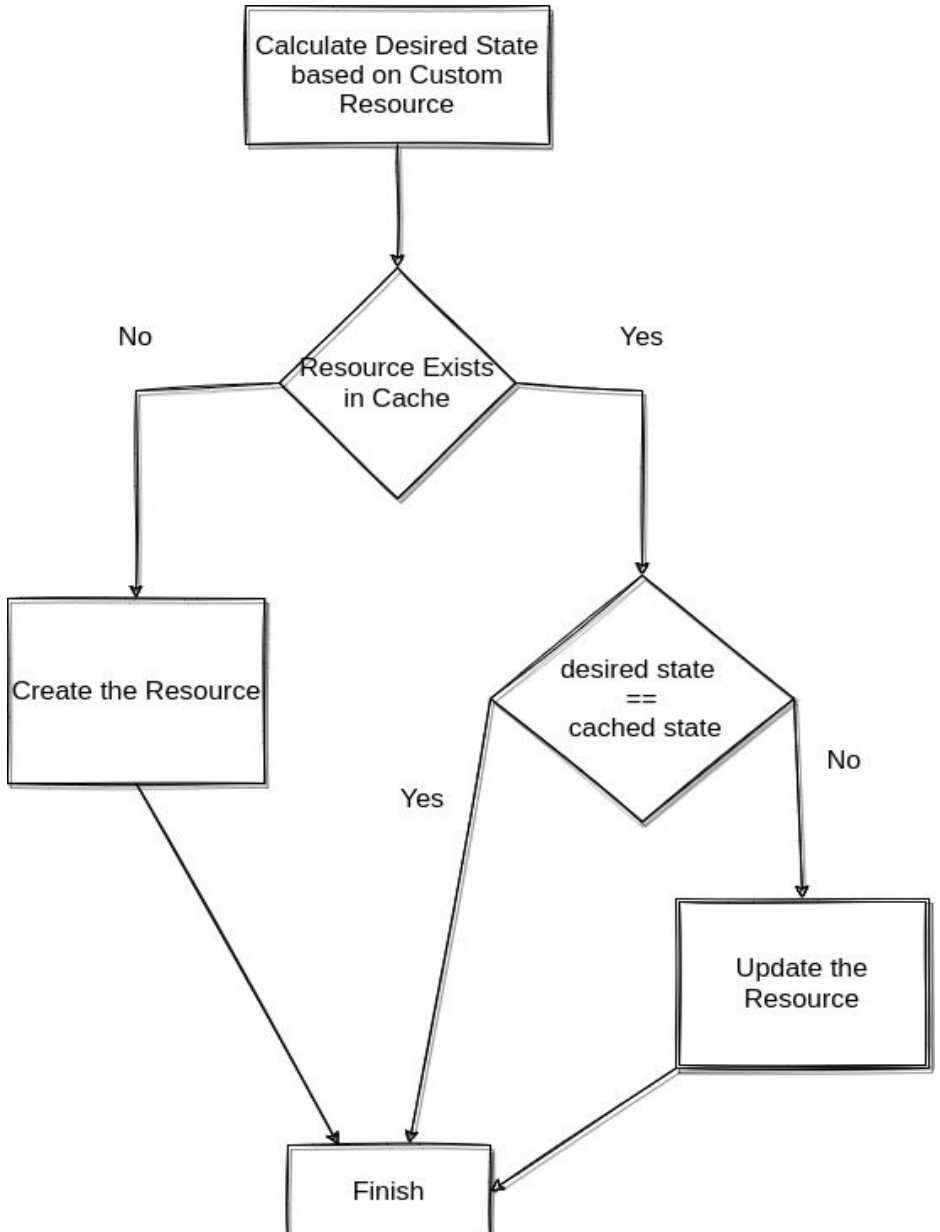
Cleaner Interface


```java
@ControllerConfiguration
public class SampleOperatorReconciler implements Reconciler<SampleCustomResource>,
        Cleaner<SampleCustomResource> {

    public UpdateControl<SampleCustomResource> reconcile(
            SampleCustomResource primary, Context<SampleCustomResource> context) {

        return UpdateControl.patchStatus(primary);
    }

    @Override
    public DeleteControl cleanup(SampleCustomResource sampleCustomResource,
                                 Context<SampleCustomResource> context) {

        // cleanup logic

        return DeleteControl.defaultDelete();
    }
}
```

Dependent Resources

- Abstraction for reconciliation of a single resource
  - Kubernetes and/or External resources
- Desired state!

- Kubernetes and External Resources
- Highly extensible and customizable
- Much more to it:
  - Trait Interfaces
  - Bulk resources
  - External resource with state
  - SSA / no SSA
  - …



ConfigMap Dependent Resource


```java
public class ConfigMapDependentResource
        extends CRUDKubernetesDependentResource<ConfigMap, SampleCustomResource> {

    public ConfigMapDependentResource() {
        super(ConfigMap.class);
    }

    @Override
    protected ConfigMap desired(SampleCustomResource primary,
                                Context<SampleCustomResource> context) {
        return new ConfigMapBuilder()
                .withMetadata(new ObjectMetaBuilder()
                                .withName(primary.getMetadata().getName())
                                .withNamespace(primary.getMetadata().getNamespace())
                                .build())
                .withData(Map.of("data", primary.getSpec().getValue()))
                .build();
    }
}
```

Using Dependent Resource


```java
@ControllerConfiguration(dependents = {
        @Dependent(type = ConfigMapDependentResource.class)})
public class SampleOperatorReconciler implements Reconciler<SampleCustomResource> {

    public UpdateControl<SampleCustomResource> reconcile(
            SampleCustomResource primary, Context<SampleCustomResource> context) {

        // addtional logic

        return UpdateControl.patchStatus(primary);
    }
}
```

Standalone Dependent Resource


```java
@ControllerConfiguration
public class SampleOperatorReconciler implements Reconciler<SampleCustomResource> {

    private KubernetesDependentResource<ConfigMap, SampleCustomResource> configMapDR;

    public UpdateControl<SampleCustomResource> reconcile(
            SampleCustomResource primary, Context<SampleCustomResource> context) {

        configMapDR.reconcile(primary,context);

        // other logic

        return UpdateControl.patchStatus(primary);
    }

    // omitted code

}
```

Workflow Sample


```java
@ControllerConfiguration(
    dependents = {
        @Dependent(name = "first", type = FirstService.class)
        @Dependent(name = "second", type = SecondService.class)
        @Dependent(name = "statefulset", type = FirstStatefulSet.class,
            dependsOn = {"first"},
            readyPostcondition = StatefulSetReadyCondition.class),
        @Dependent(name = "second",
            type = SecondStatefulSet.class,
            dependsOn = {"second", "statefulset"},
            reconcilePrecondition = MyPrecondition.class)
    })
public class ComplexDependent}Reconciler implements Reconciler<ComplexDependentCustomResource>,
    EventSourceInitializer<ComplexDependentCustomResource> {

// ...

}
```

Workflows

- Handles reconciliation of multiple dependent resources
- Will ensure optimal reconciliation execution
  - Async
  - Concurrent
- “Depends on” relation
  - Ordering
- Conditions
  - Reconcile precondition
  - Ready postcondition
  - …

Dependent Resources and Workflows

- Terraform / IaaC analogy
- … just adopted to K8S operator landscape
- Cache and eventing optimizations
- Results in easy, correct and optimal implement of reconciliation


KubeCon | CloudNativeCon

North America 2023

Thank you! Q&A