KubeCon | CloudNativeCon

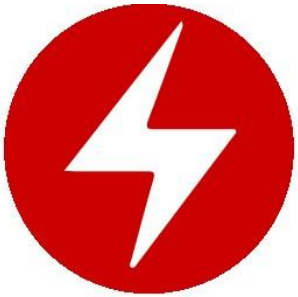
North America 2024

What's New in Operator Framework?

Bryce Palmer, Red Hat
Lalatendu Mohanty, Red Hat
Rashmi Gottipati, Red Hat
Attila Mészáros, Apple

Operator Lifecycle Manager (OLM)

- Designed to simplify the management of Kubernetes native applications.
- We have 370+ operators from various vendors in operatorhub.io
- OLM v0 github release has been downloaded more than ~9.5 million
- The github project has around 1.7K stars.



OPERATOR FRAMEWORK

Current State Of The Project

- The community has been working on a new major version of the project coined “OLM v1”
- The existing version of the project, “OLM v0”, is currently in maintenance
  - Project is not accepting new features
  - Blocker/ Critical issues and critical CVEs will be addressed on a best-effort basis
  - More information in the project readme

Why OLM v1?

- Simpler API and Mental Model
  a. Streamlined APIs
     i. OLM v0 has 9 APIs vs OLMv1 has 2
  b. Intuitive design
- Security by Default
  a. Enhanced security features out-of-the-box, reducing vulnerabilities.

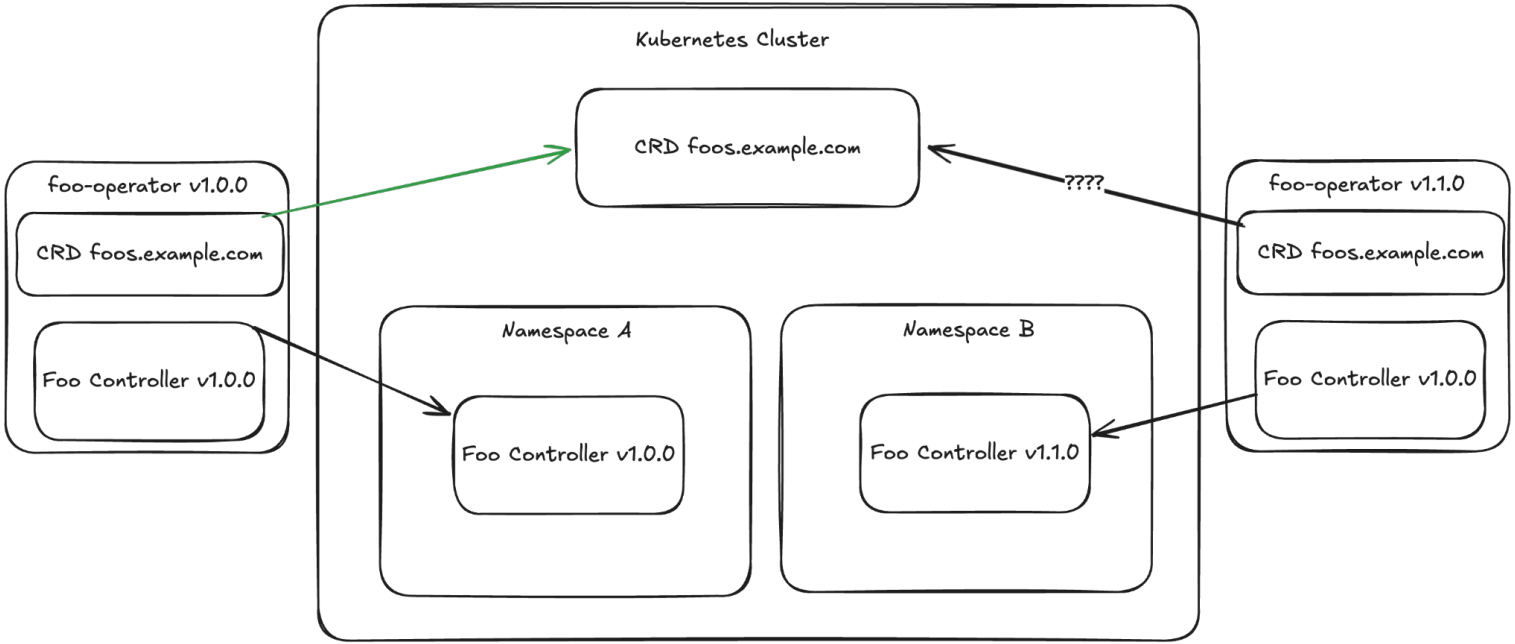
Why OLM v1? (Continued ..)

- Greater Flexibility
  a. Less rigid automation, allowing for more customization and broader use cases
  b. Beyond Operators
     i. Support for a wider range of Kubernetes applications, not limited to Operators
  c. Support for popular packaging formats (e.g. Helm)

OLM v1 Design Decisions

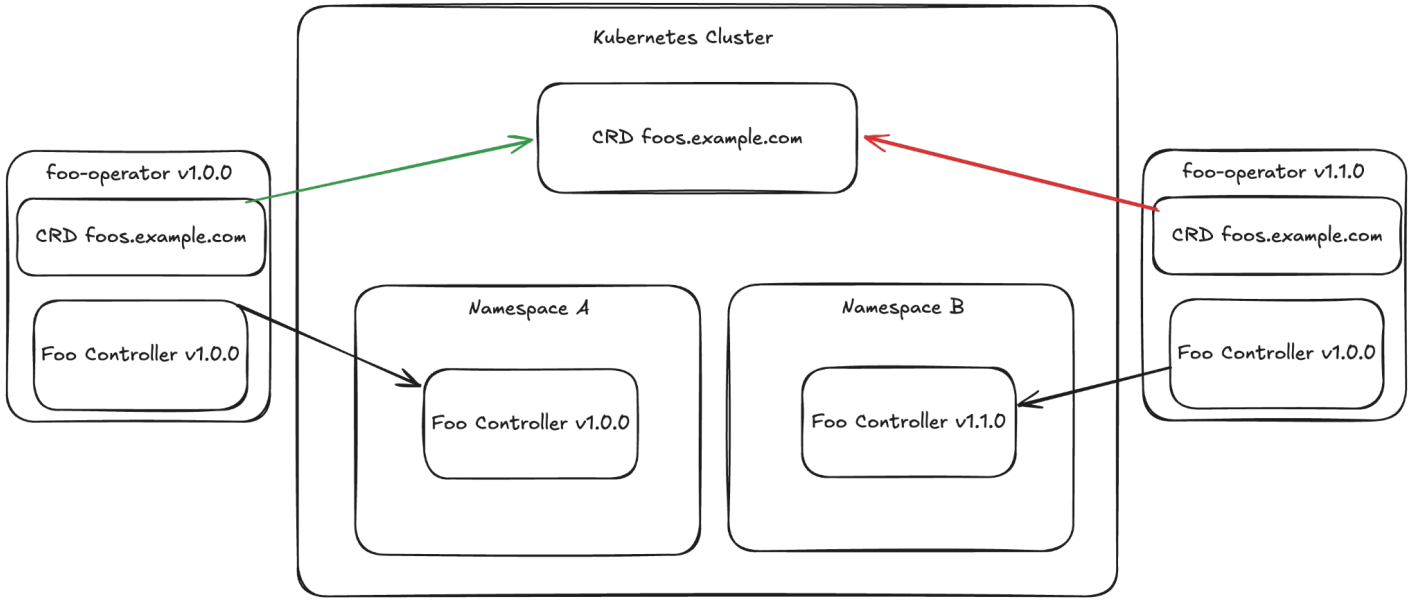
Don’t Fight Kubernetes

OLM v0 attempted to implement shared management of CRDs



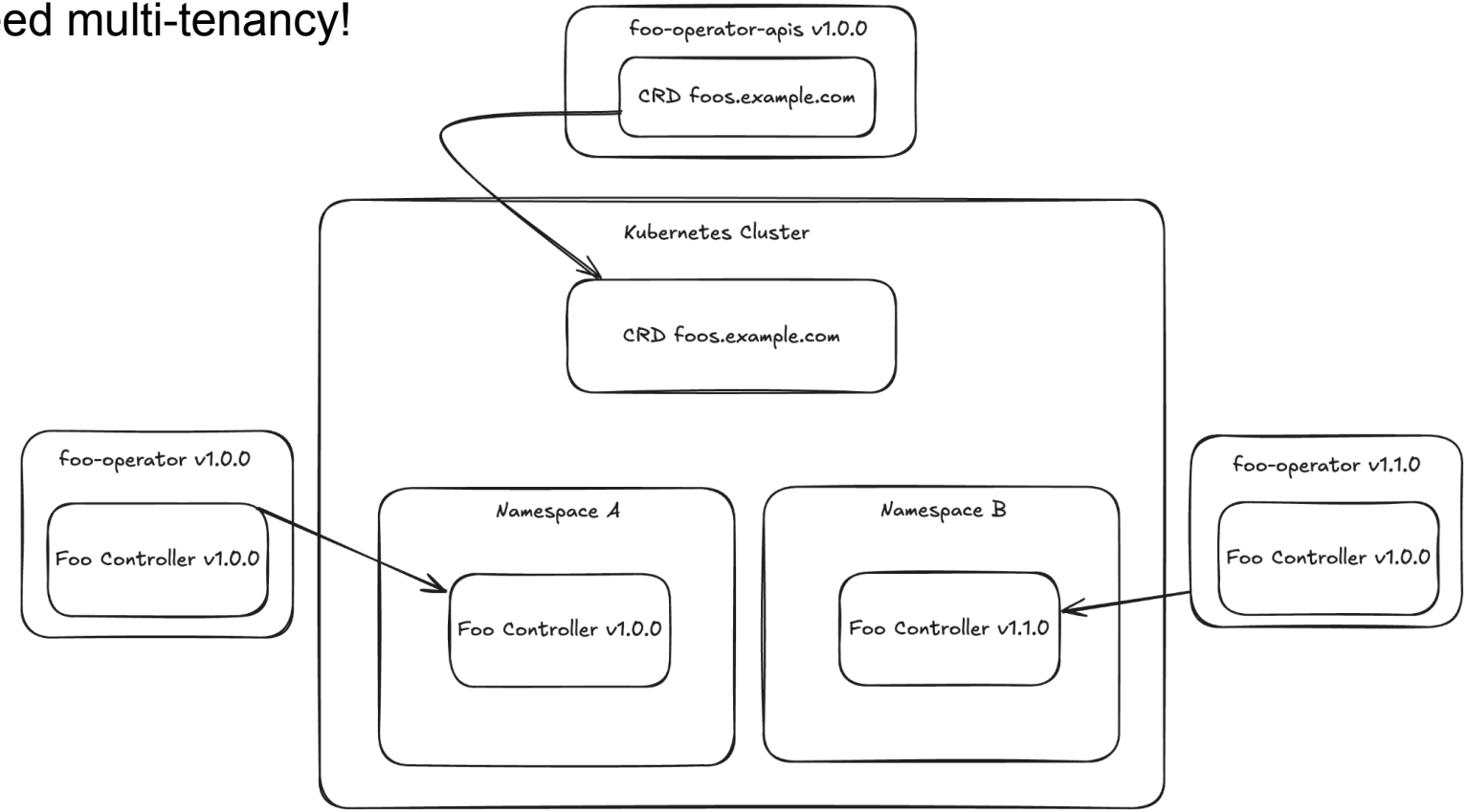
Don’t Fight Kubernetes (cont.)

- Extensions are meant to be cluster-wide
- Only one cluster-wide extension can own resource



Don’t Fight Kubernetes (cont.)

- But I need multi-tenancy!



Secure By Default

- OLM v0 has cluster-admin equivalent permissions
  - Prone to being used as a privilege escalation vector
- OLM v1 requires a ServiceAccount to be provided at install time
  - Cluster Admin explicitly controls permissions used to install extensions
  - Prevents automatic upgrades that would result in granting an extension escalated privileges

Simple, Predictable Install/Upgrade/Delete

- Declarative, GitOps friendly APIs
  - Currently, one for representing a catalog of content and one for intent to install content
- Pinning to specific versions, channels, version ranges
- Opinionated guardrails by default
  - Escape hatches to disable these guardrails
- Managed content deleted when the OLM object that represents it is deleted

APIs and Behaviors for Common Controller Patterns

- OLM v0 very opinionated as to the shape of bundles
- OLM v1 will be un-opinionated on bundle content
- Bundles that contain common controller-related resources will have special handling
  - Pre-flight checks to prevent CRD upgrades breaking clusters, clients, etc.
  - Special knowledge and handling of webhooks
  - and more….
- OLM v1 APIs may contain, optional, controller-specific fields

Constraint Checking

- No dependency resolution
- Constraints will be based on available cluster state
  - If met, installation/upgrade proceeds
  - If not met, constraints are reported and installation/upgrade will be blocked

Client Tooling Contributes to User Experience

- OLM v1 will provide official CLI tooling
- On-cluster APIs can be used to manage extensions in 100% of cases
- Official CLI will cover standard user flows, covering ~80% of use cases
- Third-party/Unofficial CLIs cover remaining ~20% of use cases


KubeCon
CloudNativeCon
North America 2024
Demo!

Current Limitations

- (For now) Only supports the existing registry+v1 bundle format
- (For now) No webhooks support
- (For now) Only supports the OLM v0 *AllNamespace* install mode
- Determining RBAC for the ServiceAccount is challenging

OLM v1 Roadmap

- Short-term
  - Direct installation of bundles (no catalog required!)
  - Tooling to help generate required RBAC / ServiceAccount
  - Expanding support for existing registry+v1 bundles
  - Tooling for content authors and cluster administrators
- Long-term
  - Support for Helm Charts
  - Reporting health conditions
  - Migration tooling to go from OLMv0 to OLMv1

Community

- Email: operator-framework-olm-dev@googlegroups.com
- Slack: #olm-dev
- Google Group: olm-gg
- Weekly in Person Working Group Meeting
- More information at https://operator-framework.github.io/operator-controller/

Java Operator SDK v5

Summary

- >50 issues / features / improvements
- Coming in few weeks
- Quarkus Operator SDK released shortly after
- Collection of issues that required API changes
- Migration should be fairly easy
- Few renaming and refactoring that impacts everyone
- Removed deprecated APIs
- Java version >= 17

Server Side Apply

- Already used for Dependent Resources
- Now also by default for:
  - Adding finalizer
  - Patching status (UpdateControl)
- Test migration!
- Feature flag “*useSSAToPatchPrimaryResource*” to use former approach

Server Side Apply - Patch done correctly

- Patch should use “fresh” resource

```
public static WebPage createWebPageForStatusUpdate(WebPage webPage,
String configMapName) {

  WebPage res = new WebPage();
  res.setMetadata(new ObjectMetaBuilder()
      .withName(webPage.getMetadata().getName())
      .withNamespace(webPage.getMetadata().getNamespace())
      .build());
  res.setStatus(createStatus(configMapName));

  return res;
}
```

Multi-cluster support for InformerEventSource

- Possible to pass other client instance

```
@Override
public List<EventSource<?, InformerRemoteClusterCustomResource>> prepareEventSources(
    EventSourceContext<InformerRemoteClusterCustomResource> context) {

  var es = new InformerEventSource<>(InformerEventSourceConfiguration
      .from(ConfigMap.class, InformerRemoteClusterCustomResource.class)
      .withSecondaryToPrimaryMapper(Mappers.fromDefaultAnnotations())
      .withKubernetesClient(remoteClient)
       .build(), context);

  return List.of(es);
}
```

No default cloning for secondary resources

- Performance implications
- Be careful changing, no changes!
- Turn back on with “*cloneSecondaryResourcesWhenGettingFromCache*“ flag

```
@Override
public UpdateControl<WebPage> reconcile(WebPage webPage, Context<WebPage> context) {

  var previousConfigMap = context.getSecondaryResource(ConfigMap.class).orElse(null);

  // code omitted

}
```

Dependent Resource and Workflows


```
@Workflow(dependents = {
    @Dependent(type = ConfigMapDependentResource.class),
    @Dependent(type = DeploymentDependentResource.class),
    @Dependent(type = ServiceDependentResource.class),
    @Dependent(type = IngressDependentResource.class,
        reconcilePrecondition = ExposedIngressCondition.class)
})
public class WebPageManagedDependentsReconciler
    implements Reconciler<WebPage>, Cleaner<WebPage> {
   // code omitted
}
```

Activation condition improvements

- What is activation condition?
- Multiple Activation condition for same type
- *CRDPresentActivationCondition*

```
@Workflow(dependents = {
    @Dependent(type = RouteDependentResource.class,
        activationCondition = CRDPresentActivationCondition.class),
})
// to trigger reconciliation with metadata change
@ControllerConfiguration(generationAwareEventProcessing = false)
public class CRDPresentActivationReconciler
    implements Reconciler<CRDPresentActivationCustomResource>,
    Cleaner<CRDPresentActivationCustomResource> {
  // code omitted
}
```

Explicit workflow invocation

- Allows various pre-processing / validations before the workflow executed

```
@Workflow(explicitInvocation = true,
    dependents = @Dependent(type = ConfigMapDependent.class))
@ControllerConfiguration
public class WorkflowExplicitInvocationReconciler
    implements Reconciler<WorkflowExplicitInvocationCustomResource> {
  @Override
  public UpdateControl<WorkflowExplicitInvocationCustomResource> reconcile(
      WorkflowExplicitInvocationCustomResource resource,
      Context<WorkflowExplicitInvocationCustomResource> context) {

    context.managedWorkflowAndDependentResourceContext().reconcileManagedWorkflow();

    return UpdateControl.noUpdate();
  }
```

Other improvements on Dependent Resources

- *ResourceDiscriminator* is removed!
  - Target resource selected based on desired state
- Support for ready only *BulkDependentResource*

Something experimental

- Can check in context if already received next trigger event
- Give us feedback :)

```
@Override
public UpdateControl<NextReconciliationImminentCustomResource> reconcile(
    NextReconciliationImminentCustomResource resource,
    Context<NextReconciliationImminentCustomResource> context) {

  if (context.isNextReconciliationImminent()) {
    return UpdateControl.noUpdate(); // or ?
  }

  // other logic
}
```


KubeCon
CloudNativeCon
North America 2024
Operator SDK

Operator SDK

- CNCF incubating project
- Call for maintainers:
  - Reach out to Joe Lanford (@joelanford) and/or Bryce Palmer (@Bryce Palmer) on the Kubernetes Slack channel #operator-sdk-dev
  - Attend the bi-weekly Operator-SDK community meeting
  - Attend the monthly Operator Framework Steering Committee meeting


KubeCon
CloudNativeCon
North America 2024
Q/A
Thank you!