Building Kubernetes Operators with Ansible

Chris Short, Principal Technical Marketing Manager
Michael Hrivnak, Principal Software Engineer
Melvin Hillsman, Senior Site Reliability Engineer
Matt Dorn, Principal Software Engineer
Tim Appnel, Senior Product Manager/Evangelist

Red Hat

Module 6

Operators with Ansible

Kubernetes Operators

Operators simplify management of complex applications
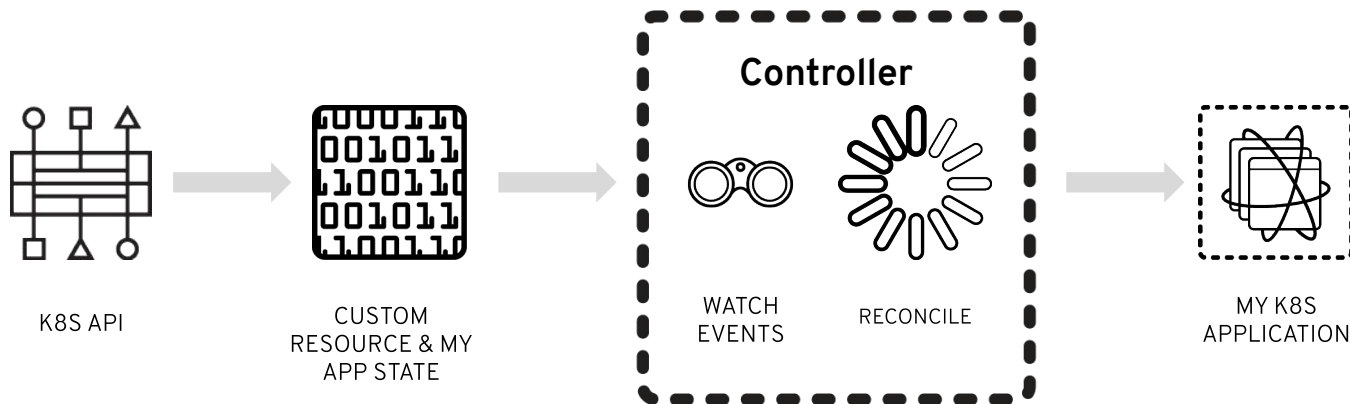on Kubernetes

OPERATOR SDK

- Encode human operational knowledge

- **Automatically patch, upgrade, recover, and tune container-based apps and services**

- Kubernetes-native

- Purpose-built for a specific application or service

- Enable "day 2" management

Red Hat

Extending Kubernetes with a
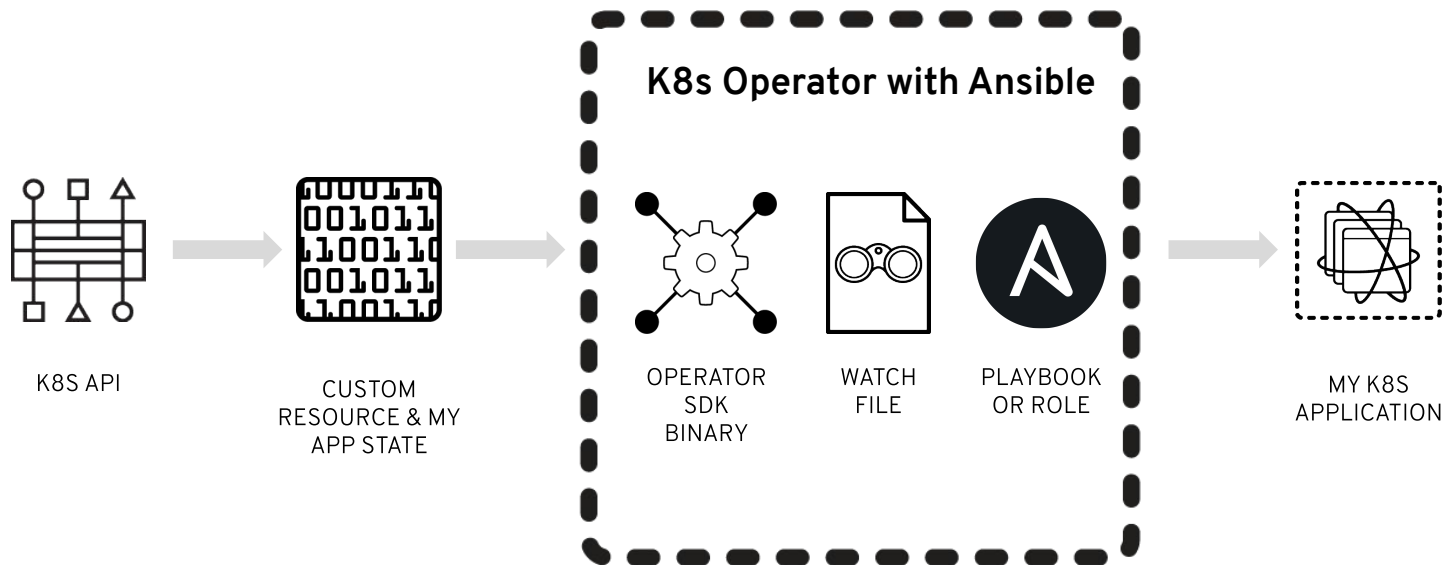# Custom Resource Definition (CRD)

Custom resources definition (CRD) is a powerful feature introduced into Kubernetes which enables users to add their own/custom objects to the Kubernetes cluster and use it like any other native Kubernetes objects.
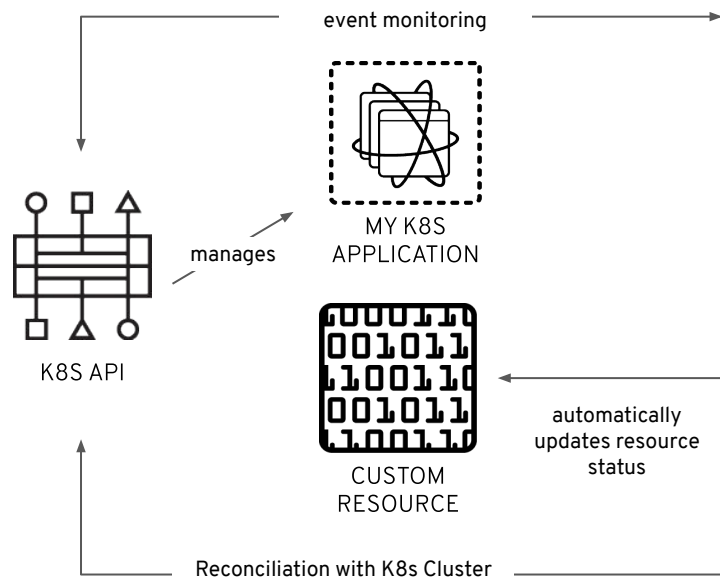
```
apiVersion: cache.example.com/v1alpha1
kind: Memcached
metadata:
  name: example-memcached
spec:
  size: 3
```

The Operator Pattern

Controller

K8S API → CUSTOM RESOURCE & MY APP STATE → WATCH EVENTS · RECONCILE → MY K8S APPLICATION

Design overview

K8s Operator with Ansible

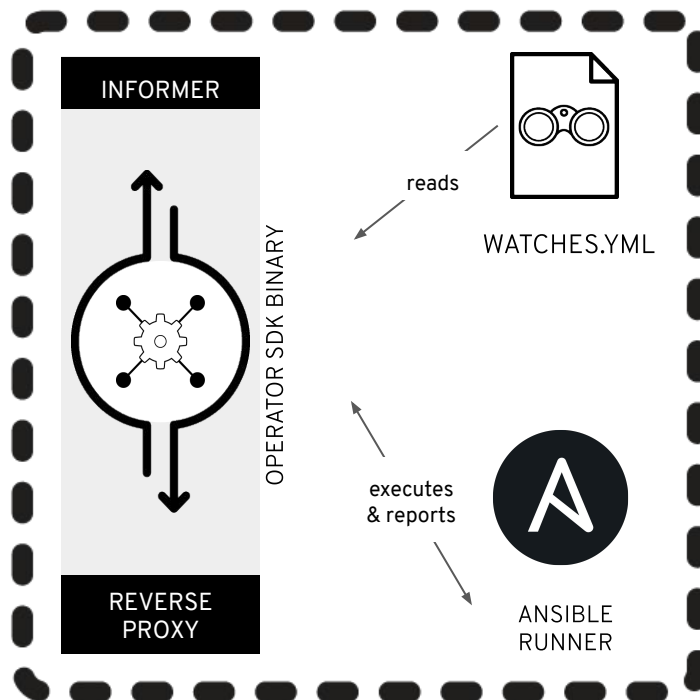K8S API → CUSTOM RESOURCE & MY APP STATE → OPERATOR SDK BINARY → WATCH FILE → PLAYBOOK OR ROLE → MY K8S APPLICATION

K8s Operator with Ansible

Mapping Kubernetes events to Ansible automation

The "watches" file (watches.yaml) maps a Kubernetes object to your Ansible automation

Associates the Kubernetes Group, Version, Kind (GVK) to an Ansible Role or Playbook

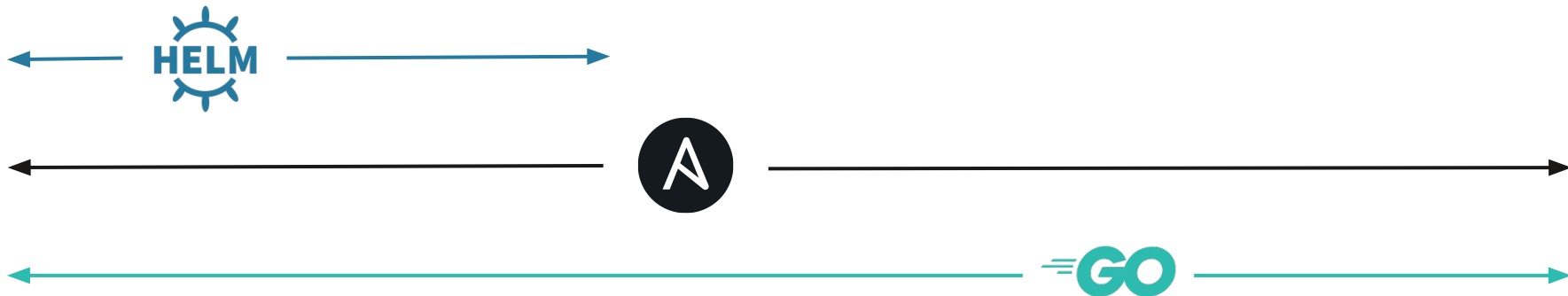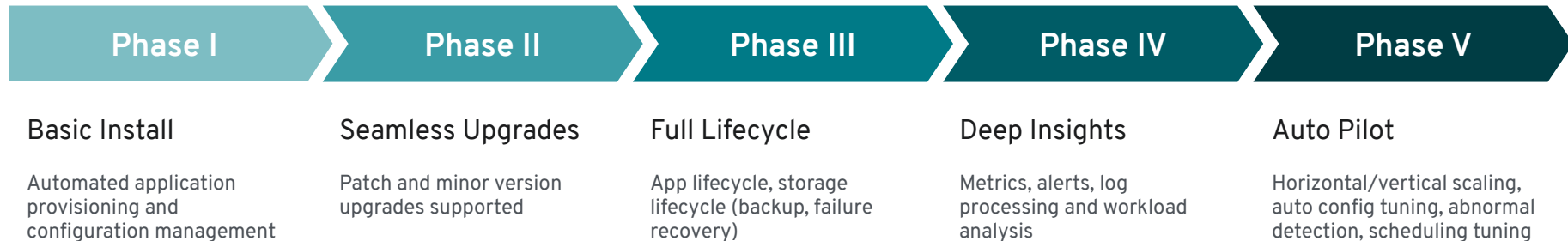The Operator SDK binary watches the cluster for matching events defined in the watches.yml

Executes the associated Ansible content when an event occurs

```
# watches.yaml
---
version: v1alpha1
group: cache.example.com
kind: Memcached
role: /path/to/role
```

OPERATOR
SDK

- Helps you create an operator

- Write using Go, Ansible, or Helm

- github.com/operator-framework/operator-sdk/

Operator capability level

Phase I	Phase II	Phase III	Phase IV	Phase V

Basic Install

Automated application provisioning and configuration management

Seamless Upgrades

Patch and minor version upgrades supported

Full Lifecycle

App lifecycle, storage lifecycle (backup, failure recovery)

Deep Insights

Metrics, alerts, log processing and workload analysis

Auto Pilot

Horizontal/vertical scaling, auto config tuning, abnormal detection, scheduling tuning

# Custom resource spec to Ansible extra variables

```
apiVersion:
app.example.com/v1alpha1
kind: PodSet
metadata:
  name: podset-sample
spec:
  eplicas: 3
  ….
status:
  <key>: <value>

  ….
```

spec: values will be translated to Ansible extra_vars.

status: will be a generic status recorded by the Ansible Operator SDK operator. This will use ansible-runner output to generate meaningful information for the user.

Anatomy of Ansible-enabled Operator image
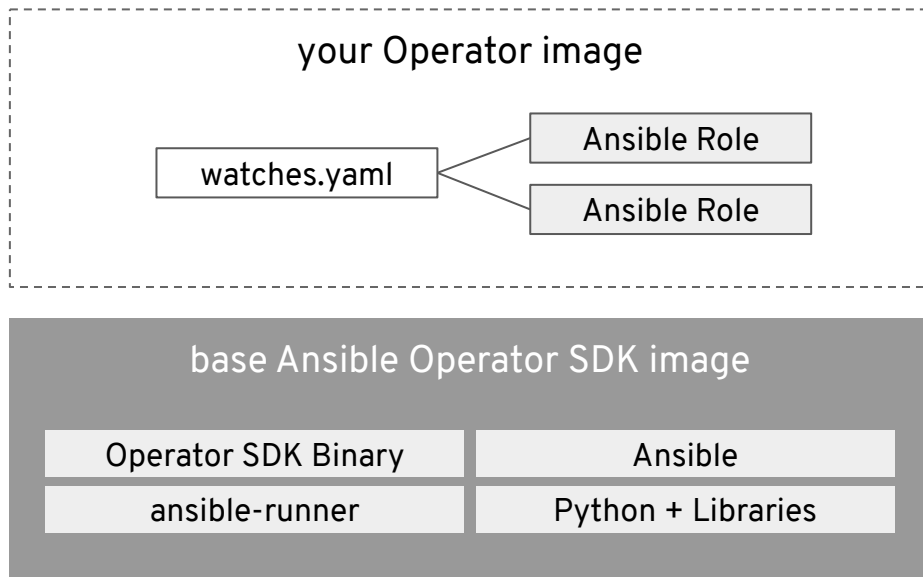
Initialize your Operator with Ansible

The new command in the Operator SDK using type=ansible will create an Ansible Role skelton, watches.yaml mapping file, CRD, deploy manifest for the Operator and basic tests using Molecule.

```
$ operator-sdk new memcached-operator
--api-version=cache.example.com/v1alpha1
--kind=Memcached --type=ansible
```

Red Hat

Hybrid Use Case

- The Ansible Operator is a first class citizen of operator-sdk.
- You can extend/change your operator with golang code to make a hybrid.
- Allows you to change, compose, or reuse the Ansible Operator.
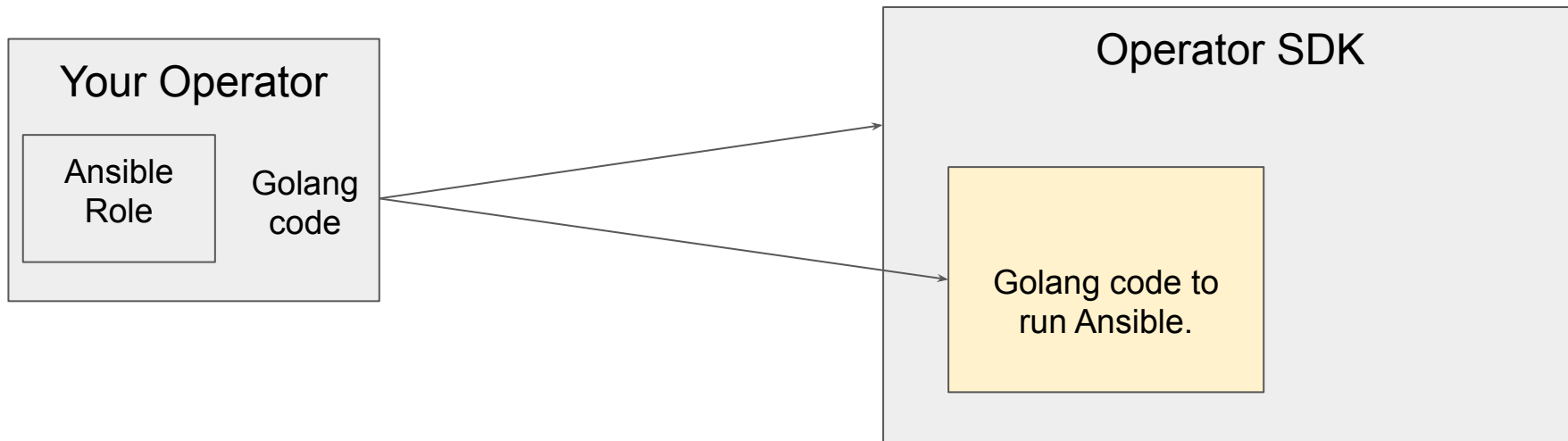
Exercise

Advanced Patterns

Finalizers

Finalizers are a way to run code before an object gets deleted.

Here the `/opt/ansible/roles/memfin` will run while an object is being deleted.

```
# watches.yaml
---
- version: v1alpha1
  group: cache.example.com
  kind: Memcached
  role: /opt/ansible/roles/memcached
  finalizer:
    name: finalizer.memcached.cache.example.com
    role: /opt/ansible/roles/memfin
```

Upgrade

Operators can be used to orchestrate upgrades of applications and services.

Use Ansible conditionals like **when** using any available variables in your tasks.

```
- when: version < 1.2
  block:
    - name: "run upgrade tool"
      shell: runupgrade.sh --version 1.2
    - name: "do more upgrade stuff"
      shell: ...
```

Backup/Restore

You can use Operators to define and perform backups, restores and other maintenance of applications and services.

Here the `EtcdBackup` CR defines how and when a backup should be created. The `backup_playbook.yaml` playbook defines workflow logic for backup.

```yaml
# watches.yaml
---
- version: v1beta2
  group: etcd.database.coreos.com
  kind: EtcdCluster
  playbook: /opt/ansible/playbook.yaml

- version: v1beta2
  group: etcd.database.coreos.com
  kind: EtcdBackup
  playbook: /opt/ansible/backup_playbook.yaml

- version: v1beta2
  group: etcd.database.coreos.com
  kind: EtcdRestore
  reconcilePeriod: 10h
  playbook: /opt/ansible/restore_playbook.yaml
```

Explore Operators

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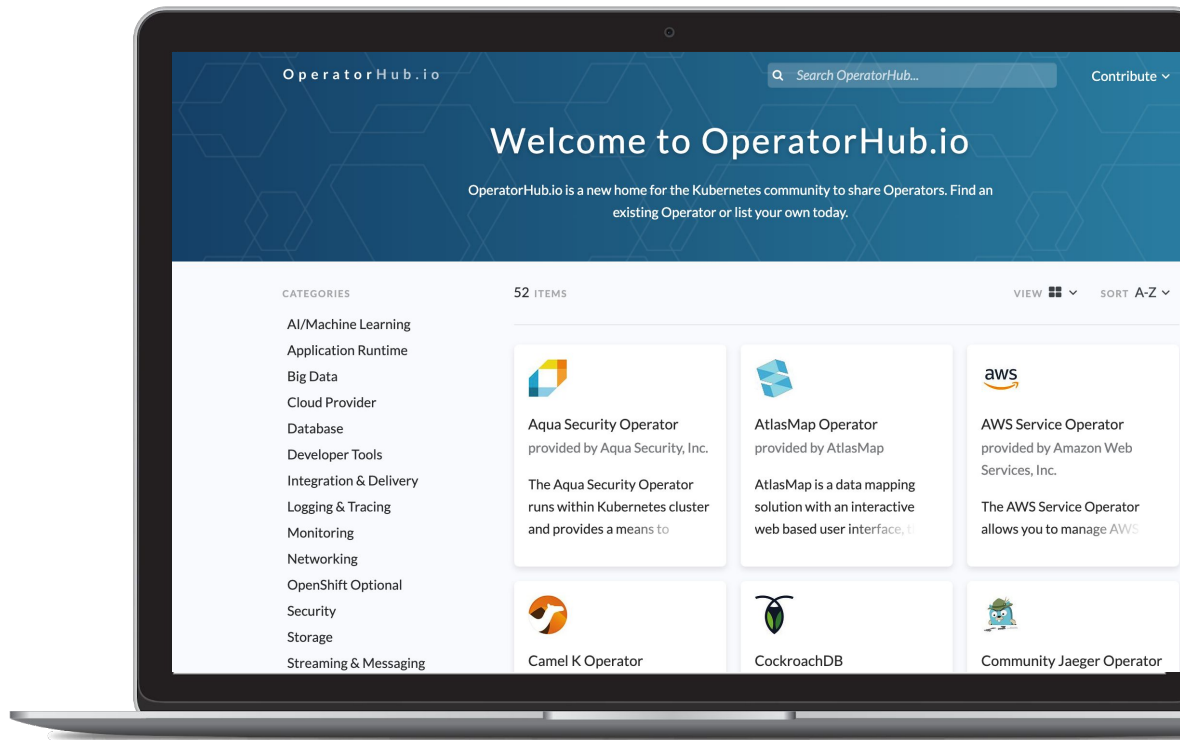
Next Up:

Module 7
Mcrouter

Red Hat