Intro to Kubernetes Operators

Kubernetes

Kubernetes is designed for automation. Out of the box, you get lots of built-in automation from the core of Kubernetes. You can use Kubernetes to automate deploying and running workloads, and you can automate how Kubernetes does that.

Operators

Kubernetes' operator pattern concept lets you extend the cluster's behaviour without modifying the code of Kubernetes itself by linking **controllers** to one or more **custom resources**.

Some of the things that you can use an operator to automate:

- deploying an application on demand
- taking and restoring backups of that application's state
- handling upgrades of the application code alongside related changes such as database schemas or extra configuration settings

Controller

A controller tracks at least one Kubernetes **resource type**. These objects have a **spec** field that represents the **desired state**. The controller(s) for that resource are responsible for making the current state come closer to that desired state.

Resources

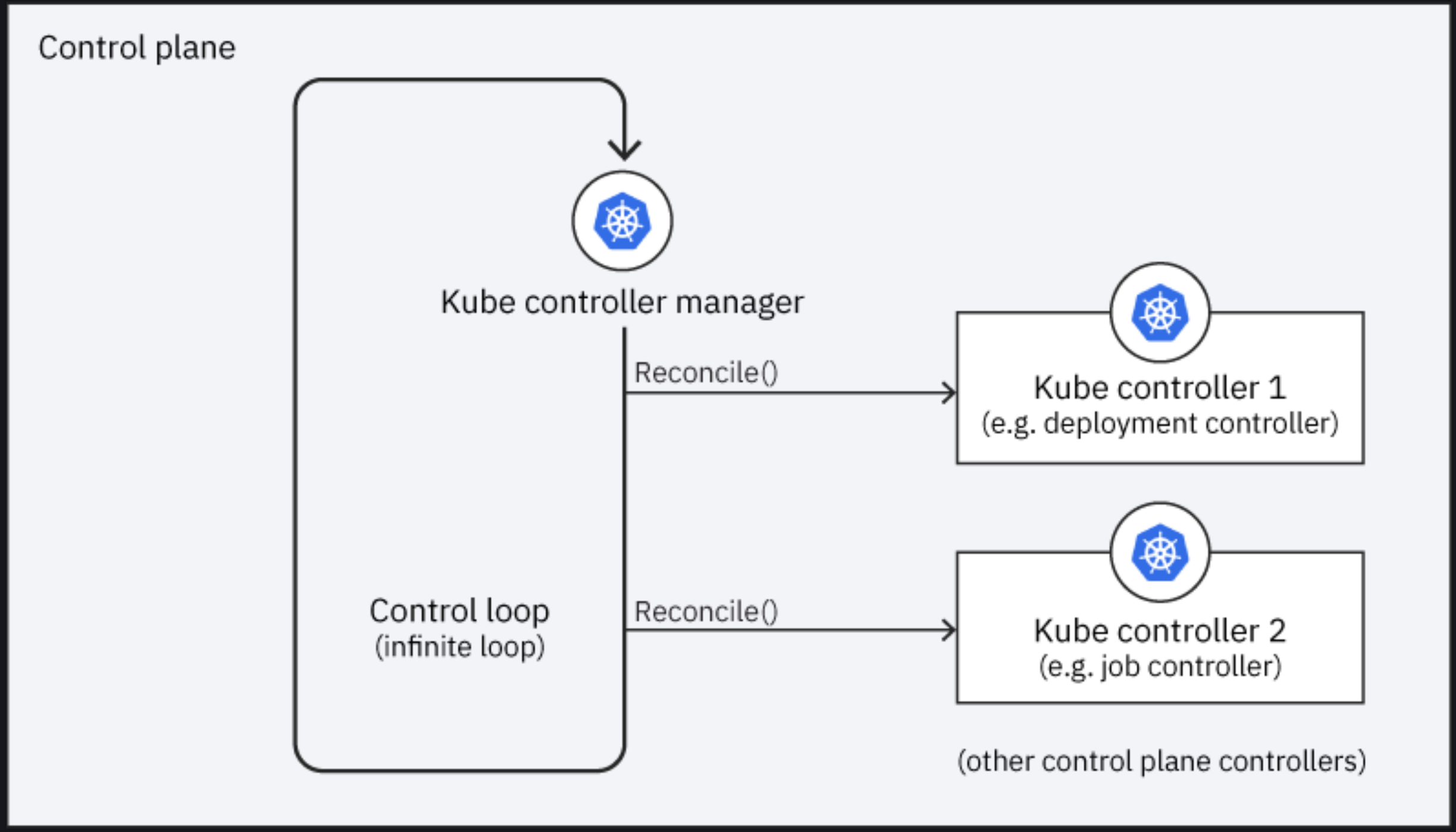
A resource is an endpoint in the Kubernetes API that stores a collection of API objects of a certain kind; for example, the built-in pods resource contains a collection of Pod objects.

Custom resources

A custom resource is an extension of the Kubernetes API that is not necessarily available in a default Kubernetes installation. It represents a customization of a particular Kubernetes installation.

Reconciliation Loop

Kubernetes is based on the concept of a declarative specification of the desired state of the cluster and the use of reconciliation loops to drive the actual state toward the desired state.


Control plane
Kube controller manager
Reconcile()
Kube controller 1
(e.g. deployment controller)
Control loop
(infinite loop)
Reconcile()
Kube controller 2
(e.g. job controller)
(other control plane controllers)

Operator SDK

https://sdk.operatorframework.io/

Show me the code

Objective

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: application-sample
---
apiVersion: minetto.dev/v1alpha1
kind: Application
metadata:
  name: application-sample
  namespace: application-sample
spec:
  image: nginx:latest
  replicas: 2
  port: 80
```

Scaffolding

```
operator-sdk init --domain minetto.dev --repo github.com/eminetto/k8s-operator-talk
operator-sdk create api --version v1alpha1 --kind Application --resource --controller
```

Add information the the Application CRD

```go
// api/v1alpha1/application_types.go
type ApplicationSpec struct {
    Image    string `json:"image,omitempty"`
    Replicas int32  `json:"replicas,omitempty"`
    Port     int32  `json:"port,omitempty"`
}
```

Generate the manifests

```
make manifests
```

Controller code

controllers/application_controller.go

Deploy

```
make docker-build docker-push IMG=registry.hub.docker.com/eminetto/k8s-operator-talk:latest
make deploy IMG=registry.hub.docker.com/eminetto/k8s-operator-talk:latest
```

Demo

OperatorHub

https://operatorhub.io/

https://github.com/eminetto/k8s-operator-talk

References

- Operator pattern
  - Controllers
- Custom Resources
- Kubernetes Operators 101, Part 2: How operators work

Contact