How to Generate a Scaffold Operator

Learn how to generate a scaffold operator project.

We'll cover the following

- Overview
- Why use kubebuilder to build operators?
- Create a scaffold operator project

Overview

We've heard about controllers. Kubernetes ships large quantities of controllers in the `kube-controller-manager`. We've also heard about operators. So, what's the difference between them? Are they the same thing, but with different names?

In the Kubernetes glossary, a **controller** refers to a control loop that keeps watching the changing status of certain objects and drives them to the desired state. For example, the `Deployment` controller keeps watching all the `Deployment` objects and turns them into `ReplicaSet` objects. Typically, controllers are part of the Kubernetes controller manager in the control plane.

On the other hand, operators always come with CRDs. They supervise the custom resources and some core resources. Essentially, operators are controllers. They build upon the basic Kubernetes controller concepts, but bring in application-specific knowledge, operational domain knowledge, and so on. We can say operators extend what controllers can do.

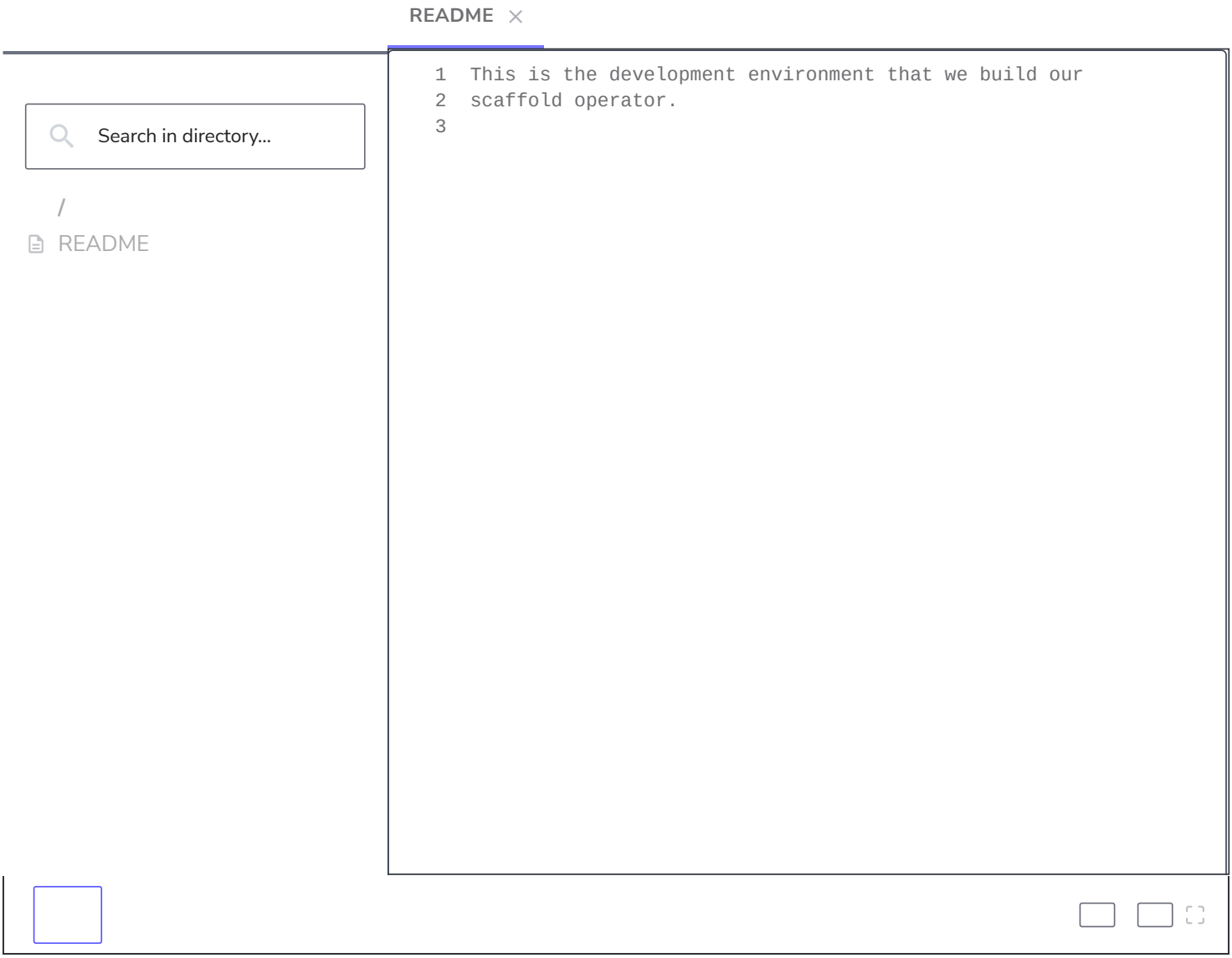
Now, in this lesson, let's learn how to scaffold an operator.

Why use `kubebuilder` to build operators?

First of all, it's not an easy task to write an operator from scratch. We need to have a deeper understanding of Kubernetes fundamentals, API implementations, how the workflows are triggered, declarative APIs, garbage collection, and even some low-level details of the Kubernetes libraries implementations. There is a steep learning curve for Kubernetes novices to write an operator that can work.

Moreover, we're not just writing a single operator, but a number of operators. They have the same reconcile loop, but for different custom resources. Thus, it's not a bad idea to abstract away this fact and build the same framework for all operators, but with different business logics. This will bring us convenience when looking for certain tasks, bugs, etc.

Normally, we use the `kubebuilder` to help us build operators.



Our development environment

Start by hitting the “Run” button in the terminal above to initialize our development environment.

Create a scaffold operator project

First, we create a new folder and initialize the project. We use `educative.io` as the domain name. Run the following commands in the terminal above:

```
mkdir -p ./projects/pwk
cd ./projects/pwk
kubebuilder init --domain educative.io --repo educative.io/pwk
```

Initialize the operator project

The output will be as follows:

```
Writing kustomize manifests for you to edit...
Writing scaffold for you to edit...
Get controller runtime:
```

```
$ go get sigs.k8s.io/controller-runtime@v0.12.2
go: downloading sigs.k8s.io/controller-runtime v0.12.2
go: downloading k8s.io/apimachinery v0.24.2
go: downloading github.com/gogo/protobuf v1.3.2
go: downloading github.com/go-logr/logr v1.2.0
go: downloading k8s.io/client-go v0.24.2
go: downloading k8s.io/klog/v2 v2.60.1
go: downloading k8s.io/utils v0.0.0-20220210201930-3a6ce19ff2f9
go: downloading github.com/google/gofuzz v1.1.0
go: downloading github.com/prometheus/client_golang v1.12.1
go: downloading sigs.k8s.io/structured-merge-diff/v4 v4.2.1
go: downloading github.com/evanphx/json-patch v4.12.0+incompatible
go: downloading golang.org/x/time v0.0.0-20220210224613-90d013bbcef8
go: downloading gomodules.xyz/jsonpatch/v2 v2.2.0
go: downloading k8s.io/api v0.24.2
go: downloading k8s.io/apiextensions-apiserver v0.24.2
go: downloading github.com/imdario/mergo v0.3.12
go: downloading github.com/spf13/pflag v1.0.5
go: downloading golang.org/x/term v0.0.0-20210927222741-03fcf44c2211
go: downloading golang.org/x/net v0.0.0-20220127200216-cd36cc0744dd
go: downloading k8s.io/component-base v0.24.2
go: downloading gopkg.in/inf.v0 v0.9.1
go: downloading github.com/golang/groupcache v0.0.0-20210331224755-41bb18bfe9da
go: downloading sigs.k8s.io/json v0.0.0-20211208200746-9f7c6b3444d2
go: downloading github.com/prometheus/client_model v0.2.0
go: downloading github.com/prometheus/common v0.32.1
go: downloading github.com/json-iterator/go v1.1.12
go: downloading gopkg.in/yaml.v2 v2.4.0
```

The output of kubebuilder init

Next, let's scaffold a Kubernetes API with a corresponding operator. We create a non-namespaced API by specifying the flag `--namespaced=false`. This API has the version `v1beta1` in the group `apps.educative.io`. By specifying `--controller=true`, the `kubebuilder` will automatically generate a scaffold controller for us. Let's run the following command in the terminal above:

```
kubebuilder create api --group apps --version v1beta1 --kind Foo --controller=true --make --resource --r
```

Create the scaffold API and controller

The output will be as follows:

```
Writing kustomize manifests for you to edit...
Writing scaffold for you to edit...
api/v1beta1/foo_types.go
controllers/foo_controller.go
Update dependencies:
$ go mod tidy
go: downloading github.com/onsi/ginkgo/v2 v2.0.0
Running make:
$ make generate
mkdir -p /home/ubuntu/pwk-demo-operator/bin
test -s /home/ubuntu/pwk-demo-operator/bin/controller-gen || GOBIN=/home/ubuntu/pwk-demo-operator/bin go
go: downloading sigs.k8s.io/controller-tools v0.9.2
go: downloading github.com/spf13/cobra v1.4.0
go: downloading github.com/gobuffalo/flect v0.2.5
go: downloading k8s.io/apiextensions-apiserver v0.24.0
go: downloading k8s.io/apimachinery v0.24.0
go: downloading golang.org/x/tools v0.1.10-0.20220218145154-897bd77cd717
go: downloading github.com/fatih/color v1.12.0
go: downloading k8s.io/api v0.24.0
go: downloading github.com/mattn/go-colorable v0.1.8
```

```
20  go: downloading github.com/mattn/go-colorable v0.1.8
21  go: downloading github.com/mattn/go-isatty v0.0.12
22  go: downloading golang.org/x/mod v0.6.0-dev.0.20220106191415-9b9b3d81d5e3
23  /home/ubuntu/pwk-demo-operator/bin/controller-gen object:headerFile="hack/boilerplate.go.txt" paths="./
24  Next: implement your new API and generate the manifests (e.g. CRDs,CRs) with:
25  $ make manifests
```

The output of kubebuilder create

If we list the contents of our scaffold project director by running the command `tree` in the terminal above, we can see all the generated files, as shown below:

```
 1  .
 2  |-- Dockerfile
 3  |-- Makefile
 4  |-- PROJECT
 5  |-- README.md
 6  |-- api
 7  |   `-- v1beta1
 8  |       |-- foo_types.go
 9  |       |-- groupversion_info.go
10  |       `-- zz_generated.deepcopy.go
11  |-- bin
12  |   `-- controller-gen
13  |-- config
14  |   |-- crd
15  |   |   |-- kustomization.yaml
16  |   |   |-- kustomizeconfig.yaml
17  |   |   `-- patches
18  |   |       |-- cainjection_in_foos.yaml
19  |   |       `-- webhook_in_foos.yaml
20  |   |-- default
21  |   |   |-- kustomization.yaml
22  |   |   |-- manager_auth_proxy_patch.yaml
23  |   |   `-- manager_config_patch.yaml
24  |   |-- manager
25  |   |   |-- controller_manager_config.yaml
26  |   |   |-- kustomization.yaml
27  |   |   `-- manager.yaml
28  |   |-- prometheus
29  |   |   |-- kustomization.yaml
30  |   |   `-- monitor.yaml
31  |   |-- rbac
```

The contents of our scaffold project directory

Now, in the current project folder, we only need to modify two files.

- All the API definitions are placed at the file `api/v1beta1/foo_types.go`. After modifying this file, we can run `make generate` to help generate codes containing the `DeepCopy`, `DeepCopyInto`, and `DeepCopyObject` method implementations. Moreover, all the manifest files including CustomResourceDefinition objects can also be auto-generated by running `make manifests`. Remember to run these two commands after modifying the file `api/v1beta1/foo_types.go`.

- The core operator logic is placed at the file `controllers/foo_controller.go`. In this file, we only need to add our logic in the function `Reconcile(ctx context.Context, req ctrl.Request)`, where we handle the event and process it.

In this project, we have `Makefile` and `Dockerfile`, which can be used to make container images, make binaries, deploy to cluster, and so on.

With `kubebuilder`, we can build a production-ready operator with ease.

Back

Quiz on kubectl Plugins

Next

Best Practices for Building Kubernetes Operators