Golang Operators Ext

HICP

Exported on 09/13/2024

Table of Contents

1 Install Go on WSL 2

1.1 Remove old Go versions

```
sudo rm -rf /usr/local/go* && sudo rm -rf /usr/local/go && sudo rm -rf /usr/bin/go
```

1.2 Install go 1.21

At the moment, go 1.21 is supported for the Operator SDK 1.31

```
VERSION=1.21.13
OS=linux
ARCH=amd64

cd $HOME
wget https://go.dev/dl/go$VERSION.$OS-$ARCH.tar.gz
tar -xvf go$VERSION.$OS-$ARCH.tar.gz
mv go go-$VERSION
sudo mv go-$VERSION /usr/local
```

Add Path to .bashrc

```
export GOROOT=/usr/local/go-1.21.13
export GOPATH=$HOME/projects/go
export GOBIN=$GOPATH/bin
export PATH=$PATH:$GOROOT/bin
export PATH=$PATH:$HOME/projects/go/bin
```

Check installation with

```
go version
```

2 Install go operator sdk

https://sdk.operatorframework.io/docs/installation/

The sdk is just a binary

The currently used version (1.36.1) requires go 1.21 and the exact installation steps for this version are here:

`https://v1-36-x.sdk.operatorframework.io/docs/installation/`[1]

1 https://v1-27-x.sdk.operatorframework.io/docs/installation/

3 Prerequisites

3.1 create softlink (ln -s) as root for kubectl (to oc) in your bin folder

```
ln -s oc kubectl
```

3.2 create softlink as root for docker (to podman) in your folder

```
ln -s podman docker
```

3.3 activate module support

*This might not be necessary anymore

```
export GO111MODULE=on
```

4 Read best practices

When you got the time, read the best practices:

https://sdk.operatorframework.io/docs/best-practices/

https://cloud.redhat.com/blog/7-best-practices-for-writing-kubernetes-operators-an-sre-perspective

5 Initialize new Project

5.1 Create repo in Bitbucket

5.2 Create directory and change into it

5.3 Create structure via scaffolding

```
operator-sdk init --domain <domain> --repo github.com/<user>/<repo>
operator-sdk init --domain example.com --repo github.com/shotty01/example-operator
```

6 Create new endpoint

If CRD is namespaced, then set namespaced=true

```
operator-sdk create api --group <apigroup> --version <apiversion> --kind <CRD-Name>
--resource --controller --namespaced=false
operator-sdk create api --group test --version v1beta1 --kind ExampleConfig --
resource --controller --namespaced=false
```

6.1 Edit your CRD type file under api/v1beta1

Define the fields you want in your crd

Complex fields have to be structs

If you want to rely on nil values you have to use *<Type>

There are kubebuilder annotations which do something when generating

Reference: https://book.kubebuilder.io/reference/markers.html

6.2 generate CRDs

With the following commands, the crd yaml file is generated in config/crs/bases

```
make generate
make manifests
```

6.3 check CRD in test cluster

Log in to the testcluster and apply the CRD with:

```
make install
```

Look for and fix errors and try to create an instance

you can fill the values for the sample in config/samples/ and apply it to the cluster:

```
oc apply -f config/samples/<samplefile>
```

6.4 Implement Operator logic in controller

Implement your logic in the Reconcile method - bear in mind it should be idempotent.

The resource the Reconcile is watching is defined in the SetupWithManager method in the For bracket

the complete api group + version is used, for this example (and the following ones) the api group is test and the api version is v1beta1

```
return ctrl.NewControllerManagedBy(mgr).
    For(&testv1beta1.ExampleConfig{}).
    Complete(r)
```

Only one resource is given to the method at one time. The unique identifier is given with the request (re.NamespacedName)

Example for NodeMaintenace:

```
maintenance := &testv1beta1.ExampleConfig{}
err := r.Get(ctx, req.NamespacedName, maintenance)
```

The following are a few possible return options for a Reconciler:

```
// With the error:
return ctrl.Result{}, err
// Without an error:
return ctrl.Result{Requeue: true}, nil
// Therefore, to stop the Reconcile, use:
return ctrl.Result{}, nil
// Reconcile again after X time:
 return ctrl.Result{RequeueAfter: nextRun.Sub(r.Now())}, nil
```

For more details, check the Reconcile godoc[2].

ATTENTION: if you update the CRD you are watching of course it will be requeued since it changed and you are watching it

Therefore pay attention to NOT update if there is NO CHANGE.

If you want to create events, you can read up here:

https://book-v1.book.kubebuilder.io/beyond_basics/creating_events.html

Please keep in mind that events get deleted after one hour, so you eventually need to refresh them.

2 https://pkg.go.dev/sigs.k8s.io/controller-runtime/pkg/reconcile

6.4.1 Using Third-party Custom Resources

To use third party custom resources you first need to import them into the desired go file

```
import (
    ...
    hivev1 "github.com/openshift/hive/apis/hive/v1"
    ...
)
```

Also, it is not enough that a resource is defined via a CRD in the cluster. You also need to add the schema of the third-party CRD in the main.go file

```
func init() {
    utilruntime.Must(clientgoscheme.AddToScheme(scheme))

    ...
    utilruntime.Must(hivev1.AddToScheme(scheme))
    ...

    utilruntime.Must(icpv1beta1.AddToScheme(scheme))
    //+kubebuilder:scaffold:scheme
}
```

Also, it is recommended to use https://pkg.go.dev/ to navigate through the structure of a CRD. This is much faster than clicking through the files on Github 🙂

6.4.2 Finalizer Handling

A finalizer can be added by using the provided controllerutil methods AddFinalizer() and RemoveFinalizer(). AddFinalizer will not add the finalizer again if it is already set on the object.

```
// Finalizer handling - examine DeletionTimestamp to determine if object is under
deletion
if !instance.ObjectMeta.DeletionTimestamp.IsZero() {
    // Object is being deleted
    <your clean-up code>
    controllerutil.RemoveFinalizer(instance, <finalizer string>)
} else {
    // If object is not being deleted add finalizer
    controllerutil.AddFinalizer(instance, <finalizer string>)
}
```

6.5 Run the Operator against the cluster

We disable webhooks or we would have to generate certificates and stuff

This does not install the operator on the cluster, it runs locally

```
make run ENABLE_WEBHOOKS=false
```

This obviously runs the operator with your logged in user, so it's for testing functionality, NOT access rights

6.6 Try to install it in the testcluster

This does NOT create a subscription. Use for tests only

Also, pay attention to install in a namespace that nothing else is in

```
export IMG=<internalregistry>/<namespace>/<operatorname>:<version>
make docker-build IMG=$IMG
make docker-push IMG=$IMG
make deploy IMG=$IMG
```

This will create two Imagestream in the namespace, one for the operator one for the bundle

6.7 Uninstall

Delete your CRDs also if you don't need them anymore

Uninstall operator:

```
make undeploy
```

Attention !!! This also deletes the namespace the operator was in!!!

7 Testing

The Operator SDK project recommends using controller-runtime's envtest[3] to write tests for your Operators projects (instead of its own test framework... wow).

In your generated file `suite_test.go` you will use envtest to create a local Kubernetes API server, instantiate and run your controllers, and then write additional `*_test.go` files to test it using Ginkgo[4].

~~Please edit the Makefile to use ENVTEST_K8S_VERSION=1.19 instead of the newest one since it has a bug and etcd hangs while starting.~~

`make test` automatically installs the necessary kube-apiserver and etcd binaries.

7.1 Testing the CRDs

The sdk only generates a suite_test.go for the controllers, and since the CRDs are generated it has no real value testing those

7.2 Registering the Controllers to the local Testcluster

If you have more than one controller, register all Reconcilers!

This is very similar to how the Reconcilers are started in the main.go File.

Just by registering your controllers, you get code coverage

```
    err = (&ExampleConfigReconciler{
        Client: k8sManager.GetClient(),
        Scheme: k8sManager.GetScheme(),
    }).SetupWithManager(k8sManager)
    Expect(err).ToNot(HaveOccurred())
```

While the code initializes also a 'live' k8s client, we use the managers cache for the Reconcilers, so it behaves like in production and it can use the cache features

7.3 Add Third-party CRDs to envtest

If you are using third-party CRDs in your operator they also need to be registered for the test suite.

This can be done in a similar way as in the main.go file described above. This needs to be added in the suite_test.go file in your "controllers" folder.

3 https://pkg.go.dev/sigs.k8s.io/controller-runtime/pkg/envtest
4 http://onsi.github.io/ginkgo

```
...
    var err error
    // cfg is defined in this file globally.
    cfg, err = testEnv.Start()
    Expect(err).NotTo(HaveOccurred())
    Expect(cfg).NotTo(BeNil())

    err = acmclusterv1.AddToScheme(scheme.Scheme) // third-party CRD
    Expect(err).NotTo(HaveOccurred())

    err = hivev1.AddToScheme(scheme.Scheme) // third-party CRD
    Expect(err).NotTo(HaveOccurred())

    err = acmagentv1.AddToScheme(scheme.Scheme) // third-party CRD
    Expect(err).NotTo(HaveOccurred())

    err = icpv1beta1.AddToScheme(scheme.Scheme)
    Expect(err).NotTo(HaveOccurred())

    //+kubebuilder:scaffold:scheme
...
```

The second step is to point env test to the third-party CRDs so envtest knows what CRDs need to be installed in the local test kubernetes environment. This can be done by adding additional file paths in the suite_test.go file.

```
...
    By("bootstrapping test environment")
    testEnv = &envtest.Environment{
        CRDDirectoryPaths: []string{
            filepath.Join("..", "config", "crd", "bases"), // by default available
            filepath.Join(build.Default.GOPATH, "pkg", "mod", "github.com",
"openshift", "hive@v1.1.16", "config", "crds"), // third-party CRD
            filepath.Join(build.Default.GOPATH, "pkg", "mod", "github.com",
"stolostron", "klusterlet-addon-controller@v0.0.0-20220714103120-43778f205c45",
"deploy"), // third-party CRD
            filepath.Join(build.Default.GOPATH, "pkg", "mod", "github.com", "open-
cluster-management", "api@v0.0.0-20210527013639-a6845f2ebcb1", "cluster", "v1"), //
third-party CRD
        },
        ErrorIfCRDPathMissing: true,
    }
...
```

7.4 Write Test for controller

For each controller, a file <kind>_controller_test.go should be added which holds tests for that controller.

https://book.kubebuilder.io/cronjob-tutorial/writing-tests.html

8 Deploying with OLM

8.1 Set the environment variables to deploy to image registry:

Adapt the variables to fit our needs. It might come handy to put these in a file (e.g. .params) and source it.

```
export USERNAME=vihkeimchri
export VERSION=0.0.1
export OPERATOR=example-operator
export ARTIFACTORY_IMG=<artifactory-registry>/<folder>/$OPERATOR:v$VERSION
export IMG=<image-registry>/<namespace>/$OPERATOR:v$VERSION
export BUNDLE_IMG=<image-registry>/<namespace>/$OPERATOR-bundle:v$VERSION
export CHANNELS=alpha,stable
export DEFAULT_CHANNEL=stable
```

The ARTIFACTORY_IMG is the url the operator will pull from

CHANNELS and DEFAULT_CHANNEL define the subscription channel the version will be available for

If you only need to push the operator to the alpha channel, remove the stable channel from the channel list.

When you are ready to release it to the stable channel, use the following command and build and push the bundle again:

```
make bundle CHANNELS=alpha,stable
```

8.2 Login to registry

```
oc whoami --show-token
docker login -p <token> -u unused <image-registry>
```

8.3 Create build & bundle

Building the image in WSL is EXTREMELY slow → this should be changed to a pipeline build in the openshift image factory.

8.3.1 Modify Makefile docker build commands to use docker format and image for bundle to use ARTIFACTORY_IMG

Our company Artifactory only understands docker format so our builds have to be in that format.

```
docker-build: test
    docker build --format=docker -t ${IMG} .

.....

bundle: manifests kustomize
    operator-sdk generate kustomize manifests -q
    cd config/manager && $(KUSTOMIZE) edit set image controller=$(ARTIFACTORY_IMG)
    $(KUSTOMIZE) build config/manifests | operator-sdk generate bundle -q --overwrite
--version $(VERSION) $(BUNDLE_METADATA_OPTS)
    operator-sdk bundle validate ./bundle

....

.PHONY: bundle-build
bundle-build:
    docker build --format=docker -f bundle.Dockerfile -t $(BUNDLE_IMG) .
```

If you forget this step, then Artifactory will have a different hash than the image in the cluster you pushed!!!

8.3.2 create build

Build the docker image

```
make docker-build docker-push IMG=$IMG
```

8.3.3 Eventually, replace kube-rbac-proxy image

We need to replace the kube-rbac-proxy image because the gcr.io one will not work in CHINA.

Replace in manager_auth_proxy_patch.yaml file:

```
image: gcr.io/kubebuilder/kube-rbac-proxy:v0.16.0
```

with

```
image: quay.io/brancz/kube-rbac-proxy:v0.16.0
```

8.3.4 Prepare for deployment

To make sure the deployed operator looks good in the Operator Hub and can be found and deployed easily, there are some changes necessary to the following file: config/*manifests/bases/your-operator.clusterserviceversion.yaml*

8.3.4.1 Adjust versions

Make sure that all versions are set correctly

8.3.4.2 Adjust descriptions

Add/adjust the descriptions of the operator and the owned CRDs

8.3.4.3 Set install mode

Set support for all install modes to false, except for OwnNamespace.

Install modes

```
installModes:
 - supported: true
   type: OwnNamespace
 - supported: false
   type: SingleNamespace
 - supported: false
   type: MultiNamespace
 - supported: false
   type: AllNamespaces
```

8.3.4.4 Add icon

Icon

```
 icon:
 - base64data: # Replace with your icon as base64
   mediatype: image/svg+xml
```

8.3.4.5 Add other useful information

Add keywords, the link to the repository, the maintainers, the maturity (channel), and the provider (your company) with their website.

8.3.5 make bundle

Check your config/manifests/bases directory and modify the clusterserviceversion as needed (installModes or add an image for example)

Afterwards, run make bundle again to update the generated bundle/manifests clusterserviceversion

This also validates the bundle

```
make bundle
```

8.3.6 Build and push bundle

```
make bundle-build bundle-push
```

8.4 Install with olm in a cluster

Installs the operator sdk to the cluster you are logged into, in the current namespace

And tries to get it from BUNDLE_IMG !!

```
operator-sdk run bundle $BUNDLE_IMG
```

This also creates a operator-bundle pod

8.5 Uninstall from cluster with cleanup

This works only if the operator was installed with run bundle (there is the pod present)

```
operator-sdk cleanup $OPERATOR
```

This will not delete the namespace, fortunately! Also, there are several flags you might want to consider. By default, no CRDs and CRs are deleted

https://sdk.operatorframework.io/docs/olm-integration/testing-deployment/#operator-sdk-cleanup-command-overview

8.6 Trigger the pipeline to distribute images

8.6.1 Install Tekton CLI if not present

```
# Get the tar.xz
curl -LO https://github.com/tektoncd/cli/releases/download/v0.22.0/
tkn_0.22.0_Linux_x86_64.tar.gz
# Extract tkn to your PATH (e.g. /usr/local/bin)
sudo tar xvzf tkn_0.22.0_Linux_x86_64.tar.gz -C /usr/local/bin/ tkn
```

8.6.2 Trigger pipeline

Trigger pipeline with name of the operator and version

```
oc project icp-image-factory
tkn pipeline start go-operator-pipeline -p operator=$OPERATOR -p version=v$VERSION -w
name=images-url,emptyDir=
```

This pipeline takes the image and the bundle-image and pushes it to artifactory, labels the imagestream of the bundle and starts the pipeline to update the registry

8.7 Deploy with subscription (Ansible Playbook)

Adapt this code to create a subscription for your operator

```
- name: "Infrastructure Services - Example Operator"
  hosts: 127.0.0.1
  connection: local
  tasks:

    - name: Create example-operator alpha subscription
      k8s:
        state: present
        definition:
          apiVersion: operators.coreos.com/v1alpha1
          kind: Subscription
          metadata:
            name: example-operator
            namespace: example-operator
          spec:
            channel: alpha
            installPlanApproval: Automatic
```

```
          name: example-operator
          source: <catalogsourcename>
          sourceNamespace: openshift-marketplace

    - name: Wait for NodeMaintenance CRD to be available
      k8s_info:
        api_version: apiextensions.k8s.io/v1
        kind: CustomResourceDefinition
        name: exampleconfigs.test.example.com
      register: result
      until: result.resources is defined and result.resources | length > 0
      delay: 10
      retries: 60

    - name: Create/update example config test-example-config
      k8s:
        state: present
        api_version: test.example.com/v1beta1
        kind: ExampleConfig
        name: test-example-config
        definition:
          spec:
            eventCount: 3
            messageContent: "bla"
```

If you install the operator in its own namespace, you have to create it first AND create a OperatorGroup or OLM will not be able to install.

Check if the namespace already has an OperatorGroup. Please pay attention that only one OperatorGroup is allowed in one namespace.

Installing the Operator from the OperatorHub will create a OperatorGroup automatically if none is present. Delete it before installing via ansible because the name generated is randomized.

```
    - name: Create example-operator namespace
      k8s:
        state: present
        merge_type: ["merge"]
        definition:
          api_version: v1
          kind: Namespace
          metadata:
            name: example-operator
            annotations:
              openshift.io/description: Hosts Example Operator
              openshift.io/display-name: example-operator

    - name: Create operator group example-operators
      k8s:
        state: present
```

```
        definition:
          apiVersion: operators.coreos.com/v1
          kind: OperatorGroup
          metadata:
            name: example-operator
            namespace: example-operator
          spec:
            targetNamespaces:
            - example-operator
```

8.8 Uninstall with Ansible Playbook

Delete subscription and CSV (with currently installed version)

Optional: Delete CRDs (deletes also instances)

```
- name: "Infrastructure Services - Example Operator"
  hosts: 127.0.0.1
  connection: local
  tasks:

    - name: Delete example-operator beta subscription
      k8s:
        state: absent
        definition:
          apiVersion: operators.coreos.com/v1alpha1
          kind: Subscription
          metadata:
            name: example-operator
            namespace: example-operator

    - name: Delete CSV
      k8s:
        state: absent
        definition:
          apiVersion: operators.coreos.com/v1alpha1
          kind: ClusterServiceVersion
          metadata:
            name: example-operator.v0.1.13
            namespace: example-operator

    - name: Delete CRD ExampleConfig
      k8s:
         state: absent
         definition:
           apiVersion: apiextensions.k8s.io/v1
           kind: CustomResourceDefinition
           metadata:
             name: exampleconfigs.test.example.com
```

Call via ansible playbook

9 Troubleshoot

9.1 missing go.sum entry for module providing package

Generate it anew

```
go mod tidy
```

9.2 Timeout error

If you get a "timeout waiting for process kube-apiserver to stop" adapt suite.go according to https://github.com/kubernetes-sigs/controller-runtime/issues/1571

9.3 Unittest build log output

Log output with "Info" level is missing in the log of unittest builds. To see those and additional verbose build information that might be helpful set KUBEBUILDER_ATTACH_CONTROL_PLANE_OUTPUT=true.

9.4 VS-Code unittest debugging

To be able to debug unittests in VS-Code create a configuration with

```
    "name": "<name>",
    "type": "go",
    "request": "launch",
    "mode": "test",
    "program": "${fileDirname}",
    "showLog": true
```

10 Pros and Cons

10.1 API Cache

Go Operators use a cache, they do not go directly to the API. The cache is kept up to date by watching the relevant resources.

You can add fields which are of relevance to query for you as an index for fast retrieval. Only Namespace is indexed by default, but you can index anything! That's more powerful/flexible than direct API calls

10.2 CRDs and CSV automatically generated, as well as RBACs

Less prone to error

10.3 Automatic Lifeness and Readiness Probes, as well as metric exposure

I haven't yet explored how to add a custom metric, but lifeness and readiness probes are way more reliable than the file check we use in python

10.4 One controller focuses on one resource

The reconcile method always concerns one resource (most of the time a CR) which is watched.

This leads to easier code.

10.5 Controller run in one container as threads and listen to changes of their resources

The reconcile method is called almost immediately if a resource which is in the cache is changed

10.6 Powerful online and offline tests

Envtest can be configured to run against an existing cluster or a local etcd. The testing framework is extensive, but tests are a blessing in the long run

10.7 Implement in Go

While Go has few keywords and is therefore thought to be easy to learn, it is a compiled language and is very type strict. Converting types can get confusing.

Also, since some expressions (like while for example) are missing and have to be done with the basic for loop, some expressions are not really intuitive

For someone who is used to an interpreted language like python, using go can be frustrating at first, since it must compile correctly before it can run.

While in python one can print out the type at runtime for example you cannot in Go if you do convert the type correctly. On the bright side, compiling is pretty fast.

The editors (VSCode and GoLand) are apparently very opinionated and format your code as well as deletes imports which are not used when saving.

Exception handling is cumbersome. You always have to check for the error variable. For people who are used to try catch and propagation of exceptions, it's a bit of a turn off.

Defer is kind of a finally for a function (executed before returning from it), but can be called everywhere in said function. That makes it less readable. Maybe try not to use it that often...

11 Open Questions

11.1 Build in cluster or local?

Local is faster but builds image from source which might not be committed

11.2 How to use custom image in CSV?

In the python CSV there is an image placeholder, where is this filled?

11.3 How to define multiple channels? Or will test strategy change and only one channel is needed?

Channel are defined in the bundle/metadata/annotations.yaml

This is probably generated by the bundle.Dockerfile

or is this even done in the Makefile?

11.4 There is a tool called opm supported that can create catalog images.

Catalog can be build (with a given comma spearated list of BUNDLE_IMGS) and pushed.

Images have to be pullable or it will fail