KCD

GROWING CLOUD NATIVE TOGETHER

BECOMING A KUBERNETES DEVELOPER

BUILDING OPERATOR FROM SCRATCH

DHAKA

KCD

GROWING CLOUD NATIVE TOGETHER

Al-Amin Talukdar (Timam)

Lead DevOps @ CHEQ Lifestyle Inc

WHAT IS AN OPERATOR?

A controller is a loop that reads desired state ("spec"), observed cluster state (others' "status"), and external state, and the reconciles cluster state and external state with the desired state, writing any observations down (to our own "status").

An operator is a controller that encodes human operational knowledge: how do I run and manage a specific piece of complex software.

All operators are controllers, but not all controllers are operators.

WAYS OF BUILDING OPERATOR

- Charmed Operator Framework
- Java Operator SDK
- Kopf (Kubernetes Operator Pythonic Framework)
- kube-rs (Rust)
- kubebuilder
- KubeOps (.NET operator SDK)
- KUDO (Kubernetes Universal Declarative Operator)
- Mast
- Metacontroller along with WebHooks that you implement yourself
- Operator Framework
- shell-operator

https://kubernetes.io/docs/concepts/extend-kubernetes/operator/#writing-operator

OPERATOR SDK

The Operator SDK makes it easier to build Kubernetes native applications, a process that can require deep, application-specific operational knowledge.

Build an Operator

WHAT IS OPERATOR SDK?

This project is a component of the Operator Framework, an open source toolkit to manage Kubernetes native applications, called Operators, in an effective, automated, and scalable way.

WHAT CAN I DO WITH OPERATOR SDK?

The Operator SDK provides the tools to build, test, and package Operators. Initially, the SDK facilitates the marriage of an application's business logic (for example, how to scale, upgrade, or backup) with the Kubernetes API to execute those operations. Over time, the SDK can allow engineers to make applications smarter and have the user experience of cloud services. Leading practices and code patterns that are shared across Operators are included in the SDK to help prevent reinventing the wheel.

The Operator SDK is a framework that uses the controller-runtime library to make writing operators easier by providing:

- High level APIs and abstractions to write the operational logic more intuitively
- Tools for scaffolding and code generation to bootstrap a new project fast
- Extensions to cover common Operator use cases

Build an Operator →

DEVELOP IN GO, ANSIBLE, OR HELM

GO

Create a new operator project using the SDK Command Line Interface (CLI)
Define new resource APIs by adding Custom Resource Definitions (CRD)
Define Controllers to watch and reconcile resources
Write the reconciling logic for your Controller using the SDK and controller-runtime APIs
Use the SDK CLI to build and generate the operator deployment manifests

Develop with Go →

ANSIBLE

Create a new operator project using the SDK Command Line Interface (CLI)
Write the reconciling logic for your object using ansible playbooks and roles
Use the SDK CLI to build and generate the operator deployment manifests
Optionally add additional CRD's using the SDK CLI and repeat steps 2 and 3

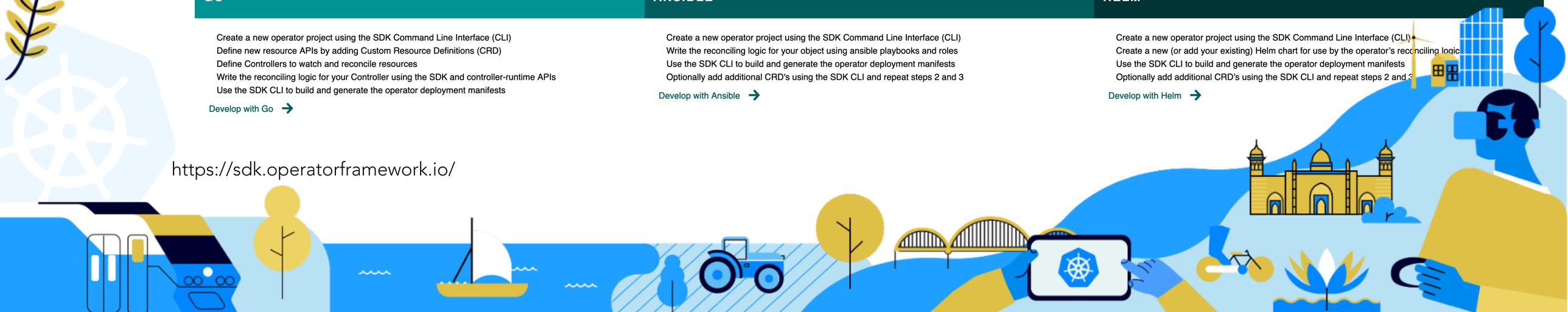
Develop with Ansible →

HELM

Create a new operator project using the SDK Command Line Interface (CLI)
Create a new (or add your existing) Helm chart for use by the operator's reconciling logic
Use the SDK CLI to build and generate the operator deployment manifests
Optionally add additional CRD's using the SDK CLI and repeat steps 2 and 3

Develop with Helm →

https://sdk.operatorframework.io/



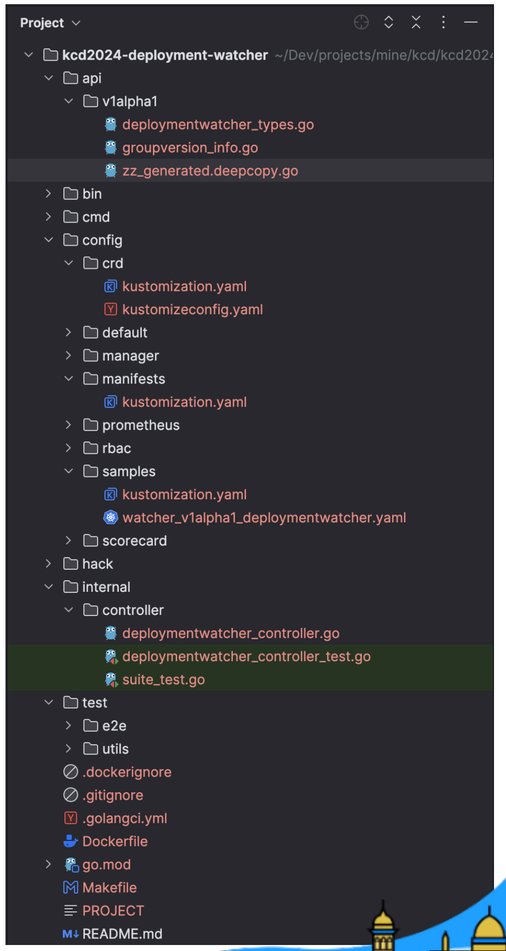
GETTING STARTED

```
timam@MacBook-Pro kcd2024-deployment-watcher % operator-sdk version
operator-sdk version: "v1.34.1", commit: "edaed1e5057db0349568e0b02df3743051b54e68", kubernetes
version: "v1.28.0", go version: "go1.21.7", GOOS: "darwin", GOARCH: "arm64"
timam@MacBook-Pro kcd2024-deployment-watcher % go version
go version go1.21.7 darwin/arm64
timam@MacBook-Pro kcd2024-deployment-watcher % kustomize version
v5.1.1
```

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

BOILERPLATE

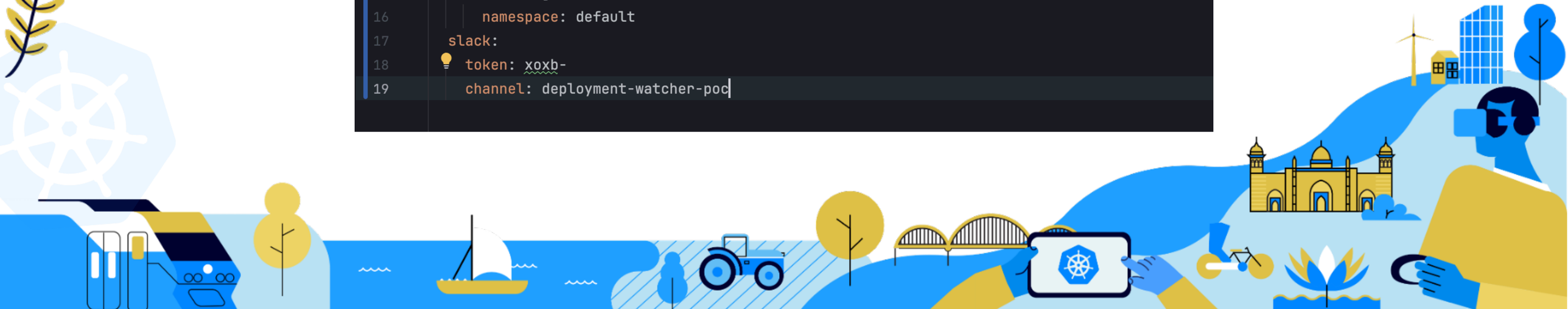
```
timam@MacBook-Pro kcd2024-deployment-watcher % operator-sdk init --domain operators.kcddhaka.org --repo
github.com/kcddhaka/kcd2024-deployment-watcher
INFO[0000] Writing kustomize manifests for you to edit...
INFO[0000] Writing scaffold for you to edit...
INFO[0000] Get controller runtime:
$ go get sigs.k8s.io/controller-runtime@v0.16.3
INFO[0002] Update dependencies:
$ go mod tidy
Next: define a resource with:
$ operator-sdk create api
timam@MacBook-Pro kcd2024-deployment-watcher %
timam@MacBook-Pro kcd2024-deployment-watcher % operator-sdk create api --group watcher --version
v1alpha1 --kind DeploymentWatcher --resource --controller
INFO[0000] Writing kustomize manifests for you to edit...
INFO[0000] Writing scaffold for you to edit...
INFO[0000] api/v1alpha1/deploymentwatcher_types.go
INFO[0000] api/v1alpha1/groupversion_info.go
INFO[0000] internal/controller/suite_test.go
INFO[0000] internal/controller/deploymentwatcher_controller.go
INFO[0000] internal/controller/deploymentwatcher_controller_test.go
INFO[0000] Update dependencies:
$ go mod tidy
INFO[0000] Running make:
$ make generate
mkdir -p /Users/timam/Dev/projects/mine/kcd/kcd2024-deployment-watcher/bin
test -s /Users/timam/Dev/projects/mine/kcd/kcd2024-deployment-watcher/bin/controller-gen && /Users/
timam/Dev/projects/mine/kcd/kcd2024-deployment-watcher/bin/controller-gen --version | grep -q v0.13.0
|| \
        GOBIN=/Users/timam/Dev/projects/mine/kcd/kcd2024-deployment-watcher/bin go install sigs.k8s.io/
controller-tools/cmd/controller-gen@v0.13.0
/Users/timam/Dev/projects/mine/kcd/kcd2024-deployment-watcher/bin/controller-gen
object:headerFile="hack/boilerplate.go.txt" paths="./..."
Next: implement your new API and generate the manifests (e.g. CRDs,CRs) with:
$ make manifests
```



PREPARE A SAMPLE

watcher_v1alpha1_deploymentwatcher.yaml

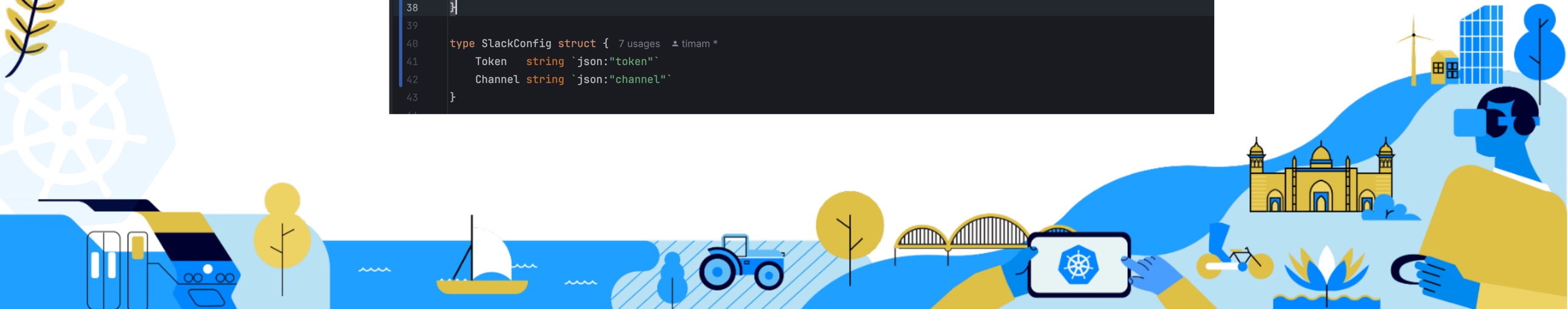
```yaml
apiVersion: watcher.operators.kcddhaka.org/v1alpha1
kind: DeploymentWatcher
metadata:
  labels:
    app.kubernetes.io/name: deploymentwatcher
    app.kubernetes.io/instance: deploymentwatcher-sample
    app.kubernetes.io/part-of: kcd2024-deployment-watcher
    app.kubernetes.io/managed-by: kustomize
    app.kubernetes.io/created-by: kcd2024-deployment-watcher
  name: deploymentwatcher-sample
spec:
  deployments:
    - name: nginx1
      namespace: default
    - name: nginx2
      namespace: default
  slack:
    token: xoxb-
    channel: deployment-watcher-poc
```



DEFINE TYPE

deploymentwatcher_types.go

```go
package v1alpha1

import ...

// EDIT THIS FILE!  THIS IS SCAFFOLDING FOR YOU TO OWN!
// NOTE: json tags are required.  Any new fields you add must have json tags for the fields to be serialized.

// DeploymentWatcherSpec defines the desired state of DeploymentWatcher
type DeploymentWatcherSpec struct { 8 usages  timam *
    // INSERT ADDITIONAL SPEC FIELDS - desired state of cluster
    // Important: Run "make" to regenerate code after modifying this file

    Deployments []NamespacedName `json:"deployments"`
    Slack       SlackConfig      `json:"slack"`
}

type NamespacedName struct { 8 usages  new *
    Name      string `json:"name"`
    Namespace string `json:"namespace"`
}

type SlackConfig struct { 7 usages  timam *
    Token   string `json:"token"`
    Channel string `json:"channel"`
}
```

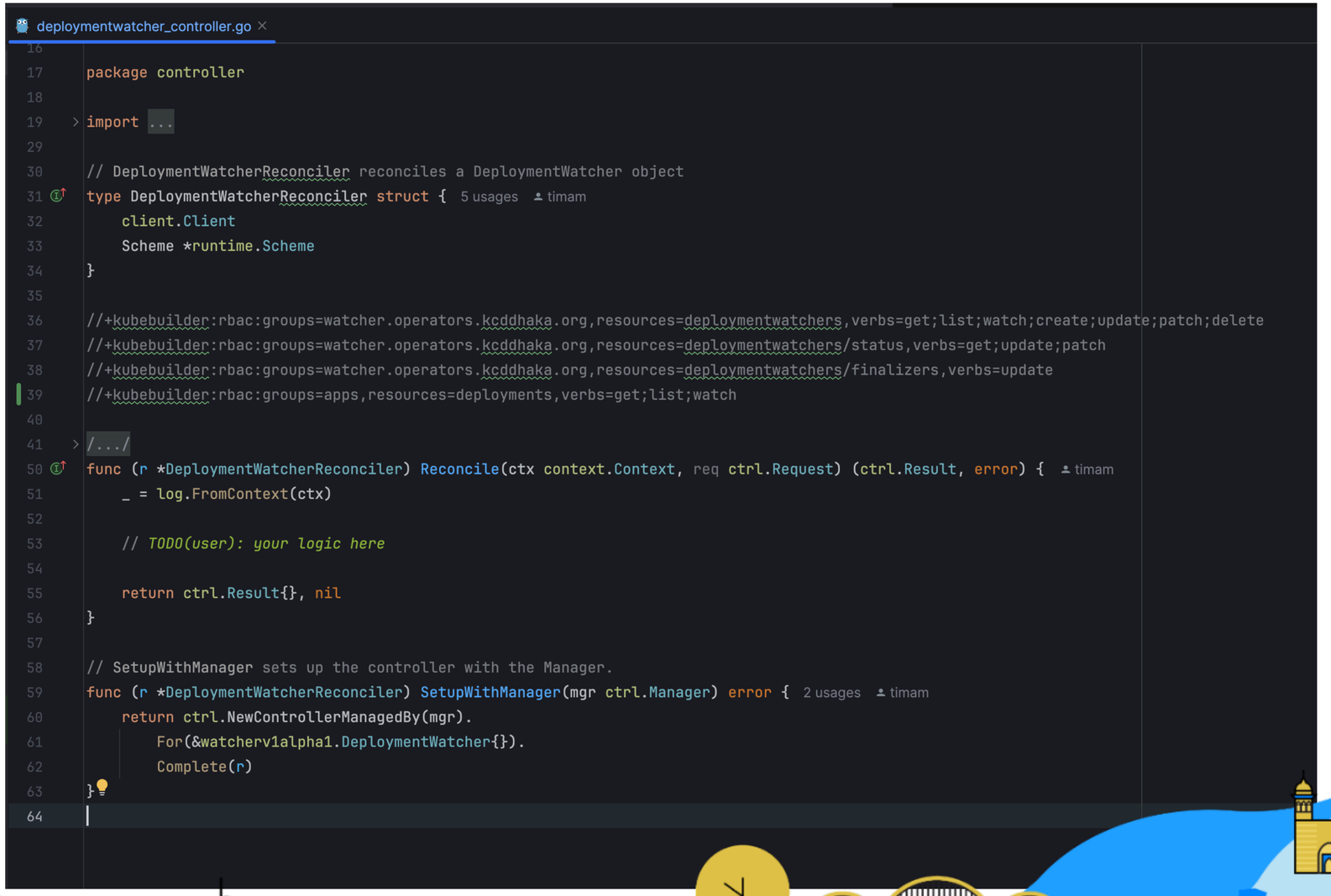


MAKE GENERATE

zz_generated.deepcopy.go

```go
package v1alpha1

import ...

// DeepCopyInto is an autogenerated deepcopy function, copying the receiver, writing into out. in must be non-nil.
func (in *DeploymentWatcher) DeepCopyInto(out *DeploymentWatcher) {  timam *
    *out = *in
    out.TypeMeta = in.TypeMeta
    in.ObjectMeta.DeepCopyInto(&out.ObjectMeta)
    in.Spec.DeepCopyInto(&out.Spec)
    out.Status = in.Status
}

// DeepCopy is an autogenerated deepcopy function, copying the receiver, creating a new DeploymentWatcher.
func (in *DeploymentWatcher) DeepCopy() *DeploymentWatcher {...}

// DeepCopyObject is an autogenerated deepcopy function, copying the receiver, creating a new runtime.Object.
func (in *DeploymentWatcher) DeepCopyObject() runtime.Object {...}

// DeepCopyInto is an autogenerated deepcopy function, copying the receiver, writing into out. in must be non-nil.
func (in *DeploymentWatcherList) DeepCopyInto(out *DeploymentWatcherList) {...}

// DeepCopy is an autogenerated deepcopy function, copying the receiver, creating a new DeploymentWatcherList.
func (in *DeploymentWatcherList) DeepCopy() *DeploymentWatcherList {...}

// DeepCopyObject is an autogenerated deepcopy function, copying the receiver, creating a new runtime.Object.
func (in *DeploymentWatcherList) DeepCopyObject() runtime.Object {...}

// DeepCopyInto is an autogenerated deepcopy function, copying the receiver, writing into out. in must be non-nil.
func (in *DeploymentWatcherSpec) DeepCopyInto(out *DeploymentWatcherSpec) {  timam *
    *out = *in
    if in.Deployments != nil {
        in, out := &in.Deployments, &out.Deployments
        *out = make([]NamespacedName, len(*in))
        copy(*out, *in)
    }
    out.Slack = in.Slack
}

// DeepCopy is an autogenerated deepcopy function, copying the receiver, creating a new DeploymentWatcherSpec.
func (in *DeploymentWatcherSpec) DeepCopy() *DeploymentWatcherSpec {...}

// DeepCopyInto is an autogenerated deepcopy function, copying the receiver, writing into out. in must be non-nil.
func (in *DeploymentWatcherStatus) DeepCopyInto(out *DeploymentWatcherStatus) {...}

// DeepCopy is an autogenerated deepcopy function, copying the receiver, creating a new DeploymentWatcherStatus.
func (in *DeploymentWatcherStatus) DeepCopy() *DeploymentWatcherStatus {...}

// DeepCopyInto is an autogenerated deepcopy function, copying the receiver, writing into out. in must be non-nil.
func (in *NamespacedName) DeepCopyInto(out *NamespacedName) {  new *
    *out = *in
}

// DeepCopy is an autogenerated deepcopy function, copying the receiver, creating a new NamespacedName.
func (in *NamespacedName) DeepCopy() *NamespacedName {  new *
    if in == nil { return nil }
    out := new(NamespacedName)
    in.DeepCopyInto(out)
    return out
}

// DeepCopyInto is an autogenerated deepcopy function, copying the receiver, writing into out. in must be non-nil.
func (in *SlackConfig) DeepCopyInto(out *SlackConfig) {  new *
    *out = *in
}

// DeepCopy is an autogenerated deepcopy function, copying the receiver, creating a new SlackConfig.
func (in *SlackConfig) DeepCopy() *SlackConfig {  new *
    if in == nil { return nil }
    out := new(SlackConfig)
    in.DeepCopyInto(out)
    return out
}
```

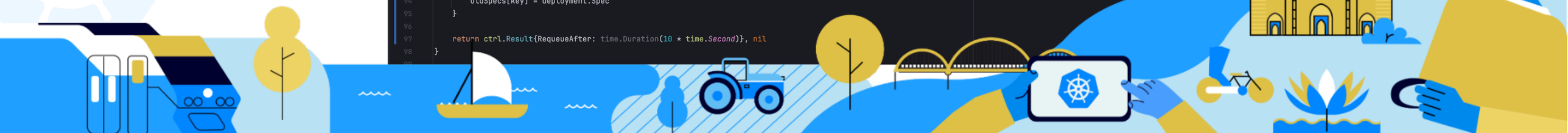
RBAC


deploymentwatcher_controller.go
16
17 package controller
18
19 import ...
29
30 // DeploymentWatcherReconciler reconciles a DeploymentWatcher object
31 type DeploymentWatcherReconciler struct { 5 usages timam
32 client.Client
33 Scheme *runtime.Scheme
34 }
35
36 //+kubebuilder:rbac:groups=watcher.operators.kcddhaka.org,resources=deploymentwatchers,verbs=get;list;watch;create;update;patch;delete
37 //+kubebuilder:rbac:groups=watcher.operators.kcddhaka.org,resources=deploymentwatchers/status,verbs=get;update;patch
38 //+kubebuilder:rbac:groups=watcher.operators.kcddhaka.org,resources=deploymentwatchers/finalizers,verbs=update
39 //+kubebuilder:rbac:groups=apps,resources=deployments,verbs=get;list;watch
40
41 /.../
50 func (r *DeploymentWatcherReconciler) Reconcile(ctx context.Context, req ctrl.Request) (ctrl.Result, error) { timam
51 _ = log.FromContext(ctx)
52
53 // TODO(user): your logic here
54
55 return ctrl.Result{}, nil
56 }
57
58 // SetupWithManager sets up the controller with the Manager.
59 func (r *DeploymentWatcherReconciler) SetupWithManager(mgr ctrl.Manager) error { 2 usages timam
60 return ctrl.NewControllerManagedBy(mgr).
61 For(&watcherv1alpha1.DeploymentWatcher{}).
62 Complete(r)
63 }
64

RECONCILE

deploymentwatcher_controller.go

```go
    watcherv1alpha1 "github.com/kcddhaka/kcd2024-deployment-watcher/api/v1alpha1"
    v1 "k8s.io/api/apps/v1"
)

// DeploymentWatcherReconciler reconciles a DeploymentWatcher object
type DeploymentWatcherReconciler struct {  5 usages  timam
    client.Client
    Scheme *runtime.Scheme
}

/.../

/.../
var oldSpecs = make(map[string]v1.DeploymentSpec)  2 usages  new *

func (r *DeploymentWatcherReconciler) Reconcile(ctx context.Context, req ctrl.Request) (ctrl.Result, error) {  timam *
    Log := log.Log.WithValues( keysAndValues...: "Request.Namespace", req.Namespace, "Request.Name", req.Name)
    //Log.Info("Reconcile Called")

    dw := &watcherv1alpha1.DeploymentWatcher{}

    err := r.Get(ctx, req.NamespacedName, dw)
    if err != nil { return ctrl.Result{}, err }

    for _, deploy := range dw.Spec.Deployments {
        deployment := &v1.Deployment{}
        err := r.Get(ctx, types.NamespacedName{
            Namespace: deploy.Namespace,
            Name:      deploy.Name,
        }, deployment)
        if err != nil { return ctrl.Result{}, err }

        key := deploy.Namespace + "/" + deploy.Name
        oldSpec, ok := oldSpecs[key]
        if ok && !reflect.DeepEqual(oldSpec, deployment.Spec) {
            Log.Info( msg: "Deployment spec has changed",  keysAndValues...: "namespace", deploy.Namespace, "name", deploy.Name)

            if !reflect.DeepEqual(oldSpec.Template.Spec.Containers[0].Image, deployment.Spec.Template.Spec.Containers[0].Image) {
                Log.Info( msg: "Deployment image has changed",  keysAndValues...: "namespace", deploy.Namespace, "name", deploy.Name)

                err = notify(deploy.Namespace, deploy.Name, deployment.Spec.Template.Spec.Containers[0].Image, dw.Spec.Slack.Token, dw.Spec.Slack.Channel)
                if err != nil { return ctrl.Result{}, err }
            } else if !reflect.DeepEqual(*oldSpec.Replicas, *deployment.Spec.Replicas) {
                Log.Info( msg: "Deployment replicas have changed",  keysAndValues...: "namespace", deploy.Namespace, "name", deploy.Name)
            }
        }

        oldSpecs[key] = deployment.Spec
    }

    return ctrl.Result{RequeueAfter: time.Duration(10 * time.Second)}, nil
}
```



ACTION

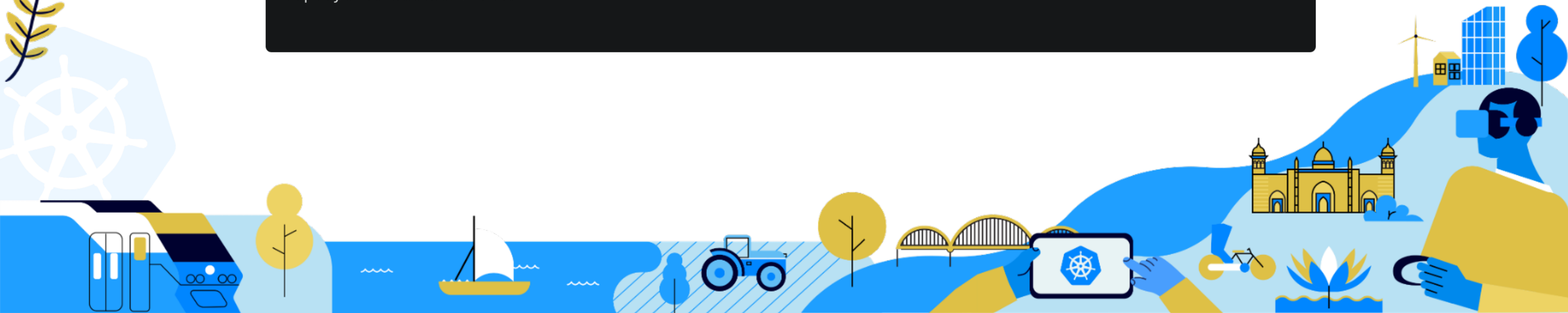
deploymentwatcher_slack.go

```go
package controller

import (
    "github.com/slack-go/slack"
    "strings"
)

func getTag(version string) string {  1 usage
    splitVersion := strings.Split(version,  sep: ":")
    return splitVersion[len(splitVersion)-1]
}

func notify(namespace, name, version, slackToken, slackChannel string) error {  1 usage

    tag := getTag(version)

    headerSection := slack.NewSectionBlock(
        slack.NewTextBlockObject( elementType: "mrkdwn",  text: ":rocket: New deployment is in progress!",  emoji: false,  verbatim: false),
        fields: nil,
        accessory: nil,
    )

    fields := make([]*slack.TextBlockObject, 0)
    fields = append(fields, slack.NewTextBlockObject( elementType: "mrkdwn",  text: "*Namespace/Deployment*",  emoji: false,  verbatim: false))
    fields = append(fields, slack.NewTextBlockObject( elementType: "mrkdwn",  text: "*Version*",  emoji: false,  verbatim: false))
    fields = append(fields, slack.NewTextBlockObject( elementType: "mrkdwn", namespace+"/"+name,  emoji: false,  verbatim: false))
    fields = append(fields, slack.NewTextBlockObject( elementType: "mrkdwn", tag,  emoji: false,  verbatim: false))

    deploymentSection := slack.NewSectionBlock(
        textObj: nil,
        fields,
        accessory: nil,
    )

    attachment := slack.Attachment{
        Blocks: slack.Blocks{
            BlockSet: []slack.Block{headerSection, deploymentSection},
        },
        Color: "#36a64f",
    }

    message := slack.MsgOptionAttachments(attachment)
    api := slack.New(slackToken)
    _, _, err := api.PostMessage(slackChannel, message)
    if err != nil { return err }
    return nil
}
```

IMAGE

```
timam@MacBook-Pro kcd2024-deployment-watcher % cd config/manager
timam@MacBook-Pro manager % kustomize edit set image controller=ghcr.io/kcddhaka/kcd2024-deployment-watcher:latest
```

```
timam@MacBook-Pro kcd2024-deployment-watcher %  make docker-buildx IMG=ghcr.io/kcddhaka/kcd2024-deployment-watcher:latest
timam@MacBook-Pro kcd2024-deployment-watcher %  make docker-push IMG=ghcr.io/kcddhaka/kcd2024-deployment-watcher:latest
```

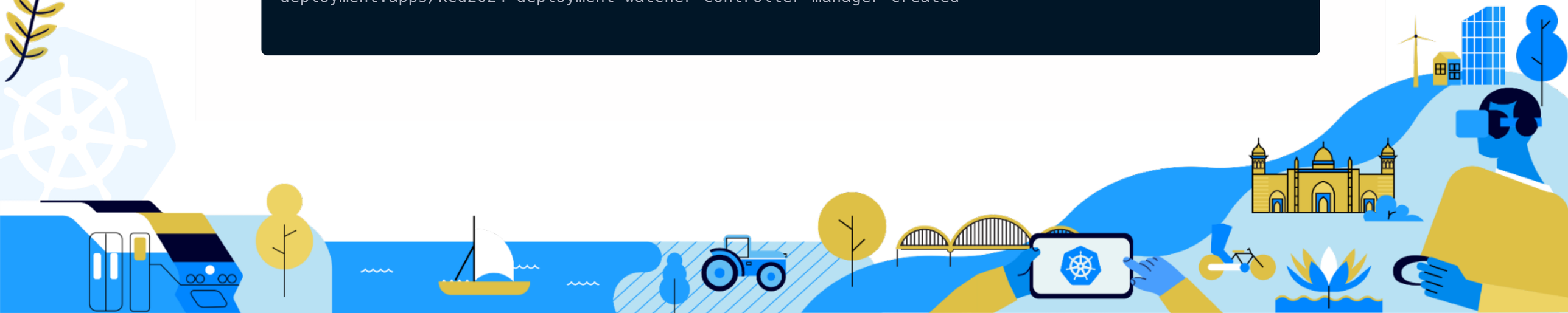


MANIFEST

```
timam@MacBook-Pro kcd2024-deployment-watcher % make manifests
/Users/timam/Dev/projects/mine/kcd/kcd2024-deployment-watcher/bin/controller-gen rbac:roleName=manager-role crd webhook paths="./..." output:crd:artifacts:config=config/crd/bases

timam@MacBook-Pro kcd2024-deployment-watcher % kustomize build config/default > ./deployment-watcher-operator.yaml
```

DEPLOY OPERATOR

```
timam@MacBook-Pro kcd2024-deployment-watcher % kubectl apply -f deployment-watcher-operator.yaml
namespace/kcd2024-deployment-watcher-system created
customresourcedefinition.apiextensions.k8s.io/deploymentwatchers.watcher.operators.kcddhaka.org created
serviceaccount/kcd2024-deployment-watcher-controller-manager created
role.rbac.authorization.k8s.io/kcd2024-deployment-watcher-leader-election-role created
clusterrole.rbac.authorization.k8s.io/kcd2024-deployment-watcher-manager-role created
clusterrole.rbac.authorization.k8s.io/kcd2024-deployment-watcher-metrics-reader created
clusterrole.rbac.authorization.k8s.io/kcd2024-deployment-watcher-proxy-role created
rolebinding.rbac.authorization.k8s.io/kcd2024-deployment-watcher-leader-election-rolebinding created
clusterrolebinding.rbac.authorization.k8s.io/kcd2024-deployment-watcher-manager-rolebinding created
clusterrolebinding.rbac.authorization.k8s.io/kcd2024-deployment-watcher-proxy-rolebinding created
service/kcd2024-deployment-watcher-controller-manager-metrics-service created
deployment.apps/kcd2024-deployment-watcher-controller-manager created
```



CR

deployment-watcher-cr.yaml

```yaml
apiVersion: watcher.operators.kcddhaka.org/v1alpha1
kind: DeploymentWatcher
metadata:
  name: demo-watcher
  namespace: default
spec:
  deployments:
    - name: nginx
      namespace: default
  slack:
    channel: dw-non-prod
    token: xoxb-
```

```
timam@MacBook-Pro kcd2024-deployment-watcher % kubectl apply -f deployment-watcher-operator.yaml
namespace/kcd2024-deployment-watcher-system created
customresourcedefinition.apiextensions.k8s.io/deploymentwatchers.watcher.operators.kcddhaka.org created
serviceaccount/kcd2024-deployment-watcher-controller-manager created
role.rbac.authorization.k8s.io/kcd2024-deployment-watcher-leader-election-role created
clusterrole.rbac.authorization.k8s.io/kcd2024-deployment-watcher-manager-role created
clusterrole.rbac.authorization.k8s.io/kcd2024-deployment-watcher-metrics-reader created
clusterrole.rbac.authorization.k8s.io/kcd2024-deployment-watcher-proxy-role created
rolebinding.rbac.authorization.k8s.io/kcd2024-deployment-watcher-leader-election-rolebinding created
clusterrolebinding.rbac.authorization.k8s.io/kcd2024-deployment-watcher-manager-rolebinding created
clusterrolebinding.rbac.authorization.k8s.io/kcd2024-deployment-watcher-proxy-rolebinding created
service/kcd2024-deployment-watcher-controller-manager-metrics-service created
deployment.apps/kcd2024-deployment-watcher-controller-manager created
```

QUESTION?

THANK YOU