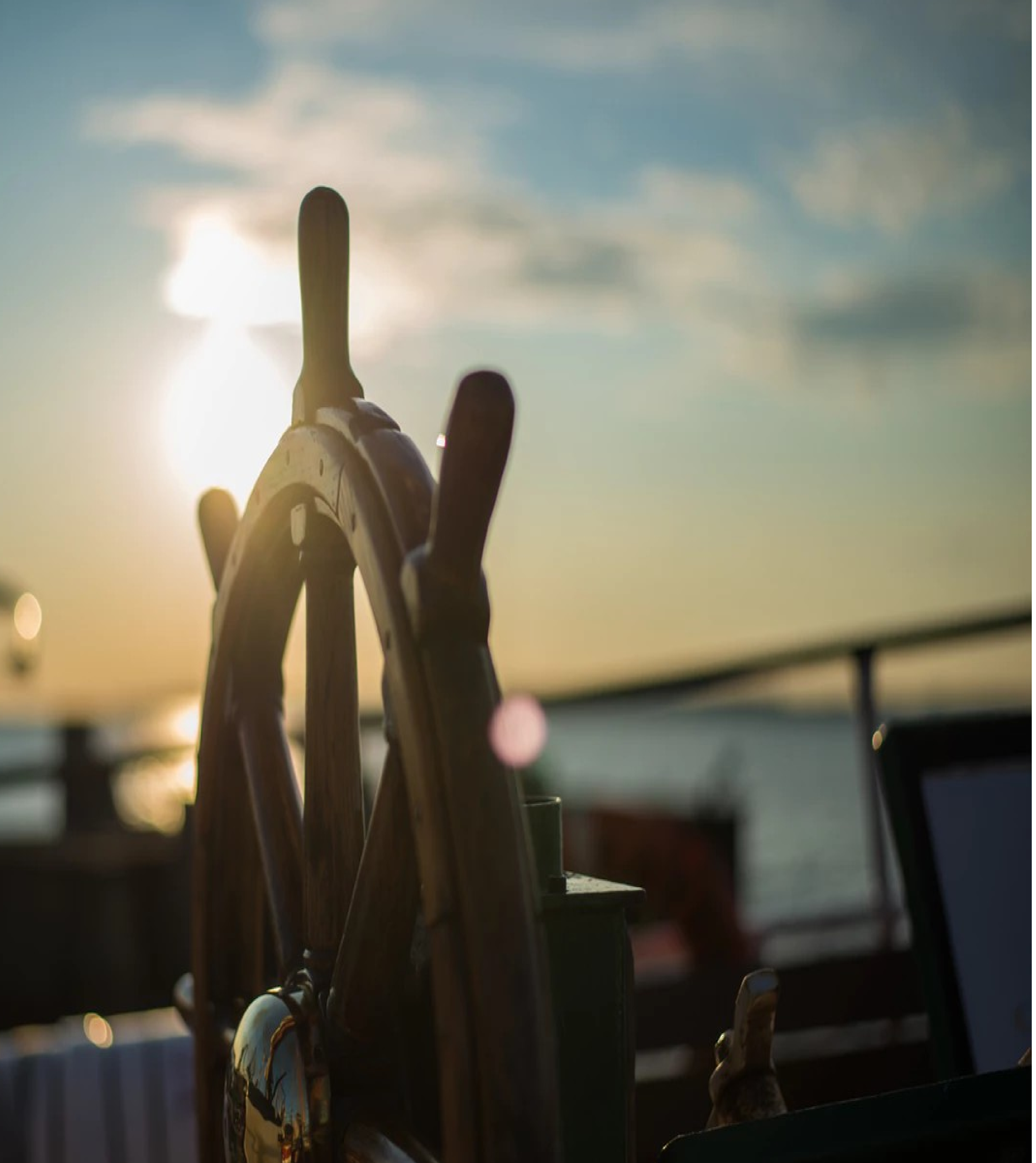
Kubernetes Custom Resources com Go.

Diego Marangoni
Engenheiro de Software @ Acesso Digital

@eusoudiego
github.com/diegomarangoni



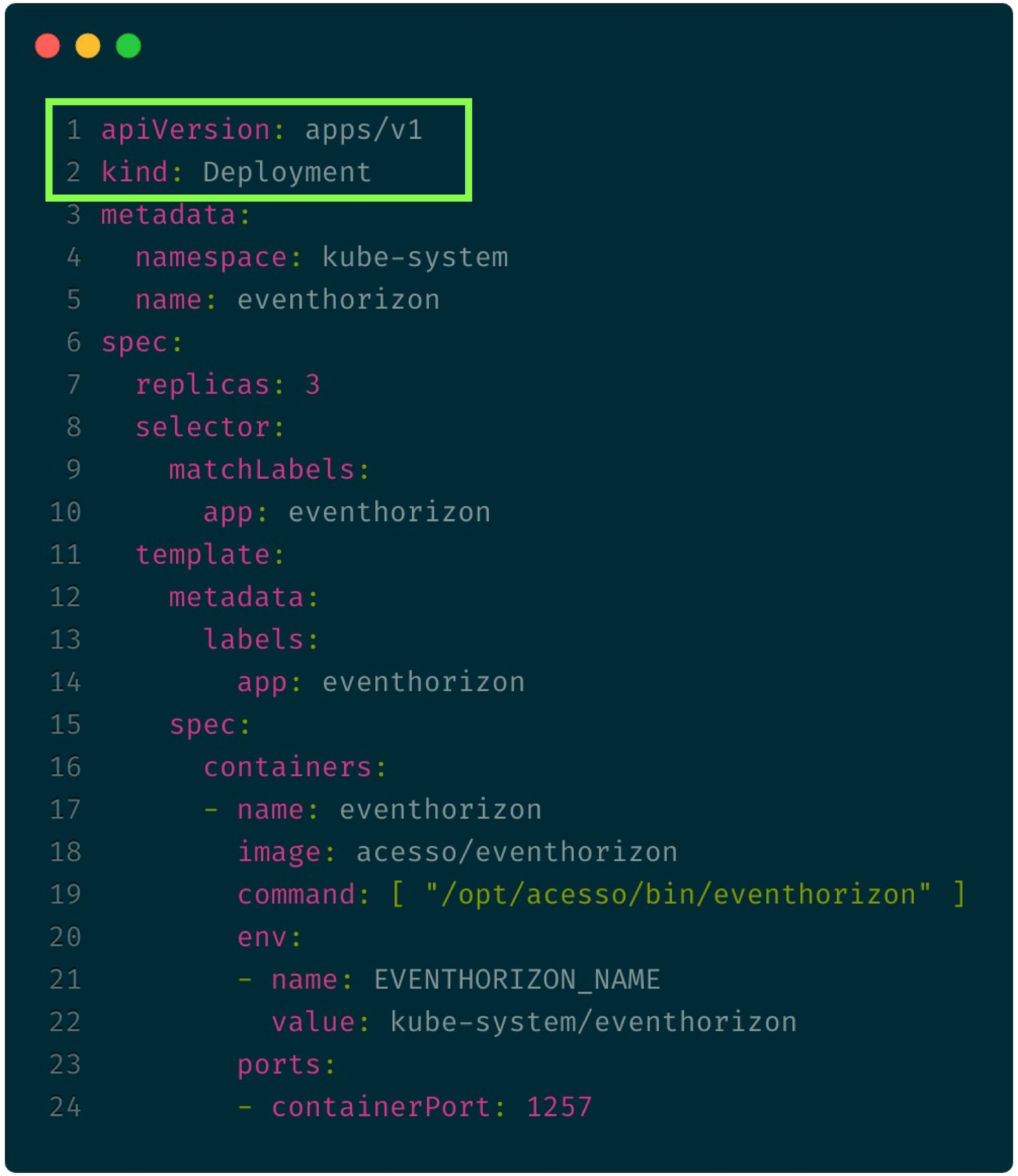
O que é Kubernetes?

"Kubernetes is a portable, extensible, open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation."

Kubernetes Resources

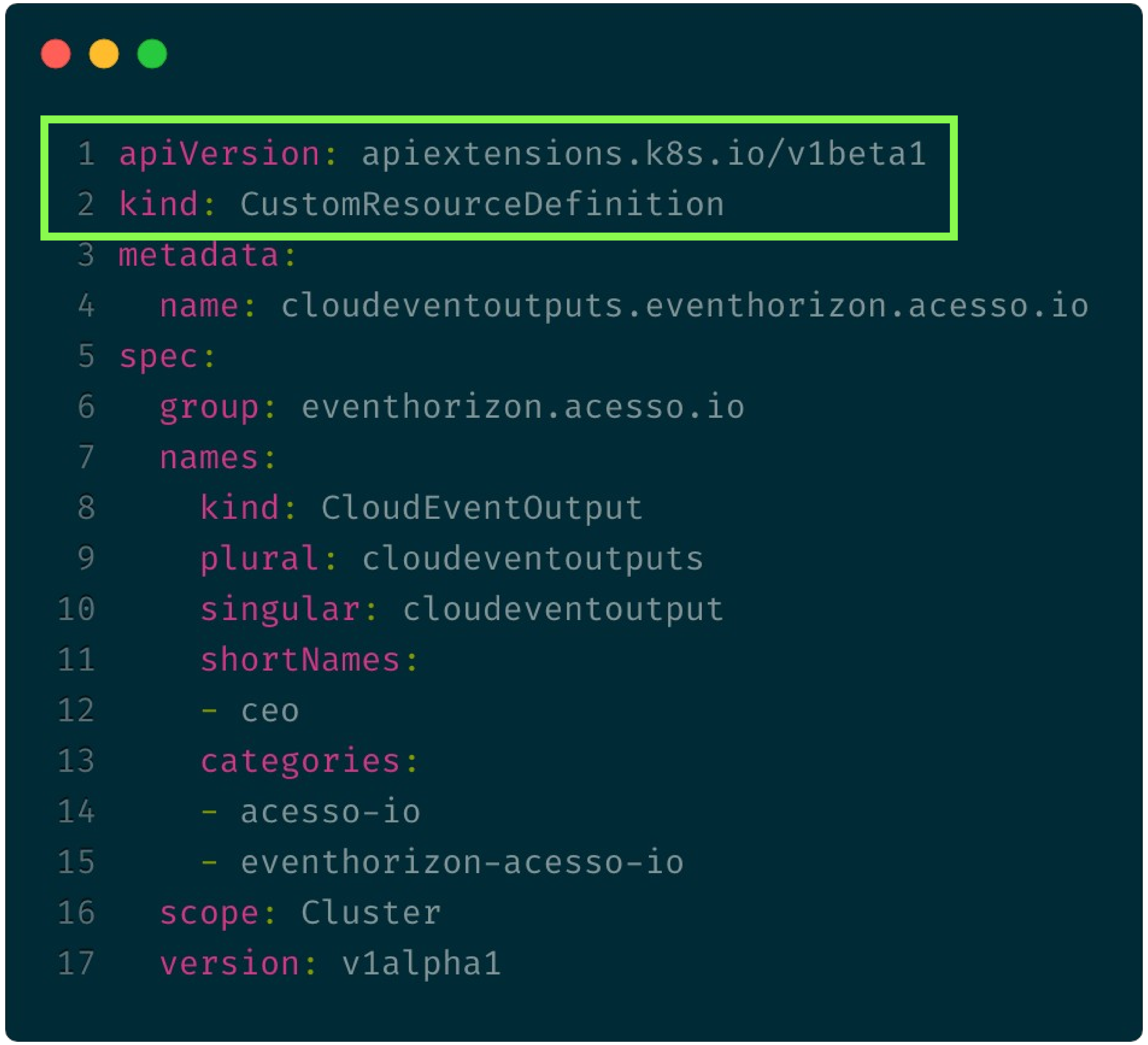
- Deployment
- ReplicaSets
- Pods
- StatefulSets
- DaemonSets
- CronJob
- Job
- CustomResourceDefinition

Deployment



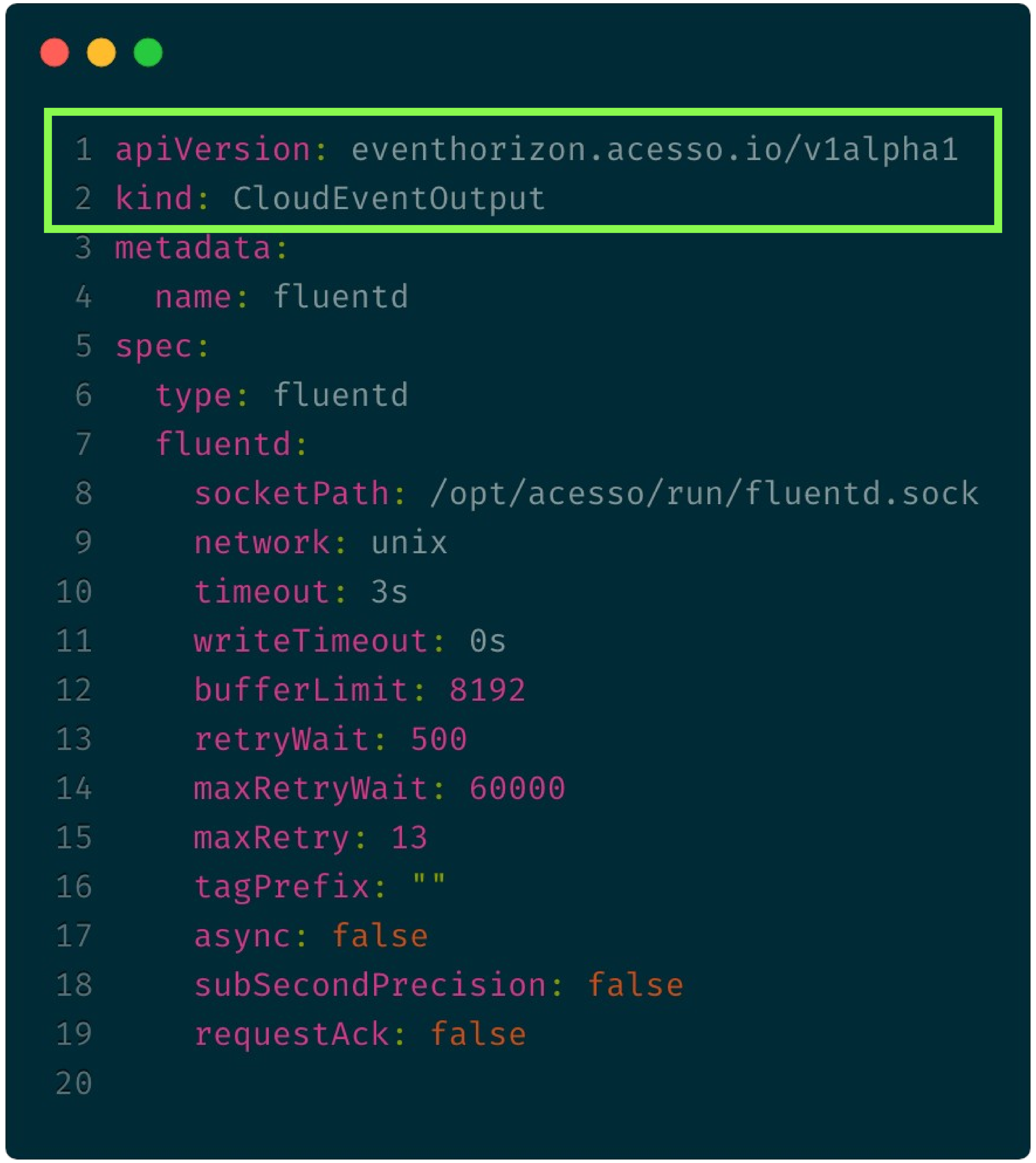
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  namespace: kube-system
  name: eventhorizon
spec:
  replicas: 3
  selector:
    matchLabels:
      app: eventhorizon
  template:
    metadata:
      labels:
        app: eventhorizon
    spec:
      containers:
      - name: eventhorizon
        image: acesso/eventhorizon
        command: [ "/opt/acesso/bin/eventhorizon" ]
        env:
        - name: EVENTHORIZON_NAME
          value: kube-system/eventhorizon
        ports:
        - containerPort: 1257
```

CustomResourcesDefinition



```yaml
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  name: cloudeventoutputs.eventhorizon.acesso.io
spec:
  group: eventhorizon.acesso.io
  names:
    kind: CloudEventOutput
    plural: cloudeventoutputs
    singular: cloudeventoutput
    shortNames:
    - ceo
    categories:
    - acesso-io
    - eventhorizon-acesso-io
  scope: Cluster
  version: v1alpha1
```

Custom Resource *CloudEventOutput*



```yaml
apiVersion: eventhorizon.acesso.io/v1alpha1
kind: CloudEventOutput
metadata:
  name: fluentd
spec:
  type: fluentd
  fluentd:
    socketPath: /opt/acesso/run/fluentd.sock
    network: unix
    timeout: 3s
    writeTimeout: 0s
    bufferLimit: 8192
    retryWait: 500
    maxRetryWait: 60000
    maxRetry: 13
    tagPrefix: ""
    async: false
    subSecondPrecision: false
    requestAck: false

```

Estrutura mínima

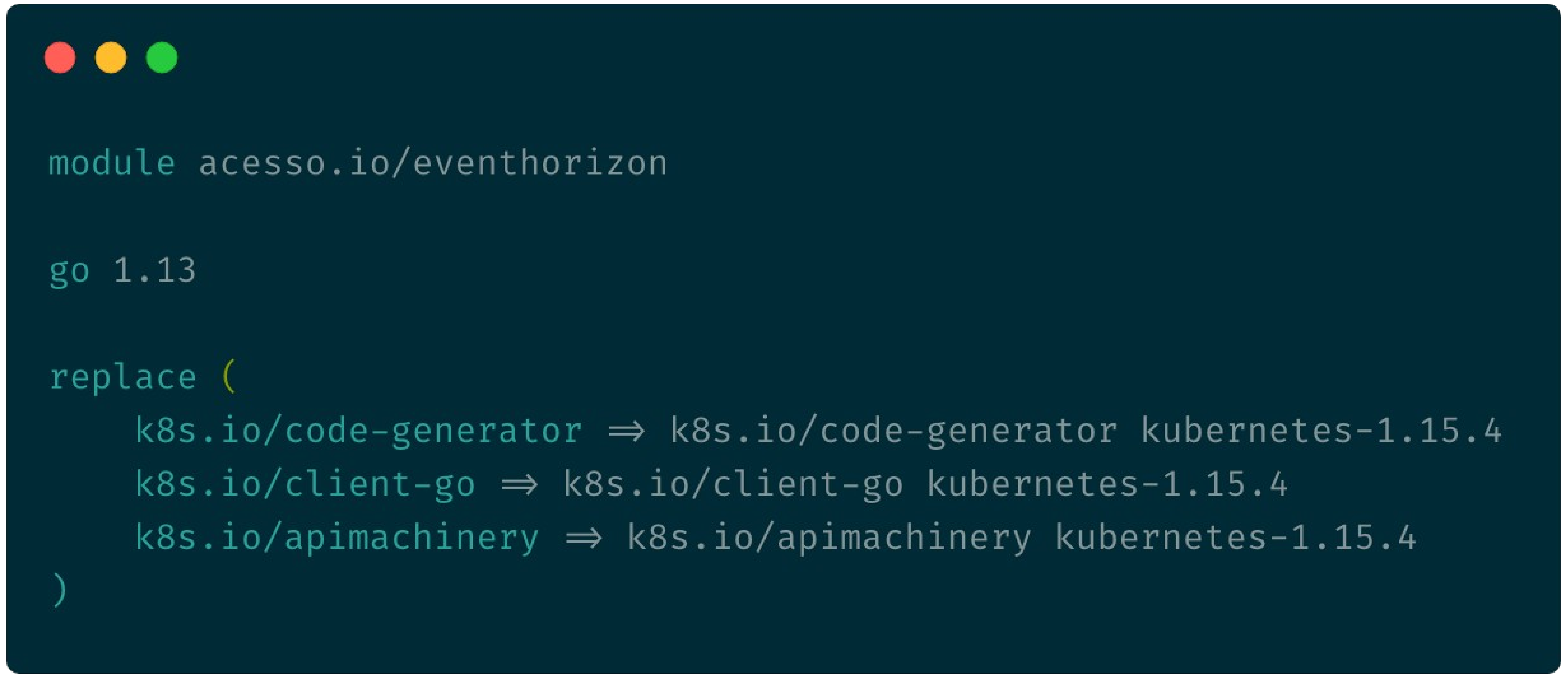
```
├── cmd
│   └── main.go
├── go.mod
├── hack
│   ├── tools.go
│   └── update-codegen.sh
├── pkg
│   ├── apis
│   │   └── eventhorizon
│   │       └── v1alpha1
│   │           ├── doc.go
│   │           ├── register.go
│   │           └── types_cloud_event_output.go
│   └── controller
│       └── controller.go
└── samples
    ├── CloudEventOutput.yml
    └── crds
        └── CloudEventOutput.yml
```

Estrutura aplicação

```
├── cmd
│   └── main.go
├── go.mod
├── hack
│   ├── tools.go
│   └── update-codegen.sh
├── pkg
│   ├── apis
│   │   └── eventhorizon
│   │       └── v1alpha1
│   │           ├── doc.go
│   │           ├── register.go
│   │           └── types_cloud_event_output.go
│   └── controller
│       └── controller.go
└── samples
    ├── CloudEventOutput.yml
    └── crds
        └── CloudEventOutput.yml
```

Estrutura mínima

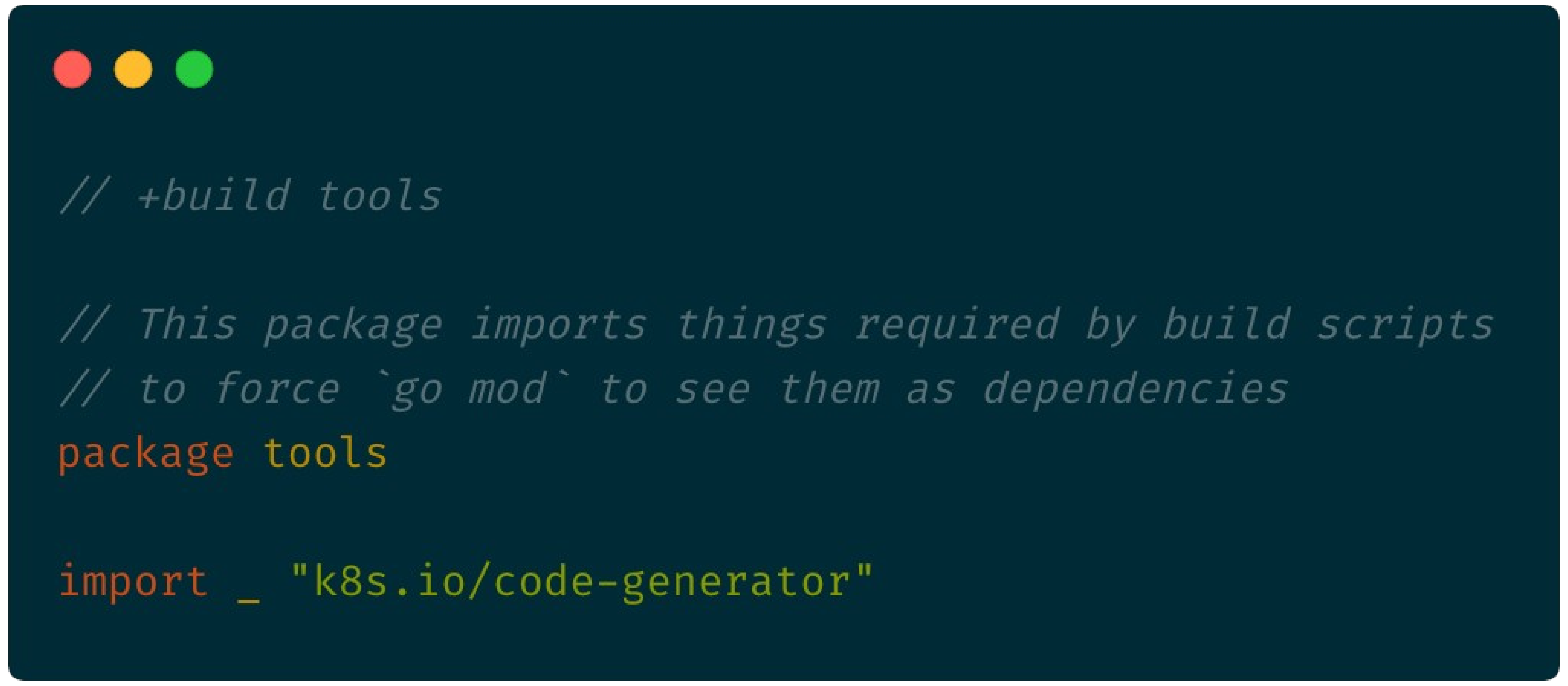
```
├── cmd
│   └── main.go
├── go.mod
├── hack
│   ├── tools.go
│   └── update-codegen.sh
├── pkg
│   ├── apis
│   │   └── eventhorizon
│   │       └── v1alpha1
│   │           ├── doc.go
│   │           ├── register.go
│   │           └── types_cloud_event_output.go
│   └── controller
│       └── controller.go
└── samples
    ├── CloudEventOutput.yml
    └── crds
        └── CloudEventOutput.yml
```



```
module acesso.io/eventhorizon

go 1.13

replace (
    k8s.io/code-generator ⇒ k8s.io/code-generator kubernetes-1.15.4
    k8s.io/client-go ⇒ k8s.io/client-go kubernetes-1.15.4
    k8s.io/apimachinery ⇒ k8s.io/apimachinery kubernetes-1.15.4
)
```

Estrutura mínima

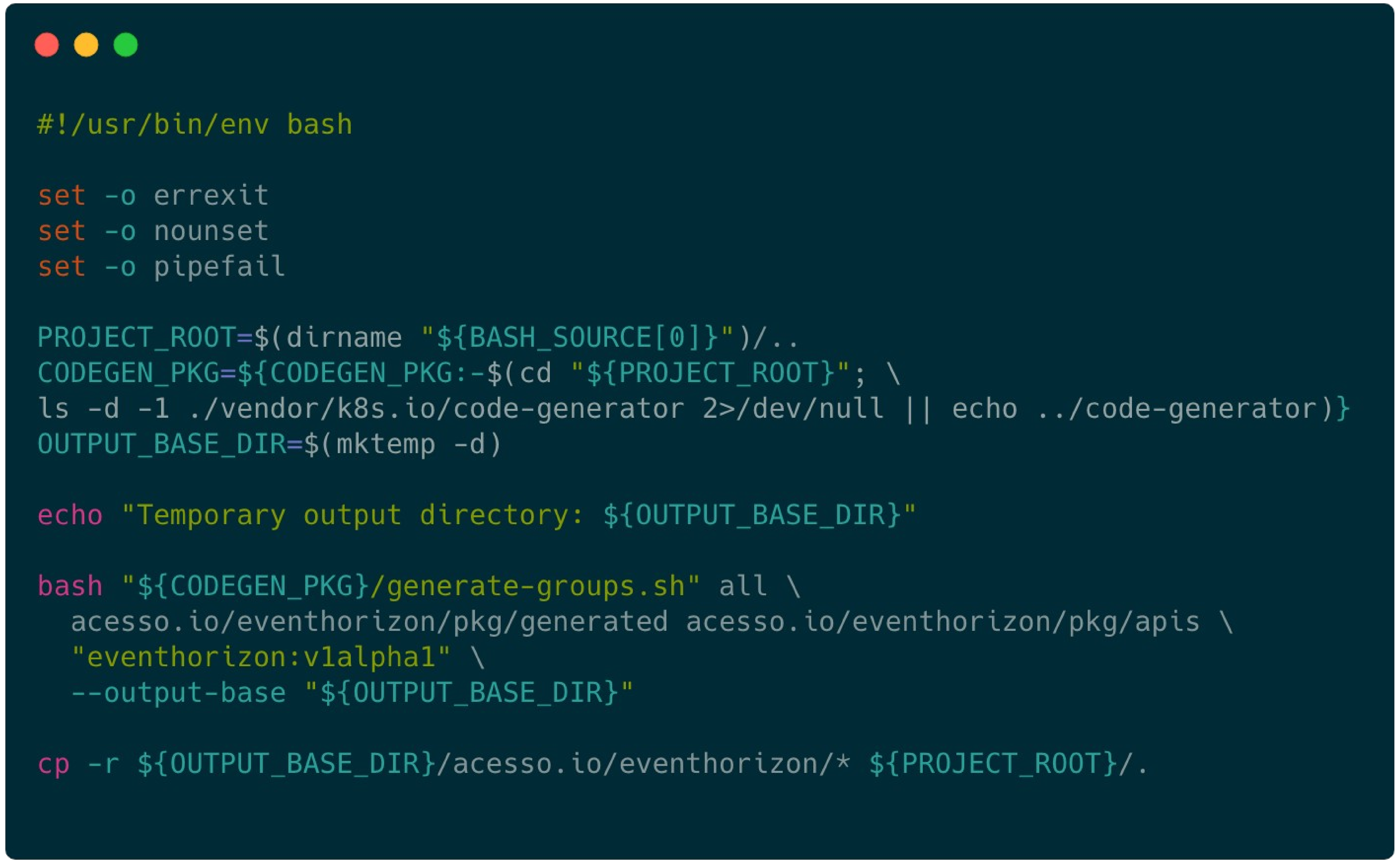
```
├── cmd
│   └── main.go
├── go.mod
├── hack
│   ├── tools.go
│   └── update-codegen.sh
├── pkg
│   ├── apis
│   │   └── eventhorizon
│   │       └── v1alpha1
│   │           ├── doc.go
│   │           ├── register.go
│   │           └── types_cloud_event_output.go
│   └── controller
│       └── controller.go
└── samples
    ├── CloudEventOutput.yml
    └── crds
        └── CloudEventOutput.yml
```



```go
// +build tools

// This package imports things required by build scripts
// to force `go mod` to see them as dependencies
package tools

import _ "k8s.io/code-generator"
```

Estrutura mínima

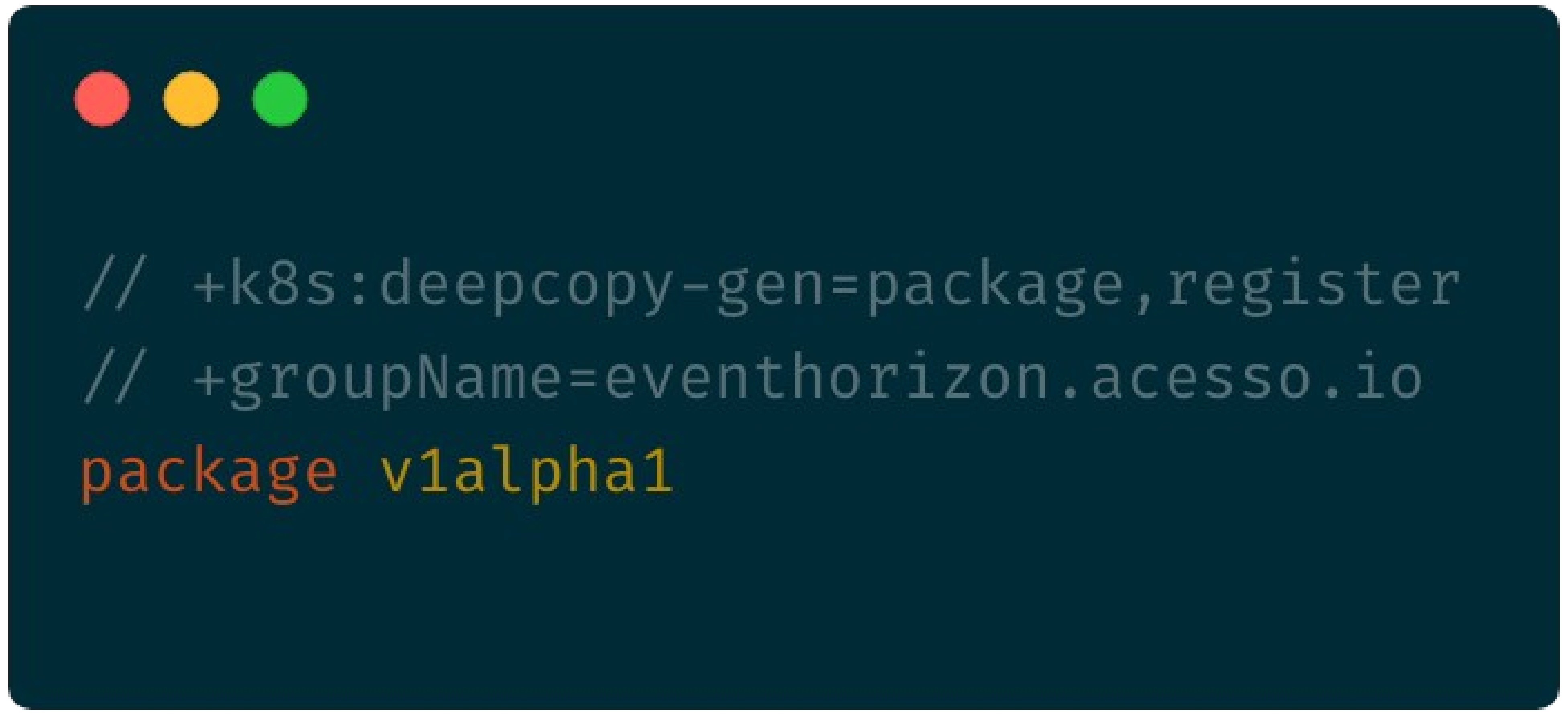
```
├── cmd
│   └── main.go
├── go.mod
├── hack
│   ├── tools.go
│   └── update-codegen.sh
├── pkg
│   ├── apis
│   │   └── eventhorizon
│   │       └── v1alpha1
│   │           ├── doc.go
│   │           ├── register.go
│   │           └── types_cloud_event_output.go
│   └── controller
│       └── controller.go
└── samples
    ├── CloudEventOutput.yml
    └── crds
        └── CloudEventOutput.yml
```



```bash
#!/usr/bin/env bash

set -o errexit
set -o nounset
set -o pipefail

PROJECT_ROOT=$(dirname "${BASH_SOURCE[0]}")/..
CODEGEN_PKG=${CODEGEN_PKG:-$(cd "${PROJECT_ROOT}"; \
ls -d -1 ./vendor/k8s.io/code-generator 2>/dev/null || echo ../code-generator)}
OUTPUT_BASE_DIR=$(mktemp -d)

echo "Temporary output directory: ${OUTPUT_BASE_DIR}"

bash "${CODEGEN_PKG}/generate-groups.sh" all \
  acesso.io/eventhorizon/pkg/generated acesso.io/eventhorizon/pkg/apis \
  "eventhorizon:v1alpha1" \
  --output-base "${OUTPUT_BASE_DIR}"

cp -r ${OUTPUT_BASE_DIR}/acesso.io/eventhorizon/* ${PROJECT_ROOT}/.
```

Estrutura mínima

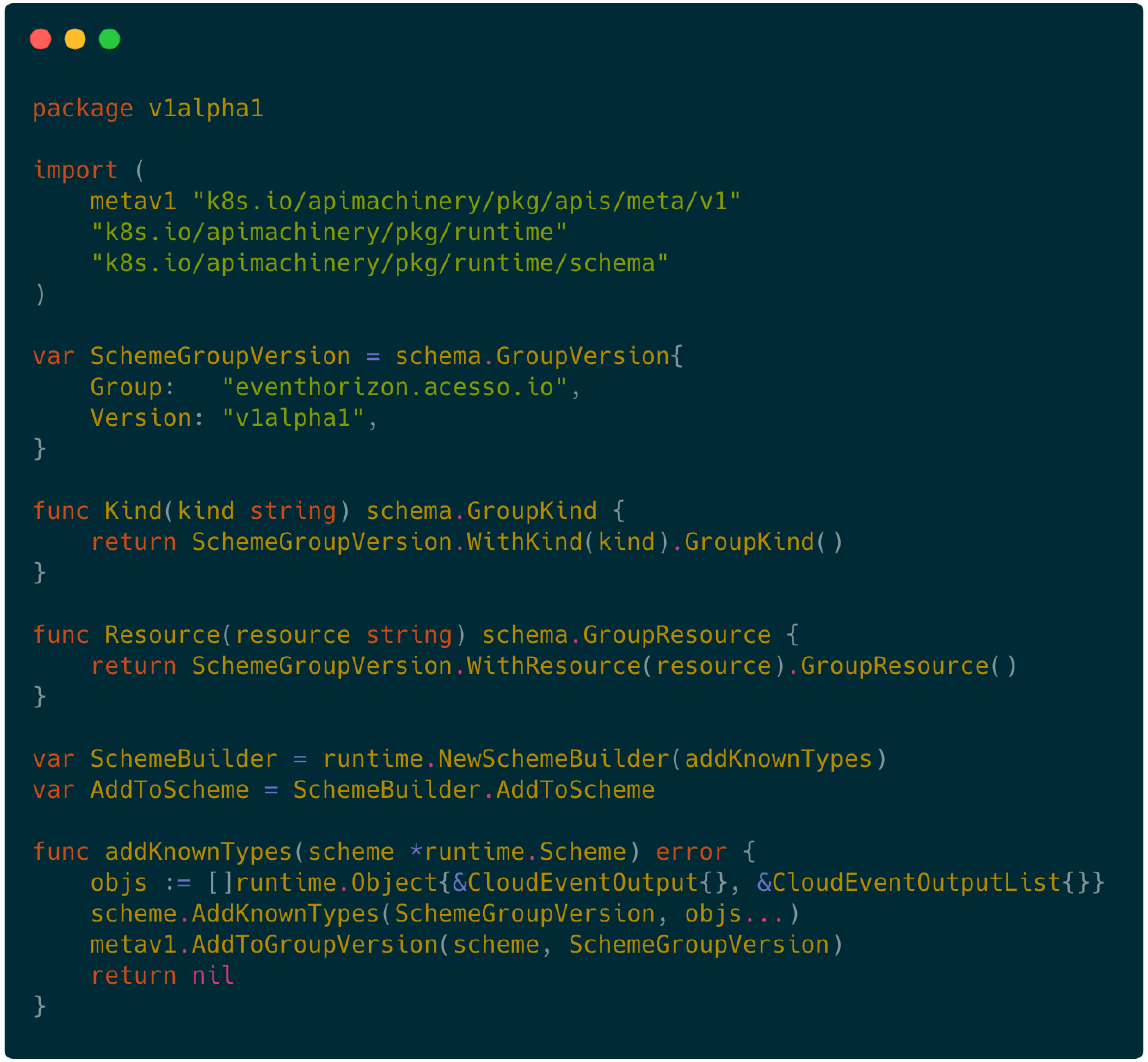
```
├── cmd
│   └── main.go
├── go.mod
├── hack
│   ├── tools.go
│   └── update-codegen.sh
├── pkg
│   ├── apis
│   │   └── eventhorizon
│   │       └── v1alpha1
│   │           ├── doc.go
│   │           ├── register.go
│   │           └── types_cloud_event_output.go
│   └── controller
│       └── controller.go
└── samples
    ├── CloudEventOutput.yml
    └── crds
        └── CloudEventOutput.yml
```



```go
// +k8s:deepcopy-gen=package,register
// +groupName=eventhorizon.acesso.io
package v1alpha1
```

Estrutura mínima

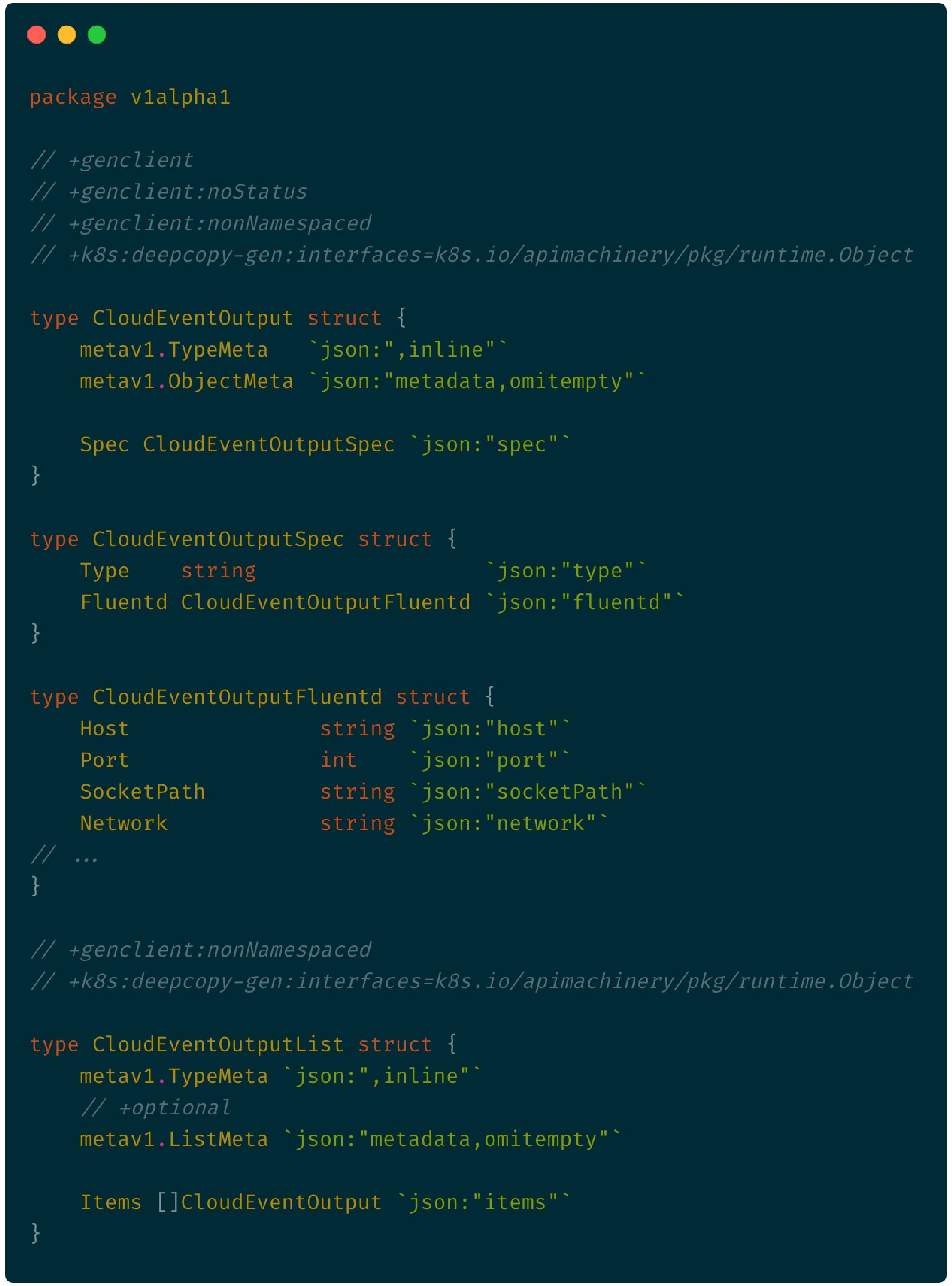
```
├── cmd
│   └── main.go
├── go.mod
├── hack
│   ├── tools.go
│   └── update-codegen.sh
├── pkg
│   ├── apis
│   │   └── eventhorizon
│   │       └── v1alpha1
│   │           ├── doc.go
│   │           ├── register.go
│   │           └── types_cloud_event_output.go
│   └── controller
│       └── controller.go
└── samples
    ├── CloudEventOutput.yml
    └── crds
        └── CloudEventOutput.yml
```



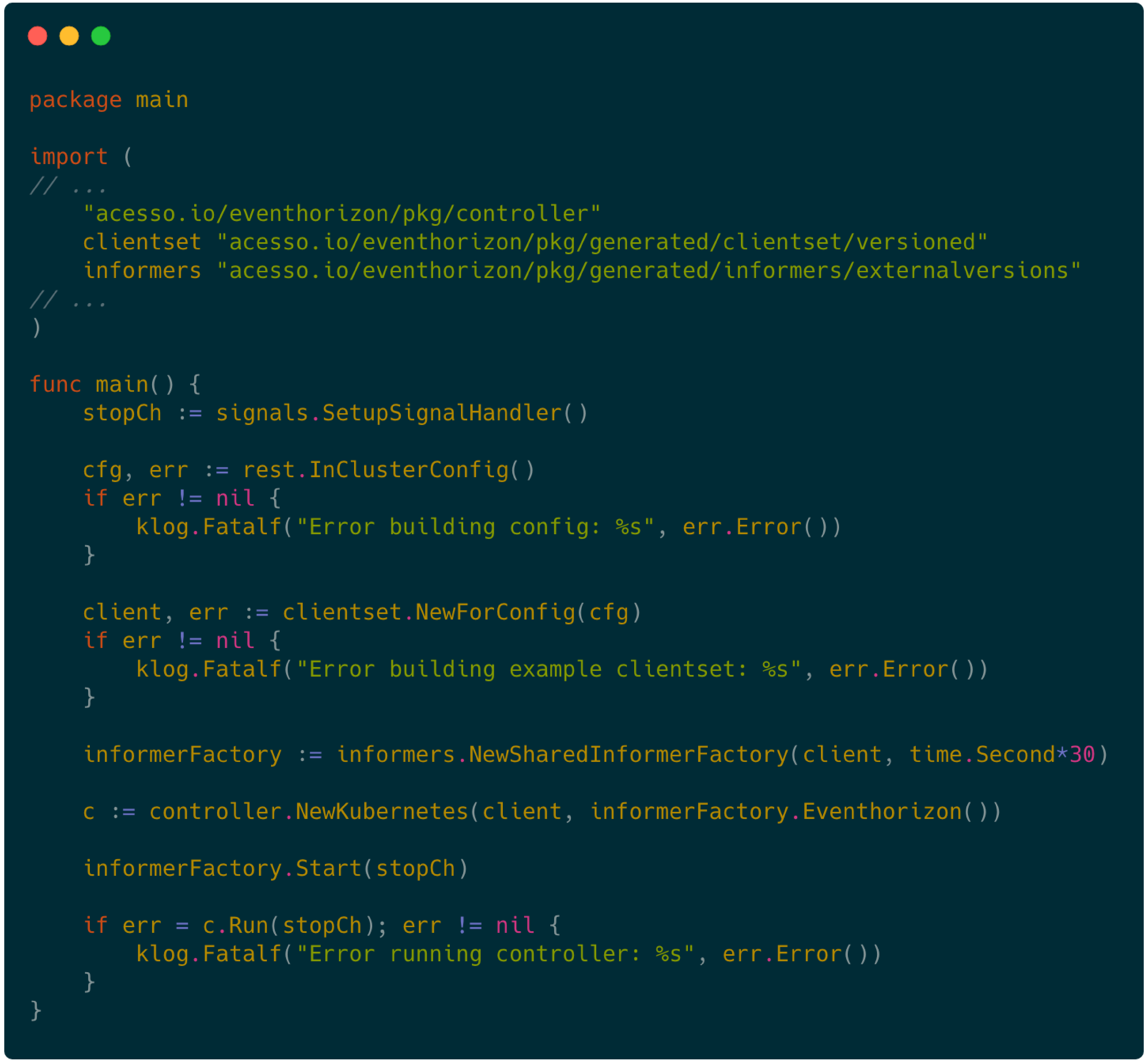
```go
package v1alpha1

import (
    metav1 "k8s.io/apimachinery/pkg/apis/meta/v1"
    "k8s.io/apimachinery/pkg/runtime"
    "k8s.io/apimachinery/pkg/runtime/schema"
)

var SchemeGroupVersion = schema.GroupVersion{
    Group:   "eventhorizon.acesso.io",
    Version: "v1alpha1",
}

func Kind(kind string) schema.GroupKind {
    return SchemeGroupVersion.WithKind(kind).GroupKind()
}

func Resource(resource string) schema.GroupResource {
    return SchemeGroupVersion.WithResource(resource).GroupResource()
}

var SchemeBuilder = runtime.NewSchemeBuilder(addKnownTypes)
var AddToScheme = SchemeBuilder.AddToScheme

func addKnownTypes(scheme *runtime.Scheme) error {
    objs := []runtime.Object{&CloudEventOutput{}, &CloudEventOutputList{}}
    scheme.AddKnownTypes(SchemeGroupVersion, objs...)
    metav1.AddToGroupVersion(scheme, SchemeGroupVersion)
    return nil
}
```

Estrutura mínima

```
├── cmd
│   └── main.go
├── go.mod
├── hack
│   ├── tools.go
│   └── update-codegen.sh
├── pkg
│   ├── apis
│   │   └── eventhorizon
│   │       └── v1alpha1
│   │           ├── doc.go
│   │           ├── register.go
│   │           └── types_cloud_event_output.go
│   └── controller
│       └── controller.go
└── samples
    ├── CloudEventOutput.yml
    └── crds
        └── CloudEventOutput.yml
```



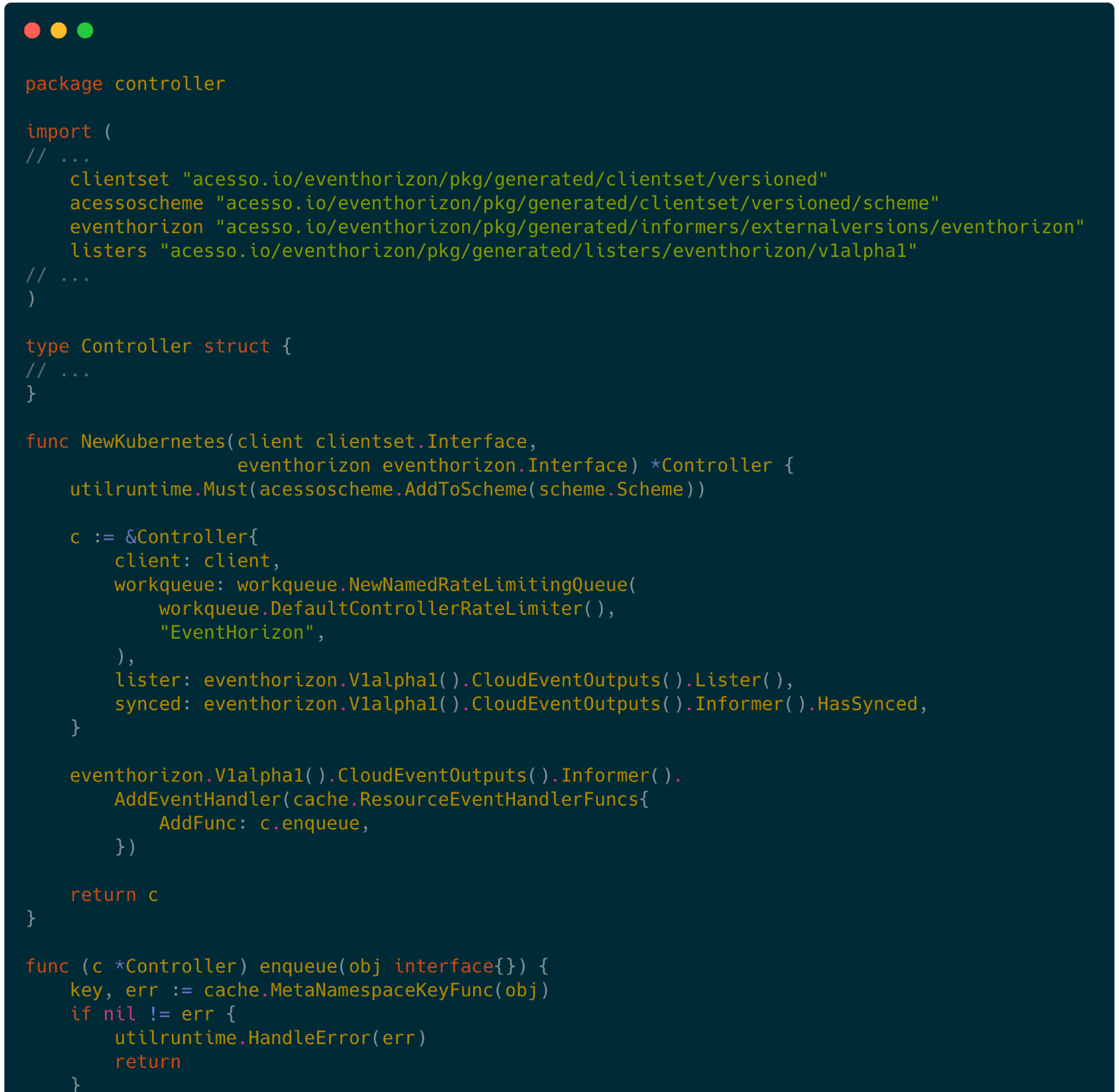
```go
package v1alpha1

// +genclient
// +genclient:noStatus
// +genclient:nonNamespaced
// +k8s:deepcopy-gen:interfaces=k8s.io/apimachinery/pkg/runtime.Object

type CloudEventOutput struct {
    metav1.TypeMeta   `json:",inline"`
    metav1.ObjectMeta `json:"metadata,omitempty"`

    Spec CloudEventOutputSpec `json:"spec"`
}

type CloudEventOutputSpec struct {
    Type    string                  `json:"type"`
    Fluentd CloudEventOutputFluentd `json:"fluentd"`
}

type CloudEventOutputFluentd struct {
    Host                string `json:"host"`
    Port                int    `json:"port"`
    SocketPath          string `json:"socketPath"`
    Network             string `json:"network"`
// ...
}

// +genclient:nonNamespaced
// +k8s:deepcopy-gen:interfaces=k8s.io/apimachinery/pkg/runtime.Object

type CloudEventOutputList struct {
    metav1.TypeMeta `json:",inline"`
    // +optional
    metav1.ListMeta `json:"metadata,omitempty"`

    Items []CloudEventOutput `json:"items"`
}
```



```go
package main

import (
// ...
    "acesso.io/eventhorizon/pkg/controller"
    clientset "acesso.io/eventhorizon/pkg/generated/clientset/versioned"
    informers "acesso.io/eventhorizon/pkg/generated/informers/externalversions"
// ...
)

func main() {
    stopCh := signals.SetupSignalHandler()

    cfg, err := rest.InClusterConfig()
    if err != nil {
        klog.Fatalf("Error building config: %s", err.Error())
    }

    client, err := clientset.NewForConfig(cfg)
    if err != nil {
        klog.Fatalf("Error building example clientset: %s", err.Error())
    }

    informerFactory := informers.NewSharedInformerFactory(client, time.Second*30)

    c := controller.NewKubernetes(client, informerFactory.Eventhorizon())

    informerFactory.Start(stopCh)

    if err = c.Run(stopCh); err != nil {
        klog.Fatalf("Error running controller: %s", err.Error())
    }
}
```

Estrutura aplicação

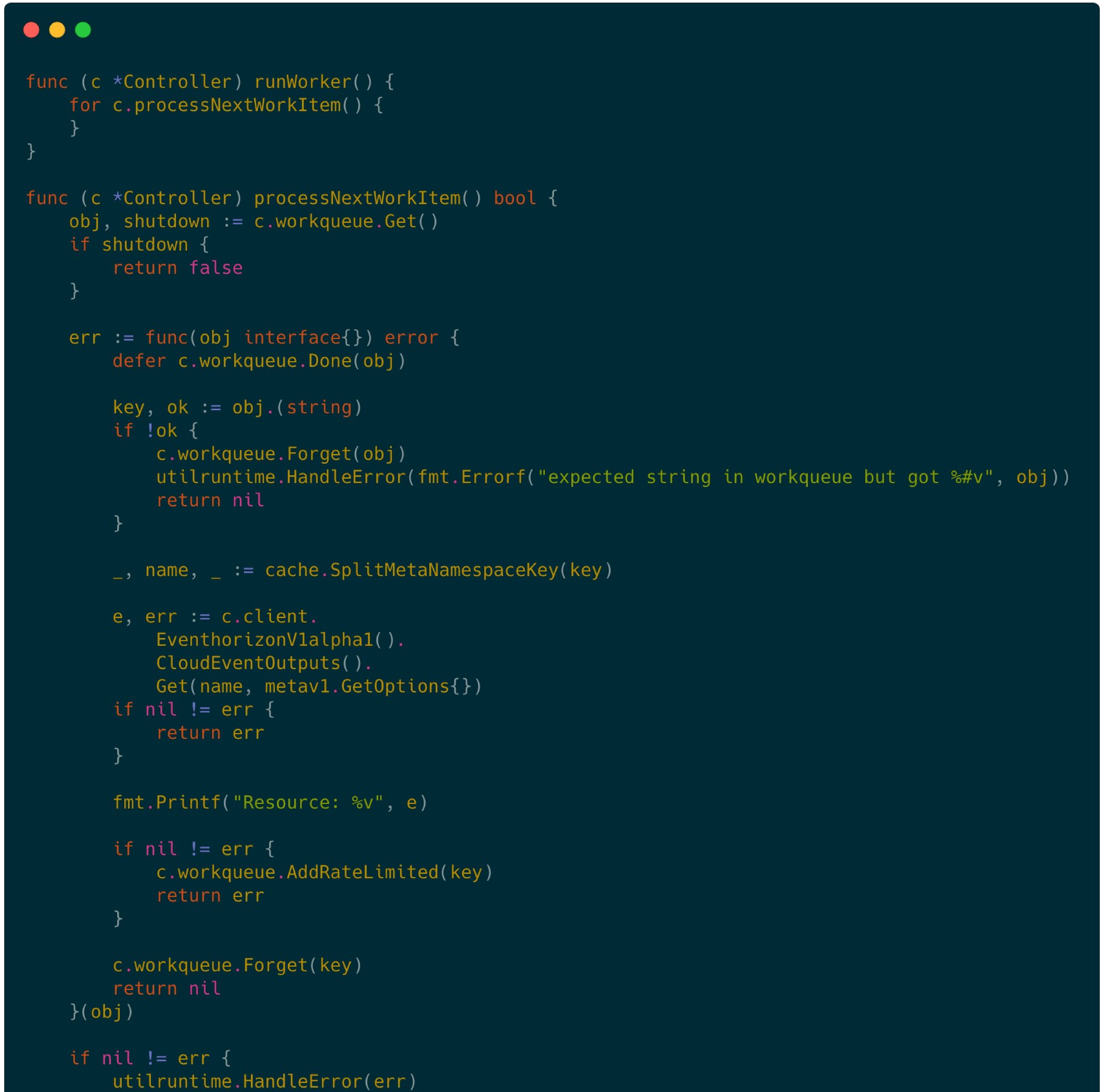
```
├── cmd
│   └── main.go
├── go.mod
├── hack
│   ├── tools.go
│   └── update-codegen.sh
├── pkg
│   ├── apis
│   │   └── eventhorizon
│   │       └── v1alpha1
│   │           ├── doc.go
│   │           ├── register.go
│   │           └── types_cloud_event_output.go
│   └── controller
│       └── controller.go
└── samples
    ├── CloudEventOutput.yml
    └── crds
        └── CloudEventOutput.yml
```



```go
package controller

import (
// ...
    clientset "acesso.io/eventhorizon/pkg/generated/clientset/versioned"
    acessoscheme "acesso.io/eventhorizon/pkg/generated/clientset/versioned/scheme"
    eventhorizon "acesso.io/eventhorizon/pkg/generated/informers/externalversions/eventhorizon"
    listers "acesso.io/eventhorizon/pkg/generated/listers/eventhorizon/v1alpha1"
// ...
)

type Controller struct {
// ...
}

func NewKubernetes(client clientset.Interface,
                   eventhorizon eventhorizon.Interface) *Controller {
    utilruntime.Must(acessoscheme.AddToScheme(scheme.Scheme))

    c := &Controller{
        client: client,
        workqueue: workqueue.NewNamedRateLimitingQueue(
            workqueue.DefaultControllerRateLimiter(),
            "EventHorizon",
        ),
        lister: eventhorizon.V1alpha1().CloudEventOutputs().Lister(),
        synced: eventhorizon.V1alpha1().CloudEventOutputs().Informer().HasSynced,
    }

    eventhorizon.V1alpha1().CloudEventOutputs().Informer().
        AddEventHandler(cache.ResourceEventHandlerFuncs{
            AddFunc: c.enqueue,
        })

    return c
}

func (c *Controller) enqueue(obj interface{}) {
    key, err := cache.MetaNamespaceKeyFunc(obj)
    if nil != err {
        utilruntime.HandleError(err)
        return
    }
```

Estrutura aplicação

```
├── cmd
│   └── main.go
├── go.mod
├── hack
│   ├── tools.go
│   └── update-codegen.sh
├── pkg
│   ├── apis
│   │   └── eventhorizon
│   │       └── v1alpha1
│   │           ├── doc.go
│   │           ├── register.go
│   │           └── types_cloud_event_output.go
│   └── controller
│       └── controller.go
└── samples
    ├── CloudEventOutput.yml
    └── crds
        └── CloudEventOutput.yml
```



```go
func (c *Controller) runWorker() {
    for c.processNextWorkItem() {
    }
}

func (c *Controller) processNextWorkItem() bool {
    obj, shutdown := c.workqueue.Get()
    if shutdown {
        return false
    }

    err := func(obj interface{}) error {
        defer c.workqueue.Done(obj)

        key, ok := obj.(string)
        if !ok {
            c.workqueue.Forget(obj)
            utilruntime.HandleError(fmt.Errorf("expected string in workqueue but got %#v", obj))
            return nil
        }

        _, name, _ := cache.SplitMetaNamespaceKey(key)

        e, err := c.client.
            EventhorizonV1alpha1().
            CloudEventOutputs().
            Get(name, metav1.GetOptions{})
        if nil != err {
            return err
        }

        fmt.Printf("Resource: %v", e)

        if nil != err {
            c.workqueue.AddRateLimited(key)
            return err
        }

        c.workqueue.Forget(key)
        return nil
    }(obj)

    if nil != err {
        utilruntime.HandleError(err)
```

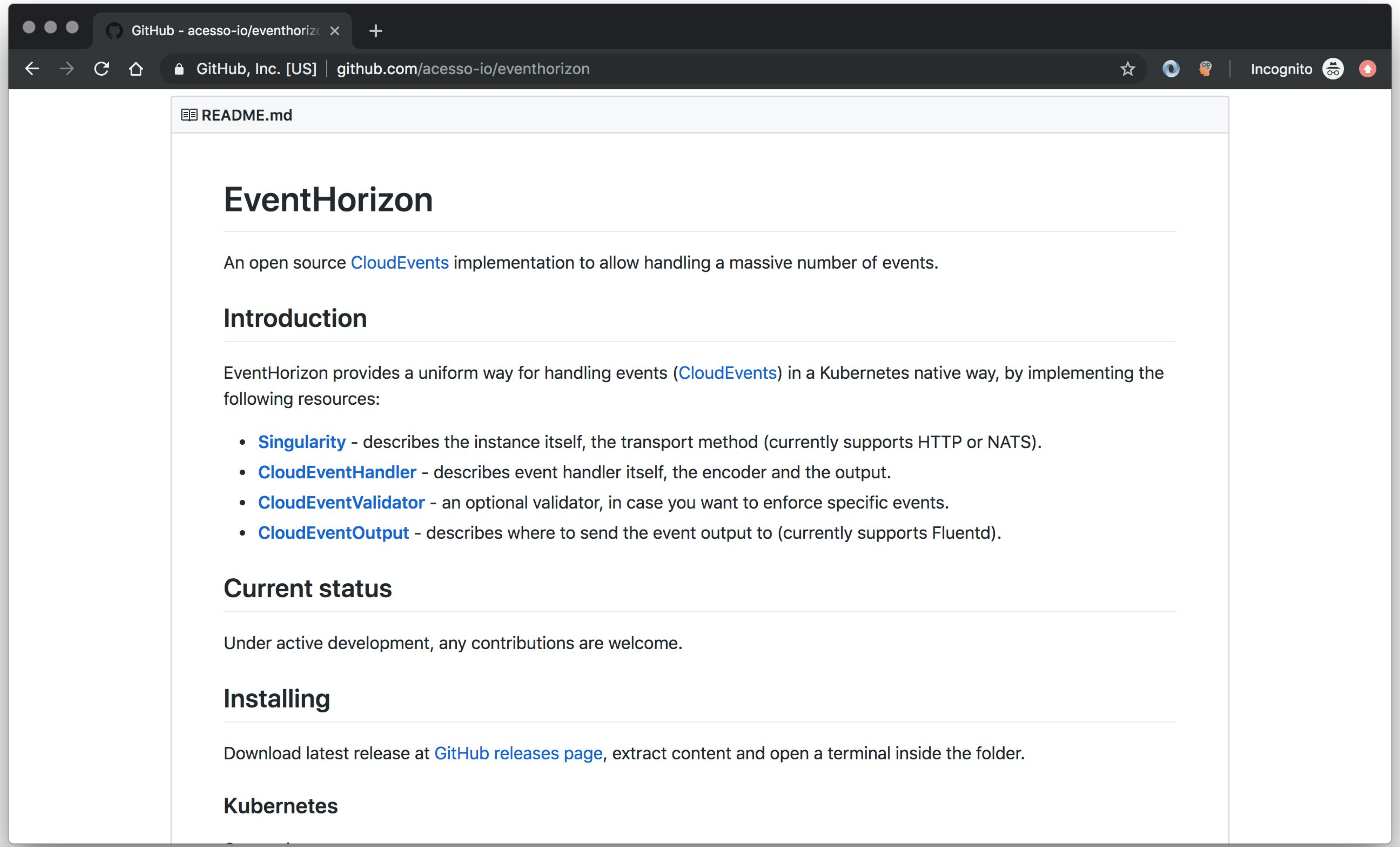
Estrutura aplicação

```
├── cmd
│   └── main.go
├── go.mod
├── hack
│   ├── tools.go
│   └── update-codegen.sh
├── pkg
│   ├── apis
│   │   └── eventhorizon
│   │       └── v1alpha1
│   │           ├── doc.go
│   │           ├── register.go
│   │           └── types_cloud_event_output.go
│   └── controller
│       └── controller.go
└── samples
    ├── CloudEventOutput.yml
    └── crds
        └── CloudEventOutput.yml
```

Exemplos de Uso

EventHorizon

https://github.com/acesso-io/eventhorizon


GitHub - acesso-io/eventhoriz
GitHub, Inc. [US] | github.com/acesso-io/eventhorizon
Incognito
README.md

EventHorizon

An open source CloudEvents implementation to allow handling a massive number of events.

Introduction

EventHorizon provides a uniform way for handling events (CloudEvents) in a Kubernetes native way, by implementing the following resources:

- **Singularity** - describes the instance itself, the transport method (currently supports HTTP or NATS).
- **CloudEventHandler** - describes event handler itself, the encoder and the output.
- **CloudEventValidator** - an optional validator, in case you want to enforce specific events.
- **CloudEventOutput** - describes where to send the event output to (currently supports Fluentd).

Current status

Under active development, any contributions are welcome.

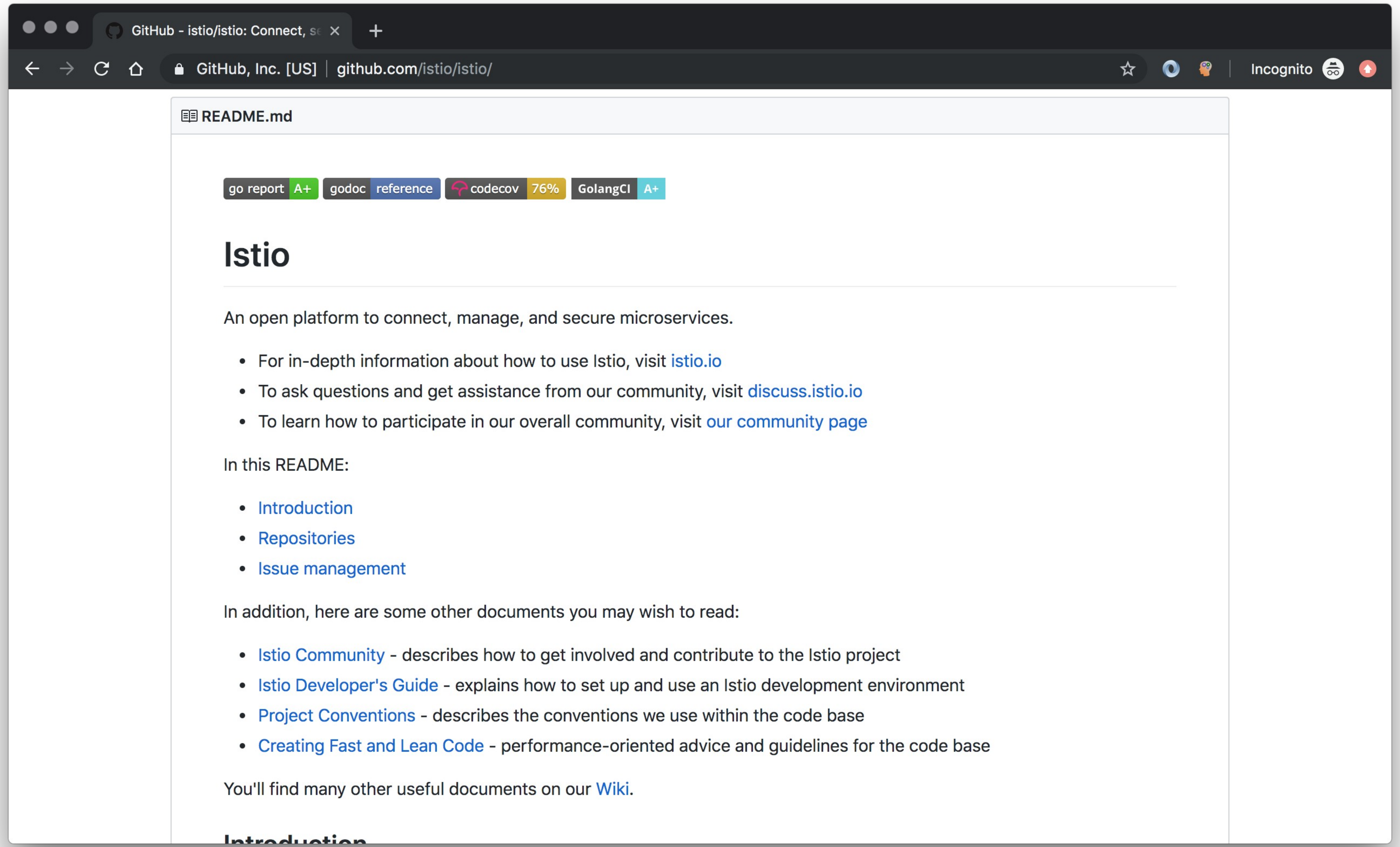
Installing

Download latest release at GitHub releases page, extract content and open a terminal inside the folder.

Kubernetes

Istio

https://github.com/istio/istio


GitHub - istio/istio: Connect, se
GitHub, Inc. [US] | github.com/istio/istio/
Incognito
README.md
go report A+
godoc reference
codecov 76%
GolangCI A+

Istio

An open platform to connect, manage, and secure microservices.

- For in-depth information about how to use Istio, visit istio.io
- To ask questions and get assistance from our community, visit discuss.istio.io
- To learn how to participate in our overall community, visit our community page

In this README:

- Introduction
- Repositories
- Issue management

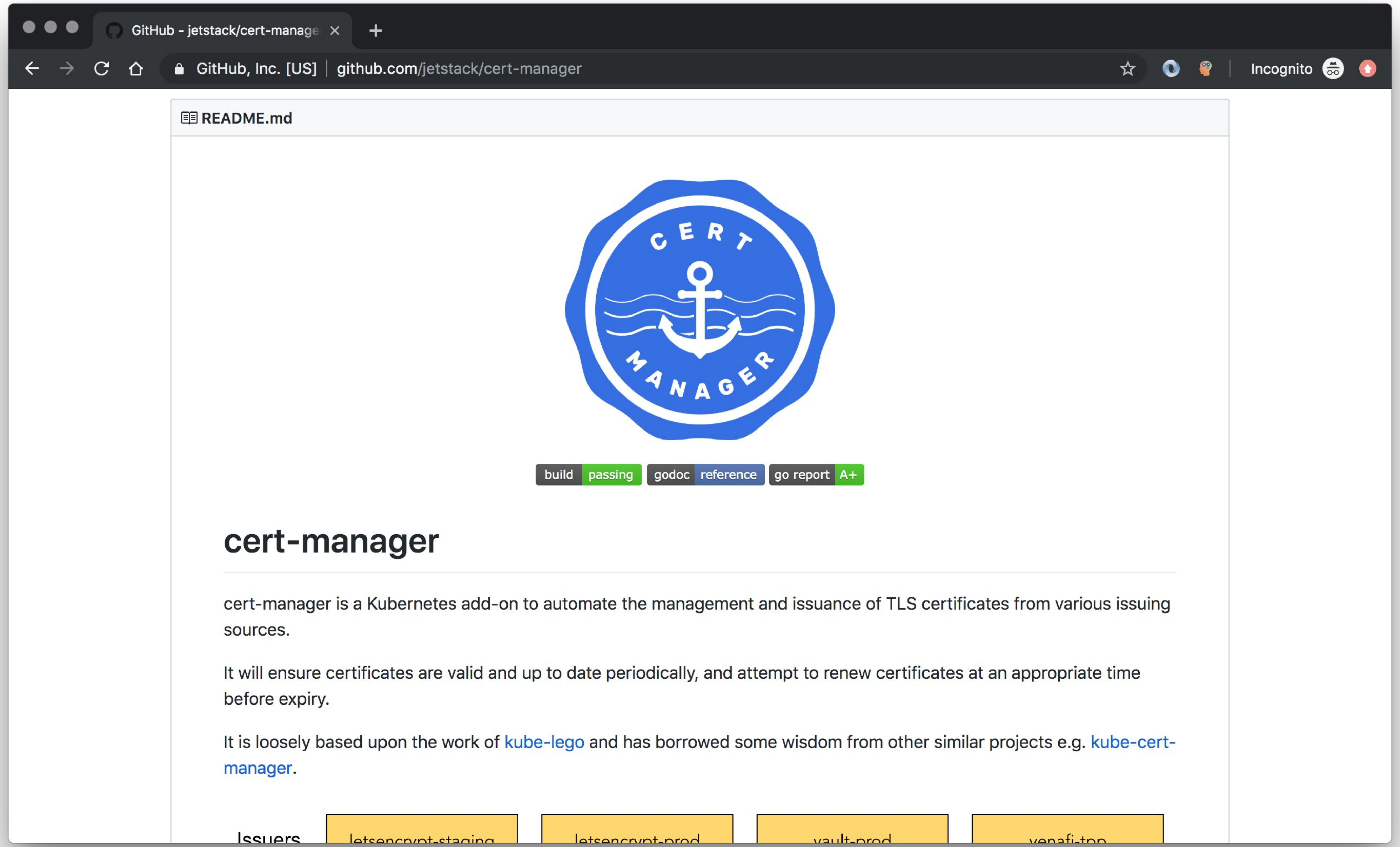
In addition, here are some other documents you may wish to read:

- Istio Community - describes how to get involved and contribute to the Istio project
- Istio Developer's Guide - explains how to set up and use an Istio development environment
- Project Conventions - describes the conventions we use within the code base
- Creating Fast and Lean Code - performance-oriented advice and guidelines for the code base

You'll find many other useful documents on our Wiki.

Cert-Manager

https://github.com/jetstack/cert-manager


GitHub - jetstack/cert-manage
GitHub, Inc. [US] | github.com/jetstack/cert-manager
Incognito
README.md
CERT MANAGER
build passing
godoc reference
go report A+

cert-manager

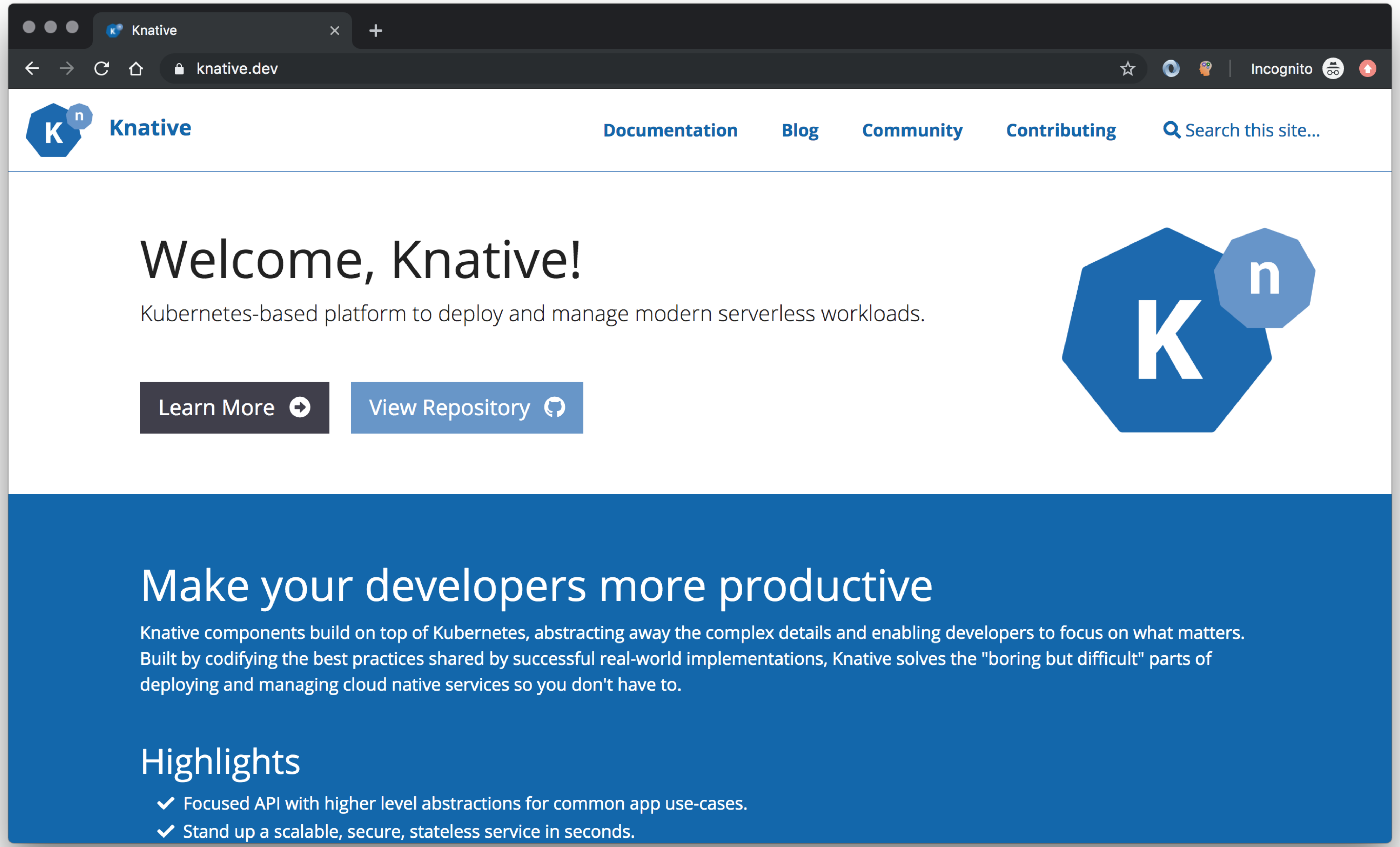
cert-manager is a Kubernetes add-on to automate the management and issuance of TLS certificates from various issuing sources.

It will ensure certificates are valid and up to date periodically, and attempt to renew certificates at an appropriate time before expiry.

It is loosely based upon the work of kube-lego and has borrowed some wisdom from other similar projects e.g. kube-cert-manager.

Knative

https://knative.dev/



Welcome, Knative!

Kubernetes-based platform to deploy and manage modern serverless workloads.

Make your developers more productive

Knative components build on top of Kubernetes, abstracting away the complex details and enabling developers to focus on what matters. Built by codifying the best practices shared by successful real-world implementations, Knative solves the "boring but difficult" parts of deploying and managing cloud native services so you don't have to.

Highlights

- ✓ Focused API with higher level abstractions for common app use-cases.
- ✓ Stand up a scalable, secure, stateless service in seconds.

Obrigado!

Diego Marangoni
@eusoudiego

github.com/acesso-io/meetup-k8s-crd-go