tech

Avito Golang meetup #4

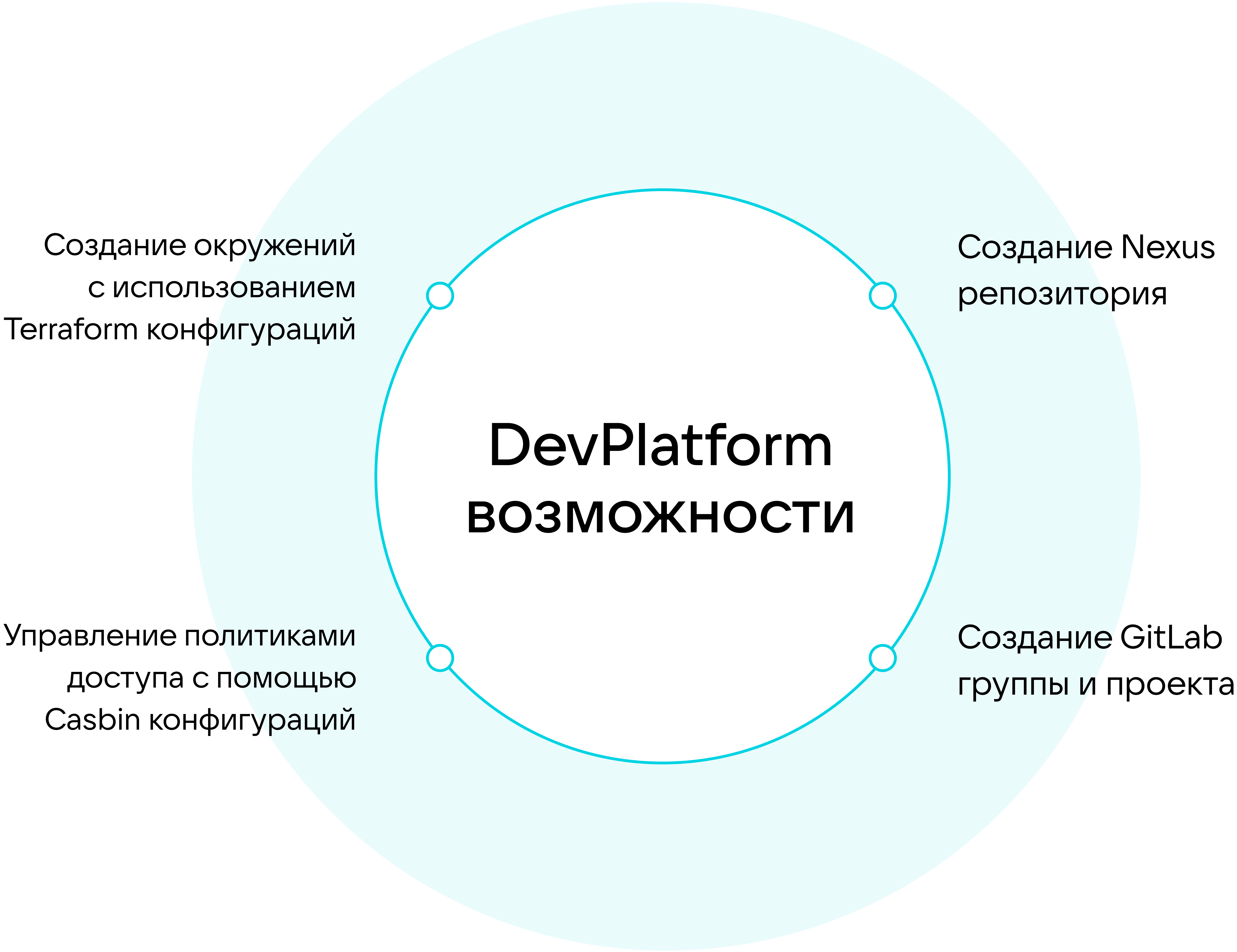
# Управление конфигурациями с помощью Kubernetes оператора в DevPlatform

Горячки Виктор,
backend разработчик

бизнес

DevPlatform служит для ускорения
и упрощения процесса разработки
программного продукта
за счет автоматизаций
и интеграций

# DevPlatform возможности



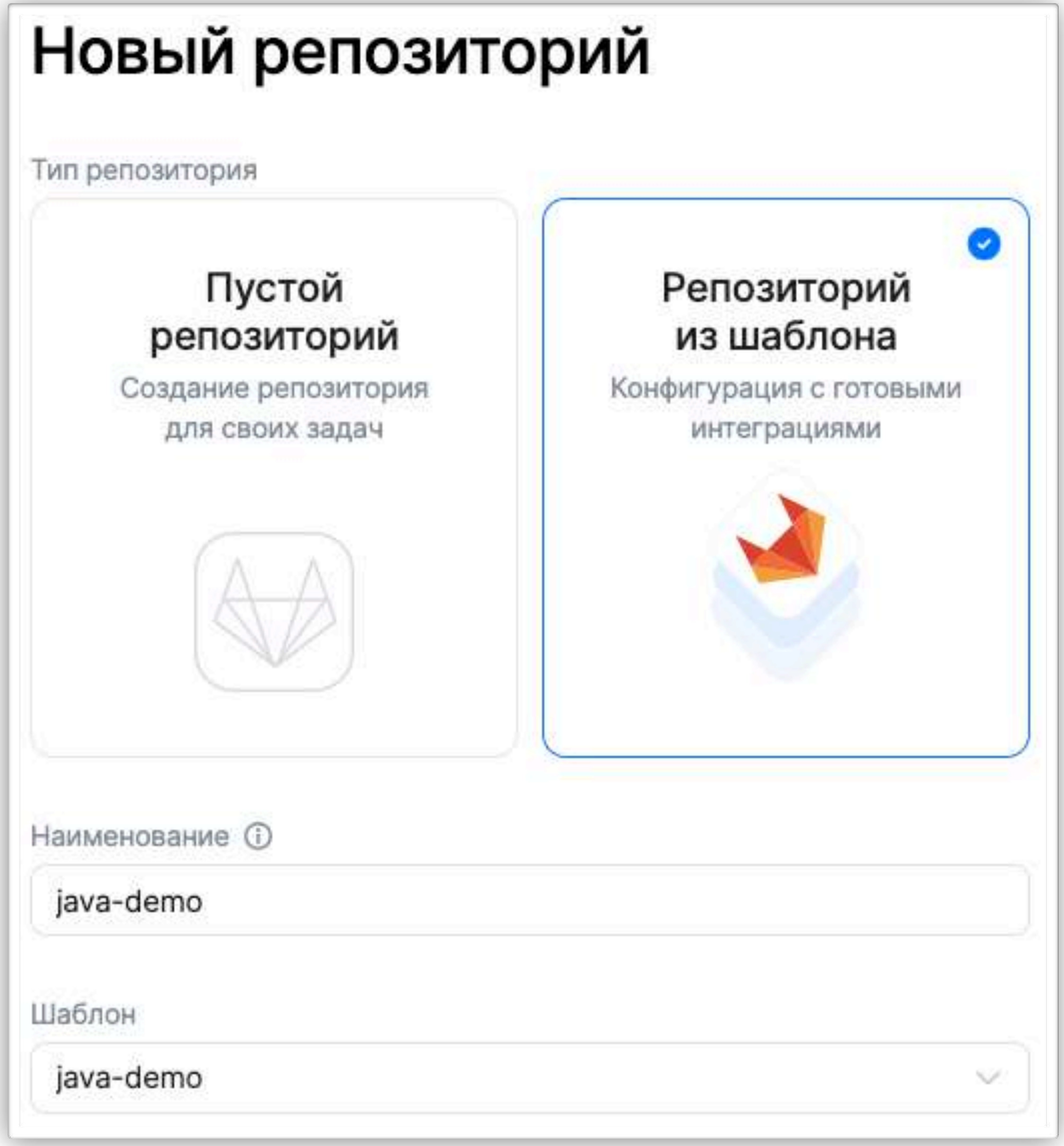
Создание окружений с использованием Terraform конфигураций

Создание Nexus репозитория

Управление политиками доступа с помощью Casbin конфигураций

Создание GitLab группы и проекта

# UI: GitLab репозитория



**Новый репозиторий**

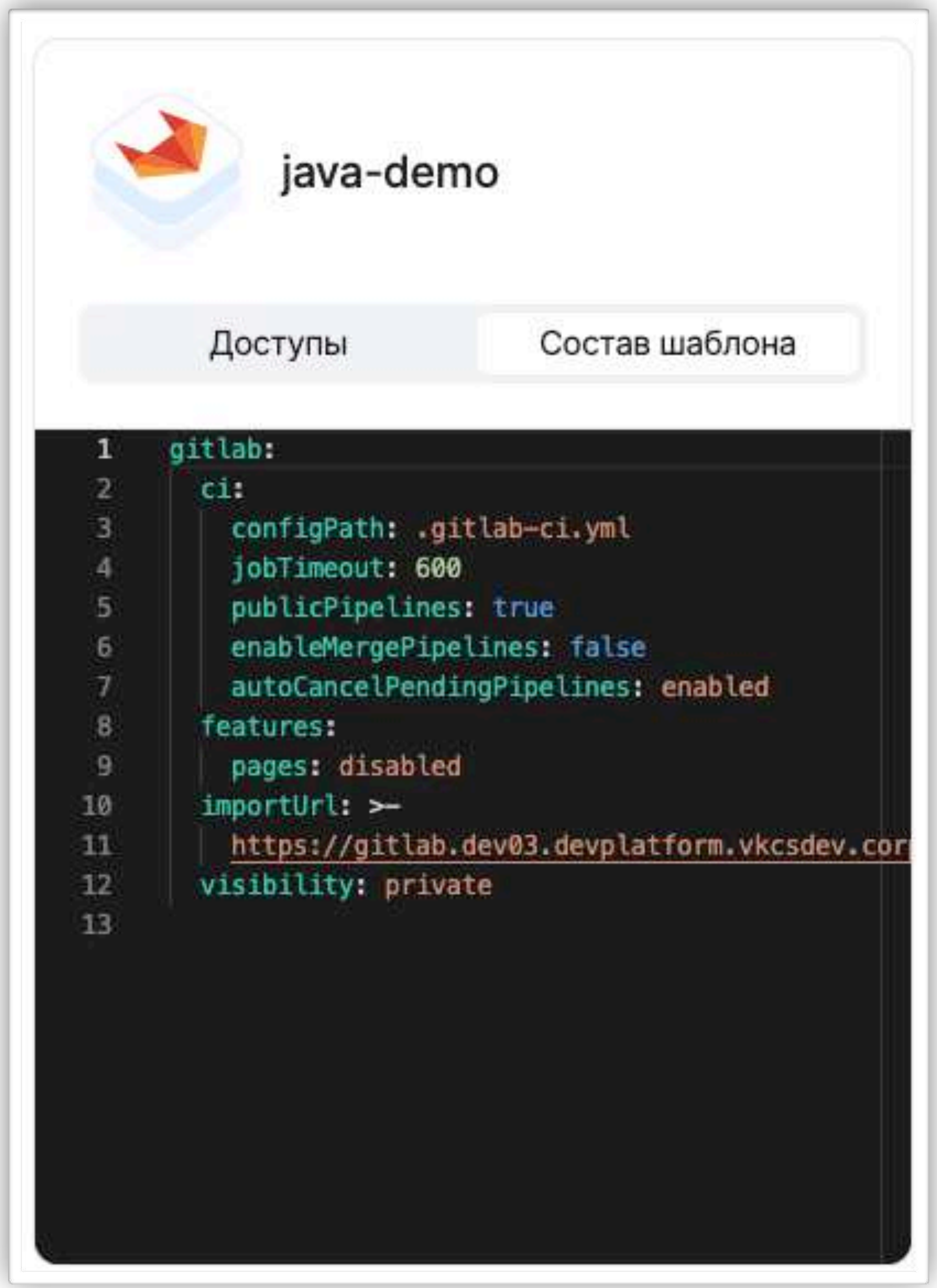
Тип репозитория

○ **Пустой репозиторий**
Создание репозитория для своих задач

● **Репозиторий из шаблона**
Конфигурация с готовыми интеграциями

Наименование: java-demo

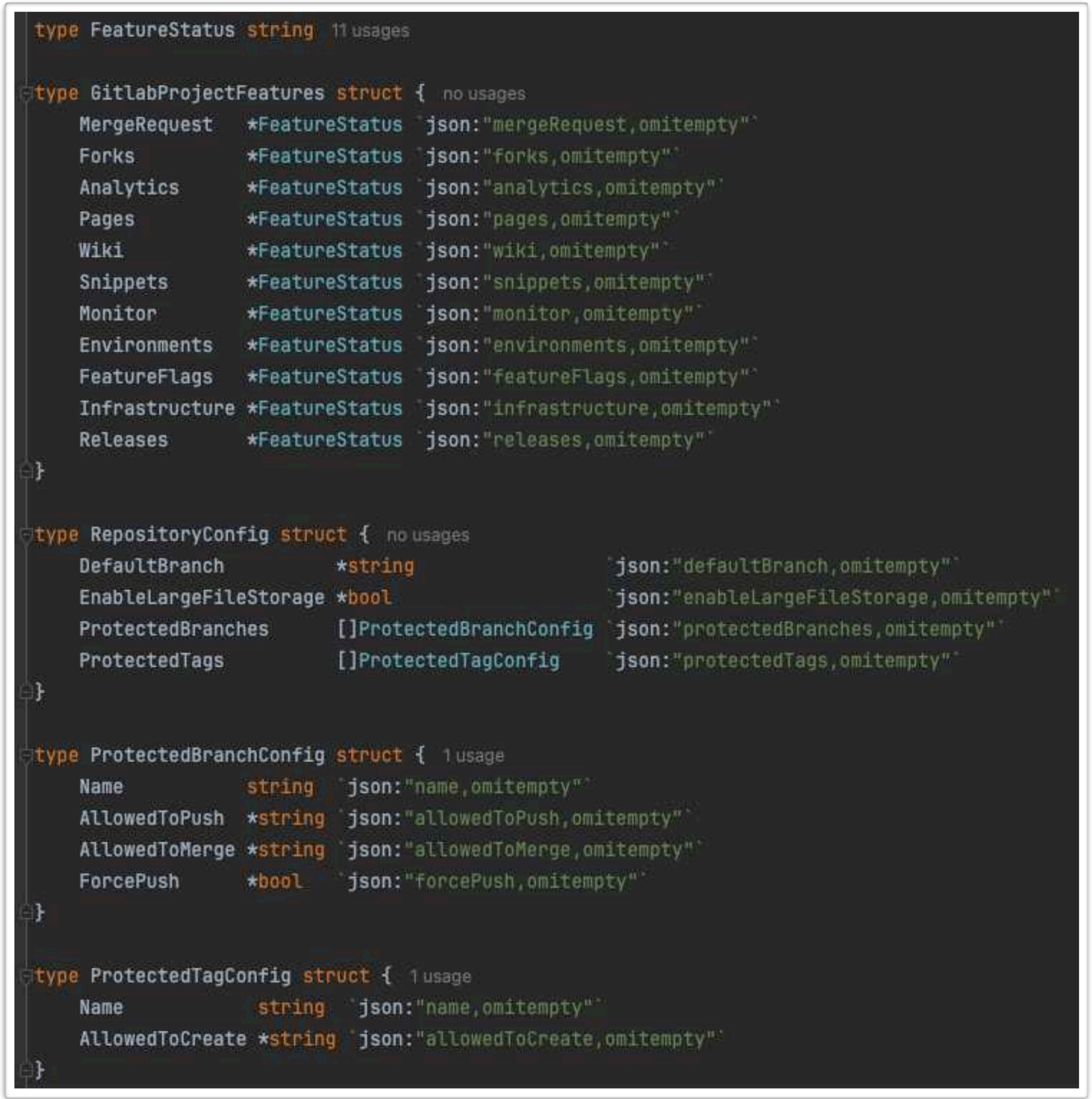
Шаблон: java-demo



java-demo

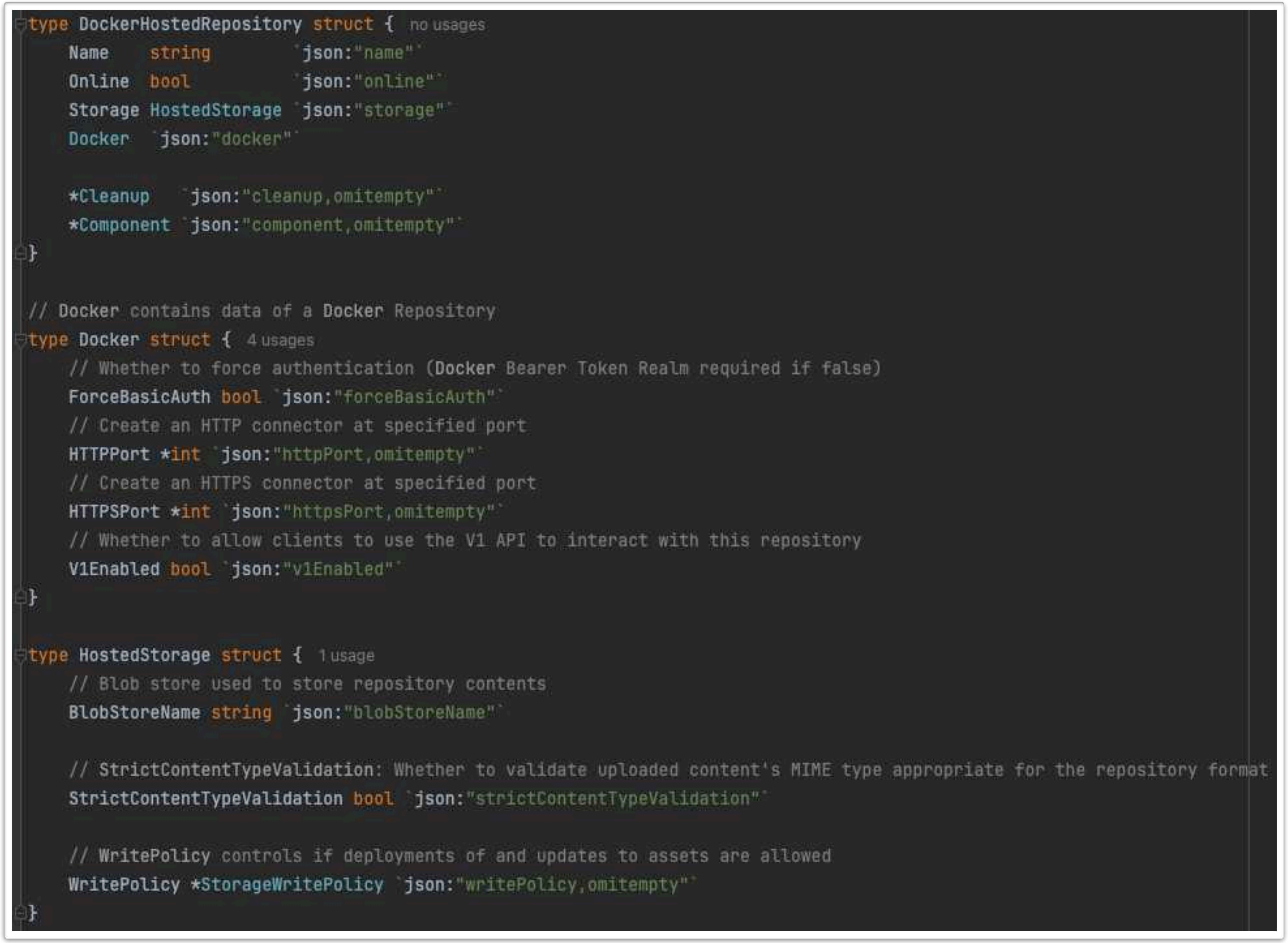
```yaml
gitlab:
  ci:
    configPath: .gitlab-ci.yml
    jobTimeout: 600
    publicPipelines: true
    enableMergePipelines: false
    autoCancelPendingPipelines: enabled
  features:
    pages: disabled
  importUrl: >-
    https://gitlab.dev03.devplatform.vkcsdev.cor
  visibility: private

```

# Конфигурация GitLab проекта



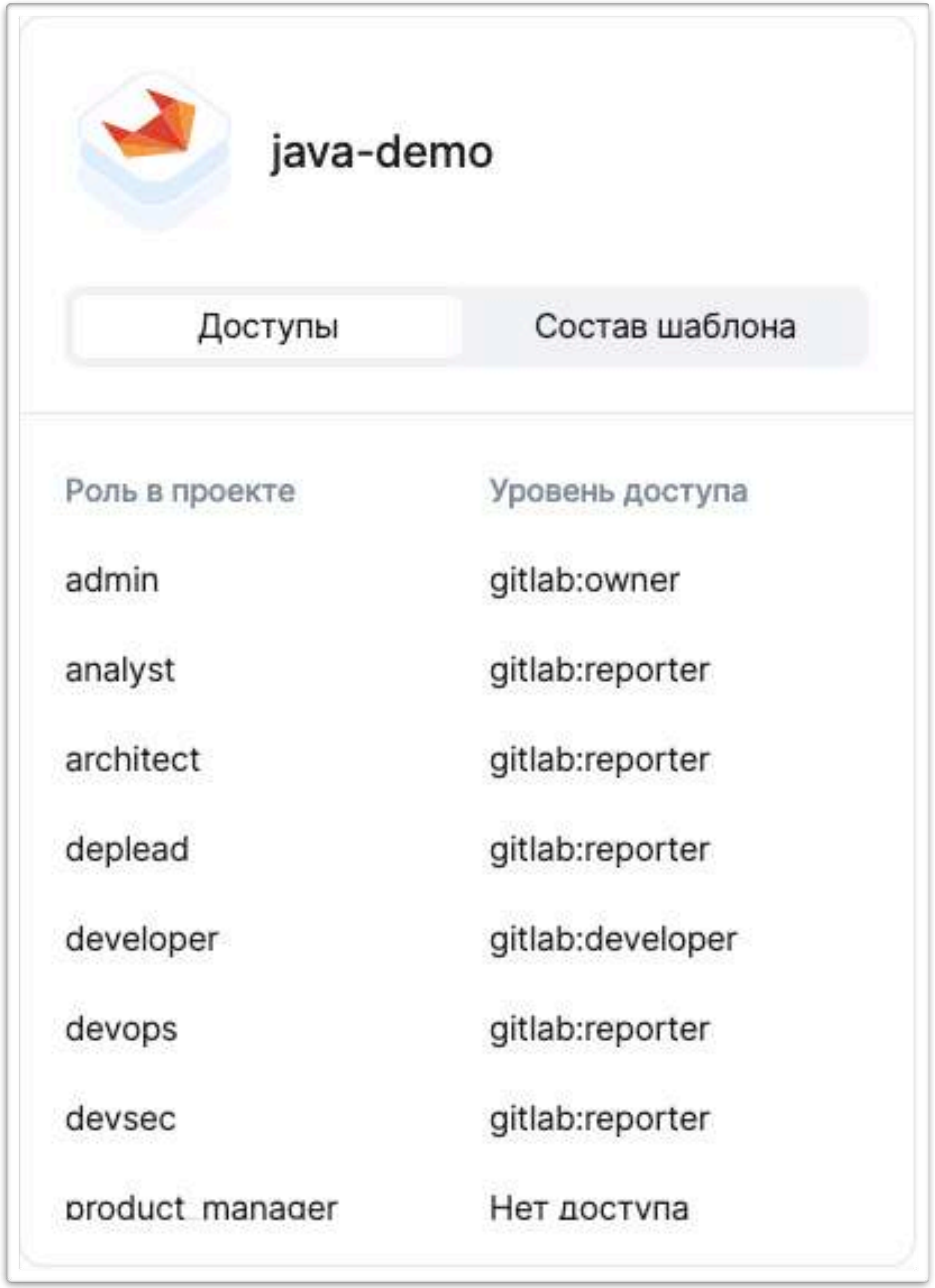
```go
type FeatureStatus string 11 usages

type GitlabProjectFeatures struct {  no usages
    MergeRequest   *FeatureStatus `json:"mergeRequest,omitempty"`
    Forks          *FeatureStatus `json:"forks,omitempty"`
    Analytics      *FeatureStatus `json:"analytics,omitempty"`
    Pages          *FeatureStatus `json:"pages,omitempty"`
    Wiki           *FeatureStatus `json:"wiki,omitempty"`
    Snippets       *FeatureStatus `json:"snippets,omitempty"`
    Monitor        *FeatureStatus `json:"monitor,omitempty"`
    Environments   *FeatureStatus `json:"environments,omitempty"`
    FeatureFlags   *FeatureStatus `json:"featureFlags,omitempty"`
    Infrastructure *FeatureStatus `json:"infrastructure,omitempty"`
    Releases       *FeatureStatus `json:"releases,omitempty"`
}

type RepositoryConfig struct {  no usages
    DefaultBranch          *string                 `json:"defaultBranch,omitempty"`
    EnableLargeFileStorage *bool                   `json:"enableLargeFileStorage,omitempty"`
    ProtectedBranches      []ProtectedBranchConfig `json:"protectedBranches,omitempty"`
    ProtectedTags          []ProtectedTagConfig    `json:"protectedTags,omitempty"`
}

type ProtectedBranchConfig struct {  1 usage
    Name          string  `json:"name,omitempty"`
    AllowedToPush *string `json:"allowedToPush,omitempty"`
    AllowedToMerge *string `json:"allowedToMerge,omitempty"`
    ForcePush     *bool   `json:"forcePush,omitempty"`
}

type ProtectedTagConfig struct {  1 usage
    Name            string  `json:"name,omitempty"`
    AllowedToCreate *string `json:"allowedToCreate,omitempty"`
}
```

# Конфигурации Nexus Docker репозитория



```
type DockerHostedRepository struct {  no usages
    Name    string        `json:"name"`
    Online  bool          `json:"online"`
    Storage HostedStorage `json:"storage"`
    Docker  `json:"docker"`

    *Cleanup   `json:"cleanup,omitempty"`
    *Component `json:"component,omitempty"`
}

// Docker contains data of a Docker Repository
type Docker struct {  4 usages
    // Whether to force authentication (Docker Bearer Token Realm required if false)
    ForceBasicAuth bool `json:"forceBasicAuth"`
    // Create an HTTP connector at specified port
    HTTPPort *int `json:"httpPort,omitempty"`
    // Create an HTTPS connector at specified port
    HTTPSPort *int `json:"httpsPort,omitempty"`
    // Whether to allow clients to use the V1 API to interact with this repository
    V1Enabled bool `json:"v1Enabled"`
}

type HostedStorage struct {  1 usage
    // Blob store used to store repository contents
    BlobStoreName string `json:"blobStoreName"`

    // StrictContentTypeValidation: Whether to validate uploaded content's MIME type appropriate for the repository format
    StrictContentTypeValidation bool `json:"strictContentTypeValidation"`

    // WritePolicy controls if deployments of and updates to assets are allowed
    WritePolicy *StorageWritePolicy `json:"writePolicy,omitempty"`
}
```

# UI: роли



| Роль в проекте | Уровень доступа |
|---|---|
| admin | gitlab:owner |
| analyst | gitlab:reporter |
| architect | gitlab:reporter |
| deplead | gitlab:reporter |
| developer | gitlab:developer |
| devops | gitlab:reporter |
| devsec | gitlab:reporter |
| product_manager | Нет доступа |

# Задачи

1

Хранение конфигураций

# Задачи

1

Хранение конфигураций

2

DRY подход в процессе создания конфигураций

# Задачи

1

Хранение конфигураций

2

DRY подход в процессе создания конфигураций

3

Валидация

# Задачи

1

Хранение конфигураций

2

DRY подход в процессе создания конфигураций

3

Валидация

4

Версионирование

# Задачи

1

Хранение конфигураций

2

DRY подход в процессе создания конфигураций

3

Валидация

4

Версионирование

5

Выполнение действий в случае изменения конфигураций

# Варианты решения

Разработать свое приложение

Использовать готовую технологию (Kubernetes operator + git)

# Определения и сокращения

CRD

Custom resource definition
определение (свойства)
пользовательского ресурса

# Определения и сокращения

CRD

Custom resource definition
определение (свойства)
пользовательского ресурса

CR

Custom resource
конкретный экземпляр
пользовательского ресурса

# Определения и сокращения

CRD

Custom resource definition
определение (свойства)
пользовательского ресурса

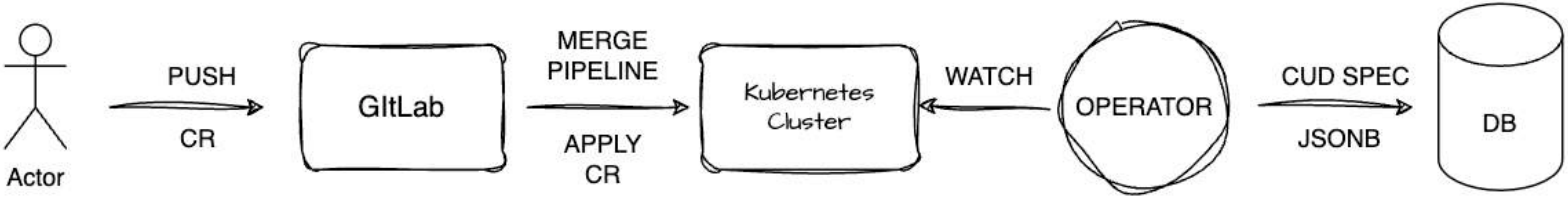
CR

Custom resource
конкретный экземпляр
пользовательского ресурса

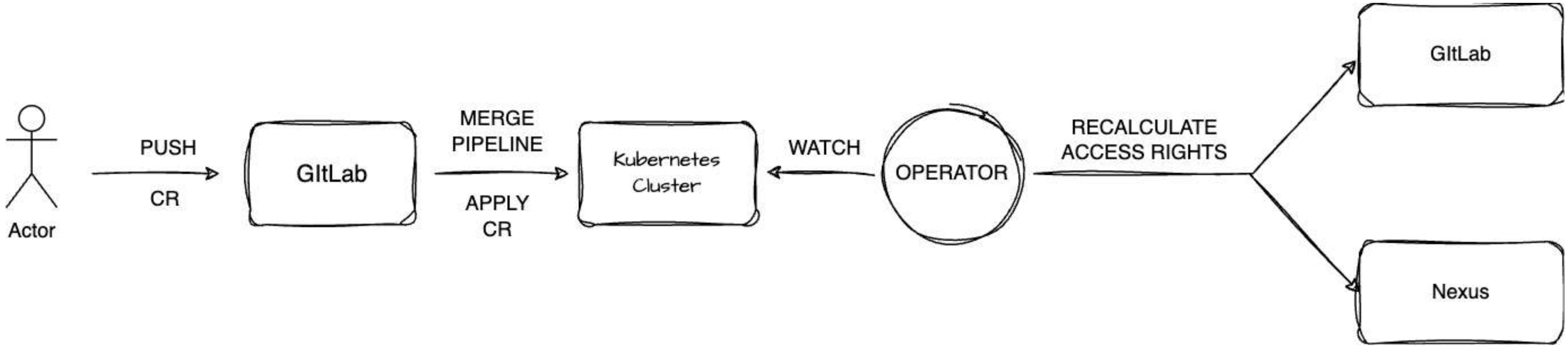
Operator

Operator
программа для управления
ресурсами на основе их целевого
описания (custom resources)

# Подход: Nexus и Gitlab конфигурации



# Подход: конфигурации политик casbin



# Инструменты для разработки оператора

- https://kubebuilder.io
  - Scaffolding проекта, api и т.д.

# Инструменты для разработки оператора

- https://kubebuilder.io
  - Scaffolding проекта, api и т.д.
- https://sdk.operatorframework.io
  - использует kubebuilder

# Инструменты для разработки оператора

- https://kubebuilder.io
  - Scaffolding проекта, api и т.д.
- https://sdk.operatorframework.io
  - использует kubebuilder
  - Operator Lifecycle Manager - https://olm.operatorframework.io

# Инструменты для разработки оператора

- https://kubebuilder.io
  - Scaffolding проекта, api и т.д.
- https://sdk.operatorframework.io
  - использует kubebuilder
  - Operator Lifecycle Manager - https://olm.operatorframework.io
  - Operator hub - https://operatorhub.io

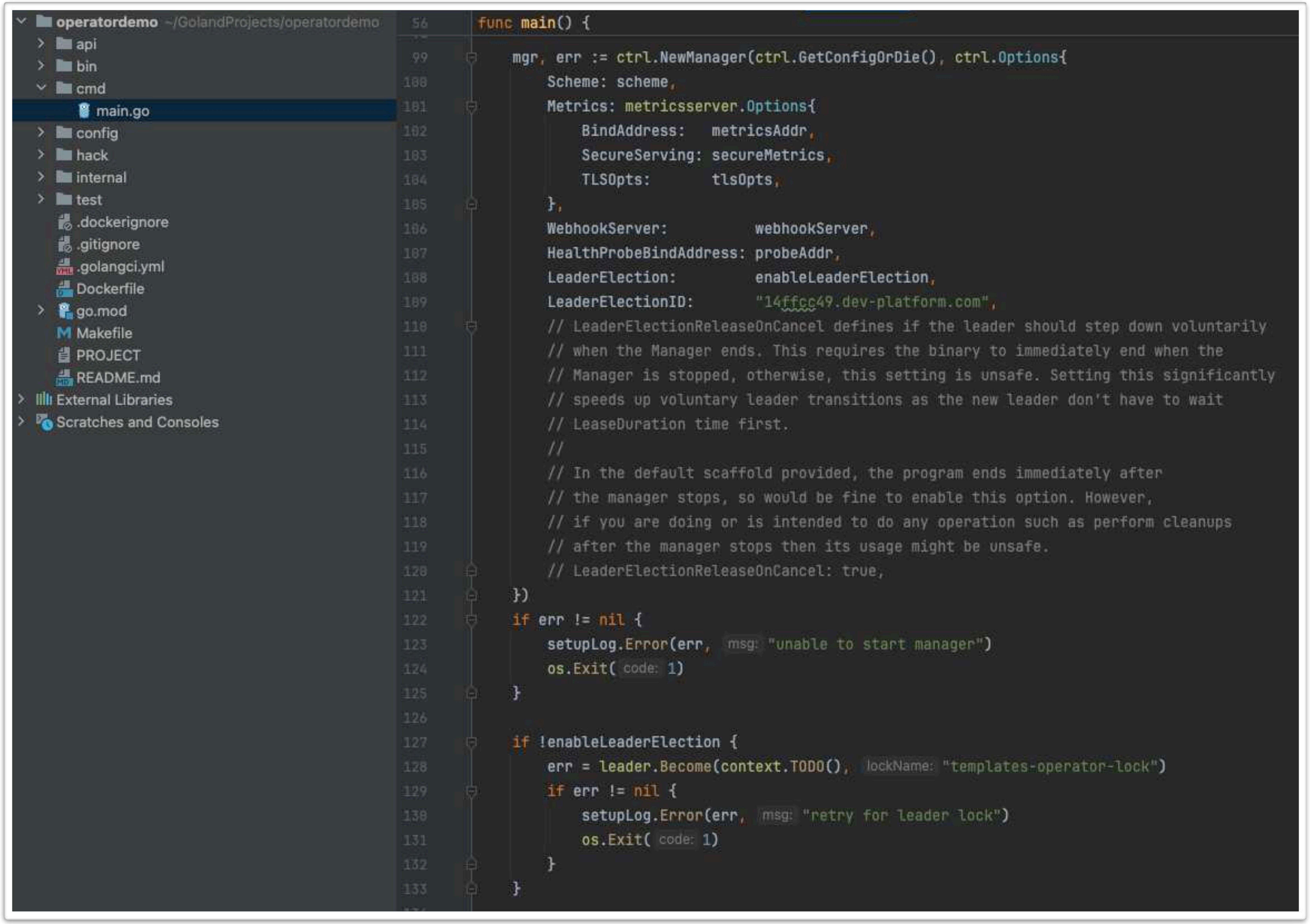
# Инструменты для разработки оператора

- https://kubebuilder.io
  - Scaffolding проекта, api и т.д.
- https://sdk.operatorframework.io
  - использует kubebuilder
  - Operator Lifecycle Manager - https://olm.operatorframework.io
  - Operator hub - https://operatorhub.io
  - Scorecard - тестирование оператора

# Инструменты для разработки оператора

- https://kubebuilder.io
  - Scaffolding проекта, api и т.д.
- https://sdk.operatorframework.io
  - использует kubebuilder
  - Operator Lifecycle Manager - https://olm.operatorframework.io
  - Operator hub - https://operatorhub.io
  - Scorecard - тестирование оператора
  - поддерживает ansible и helm операторы

# Инициализация проекта



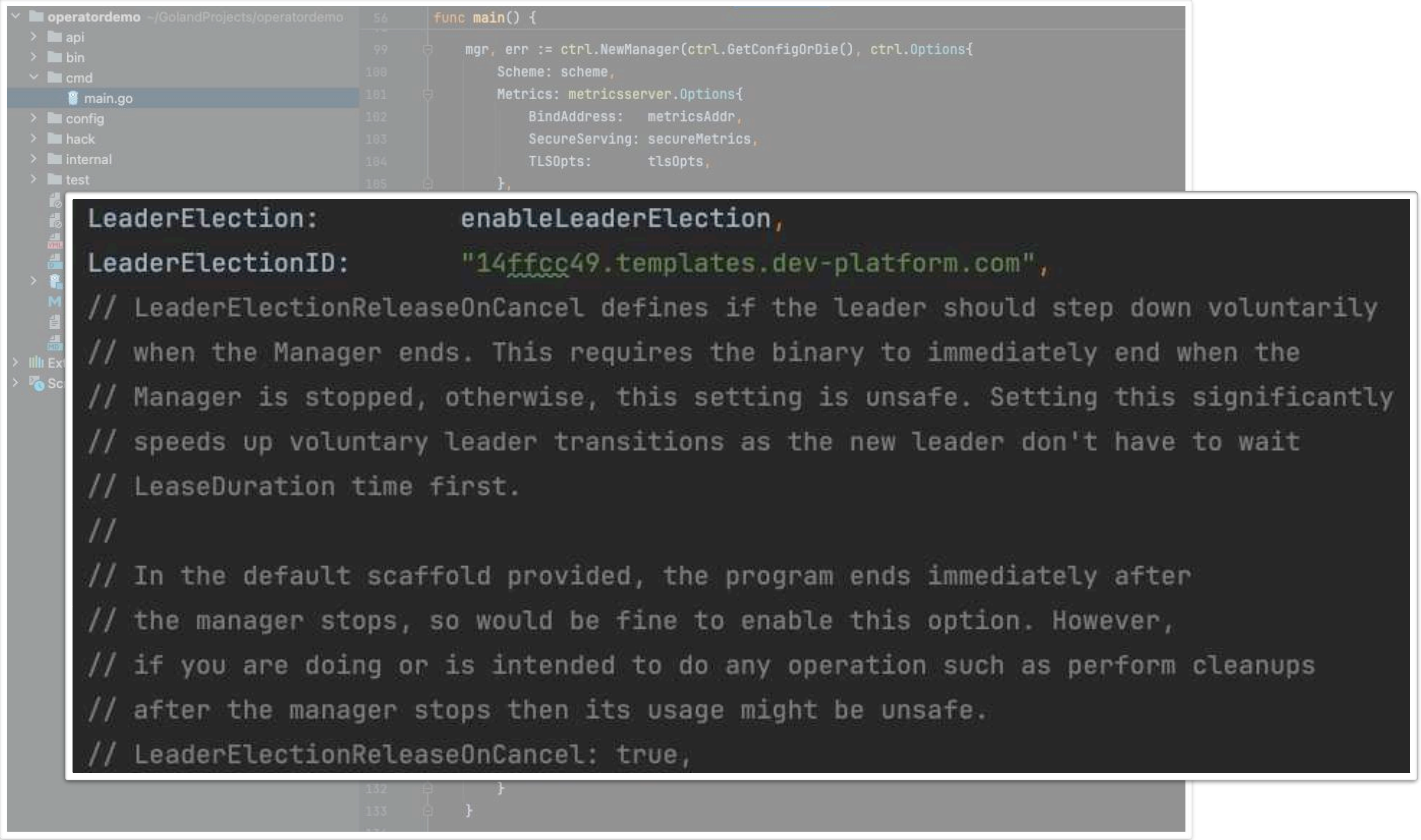
```go
func main() {
    mgr, err := ctrl.NewManager(ctrl.GetConfigOrDie(), ctrl.Options{
        Scheme: scheme,
        Metrics: metricsserver.Options{
            BindAddress:   metricsAddr,
            SecureServing: secureMetrics,
            TLSOpts:       tlsOpts,
        },
        WebhookServer:          webhookServer,
        HealthProbeBindAddress: probeAddr,
        LeaderElection:         enableLeaderElection,
        LeaderElectionID:       "14ffcc49.dev-platform.com",
        // LeaderElectionReleaseOnCancel defines if the leader should step down voluntarily
        // when the Manager ends. This requires the binary to immediately end when the
        // Manager is stopped, otherwise, this setting is unsafe. Setting this significantly
        // speeds up voluntary leader transitions as the new leader don't have to wait
        // LeaseDuration time first.
        //
        // In the default scaffold provided, the program ends immediately after
        // the manager stops, so would be fine to enable this option. However,
        // if you are doing or is intended to do any operation such as perform cleanups
        // after the manager stops then its usage might be unsafe.
        // LeaderElectionReleaseOnCancel: true,
    })
    if err != nil {
        setupLog.Error(err, msg: "unable to start manager")
        os.Exit( code: 1)
    }

    if !enableLeaderElection {
        err = leader.Become(context.TODO(), lockName: "templates-operator-lock")
        if err != nil {
            setupLog.Error(err, msg: "retry for leader lock")
            os.Exit( code: 1)
        }
    }
```

**operator-sdk init**

- -**domain** dev-platform.com

- -**repo** github.com/templates

# Leader election: leader-with-lease



```go
func main() {
    mgr, err := ctrl.NewManager(ctrl.GetConfigOrDie(), ctrl.Options{
        Scheme: scheme,
        Metrics: metricsserver.Options{
            BindAddress:   metricsAddr,
            SecureServing: secureMetrics,
            TLSOpts:       tlsOpts,
        },
```

```go
LeaderElection:          enableLeaderElection,
LeaderElectionID:        "14ffcc49.templates.dev-platform.com",
// LeaderElectionReleaseOnCancel defines if the leader should step down voluntarily
// when the Manager ends. This requires the binary to immediately end when the
// Manager is stopped, otherwise, this setting is unsafe. Setting this significantly
// speeds up voluntary leader transitions as the new leader don't have to wait
// LeaseDuration time first.
//
// In the default scaffold provided, the program ends immediately after
// the manager stops, so would be fine to enable this option. However,
// if you are doing or is intended to do any operation such as perform cleanups
// after the manager stops then its usage might be unsafe.
// LeaderElectionReleaseOnCancel: true,
```

```go
        }
    }
```

# Leader election: leader-for-life

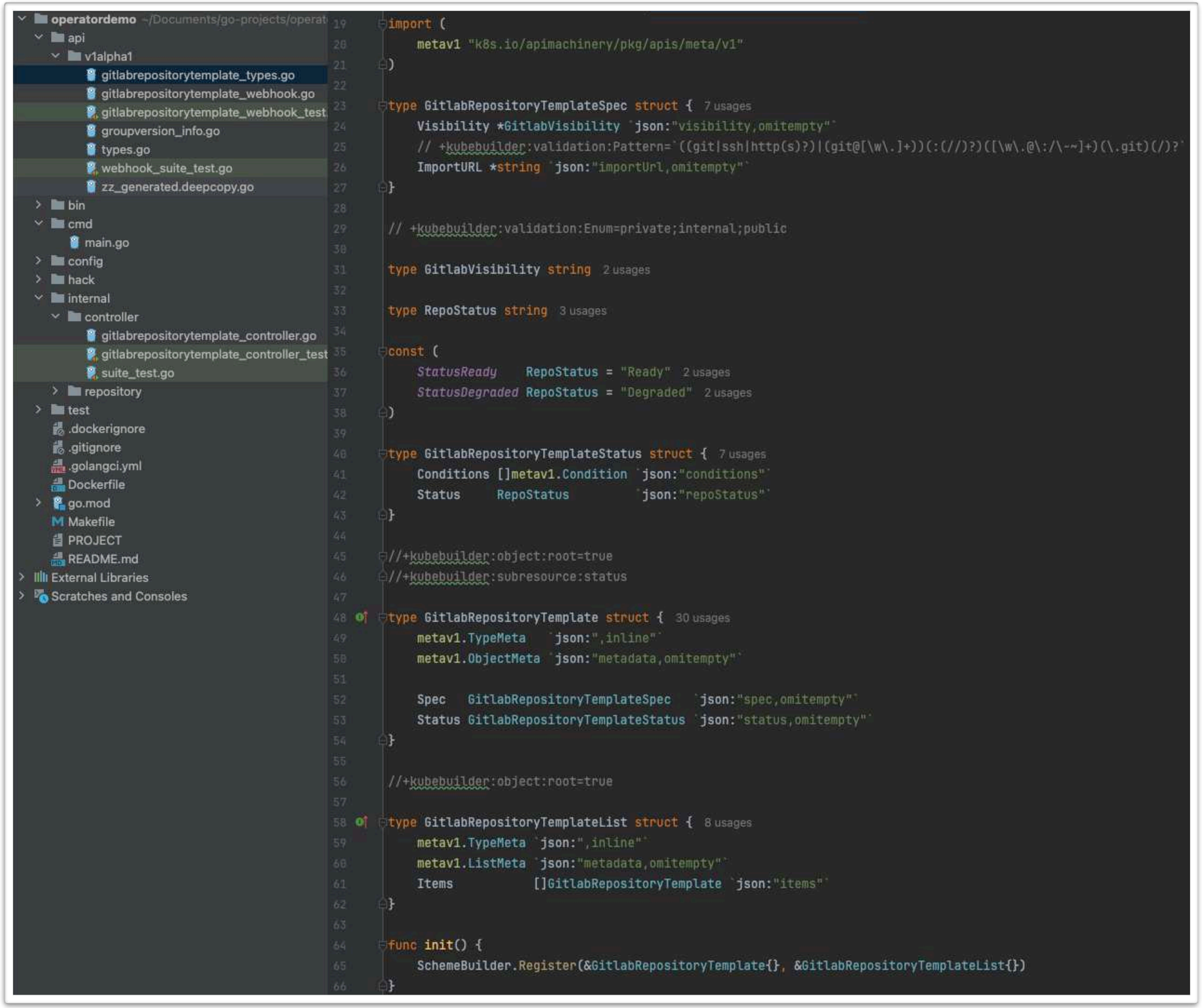
operatordemo ~/GolandProjects/operatordemo
api
bin
cmd
main.go
config
hack
internal
test
.dockerignore
.gitignore
.golangci.yml
Dockerfile
go.mod
Makefile
PROJECT

```go
func main() {
    mgr, err := ctrl.NewManager(ctrl.GetConfigOrDie(), ctrl.Options{
        Scheme: scheme,
        Metrics: metricsserver.Options{
            BindAddress:   metricsAddr,
            SecureServing: secureMetrics,
            TLSOpts:       tlsOpts,
        },
        WebhookServer:          webhookServer,
        HealthProbeBindAddress: probeAddr,
        LeaderElection:         enableLeaderElection,
        LeaderElectionID:       "14ffcc49.dev-platform.com",
        // LeaderElectionReleaseOnCancel defines if the leader should step down voluntarily
        // when the Manager ends. This requires the binary to immediately end when the
```

```go
err = leader.Become(context.TODO(), lockName: "templates-operator-lock")
if err != nil {
    setupLog.Error(err, msg: "retry for leader lock")
    os.Exit( code: 1)
}
```

```go
        err = leader.Become(context.TODO(), lockName: "templates-operator-lock")
        if err != nil {
            setupLog.Error(err, msg: "retry for leader lock")
            os.Exit( code: 1)
        }
    }
```

# Создание ресурсов и контроллера



```
operatordemo ~/Documents/go-projects/operat
  api
    v1alpha1
      gitlabrepositorytemplate_types.go
      gitlabrepositorytemplate_webhook.go
      gitlabrepositorytemplate_webhook_test.
      groupversion_info.go
      types.go
      webhook_suite_test.go
      zz_generated.deepcopy.go
  bin
  cmd
    main.go
  config
  hack
  internal
    controller
      gitlabrepositorytemplate_controller.go
      gitlabrepositorytemplate_controller_test
      suite_test.go
    repository
  test
  .dockerignore
  .gitignore
  .golangci.yml
  Dockerfile
  go.mod
  Makefile
  PROJECT
  README.md
External Libraries
Scratches and Consoles
```

```go
import (
    metav1 "k8s.io/apimachinery/pkg/apis/meta/v1"
)

type GitlabRepositoryTemplateSpec struct {  7 usages
    Visibility *GitlabVisibility `json:"visibility,omitempty"`
    // +kubebuilder:validation:Pattern=`((git|ssh|http(s)?)|(git@[\w\.]+))(:(//)?)([\w\.@\:/\-~]+)(\.git)(/)?`
    ImportURL *string `json:"importUrl,omitempty"`
}

// +kubebuilder:validation:Enum=private;internal;public

type GitlabVisibility string  2 usages

type RepoStatus string  3 usages

const (
    StatusReady    RepoStatus = "Ready"  2 usages
    StatusDegraded RepoStatus = "Degraded"  2 usages
)

type GitlabRepositoryTemplateStatus struct {  7 usages
    Conditions []metav1.Condition `json:"conditions"`
    Status     RepoStatus         `json:"repoStatus"`
}

//+kubebuilder:object:root=true
//+kubebuilder:subresource:status

type GitlabRepositoryTemplate struct {  30 usages
    metav1.TypeMeta   `json:",inline"`
    metav1.ObjectMeta `json:"metadata,omitempty"`

    Spec   GitlabRepositoryTemplateSpec   `json:"spec,omitempty"`
    Status GitlabRepositoryTemplateStatus `json:"status,omitempty"`
}

//+kubebuilder:object:root=true

type GitlabRepositoryTemplateList struct {  8 usages
    metav1.TypeMeta `json:",inline"`
    metav1.ListMeta `json:"metadata,omitempty"`
    Items           []GitlabRepositoryTemplate `json:"items"`
}

func init() {
    SchemeBuilder.Register(&GitlabRepositoryTemplate{}, &GitlabRepositoryTemplateList{})
}
```

**operator-sdk create api**

- -**group** templates

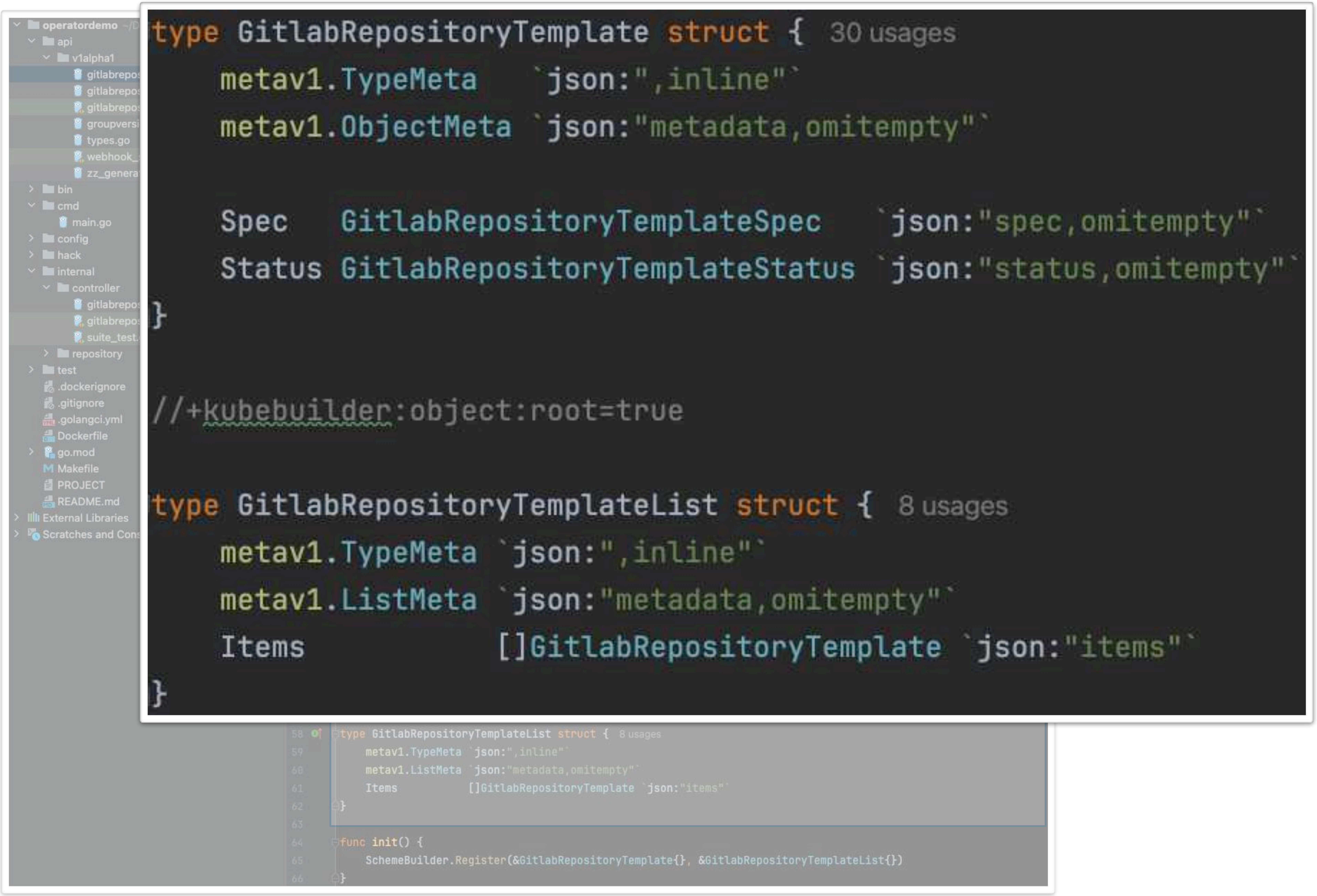
- -**version** v1alpha1

- -**kind** GitlabRepositoryTemplate

- -**resource**

- -**controller**

# Структура CRD



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

    Spec   GitlabRepositoryTemplateSpec   `json:"spec,omitempty"`
    Status GitlabRepositoryTemplateStatus `json:"status,omitempty"`
}

//+kubebuilder:object:root=true

type GitlabRepositoryTemplateList struct {
    metav1.TypeMeta `json:",inline"`
    metav1.ListMeta `json:"metadata,omitempty"`
    Items           []GitlabRepositoryTemplate `json:"items"`
}
```

```go
type GitlabRepositoryTemplateList struct {
    metav1.TypeMeta `json:",inline"`
    metav1.ListMeta `json:"metadata,omitempty"`
    Items           []GitlabRepositoryTemplate `json:"items"`
}
```

```go
func init() {
    SchemeBuilder.Register(&GitlabRepositoryTemplate{}, &GitlabRepositoryTemplateList{})
}
```

# CRD spec

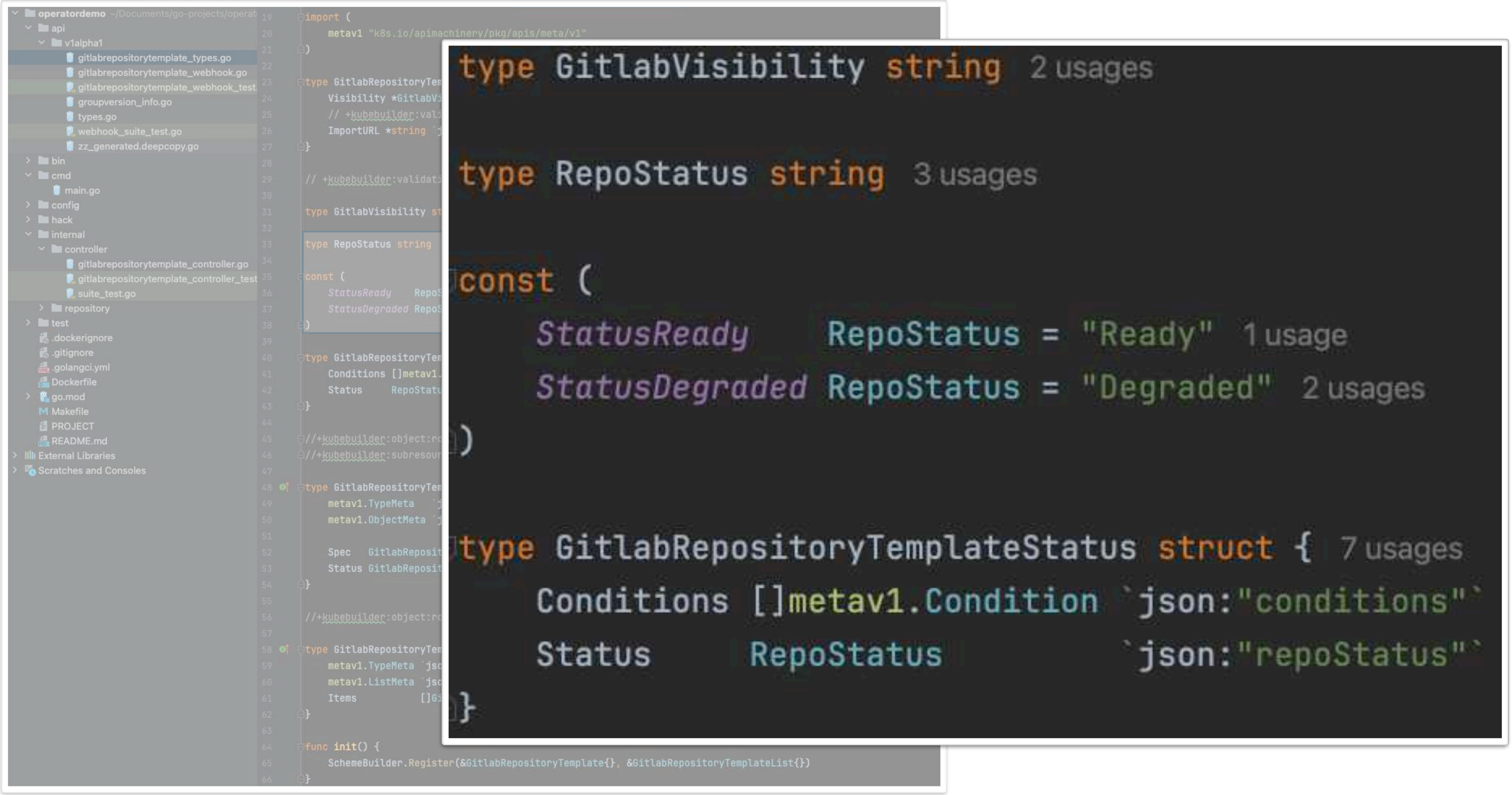
operatordemo ~/Documents/go-projects/operat
api
v1alpha1
gitlabrepositorytemplate_types.go
gitlabrepositorytemplate_webhook.go
gitlabrepositorytemplate_webhook_test
groupversion_info.go
types.go
webhook_suite_test.go
zz_generated.deepcopy.go
bin
cmd
main.go
config
hack

```go
import (
    metav1 "k8s.io/apimachinery/pkg/apis/meta/v1"
)

type GitlabRepositoryTemplateSpec struct {  7 usages
    Visibility *GitlabVisibility `json:"visibility,omitempty"`
    // +kubebuilder:validation:Pattern=`((git|ssh|http(s)?)|(git@[\w\.]+))(:(//)?)([\w\.@\:/\-~]+)(\.git)(/)?`
    ImportURL *string `json:"importUrl,omitempty"`
}

// +kubebuilder:validation:Enum=private;internal;public

type GitlabVisibility string  2 usages
```

```go
type GitlabRepositoryTemplateSpec struct {  7 usages
    Visibility *GitlabVisibility `json:"visibility,omitempty"`
    // +kubebuilder:validation:Pattern=`((git|ssh|http(s)?)|(git@[\w\.]+))(:(//)?)([\w\.@\:/\-~]+)(\.git)(/)?`
    ImportURL *string `json:"importUrl,omitempty"`
}

// +kubebuilder:validation:Enum=private;internal;public

type GitlabVisibility string  2 usages
```

```go

//+kubebuilder:object:root=true

type GitlabRepositoryTemplateList struct {  8 usages
    metav1.TypeMeta `json:",inline"`
    metav1.ListMeta `json:"metadata,omitempty"`
    Items           []GitlabRepositoryTemplate `json:"items"`
}

func init() {
    SchemeBuilder.Register(&GitlabRepositoryTemplate{}, &GitlabRepositoryTemplateList{})
}
```

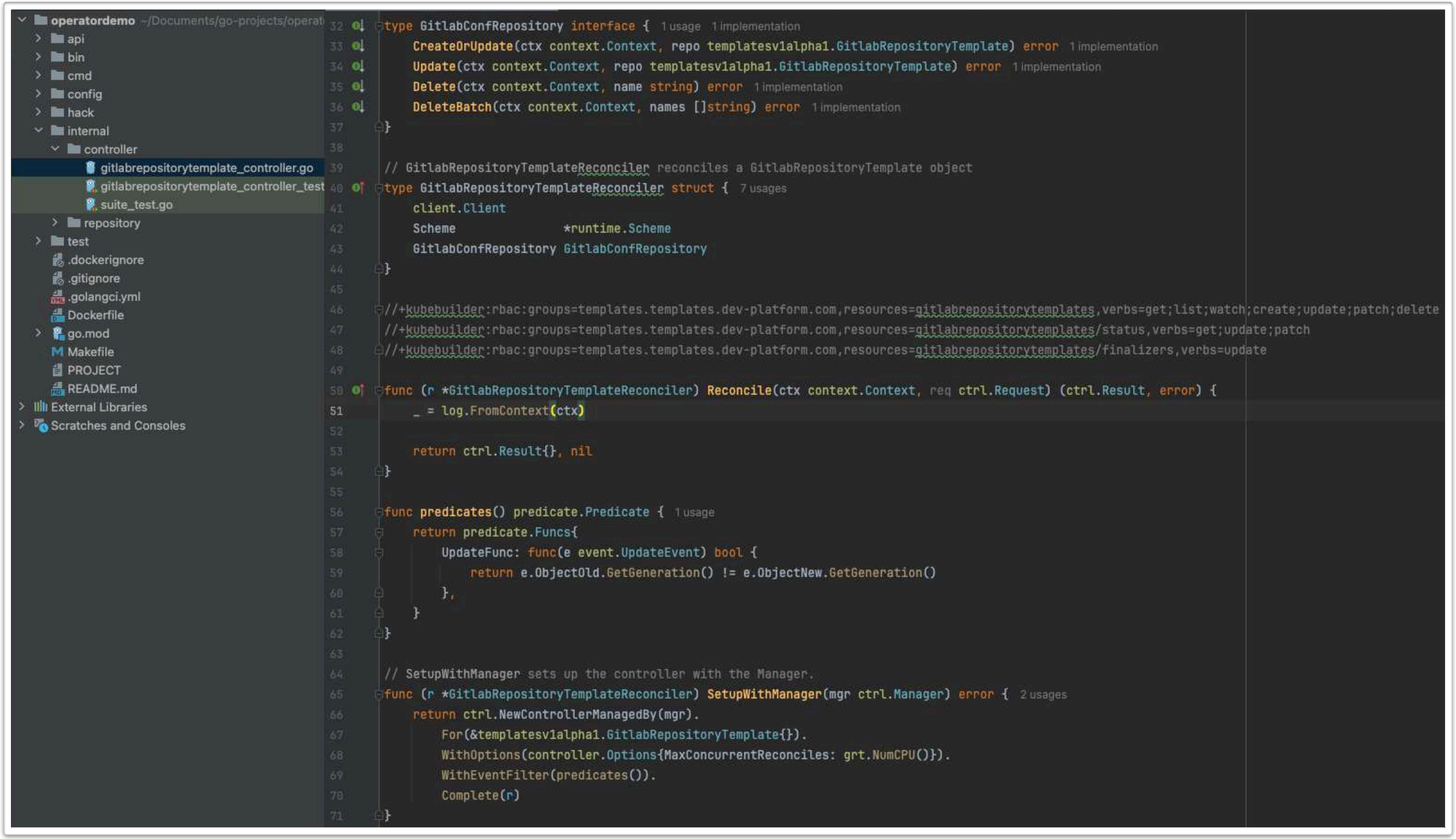
# CRD status



```go
import (
	metav1 "k8s.io/apimachinery/pkg/apis/meta/v1"
)

type GitlabRepositoryTem
	Visibility *GitlabVi
	// +kubebuilder:vali
	ImportURL *string `j
}

// +kubebuilder:validati

type GitlabVisibility st

type RepoStatus string

const (
	StatusReady    RepoS
	StatusDegraded RepoS
)

type GitlabRepositoryTem
	Conditions []metav1.
	Status     RepoStatu
}

//+kubebuilder:object:ro
//+kubebuilder:subresour

type GitlabRepositoryTem
	metav1.TypeMeta   `j
	metav1.ObjectMeta `j

	Spec   GitlabReposit
	Status GitlabReposit
}

//+kubebuilder:object:ro

type GitlabRepositoryTem
	metav1.TypeMeta `jso
	metav1.ListMeta `jso
	Items           []Gi
}

func init() {
	SchemeBuilder.Register(&GitlabRepositoryTemplate{}, &GitlabRepositoryTemplateList{})
}
```

```go
type GitlabVisibility string

type RepoStatus string

const (
	StatusReady    RepoStatus = "Ready"
	StatusDegraded RepoStatus = "Degraded"
)

type GitlabRepositoryTemplateStatus struct {
	Conditions []metav1.Condition `json:"conditions"`
	Status     RepoStatus         `json:"repoStatus"`
}
```

# Контроллер



```
operatordemo ~/Documents/go-projects/operat
  api
  bin
  cmd
  config
  hack
  internal
    controller
      gitlabrepositorytemplate_controller.go
      gitlabrepositorytemplate_controller_test
      suite_test.go
    repository
  test
  .dockerignore
  .gitignore
  .golangci.yml
  Dockerfile
  go.mod
  Makefile
  PROJECT
  README.md
External Libraries
Scratches and Consoles
```

```go
type GitlabConfRepository interface {  1 usage  1 implementation
	CreateOrUpdate(ctx context.Context, repo templatesv1alpha1.GitlabRepositoryTemplate) error  1 implementation
	Update(ctx context.Context, repo templatesv1alpha1.GitlabRepositoryTemplate) error  1 implementation
	Delete(ctx context.Context, name string) error  1 implementation
	DeleteBatch(ctx context.Context, names []string) error  1 implementation
}

// GitlabRepositoryTemplateReconciler reconciles a GitlabRepositoryTemplate object
type GitlabRepositoryTemplateReconciler struct {  7 usages
	client.Client
	Scheme               *runtime.Scheme
	GitlabConfRepository GitlabConfRepository
}

//+kubebuilder:rbac:groups=templates.templates.dev-platform.com,resources=gitlabrepositorytemplates,verbs=get;list;watch;create;update;patch;delete
//+kubebuilder:rbac:groups=templates.templates.dev-platform.com,resources=gitlabrepositorytemplates/status,verbs=get;update;patch
//+kubebuilder:rbac:groups=templates.templates.dev-platform.com,resources=gitlabrepositorytemplates/finalizers,verbs=update

func (r *GitlabRepositoryTemplateReconciler) Reconcile(ctx context.Context, req ctrl.Request) (ctrl.Result, error) {
	_ = log.FromContext(ctx)

	return ctrl.Result{}, nil
}

func predicates() predicate.Predicate {  1 usage
	return predicate.Funcs{
		UpdateFunc: func(e event.UpdateEvent) bool {
			return e.ObjectOld.GetGeneration() != e.ObjectNew.GetGeneration()
		},
	}
}

// SetupWithManager sets up the controller with the Manager.
func (r *GitlabRepositoryTemplateReconciler) SetupWithManager(mgr ctrl.Manager) error {  2 usages
	return ctrl.NewControllerManagedBy(mgr).
		For(&templatesv1alpha1.GitlabRepositoryTemplate{}).
		WithOptions(controller.Options{MaxConcurrentReconciles: grt.NumCPU()}).
		WithEventFilter(predicates()).
		Complete(r)
}
```

# RBAC

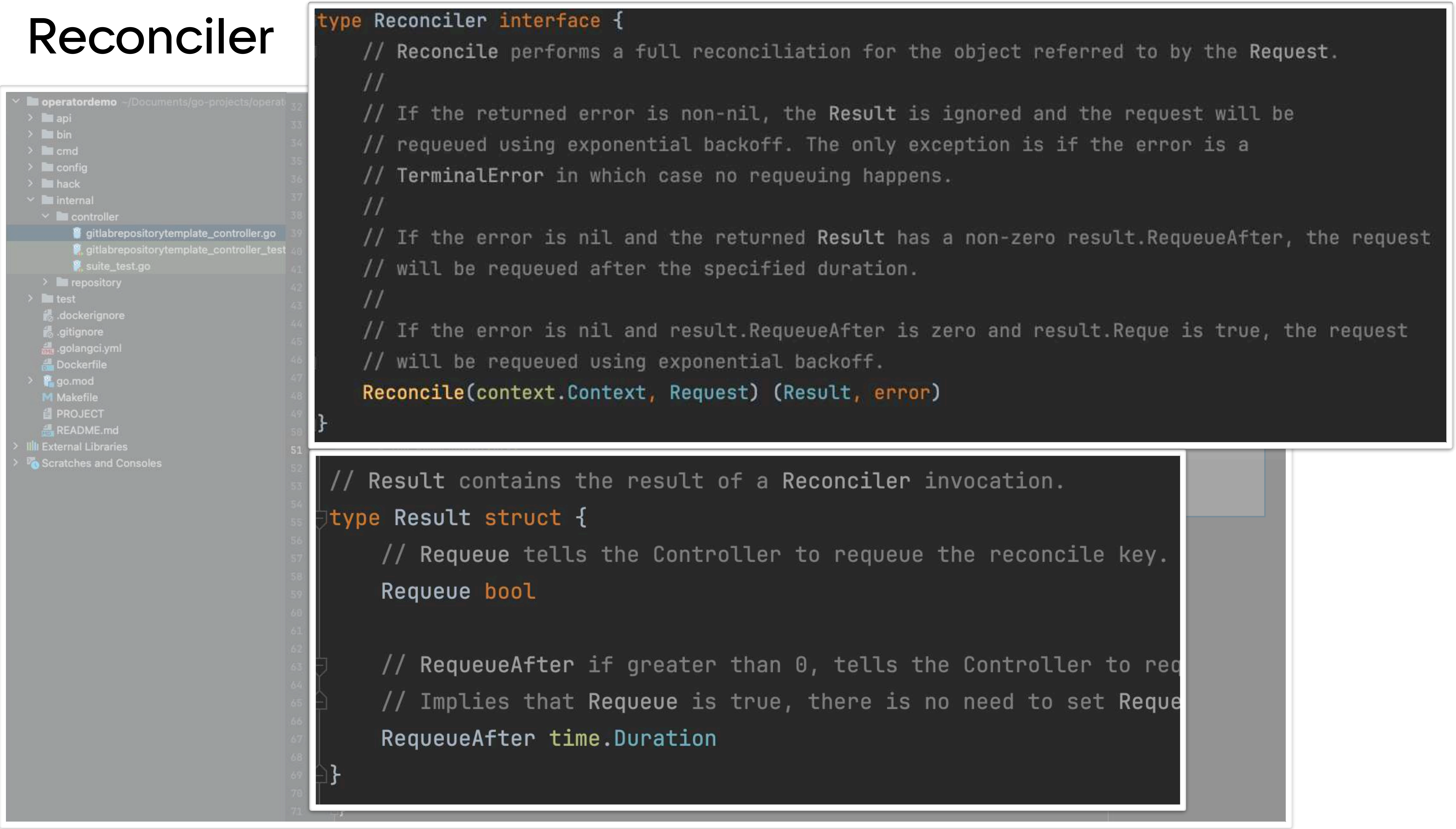
operatordemo ~/Documents/go-projects/operat
api
bin
cmd
config
hack
internal
controller
gitlabrepositorytemplate_controller.go
gitlabrepositorytemplate_controller_test
suite_test.go
repository
test
.dockerignore
.gitignore
.golangci.yml
Dockerfile
go.mod
Makefile

```go
type GitlabConfRepository interface {  1 usage  1 implementation
    CreateOrUpdate(ctx context.Context, repo templatesv1alpha1.GitlabRepositoryTemplate) error  1 implementation
    Update(ctx context.Context, repo templatesv1alpha1.GitlabRepositoryTemplate) error  1 implementation
    Delete(ctx context.Context, name string) error  1 implementation
    DeleteBatch(ctx context.Context, names []string) error  1 implementation
}

// GitlabRepositoryTemplateReconciler reconciles a GitlabRepositoryTemplate object
type GitlabRepositoryTemplateReconciler struct {  7 usages
    client.Client
    Scheme               *runtime.Scheme
    GitlabConfRepository GitlabConfRepository
}

//+kubebuilder:rbac:groups=templates.templates.dev-platform.com,resources=gitlabrepositorytemplates,verbs=get;list;watch;create;update;patch;delete
//+kubebuilder:rbac:groups=templates.templates.dev-platform.com,resources=gitlabrepositorytemplates/status,verbs=get;update;patch
//+kubebuilder:rbac:groups=templates.templates.dev-platform.com,resources=gitlabrepositorytemplates/finalizers,verbs=update
```

```
//+kubebuilder:rbac:groups=templates.templates.dev-platform.com,resources=gitlabrepositorytemplates,verbs=get;list;watch;create;update;patch;delete
//+kubebuilder:rbac:groups=templates.templates.dev-platform.com,resources=gitlabrepositorytemplates/status,verbs=get;update;patch
//+kubebuilder:rbac:groups=templates.templates.dev-platform.com,resources=gitlabrepositorytemplates/finalizers,verbs=update
```

```go
func predicates() predicate.Predicate {  1 usage
    return predicate.Funcs{
        UpdateFunc: func(e event.UpdateEvent) bool {
            return e.ObjectOld.GetGeneration() != e.ObjectNew.GetGeneration()
        },
    }
}

// SetupWithManager sets up the controller with the Manager.
func (r *GitlabRepositoryTemplateReconciler) SetupWithManager(mgr ctrl.Manager) error {  2 usages
    return ctrl.NewControllerManagedBy(mgr).
        For(&templatesv1alpha1.GitlabRepositoryTemplate{}).
        WithOptions(controller.Options{MaxConcurrentReconciles: grt.NumCPU()}).
        WithEventFilter(predicates()).
        Complete(r)
}
```

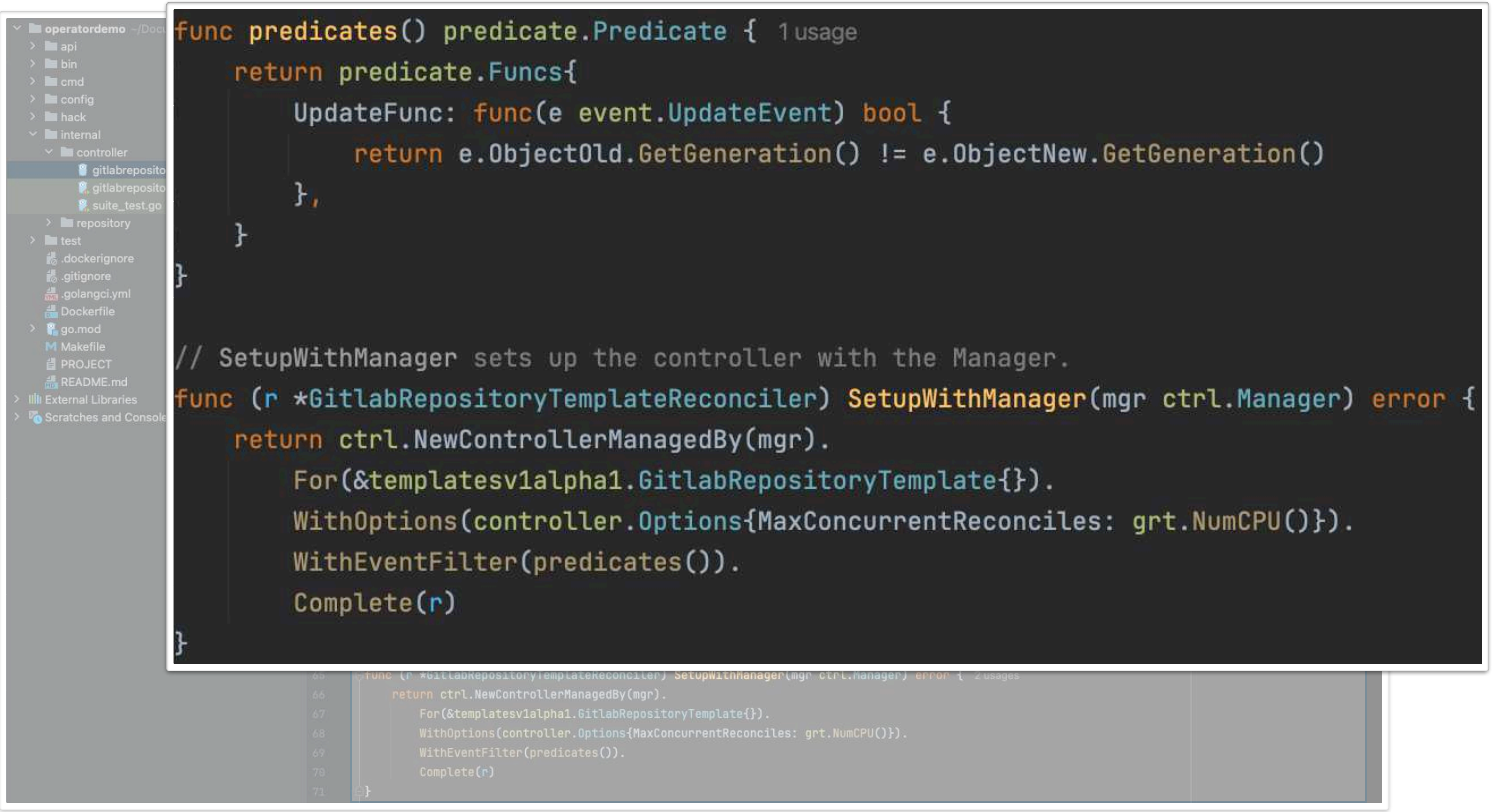
# Reconciler



```go
type Reconciler interface {
	// Reconcile performs a full reconciliation for the object referred to by the Request.
	//
	// If the returned error is non-nil, the Result is ignored and the request will be
	// requeued using exponential backoff. The only exception is if the error is a
	// TerminalError in which case no requeuing happens.
	//
	// If the error is nil and the returned Result has a non-zero result.RequeueAfter, the request
	// will be requeued after the specified duration.
	//
	// If the error is nil and result.RequeueAfter is zero and result.Reque is true, the request
	// will be requeued using exponential backoff.
	Reconcile(context.Context, Request) (Result, error)
}
```

```go
// Result contains the result of a Reconciler invocation.
type Result struct {
	// Requeue tells the Controller to requeue the reconcile key.
	Requeue bool

	// RequeueAfter if greater than 0, tells the Controller to req
	// Implies that Requeue is true, there is no need to set Reque
	RequeueAfter time.Duration
}
```

# Конфигурация



```go
func predicates() predicate.Predicate {
    return predicate.Funcs{
        UpdateFunc: func(e event.UpdateEvent) bool {
            return e.ObjectOld.GetGeneration() != e.ObjectNew.GetGeneration()
        },
    }
}

// SetupWithManager sets up the controller with the Manager.
func (r *GitlabRepositoryTemplateReconciler) SetupWithManager(mgr ctrl.Manager) error {
    return ctrl.NewControllerManagedBy(mgr).
        For(&templatesv1alpha1.GitlabRepositoryTemplate{}).
        WithOptions(controller.Options{MaxConcurrentReconciles: grt.NumCPU()}).
        WithEventFilter(predicates()).
        Complete(r)
}
```

```go
func (r *GitlabRepositoryTemplateReconciler) SetupWithManager(mgr ctrl.Manager) error {
    return ctrl.NewControllerManagedBy(mgr).
        For(&templatesv1alpha1.GitlabRepositoryTemplate{}).
        WithOptions(controller.Options{MaxConcurrentReconciles: grt.NumCPU()}).
        WithEventFilter(predicates()).
        Complete(r)
}
```

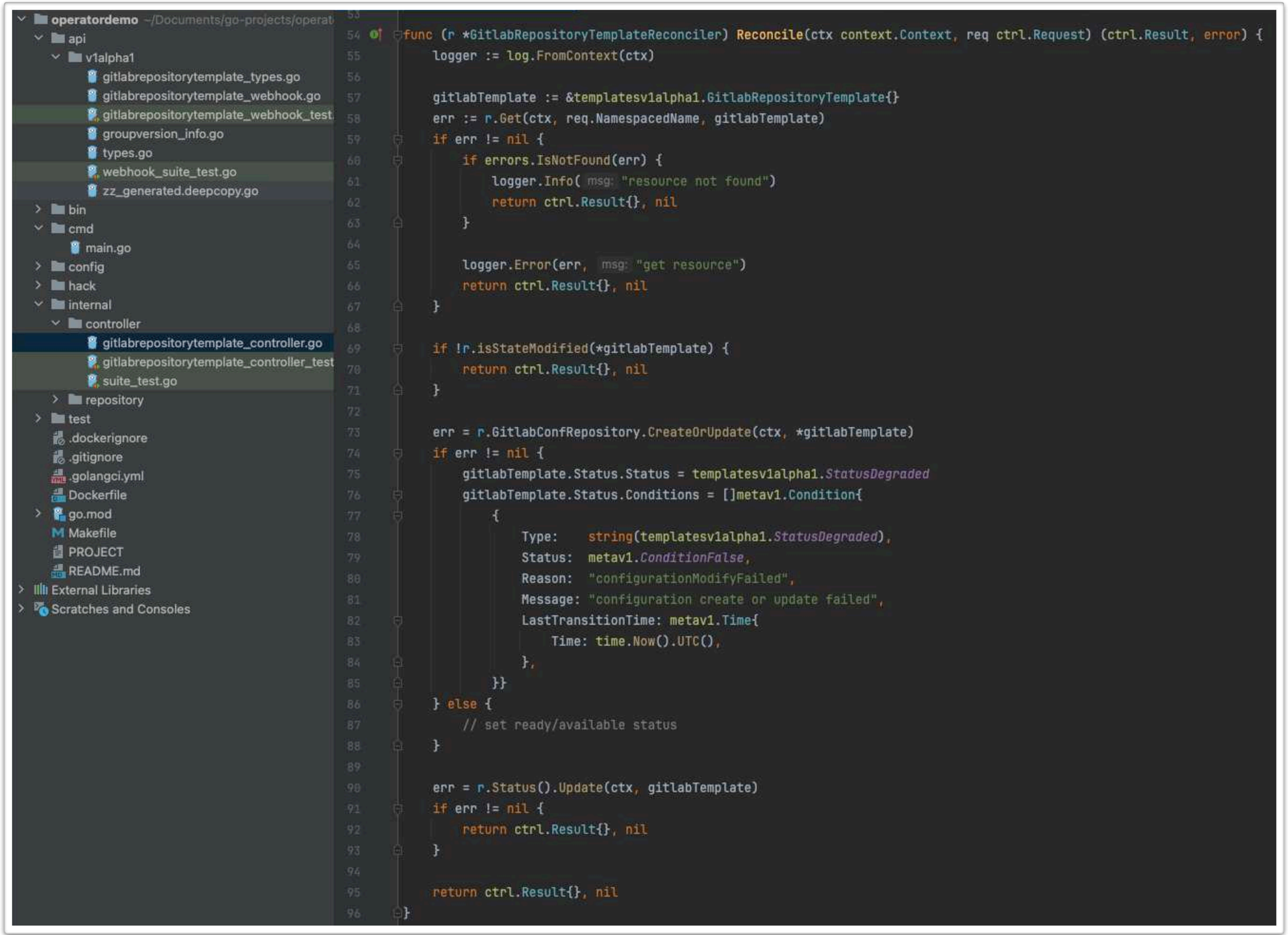
# Repository

operatordemo ~/Documents/go-projects/operat
api
bin
cmd
config
hack
internal
controller

```
type GitlabConfRepository interface {  1 usage  1 implementation
    CreateOrUpdate(ctx context.Context, repo templatesv1alpha1.GitlabRepositoryTemplate) error  1 implementation
    Update(ctx context.Context, repo templatesv1alpha1.GitlabRepositoryTemplate) error  1 implementation
    Delete(ctx context.Context, name string) error  1 implementation
    DeleteBatch(ctx context.Context, names []string) error  1 implementation
}

}

func predicates() predicate.Predicate {  1 usage
    return predicate.Funcs{
        UpdateFunc: func(e event.UpdateEvent) bool {
            return e.ObjectOld.GetGeneration() != e.ObjectNew.GetGeneration()
        },
    }
}

// SetupWithManager sets up the controller with the Manager.
func (r *GitlabRepositoryTemplateReconciler) SetupWithManager(mgr ctrl.Manager) error {  2 usages
    return ctrl.NewControllerManagedBy(mgr).
        For(&templatesv1alpha1.GitlabRepositoryTemplate{}).
        WithOptions(controller.Options{MaxConcurrentReconciles: grt.NumCPU()}).
        WithEventFilter(predicates()).
        Complete(r)
}
```

```
type GitlabConfRepository interface {  1 usage  1 implementation
    CreateOrUpdate(ctx context.Context, repo templatesv1alpha1.GitlabRepositoryTemplate) error
    Update(ctx context.Context, repo templatesv1alpha1.GitlabRepositoryTemplate) error  1 impleme
    Delete(ctx context.Context, name string) error  1 implementation
    DeleteBatch(ctx context.Context, names []string) error  1 implementation
}
```

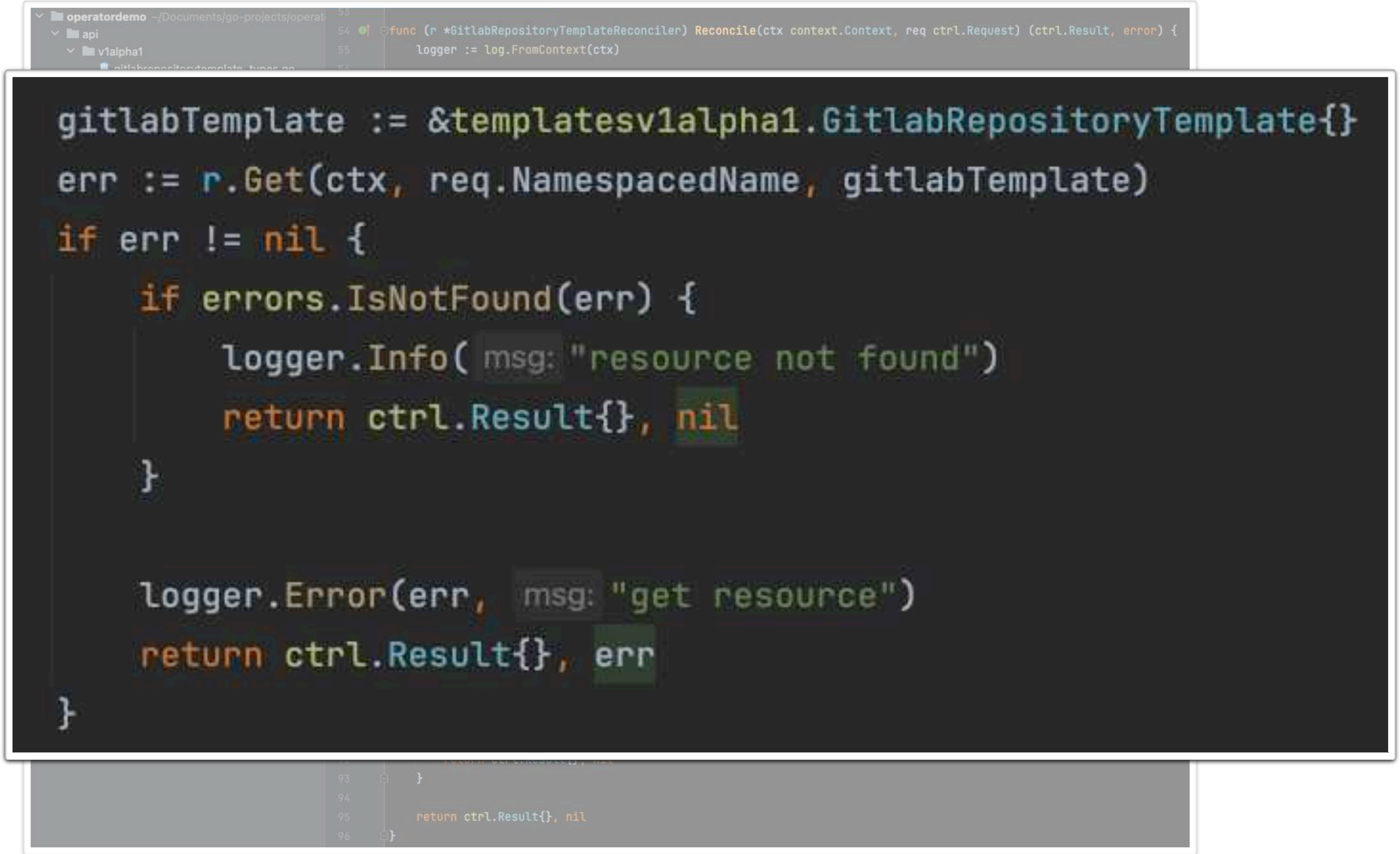
# Создание и обновление CR



- operatordemo ~/Documents/go-projects/operat
  - api
    - v1alpha1
      - gitlabrepositorytemplate_types.go
      - gitlabrepositorytemplate_webhook.go
      - gitlabrepositorytemplate_webhook_test.
      - groupversion_info.go
      - types.go
      - webhook_suite_test.go
      - zz_generated.deepcopy.go
  - bin
  - cmd
    - main.go
  - config
  - hack
  - internal
    - controller
      - gitlabrepositorytemplate_controller.go
      - gitlabrepositorytemplate_controller_test
      - suite_test.go
    - repository
  - test
  - .dockerignore
  - .gitignore
  - .golangci.yml
  - Dockerfile
  - go.mod
  - Makefile
  - PROJECT
  - README.md
- External Libraries
- Scratches and Consoles

```go
func (r *GitlabRepositoryTemplateReconciler) Reconcile(ctx context.Context, req ctrl.Request) (ctrl.Result, error) {
	logger := log.FromContext(ctx)

	gitlabTemplate := &templatesv1alpha1.GitlabRepositoryTemplate{}
	err := r.Get(ctx, req.NamespacedName, gitlabTemplate)
	if err != nil {
		if errors.IsNotFound(err) {
			logger.Info( msg: "resource not found")
			return ctrl.Result{}, nil
		}

		logger.Error(err,  msg: "get resource")
		return ctrl.Result{}, nil
	}

	if !r.isStateModified(*gitlabTemplate) {
		return ctrl.Result{}, nil
	}

	err = r.GitlabConfRepository.CreateOrUpdate(ctx, *gitlabTemplate)
	if err != nil {
		gitlabTemplate.Status.Status = templatesv1alpha1.StatusDegraded
		gitlabTemplate.Status.Conditions = []metav1.Condition{
			{
				Type:    string(templatesv1alpha1.StatusDegraded),
				Status:  metav1.ConditionFalse,
				Reason:  "configurationModifyFailed",
				Message: "configuration create or update failed",
				LastTransitionTime: metav1.Time{
					Time: time.Now().UTC(),
				},
			}}
	} else {
		// set ready/available status
	}

	err = r.Status().Update(ctx, gitlabTemplate)
	if err != nil {
		return ctrl.Result{}, nil
	}

	return ctrl.Result{}, nil
}
```

# Получение CR

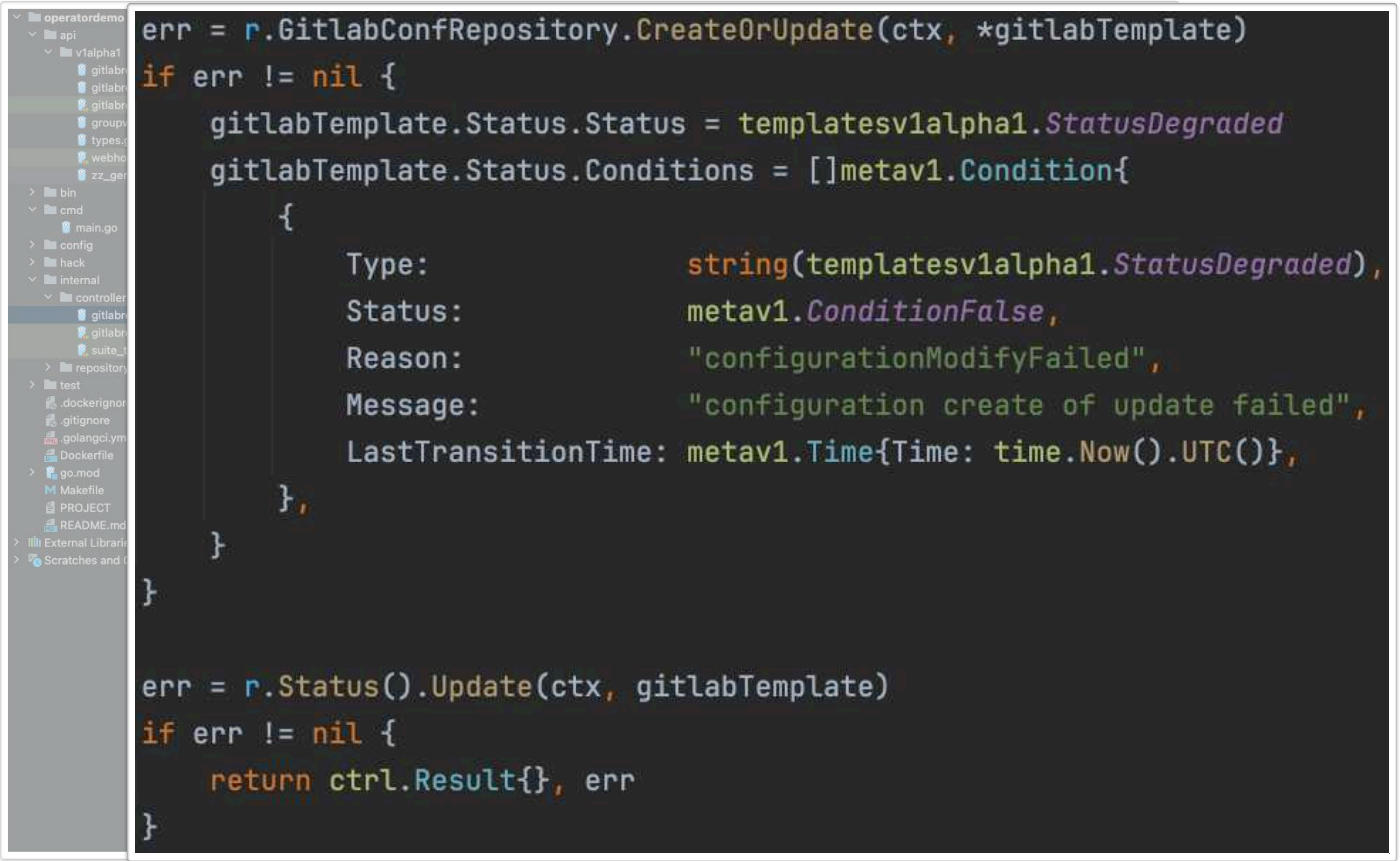


```go
func (r *GitlabRepositoryTemplateReconciler) Reconcile(ctx context.Context, req ctrl.Request) (ctrl.Result, error) {
    logger := log.FromContext(ctx)
```

```go
gitlabTemplate := &templatesv1alpha1.GitlabRepositoryTemplate{}
err := r.Get(ctx, req.NamespacedName, gitlabTemplate)
if err != nil {
    if errors.IsNotFound(err) {
        logger.Info( msg: "resource not found")
        return ctrl.Result{}, nil
    }

    logger.Error(err, msg: "get resource")
    return ctrl.Result{}, err
}
```

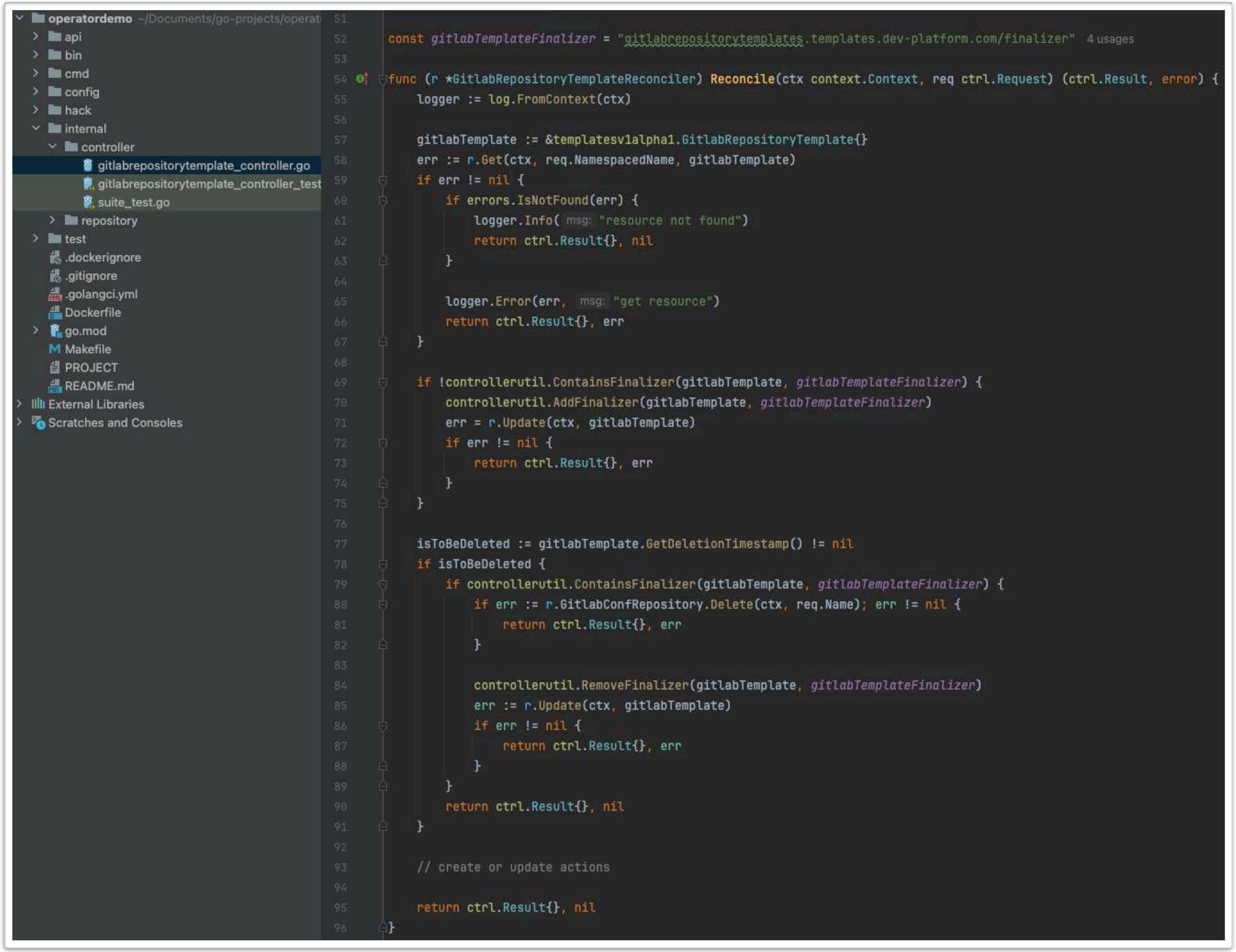
```go
    }

    return ctrl.Result{}, nil
}
```

# Conditions и status



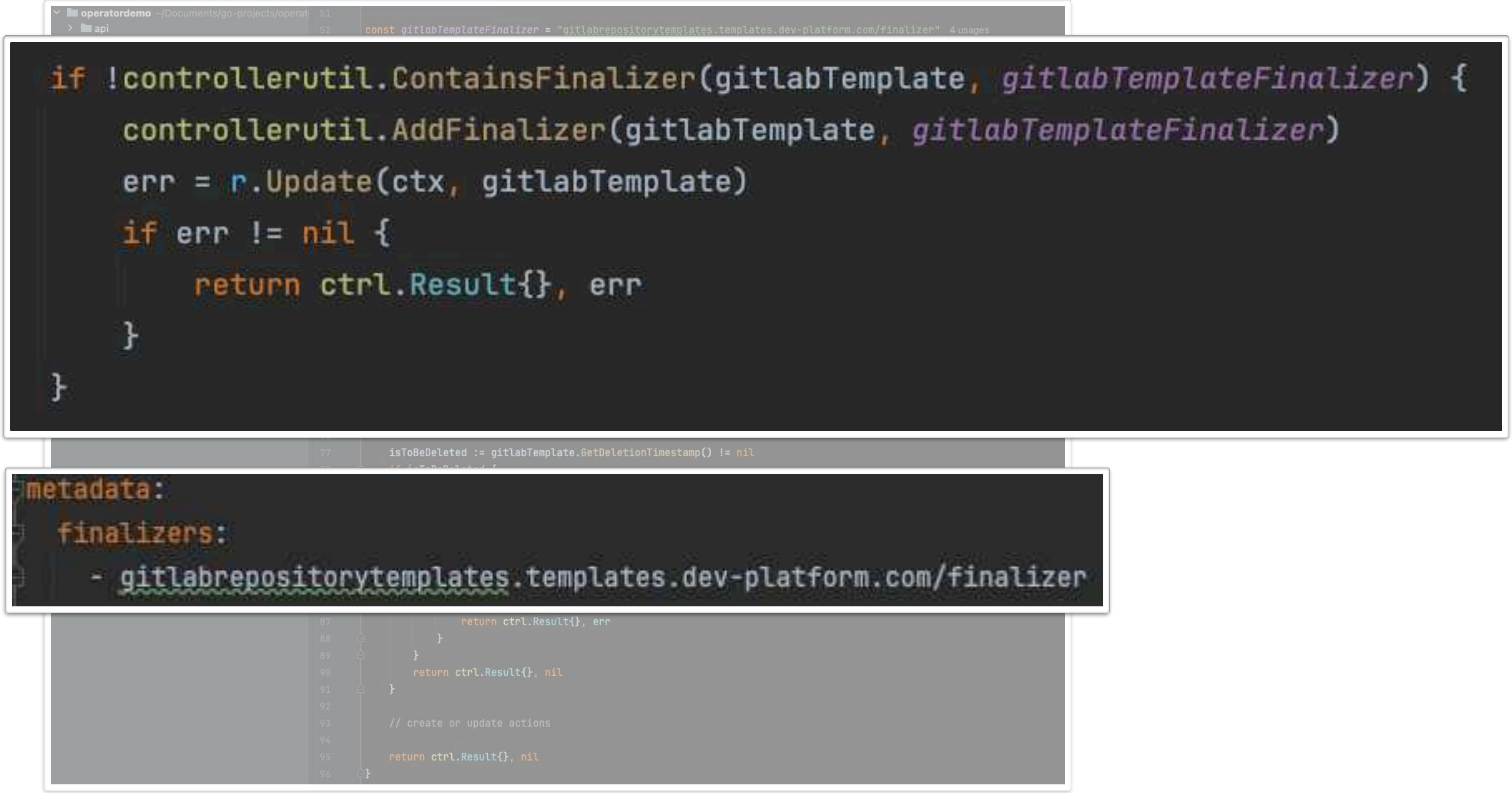
```go
err = r.GitlabConfRepository.CreateOrUpdate(ctx, *gitlabTemplate)
if err != nil {
    gitlabTemplate.Status.Status = templatesv1alpha1.StatusDegraded
    gitlabTemplate.Status.Conditions = []metav1.Condition{
        {
            Type:               string(templatesv1alpha1.StatusDegraded),
            Status:             metav1.ConditionFalse,
            Reason:             "configurationModifyFailed",
            Message:            "configuration create of update failed",
            LastTransitionTime: metav1.Time{Time: time.Now().UTC()},
        },
    }
}

err = r.Status().Update(ctx, gitlabTemplate)
if err != nil {
    return ctrl.Result{}, err
}
```

# Finalizers



```go

const gitlabTemplateFinalizer = "gitlabrepositorytemplates.templates.dev-platform.com/finalizer"  4 usages

func (r *GitlabRepositoryTemplateReconciler) Reconcile(ctx context.Context, req ctrl.Request) (ctrl.Result, error) {
    logger := log.FromContext(ctx)

    gitlabTemplate := &templatesv1alpha1.GitlabRepositoryTemplate{}
    err := r.Get(ctx, req.NamespacedName, gitlabTemplate)
    if err != nil {
        if errors.IsNotFound(err) {
            logger.Info( msg: "resource not found")
            return ctrl.Result{}, nil
        }

        logger.Error(err,  msg: "get resource")
        return ctrl.Result{}, err
    }

    if !controllerutil.ContainsFinalizer(gitlabTemplate, gitlabTemplateFinalizer) {
        controllerutil.AddFinalizer(gitlabTemplate, gitlabTemplateFinalizer)
        err = r.Update(ctx, gitlabTemplate)
        if err != nil {
            return ctrl.Result{}, err
        }
    }

    isToBeDeleted := gitlabTemplate.GetDeletionTimestamp() != nil
    if isToBeDeleted {
        if controllerutil.ContainsFinalizer(gitlabTemplate, gitlabTemplateFinalizer) {
            if err := r.GitlabConfRepository.Delete(ctx, req.Name); err != nil {
                return ctrl.Result{}, err
            }

            controllerutil.RemoveFinalizer(gitlabTemplate, gitlabTemplateFinalizer)
            err := r.Update(ctx, gitlabTemplate)
            if err != nil {
                return ctrl.Result{}, err
            }
        }
        return ctrl.Result{}, nil
    }

    // create or update actions

    return ctrl.Result{}, nil
}
```

# Finalizers



```go
const gitlabTemplateFinalizer = "gitlabrepositorytemplates.templates.dev-platform.com/finalizer"
```

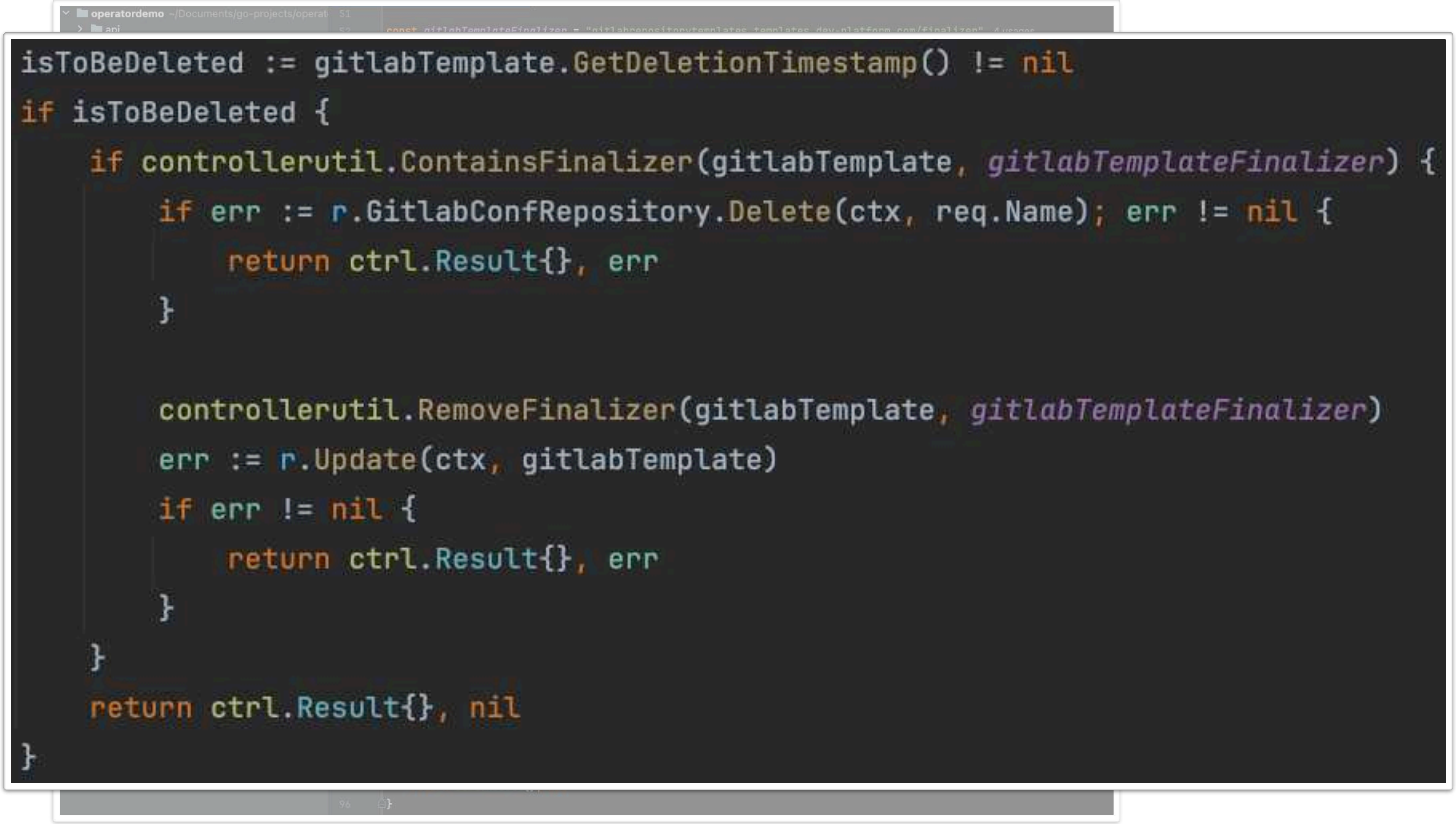
```go
if !controllerutil.ContainsFinalizer(gitlabTemplate, gitlabTemplateFinalizer) {
    controllerutil.AddFinalizer(gitlabTemplate, gitlabTemplateFinalizer)
    err = r.Update(ctx, gitlabTemplate)
    if err != nil {
        return ctrl.Result{}, err
    }
}
```

```go
    isToBeDeleted := gitlabTemplate.GetDeletionTimestamp() != nil
```

```yaml
metadata:
  finalizers:
    - gitlabrepositorytemplates.templates.dev-platform.com/finalizer
```

```go
                return ctrl.Result{}, err
            }
        }
        return ctrl.Result{}, nil
    }

    // create or update actions

    return ctrl.Result{}, nil
}
```

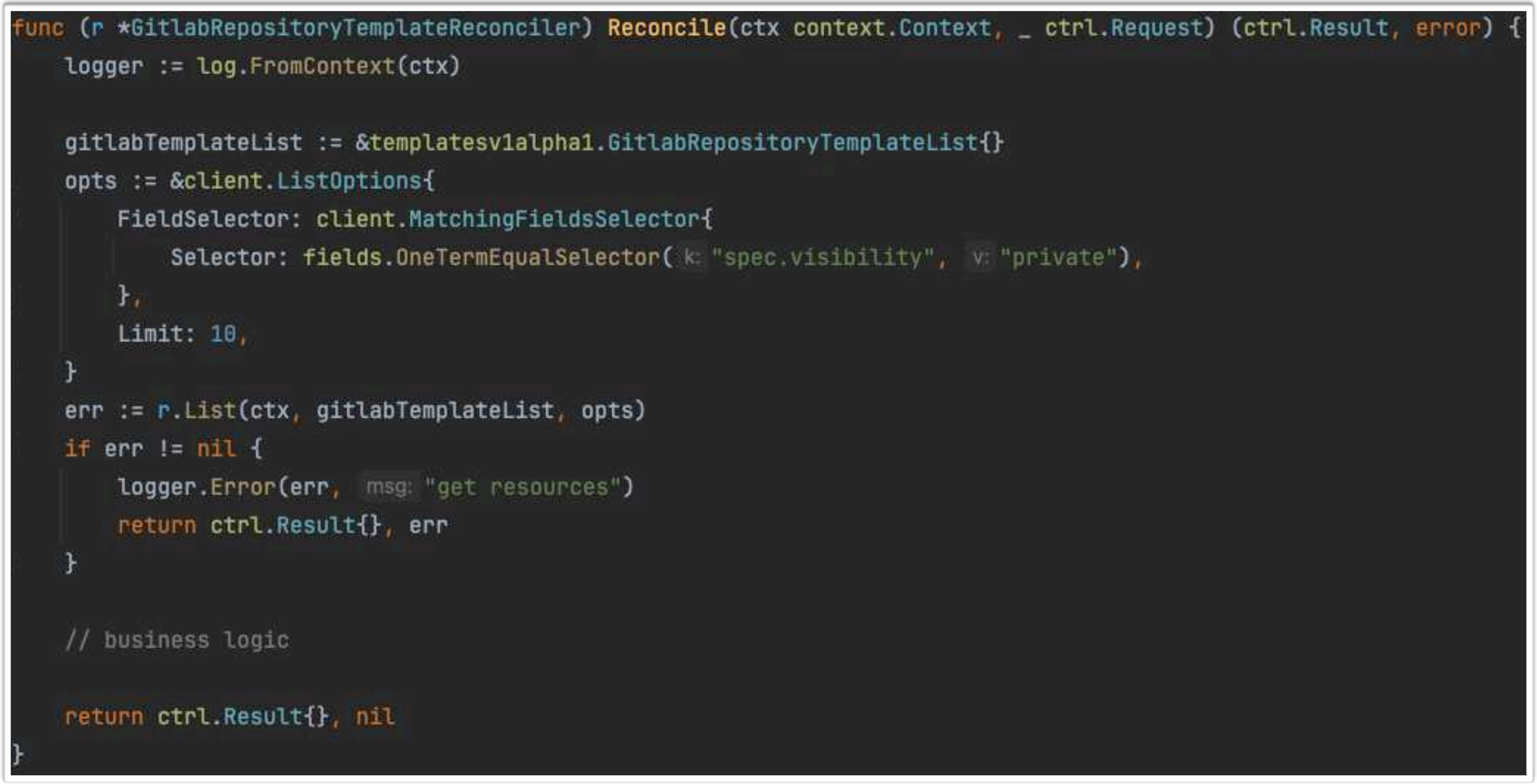
# Finalizers



```
const gitlabTemplateFinalizer = "gitlabrepositorytemplates.templates.dev-platform.com/finalizer" 4 usages
```

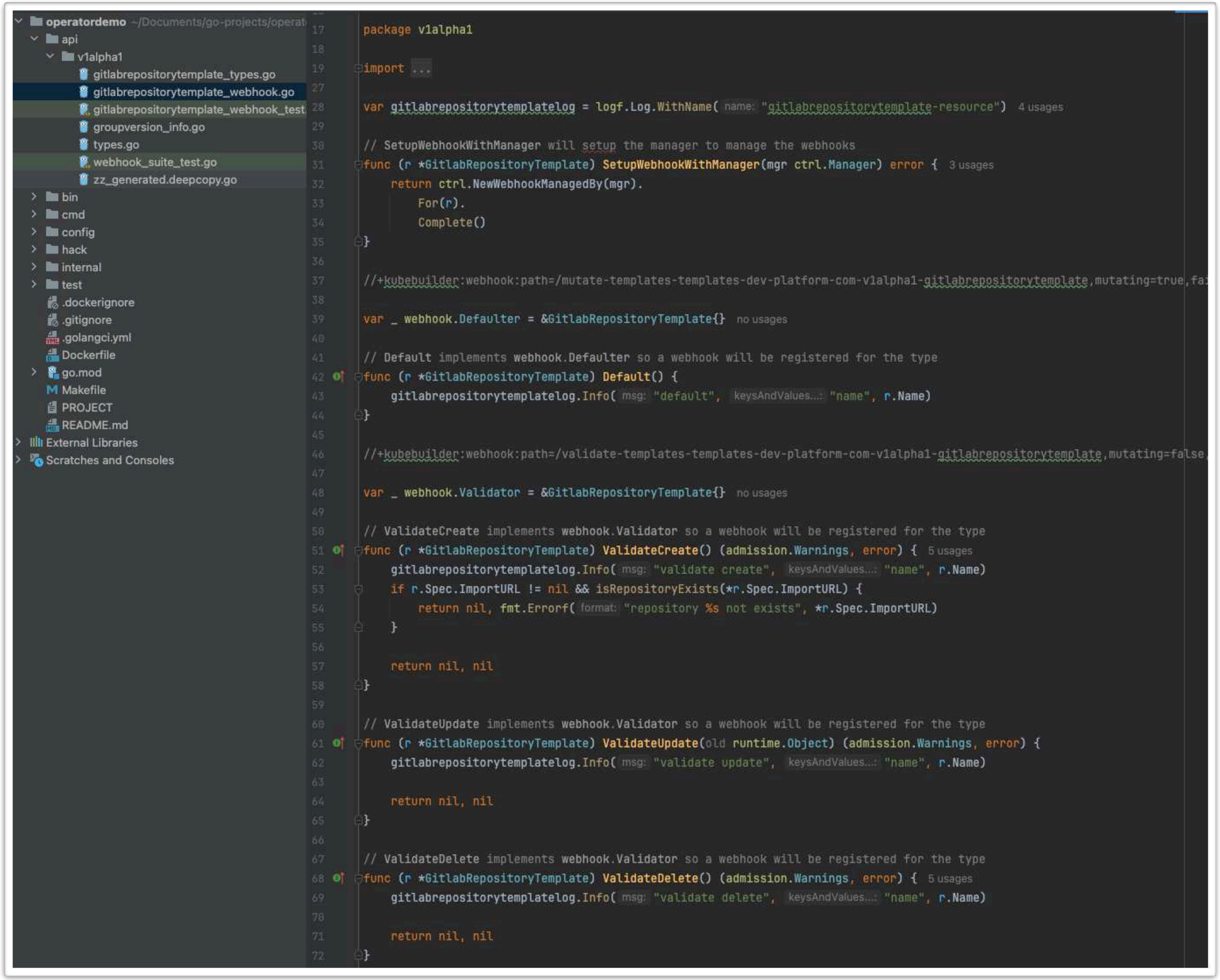
```
isToBeDeleted := gitlabTemplate.GetDeletionTimestamp() != nil
if isToBeDeleted {
    if controllerutil.ContainsFinalizer(gitlabTemplate, gitlabTemplateFinalizer) {
        if err := r.GitlabConfRepository.Delete(ctx, req.Name); err != nil {
            return ctrl.Result{}, err
        }

        controllerutil.RemoveFinalizer(gitlabTemplate, gitlabTemplateFinalizer)
        err := r.Update(ctx, gitlabTemplate)
        if err != nil {
            return ctrl.Result{}, err
        }
    }
    return ctrl.Result{}, nil
}
```

# Список ресурсов



```go
func (r *GitlabRepositoryTemplateReconciler) Reconcile(ctx context.Context, _ ctrl.Request) (ctrl.Result, error) {
    logger := log.FromContext(ctx)

    gitlabTemplateList := &templatesv1alpha1.GitlabRepositoryTemplateList{}
    opts := &client.ListOptions{
        FieldSelector: client.MatchingFieldsSelector{
            Selector: fields.OneTermEqualSelector( k: "spec.visibility",  v: "private"),
        },
        Limit: 10,
    }
    err := r.List(ctx, gitlabTemplateList, opts)
    if err != nil {
        logger.Error(err,  msg: "get resources")
        return ctrl.Result{}, err
    }

    // business logic

    return ctrl.Result{}, nil
}
```

# Webhooks



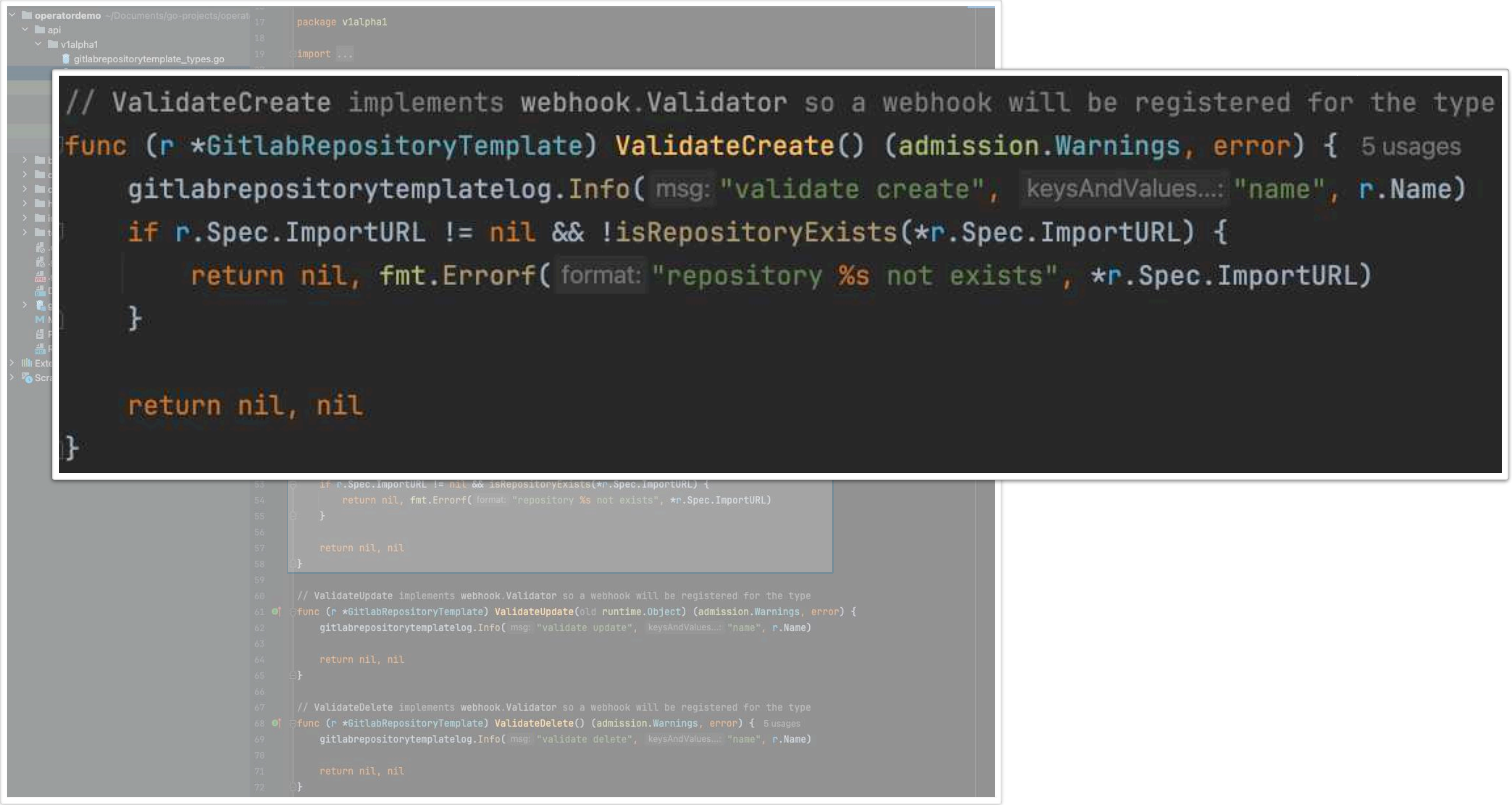
```
operatordemo ~/Documents/go-projects/operat
  api
    v1alpha1
      gitlabrepositorytemplate_types.go
      gitlabrepositorytemplate_webhook.go
      gitlabrepositorytemplate_webhook_test
      groupversion_info.go
      types.go
      webhook_suite_test.go
      zz_generated.deepcopy.go
  bin
  cmd
  config
  hack
  internal
  test
  .dockerignore
  .gitignore
  .golangci.yml
  Dockerfile
  go.mod
  Makefile
  PROJECT
  README.md
External Libraries
Scratches and Consoles
```

```go
package v1alpha1

import ...

var gitlabrepositorytemplatelog = logf.Log.WithName( name: "gitlabrepositorytemplate-resource")  4 usages

// SetupWebhookWithManager will setup the manager to manage the webhooks
func (r *GitlabRepositoryTemplate) SetupWebhookWithManager(mgr ctrl.Manager) error {  3 usages
    return ctrl.NewWebhookManagedBy(mgr).
        For(r).
        Complete()
}

//+kubebuilder:webhook:path=/mutate-templates-templates-dev-platform-com-v1alpha1-gitlabrepositorytemplate,mutating=true,fa

var _ webhook.Defaulter = &GitlabRepositoryTemplate{}  no usages

// Default implements webhook.Defaulter so a webhook will be registered for the type
func (r *GitlabRepositoryTemplate) Default() {
    gitlabrepositorytemplatelog.Info( msg: "default",  keysAndValues...: "name", r.Name)
}

//+kubebuilder:webhook:path=/validate-templates-templates-dev-platform-com-v1alpha1-gitlabrepositorytemplate,mutating=false,

var _ webhook.Validator = &GitlabRepositoryTemplate{}  no usages

// ValidateCreate implements webhook.Validator so a webhook will be registered for the type
func (r *GitlabRepositoryTemplate) ValidateCreate() (admission.Warnings, error) {  5 usages
    gitlabrepositorytemplatelog.Info( msg: "validate create",  keysAndValues...: "name", r.Name)
    if r.Spec.ImportURL != nil && isRepositoryExists(*r.Spec.ImportURL) {
        return nil, fmt.Errorf( format: "repository %s not exists", *r.Spec.ImportURL)
    }

    return nil, nil
}

// ValidateUpdate implements webhook.Validator so a webhook will be registered for the type
func (r *GitlabRepositoryTemplate) ValidateUpdate(old runtime.Object) (admission.Warnings, error) {
    gitlabrepositorytemplatelog.Info( msg: "validate update",  keysAndValues...: "name", r.Name)

    return nil, nil
}

// ValidateDelete implements webhook.Validator so a webhook will be registered for the type
func (r *GitlabRepositoryTemplate) ValidateDelete() (admission.Warnings, error) {  5 usages
    gitlabrepositorytemplatelog.Info( msg: "validate delete",  keysAndValues...: "name", r.Name)

    return nil, nil
}
```

**operator-sdk create webhook**

- -**group** templates
- -**version** v1alpha1
- -**kind** GitlabRepositoryTemplate
- -**defaulting**
- -**programmatic-validation**

# Validation webhooks


operatordemo ~/Documents/go-projects/operat
api
v1alpha1
gitlabrepositorytemplate_types.go
Exte
Scra

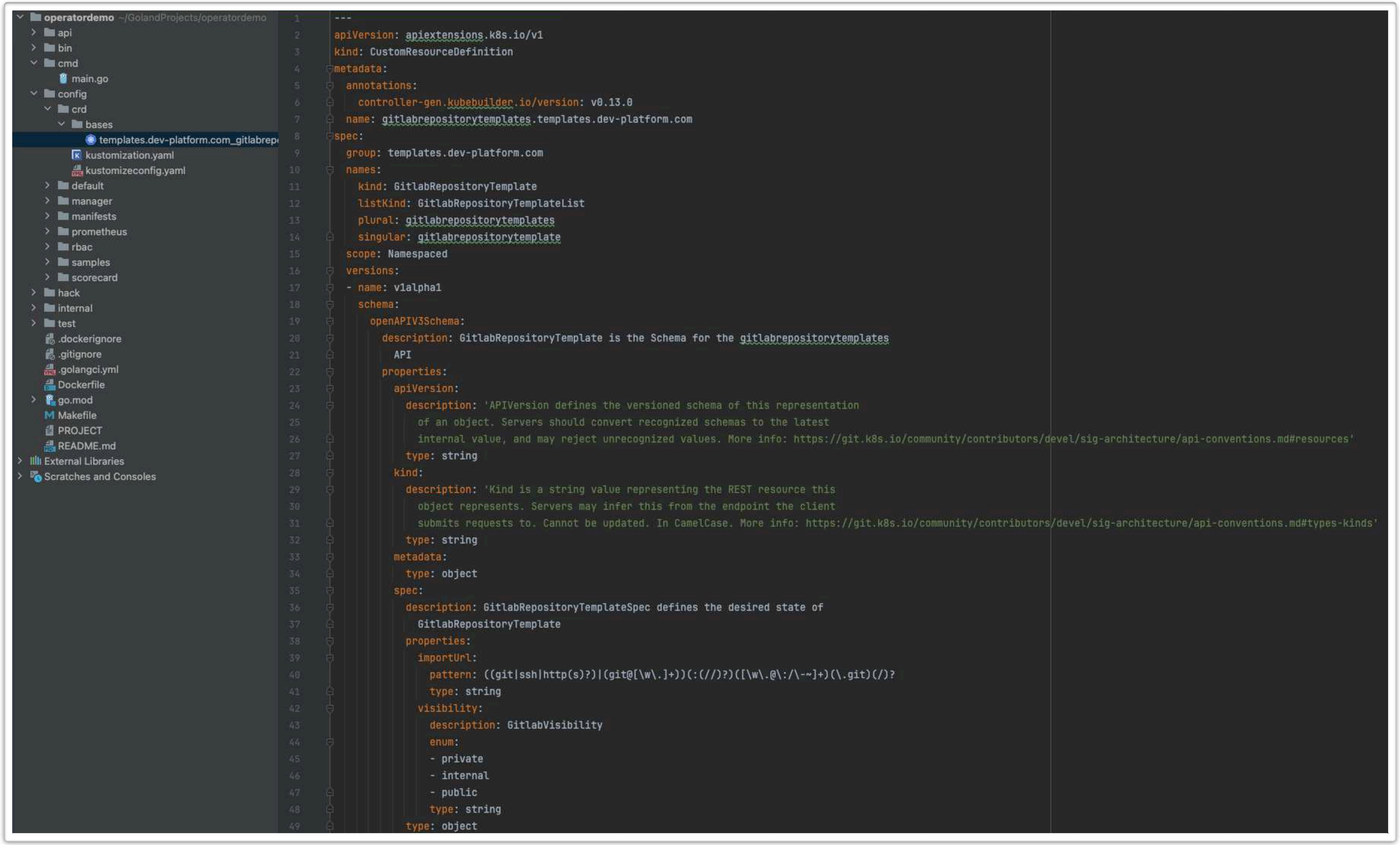
```
package v1alpha1

import ...
```

```
// ValidateCreate implements webhook.Validator so a webhook will be registered for the type
func (r *GitlabRepositoryTemplate) ValidateCreate() (admission.Warnings, error) { 5 usages
	gitlabrepositorytemplatelog.Info( msg: "validate create", keysAndValues...: "name", r.Name)
	if r.Spec.ImportURL != nil && !isRepositoryExists(*r.Spec.ImportURL) {
		return nil, fmt.Errorf( format: "repository %s not exists", *r.Spec.ImportURL)
	}

	return nil, nil
}
```

```
	if r.Spec.ImportURL != nil && isRepositoryExists(*r.Spec.ImportURL) {
		return nil, fmt.Errorf( format: "repository %s not exists", *r.Spec.ImportURL)
	}

	return nil, nil
}
```

```
// ValidateUpdate implements webhook.Validator so a webhook will be registered for the type
func (r *GitlabRepositoryTemplate) ValidateUpdate(old runtime.Object) (admission.Warnings, error) {
	gitlabrepositorytemplatelog.Info( msg: "validate update", keysAndValues...: "name", r.Name)

	return nil, nil
}

// ValidateDelete implements webhook.Validator so a webhook will be registered for the type
func (r *GitlabRepositoryTemplate) ValidateDelete() (admission.Warnings, error) { 5 usages
	gitlabrepositorytemplatelog.Info( msg: "validate delete", keysAndValues...: "name", r.Name)

	return nil, nil
}
```

# Генерация и добавление в кластер CRD



```
operatordemo ~/GolandProjects/operatordemo
  api
  bin
  cmd
    main.go
  config
    crd
      bases
        templates.dev-platform.com_gitlabrep
      kustomization.yaml
      kustomizeconfig.yaml
    default
    manager
    manifests
    prometheus
    rbac
    samples
    scorecard
  hack
  internal
  test
  .dockerignore
  .gitignore
  .golangci.yml
  Dockerfile
  go.mod
  Makefile
  PROJECT
  README.md
External Libraries
Scratches and Consoles
```

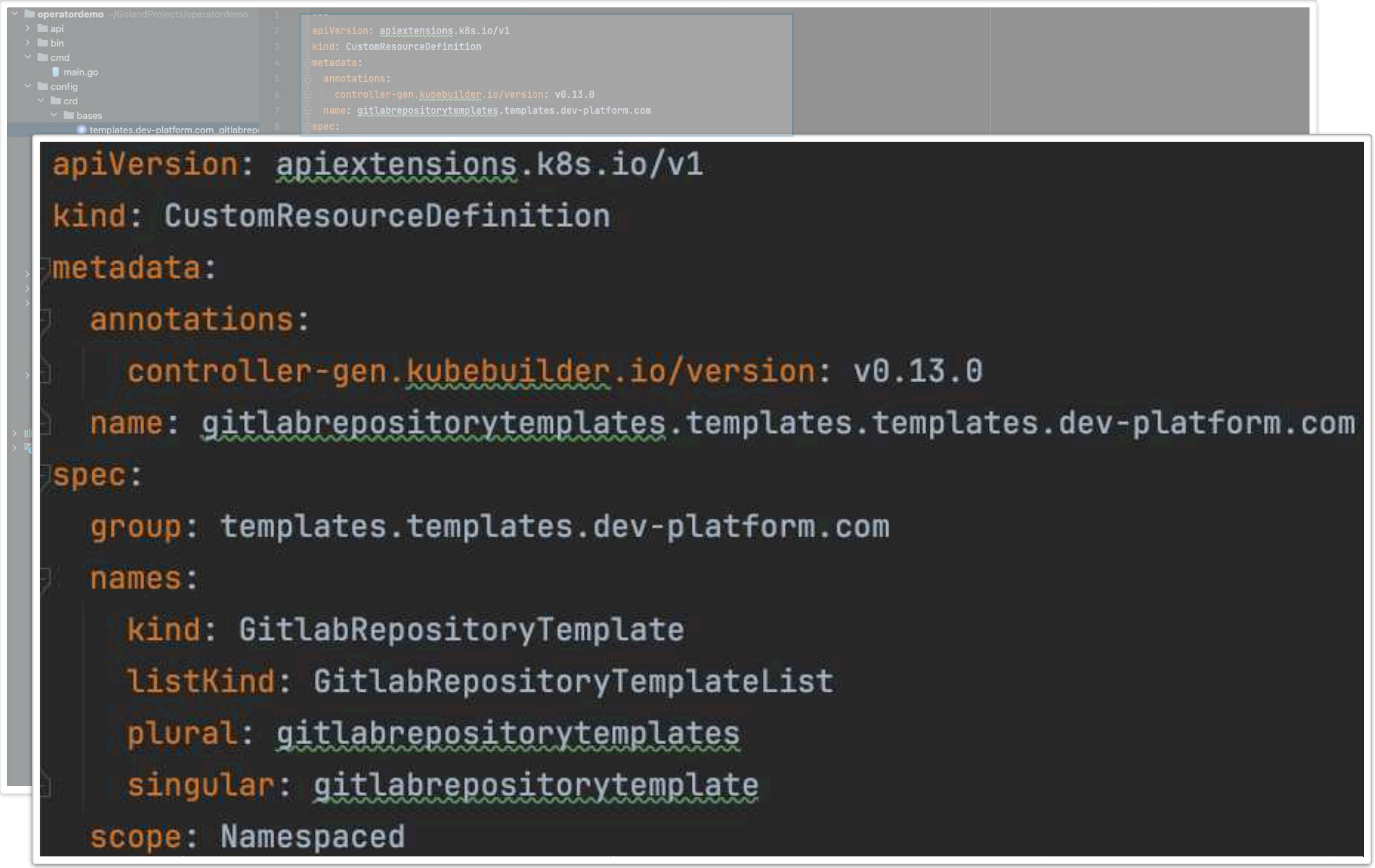
```yaml
---
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
  annotations:
    controller-gen.kubebuilder.io/version: v0.13.0
  name: gitlabrepositorytemplates.templates.dev-platform.com
spec:
  group: templates.dev-platform.com
  names:
    kind: GitlabRepositoryTemplate
    listKind: GitlabRepositoryTemplateList
    plural: gitlabrepositorytemplates
    singular: gitlabrepositorytemplate
  scope: Namespaced
  versions:
  - name: v1alpha1
    schema:
      openAPIV3Schema:
        description: GitlabRepositoryTemplate is the Schema for the gitlabrepositorytemplates
          API
        properties:
          apiVersion:
            description: 'APIVersion defines the versioned schema of this representation
              of an object. Servers should convert recognized schemas to the latest
              internal value, and may reject unrecognized values. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources'
            type: string
          kind:
            description: 'Kind is a string value representing the REST resource this
              object represents. Servers may infer this from the endpoint the client
              submits requests to. Cannot be updated. In CamelCase. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds'
            type: string
          metadata:
            type: object
          spec:
            description: GitlabRepositoryTemplateSpec defines the desired state of
              GitlabRepositoryTemplate
            properties:
              importUrl:
                pattern: ((git|ssh|http(s)?)|(git@[\w\.]+))(:(//)?)([\w\.@\:/\-~]+)(\.git)(/)?
                type: string
              visibility:
                description: GitlabVisibility
                enum:
                - private
                - internal
                - public
                type: string
            type: object
```

**Генерация CRD**:
make manifests

**Генерация DeepCopy**:
make generate

**Добавление в кластер**:
make install

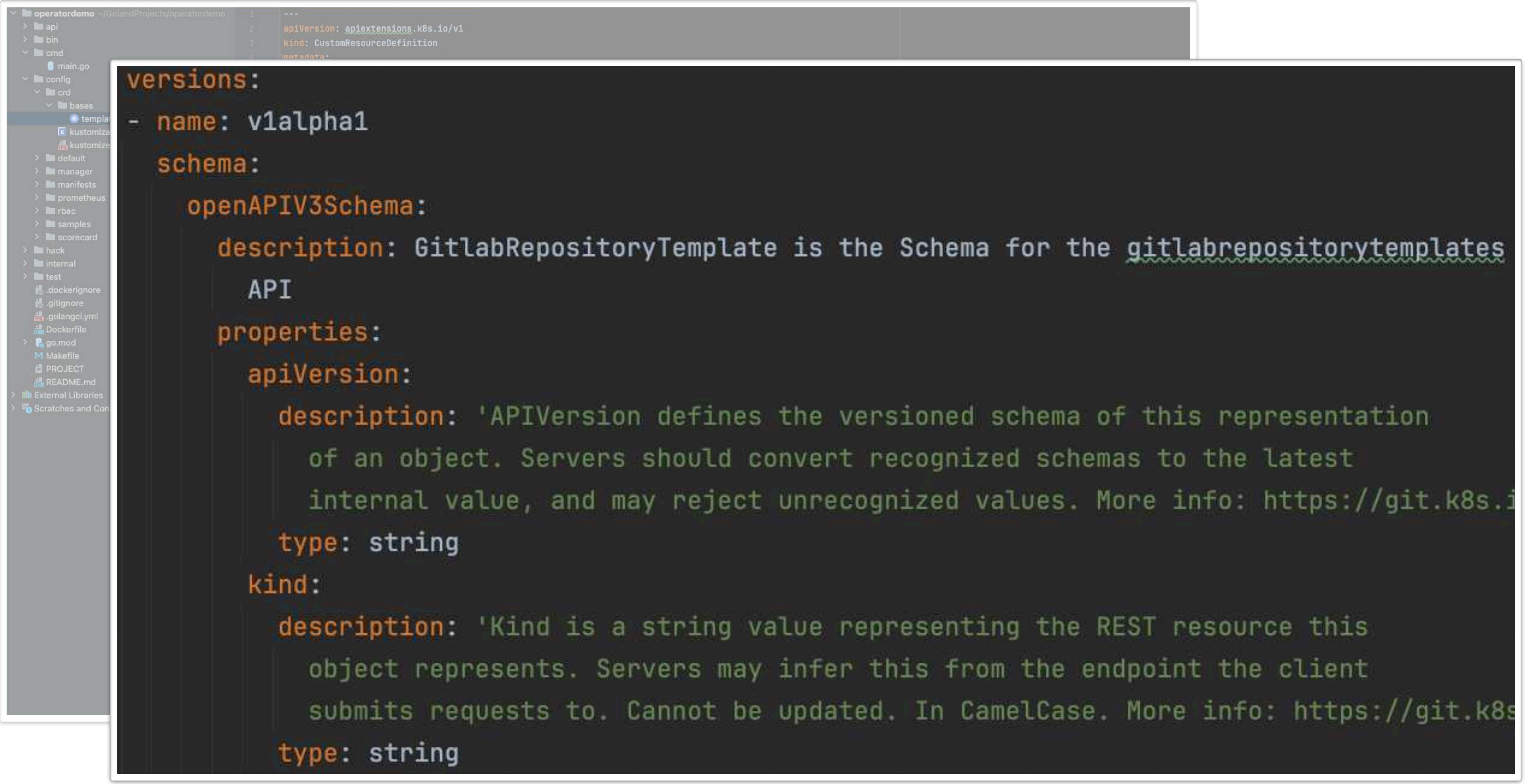
# Metadata и spec



operatordemo ~/GolandProjects/operatordemo
api
bin
cmd
main.go
config
crd
bases
templates.dev-platform.com_gitlabrep

```yaml
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
  annotations:
    controller-gen.kubebuilder.io/version: v0.13.0
  name: gitlabrepositorytemplates.templates.templates.dev-platform.com
spec:
  group: templates.templates.dev-platform.com
  names:
    kind: GitlabRepositoryTemplate
    listKind: GitlabRepositoryTemplateList
    plural: gitlabrepositorytemplates
    singular: gitlabrepositorytemplate
  scope: Namespaced
```

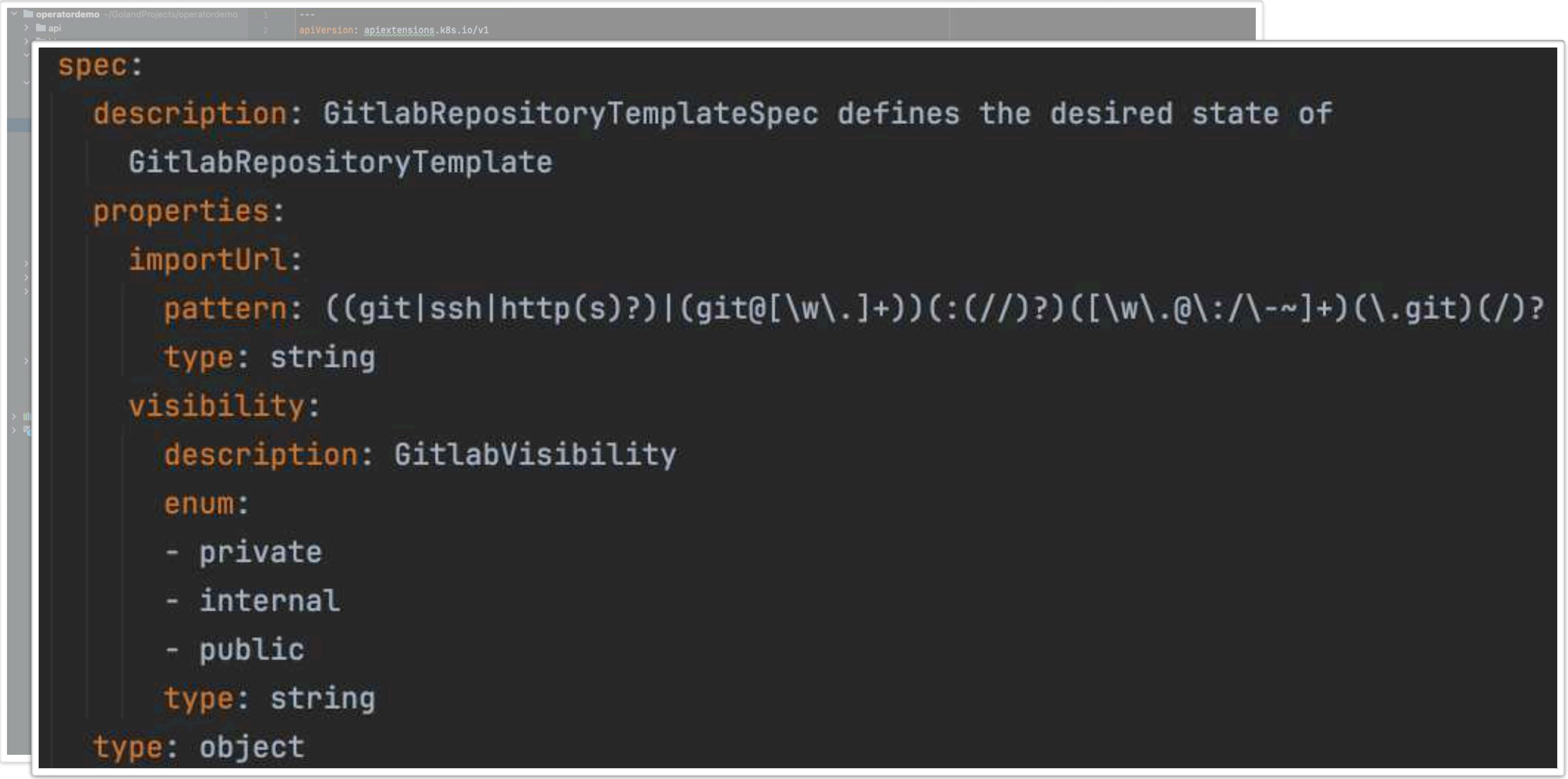
# OpenAPI V3



```yaml
---
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
```

```yaml
versions:
- name: v1alpha1
  schema:
    openAPIV3Schema:
      description: GitlabRepositoryTemplate is the Schema for the gitlabrepositorytemplates
        API
      properties:
        apiVersion:
          description: 'APIVersion defines the versioned schema of this representation
            of an object. Servers should convert recognized schemas to the latest
            internal value, and may reject unrecognized values. More info: https://git.k8s.i
          type: string
        kind:
          description: 'Kind is a string value representing the REST resource this
            object represents. Servers may infer this from the endpoint the client
            submits requests to. Cannot be updated. In CamelCase. More info: https://git.k8s
          type: string
```

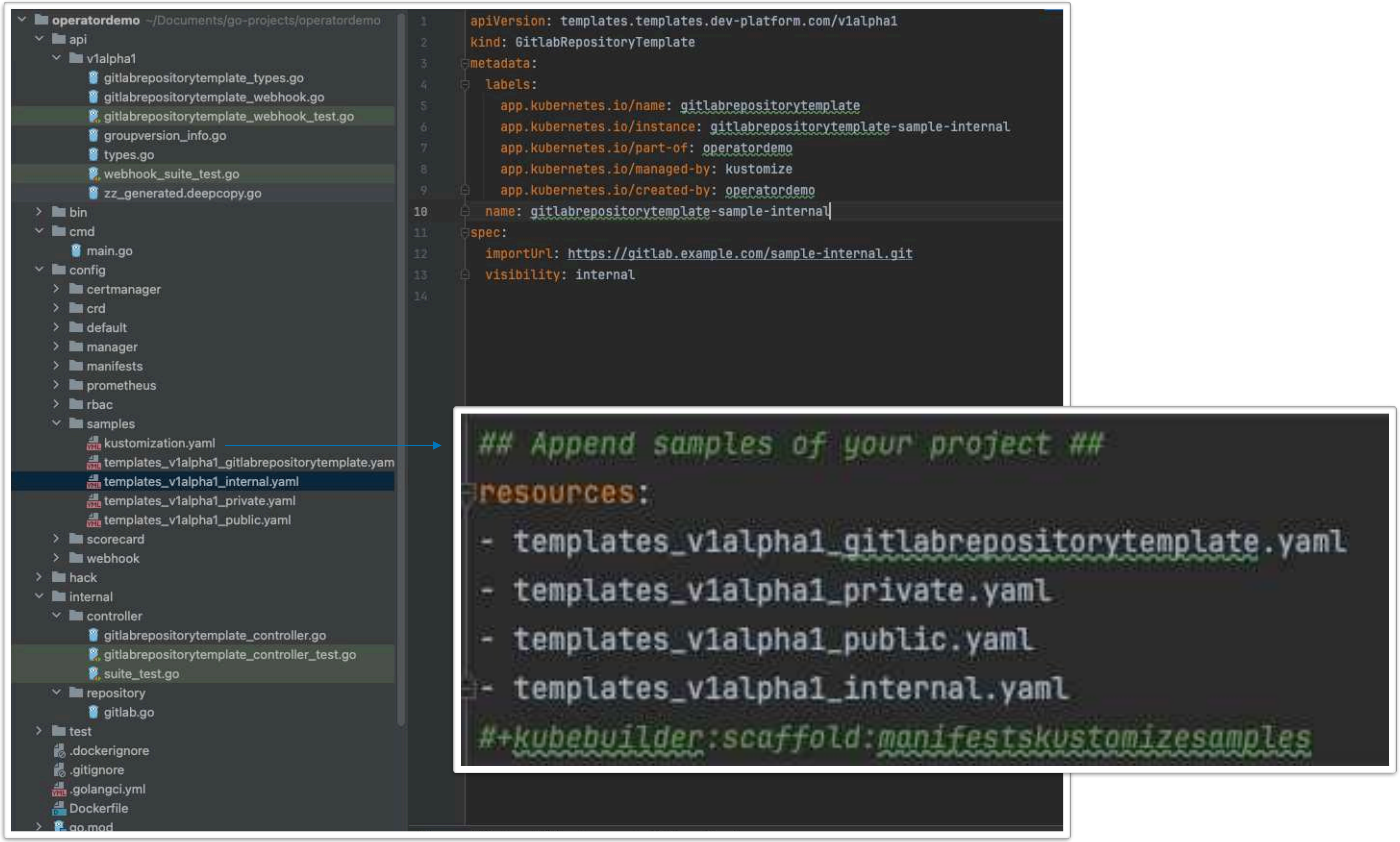
# Валидация



```
---
apiVersion: apiextensions.k8s.io/v1
```

```
spec:
  description: GitlabRepositoryTemplateSpec defines the desired state of
    GitlabRepositoryTemplate
  properties:
    importUrl:
      pattern: ((git|ssh|http(s)?)|(git@[\w\.]+))(:(//)?)([\w\.@\:/\-~]+)(\.git)(/)?
      type: string
    visibility:
      description: GitlabVisibility
      enum:
      - private
      - internal
      - public
      type: string
  type: object
```

# Добавление в кластер CR



```
operatordemo ~/Documents/go-projects/operatordemo
  api
    v1alpha1
      gitlabrepositorytemplate_types.go
      gitlabrepositorytemplate_webhook.go
      gitlabrepositorytemplate_webhook_test.go
      groupversion_info.go
      types.go
      webhook_suite_test.go
      zz_generated.deepcopy.go
  bin
  cmd
    main.go
  config
    certmanager
    crd
    default
    manager
    manifests
    prometheus
    rbac
    samples
      kustomization.yaml
      templates_v1alpha1_gitlabrepositorytemplate.yam
      templates_v1alpha1_internal.yaml
      templates_v1alpha1_private.yaml
      templates_v1alpha1_public.yaml
    scorecard
    webhook
  hack
  internal
    controller
      gitlabrepositorytemplate_controller.go
      gitlabrepositorytemplate_controller_test.go
      suite_test.go
    repository
      gitlab.go
  test
  .dockerignore
  .gitignore
  .golangci.yml
  Dockerfile
  go.mod
```

```yaml
apiVersion: templates.templates.dev-platform.com/v1alpha1
kind: GitlabRepositoryTemplate
metadata:
  labels:
    app.kubernetes.io/name: gitlabrepositorytemplate
    app.kubernetes.io/instance: gitlabrepositorytemplate-sample-internal
    app.kubernetes.io/part-of: operatordemo
    app.kubernetes.io/managed-by: kustomize
    app.kubernetes.io/created-by: operatordemo
  name: gitlabrepositorytemplate-sample-internal
spec:
  importUrl: https://gitlab.example.com/sample-internal.git
  visibility: internal
```

```yaml
## Append samples of your project ##
resources:
- templates_v1alpha1_gitlabrepositorytemplate.yaml
- templates_v1alpha1_private.yaml
- templates_v1alpha1_public.yaml
- templates_v1alpha1_internal.yaml
#+kubebuilder:scaffold:manifestskustomizesamples
```

# Почему мы выбрали оператор

1

Версионирование

# Почему мы выбрали оператор

1

Версионирование

2

Валидация

# Почему мы выбрали оператор

1

Версионирование

2

Валидация

3

Реконсиляция, остлеживание событий с ресурсами

# Почему мы выбрали оператор

1

Версионирование

2

Валидация

3

Реконсиляция, остлежнивание событий с ресурсами

4

Дополнительные возможности обслуживания (OLM) и тестирования (Scorecard)

# Почему мы выбрали оператор

1. Версионирование
2. Валидация
3. Реконсиляция, остлежнивание событий с ресурсами
4. Дополнительные возможности обслуживания (OLM) и тестирования (Scorecard)
5. DRY подход с помощью Kustomize

# Почему мы выбрали оператор

1

Версионирование

2

Валидация

3

Реконсиляция, остлежнивание событий с ресурсами

4

Дополнительные возможности обслуживания (OLM) и тестирования (Scorecard)

5

DRY подход с помощью Kustomize

6

Написан на Go

Виктор Горячкин
backend разработчик

# Спасибо за внимание