Практическая работа 8 - CI/CD

vi gitlab-runner.yaml

```
igor@ubuntu-server: ~/cicd - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
$--
apiVersion: v1
kind: ServiceAccount
metadata:
  name: gitlab-runner
  labels:
    app: gitlab-runner
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: gitlab-runner
  labels:
    app: gitlab-runner
rules:
- apiGroups: [""]
  resources: ["*"]
  verbs: ["*"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: gitlab-runner
  labels:
    app: gitlab-runner
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: gitlab-runner
subjects:
- kind: ServiceAccount
  name: gitlab-runner
  namespace: gitlab
---
apiVersion: v1
kind: Secret
metadata:
  name: gitlab-runner
  labels:
    app: gitlab-runner
type: Opaque
stringData:
  runner-registration-token: wxpLTZXVgFxdcUYb1_eg
  runner-token: ""
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: gitlab-runner
  labels:
    app: gitlab-runner
data:
  entrypoint: |
    #!/bin/bash
    set -e
    mkdir -p /home/gitlab-runner/.gitlab-runner/
    cp /scripts/config.toml /home/gitlab-runner/.gitlab-runner/
    # Register the runner
    if [[ -f /secrets/accesskey && -f /secrets/secretkey ]]; then
      export CACHE_S3_ACCESS_KEY=$(cat /secrets/accesskey)
      export CACHE_S3_SECRET_KEY=$(cat /secrets/secretkey)
    fi
    if [[ -f /secrets/gcs-applicaton-credentials-file ]]; then
      export GOOGLE_APPLICATION_CREDENTIALS="/secrets/gcs-applicaton-credentials-file"
    elif [[ -f /secrets/gcs-application-credentials-file ]]; then
      export GOOGLE_APPLICATION_CREDENTIALS="/secrets/gcs-application-credentials-file"
    else
      if [[ -f /secrets/gcs-access-id && -f /secrets/gcs-private-key ]]; then
        export CACHE_GCS_ACCESS_ID=$(cat /secrets/gcs-access-id)
        # echo -e used to make private key multiline (in google json auth key private key is oneline with \n)
        export CACHE_GCS_PRIVATE_KEY=$(echo -e $(cat /secrets/gcs-private-key))
      fi
    fi
                                                                                    1,1          Наверху
```

kubectl get ns

```
igor@ubuntu-server: ~/cicd - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
igor@ubuntu-server:~/cicd$ kubectl get ns
NAME              STATUS   AGE
default           Active   15d
gitlab            Active   20h
ingress-nginx     Active   7d21h
kube-node-lease   Active   15d
kube-public       Active   15d
kube-system       Active   15d
prod              Active   20h
redmine           Active   8d
stage             Active   20h
igor@ubuntu-server:~/cicd$
```

kubectl create sa deploy --namespace stage

```
igor@ubuntu-server: ~/cicd - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
igor@ubuntu-server:~/cicd$ kubectl create sa deploy --namespace stage
serviceaccount/deploy created
igor@ubuntu-server:~/cicd$
```

kubectl create rolebinding deploy --serviceaccount stage:deploy --clusterrole edit --namespace stage

```
igor@ubuntu-server: ~/cicd - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
igor@ubuntu-server:~/cicd$ kubectl create rolebinding deploy --serviceaccount stage:deploy --clusterrole edit --namespace stage
rolebinding.rbac.authorization.k8s.io/deploy created
igor@ubuntu-server:~/cicd$
```

kubectl create sa deploy --namespace prod

```
igor@ubuntu-server: ~/cicd - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
igor@ubuntu-server:~/cicd$ kubectl create sa deploy --namespace prod
serviceaccount/deploy created
igor@ubuntu-server:~/cicd$
```

kubectl create rolebinding deploy --serviceaccount prod:deploy --clusterrole edit --namespace prod

```
igor@ubuntu-server: ~/cicd - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
serviceaccount/deploy created
igor@ubuntu-server:~/cicd$ kubectl create rolebinding deploy --serviceaccount prod:deploy --clusterrole edit --namespace prod
rolebinding.rbac.authorization.k8s.io/deploy created
igor@ubuntu-server:~/cicd$
```

export NAMESPACE=stage
kubectl get secret $(kubectl get sa deploy --namespace $NAMESPACE -o jsonpath='{.secrets[0].name}') --namespace $NAMESPACE -o jsonpath='{.data.token}'

```
igor@ubuntu-server: ~/cicd - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
igor@ubuntu-server:~/cicd$ export NAMESPACE=stage
igor@ubuntu-server:~/cicd$ kubectl get secret $(kubectl get sa deploy --namespace $NAMESPACE -o jsonpath='{.secrets[0].name}') --namespace $NAMES
PACE -o jsonpath='{.data.token}'
ZXlKaGJHY2lPaUpTVXpJMU5pSXNJbXRwWkNJNklsSkxTMFI2YWpOWVh6UkVkalJqT0hReGVTMVhkRzUxU21oQmRtMXpTbWRCTVc1UVVWRlhZMngzYkVFaWZRLmV5SnBjM01pT2lKcmRXSmxjb
TVsZEdWekwzTmxjblpwWTJWaFkyTnZkVzUwSWl3aWEzVmlaWEp1WlhSbGN5NXBieTl6WlhKMmFXTmxZV05qYjNWdWRDOXVZVzFsYzNCaFkyVWlPaUp6ZEdGblpTSXNJbXQxWW1WeWJtVjBaWE
11YVc4dmMyVnlkbWxqWldGalkyOTFiblF2YzJWamNtVjBMbTVoYldVaU9pSmtaWEJzYjNrdGRHOXJaVzR0Ympkak1uY2lMQ0pyZFdKbGNtNWxkR1Z6TG1sdkwzTmxjblpwWTJWaFkyTnZkVzU
wTDNObGNuWnBZMlV0WVdOamIzVnVkQzV1WVcxbElqb2laR1Z3Ykc5NUlpd2lhM1ZpWlhKdVpYUmxjeTVwYnk5elpYSjJhV05sWVdOamIzVnVkQzl6WlhKMmFXTmxMV0ZqWTI5MWJuUXVkV2xr
SWpvaU9UVTVNMlprT0RVdE1ERXdZaTAwTjJRM0xUazBOekF0WVRnMlpqQmhNR1ZpTldKbElpd2ljM1ZpSWpvaWMzbHpkR1Z0T25ObGNuWnBZMlZoWTJOdmRXNTBPbk4wWVdkbE9tUmxjR3h2Z
VNKOS5XTE9RY1FVZFNKeXZPLUdVTHEwQVc2R1o5UmE1SWh6ZHpMaDJOZFBVNFZHZkJKQzBBbTMyem5HMzZFblRGR0dsX0k0UHAzY01NMmlPcDZRaEU5eG5sNDIwT1hzaWhqWW5lOEJnMi1Xbj
JfTGp3VlFyODNPajZSdlowazZPbG9Rc0I4ZWxmdkVNejBudlBVVk04bG5vMnlnMDUxS1E2SXZ4SzRNMmZUQ2FhOEhBLVRwTTVTVDB2aEU0VFlod1ctekxWWFdtX0VIU1hZZkYxTVFkSWZ1N3h
aajY1bVVodHdXWHl6cXlXbmk4RUcxOTBKSi1ueXg1dXVaWWdNLVNjNllkTVlxZnl0ekFheW1DelJmZXdsNGNCaEE5RkhOZ3lwVjBaWklCUTBrNjdMVmhOVHFpWWVtclpMZXh6SXNaZWU5aHNF
igor@ubuntu-server:~/cicd$
```

export NAMESPACE=prod; kubectl get secret $(kubectl get sa deploy --namespace $NAMESPACE -o jsonpath='{.secrets[0].name}') --namespace $NAMESPACE -o jsonpath='{.data.token}'

```
igor@ubuntu-server: ~/cicd - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
aajY1bVVodHdXWHl6cXlXbmk4RUcxOTBKSi1ueXg1dXVaWWdNLVNjNllkTVlxZnl0ekFheW1DelJmZXdsNGNCaEE5RkhOZ3lwVjBaWklCUTBrNjdMVmhOVHFpWWVtclpMZXh6SXNaZWU5aHNF
igor@ubuntu-server:~/cicd$ export NAMESPACE=prod; kubectl get secret $(kubectl get sa deploy --namespace $NAMESPACE -o jsonpath='{.secrets[0].nam
e}') --namespace $NAMESPACE -o jsonpath='{.data.token}'
ZXlKaGJHY2lPaUpTVXpJMU5pSXNJbXRwWkNJNklsSkxTMFI2YWpOWVh6UkVkalJqT0hReGVTMVhkRzUxU21oQmRtMXpTbWRCTVc1UVVWRlhZMngzYkVFaWZRLmV5SnBjM01pT2lKcmRXSmxjb
TVsZEdWekwzTmxjblpwWTJWaFkyTnZkVzUwSWl3aWEzVmlaWEp1WlhSbGN5NXBieTl6WlhKMmFXTmxZV05qYjNWdWRDOXVZVzFsYzNCaFkyVWlPaUp3Y205a0lpd2lhM1ZpWlhKdVpYUmxjeT
VwYnk5elpYSjJhV05sWVdOamIzVnVkQzl6WldOeVpYUXVibUZ0WlNJNklsUmxjR3h2ZVMxMGIydGxiaTFyY0dKMlp5SXNJbXQxWW1WeWJtVjBaWE11YVc4dmMyVnlkbWxqWldGalkyOTFiblF
2YzJWeWRtbGpaUzFoWTJOdmRXNTBMbTVoYldVaU9pSmtaWEJzYjNraUxDSnJkV0psY201bGRHVnpMbWx2TDNObGNuWnBZMlZoWTJOdmRXNTBMM05sY25acFkyVXRZV05qYjNWdWRDNTFhV1Fp
T2lJNE1EZzBaak00TkMxaU16VTBMVFEzWkdVdFlUUXdZaTA0TlRNellqUTJNRE0wTVdJaUxDSnpkV0lpT2lKemVYTjBaVzA2YzJWeWRtbGpaV0ZqWTI5MWJuUTZjSEp2WkRwa1pYQnNiM2tpZ
lEuVnJwLXpITV93REVaLUVnV21WVk1KZll4enBNTWppUjR3U3haTkxQMElFUzVXMXoxX3JhZUhFV1NJOUFkdkRjcFRUOXQxbjBZdFVqbE5yOGtfSUdUTkg2WEMxNWFMR3ZHaGhfdFZsbzhSYU
xUTUpwMEZVUk01U2h1S0RaVTlCYVdOWUlPY01NaXBVdjFDTlRmTUJYeEFYLWRXdktfVHZaZGs4RXZMbEJhR09rWjdjZndVS3FCV3hhVW80MUtXb2pRcHdXSHQ3YTFvTE83TFRIaUo0S3h0YWJ
tenNUdVRIOGRYZnYxcUFNRGpaa01YRDRINjVtMlUyQmdRMEo4eW1rVF81QmlJZ3Jhb3BkTTZhby1pVnM1dUNiaWJhZTMtTEJpc0hWQklkalU3MTFNYm52bkZhS2tkdHhyOVNTWWhYSHQ1Rm42
S0JobEhGek9lb1FLZmlJTWFnigor@ubuntu-server:~/cicd$
```

geekbrains stage
iRDJ_ezVXDN281fPVUvU
geekbrains prod
3KUGPiDWRQdsiEbck34u

kubectl create secret docker-registry gitlab-registry --docker-server=registry.gitlab.com --docker-username=geekbrains --docker-password=iRDJ_ezVXDN281fPVUvU --docker-email=admin@admin.admin --namespace stage

```
igor@ubuntu-server: ~/cicd - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
igor@ubuntu-server:~/cicd$ kubectl create secret docker-registry gitlab-registry --docker-server=registry.gitlab.com --docker-username=geekbrains --docker-password
=iRDJ_ezVXDN281fPVUvU --docker-email=admin@admin.admin --namespace stage
secret/gitlab-registry created
igor@ubuntu-server:~/cicd$
```

kubectl create secret docker-registry gitlab-registry --docker-server=registry.gitlab.com --docker-username=geekbrains --docker-password=3KUGPiDWRQdsiEbck34u --docker-email=admin@admin.admin --namespace prod

```
igor@ubuntu-server: ~/cicd - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
igor@ubuntu-server:~/cicd$ kubectl create secret docker-registry gitlab-registry --docker-server=registry.gitlab.com --docker-username=geekbrains --docker-password
=3KUGPiDWRQdsiEbck34u --docker-email=admin@admin.admin --namespace prod
secret/gitlab-registry created
igor@ubuntu-server:~/cicd$
```

kubectl patch serviceaccount default -p '{"imagePullSecrets": [{"name": "gitlab-registry"}]}' -n stage

```
igor@ubuntu-server: ~/cicd - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
igor@ubuntu-server:~/cicd$ kubectl patch serviceaccount default -p '{"imagePullSecrets": [{"name": "gitlab-registry"}]}' -n stage
serviceaccount/default patched
igor@ubuntu-server:~/cicd$
```

kubectl patch serviceaccount default -p '{"imagePullSecrets": [{"name": "gitlab-registry"}]}' -n prod

```
igor@ubuntu-server: ~/cicd - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
igor@ubuntu-server:~/cicd$ kubectl patch serviceaccount default -p '{"imagePullSecrets": [{"name": "gitlab-registry"}]}' -n prod
serviceaccount/default patched
igor@ubuntu-server:~/cicd$
```

cd app

```
igor@ubuntu-server: ~/app - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
igor@ubuntu-server:~$ cd app
igor@ubuntu-server:~/app$ ls
igor@ubuntu-server:~/app$
```

vi Dockerfile

```
igor@ubuntu-server: ~/app - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
FROM golang:1.14 as builder

RUN mkdir /app

COPY . /app

WORKDIR /app
RUN CGO_ENABLED=0 GOOS=linux GOARCH=amd64 go build -o server .


FROM scratch

COPY --from=builder /app/server /

EXPOSE 8000
CMD ["/server"]
~
:wq
```

igor@ubuntu-server: ~/app/app - Терминал

Файл Правка Вид Терминал Вкладки Справка

```
	}

	a.DB = model.DBMigrate(db)
	a.Router = mux.NewRouter()
	a.setRouters()
}

// Set all required routers
func (a *App) setRouters() {
	a.Get("/users", a.GetAllUsers)
	a.Post("/users", a.CreateUser)
	a.Get("/users/{title}", a.GetUser)
	a.Put("/users/{title}", a.UpdateUser)
	a.Delete("/users/{title}", a.DeleteUser)
	a.Put("/users/{title}/disable", a.DisableUser)
	a.Put("/users/{title}/enable", a.EnableUser)
}

// Wrap the router for GET method
func (a *App) Get(path string, f func(w http.ResponseWriter, r *http.Request)) {
	a.Router.HandleFunc(path, f).Methods("GET")
}

// Wrap the router for POST method
func (a *App) Post(path string, f func(w http.ResponseWriter, r *http.Request)) {
	a.Router.HandleFunc(path, f).Methods("POST")
}

// Wrap the router for PUT method
func (a *App) Put(path string, f func(w http.ResponseWriter, r *http.Request)) {
	a.Router.HandleFunc(path, f).Methods("PUT")
}

// Wrap the router for DELETE method
func (a *App) Delete(path string, f func(w http.ResponseWriter, r *http.Request)) {
	a.Router.HandleFunc(path, f).Methods("DELETE")
}

// Handlers to manage User Data
func (a *App) GetAllUsers(w http.ResponseWriter, r *http.Request) {
	handler.GetAllUsers(a.DB, w, r)
}

func (a *App) CreateUser(w http.ResponseWriter, r *http.Request) {
	handler.CreateUser(a.DB, w, r)
}

func (a *App) GetUser(w http.ResponseWriter, r *http.Request) {
	handler.GetUser(a.DB, w, r)
}

func (a *App) UpdateUser(w http.ResponseWriter, r *http.Request) {
	handler.UpdateUser(a.DB, w, r)
}

func (a *App) DeleteUser(w http.ResponseWriter, r *http.Request) {
	handler.DeleteUser(a.DB, w, r)
}

func (a *App) DisableUser(w http.ResponseWriter, r *http.Request) {
	handler.DisableUser(a.DB, w, r)
}

func (a *App) EnableUser(w http.ResponseWriter, r *http.Request) {
	handler.EnableUser(a.DB, w, r)
}

// Run the app on it's router
func (a *App) Run(host string) {
	log.Fatal(http.ListenAndServe(host, a.Router))
}
~
~
:wq
```

vi config.go

igor@ubuntu-server: ~/app/config - Терминал

Файл Правка Вид Терминал Вкладки Справка

```go
package config

import "os"

type Config struct {
	DB *DBConfig
}

type DBConfig struct {
	Host     string
	Port     string
	Username string
	Password string
	Name     string
}

func GetConfig() *Config {
	return &Config{
		DB: &DBConfig{
			Host:     os.Getenv("DB_HOST"),
			Port:     os.Getenv("DB_PORT"),
			Username: os.Getenv("DB_USER"),
			Password: os.Getenv("DB_PASSWORD"),
			Name:     os.Getenv("DB_NAME"),
		},
	}
}
:q!
```

vi go.mod

igor@ubuntu-server: ~/app - Терминал

Файл Правка Вид Терминал Вкладки Справка

```go
module github.com/pauljamm/geekbrains-conteinerization/practice/8.ci-cd/app

go 1.14

require (
	github.com/gorilla/mux v1.7.4
	github.com/jinzhu/gorm v1.9.15
)
:wq
```

vi go.sum

```
igor@ubuntu-server: ~/app - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
github.com/PuerkitoBio/goquery v1.5.1/go.mod h1:GsLWisAFVj4WgDibEWF4pvYnkVQBpKBKeU+7zCJoLcc=
github.com/andybalholm/cascadia v1.1.0/go.mod h1:GsXiBklL0woXo1j/WYWtSYYC4ouU9PqHO0sqidkEA4Y=
github.com/denisenkom/go-mssqldb v0.0.0-20191124224453-732737034ffd/go.mod h1:xbL0rPBG9cCiLr28tMa8zpbdarY27NDyej4t/EjAShU=
github.com/erikstmartin/go-testdb v0.0.0-20160219214506-8d10e4a1bae5/go.mod h1:a2zkGnVExMxdzMo3M0Hi/3sEU+cWnZpSni0O6/Yb/P0=
github.com/go-sql-driver/mysql v1.5.0 h1:ozyZYNQW3x3HtqT1jira07DN2PArx2v7/mN66gGcHOs=
github.com/go-sql-driver/mysql v1.5.0/go.mod h1:DCzpHaOWr8IXmIStZouvnhqoel9Qv2LBy8hT2VhHyBg=
github.com/golang-sql/civil v0.0.0-20190719163853-cb61b32ac6fe/go.mod h1:8vg3r2VgvsThLBIFL93Qb5yWzgyZWhEmBwUJWevAkK0=
github.com/gorilla/mux v1.7.4 h1:VuZ8uybHlWmqV03+zRzdwKL4tUnIp1MAQtp1mIFE1bc=
github.com/gorilla/mux v1.7.4/go.mod h1:DVbg23sWSpFRCP0SfiEN6jmj59UnW/n46BH5rLB71So=
github.com/jinzhu/gorm v1.9.15 h1:OdR1qFvtXktlxk73XFYMiYn9ywzTwytqe4QkuMRqc38=
github.com/jinzhu/gorm v1.9.15/go.mod h1:G3LB3wezTOWM2ITLzPxEXgSkOXAntiLHS7UdBefADcs=
github.com/jinzhu/inflection v1.0.0 h1:K317FqzuhWc8YvSVlFMCCUb36O/S9MCKRDI7QkRKD/E=
github.com/jinzhu/inflection v1.0.0/go.mod h1:h+uFLlag+Qp1Va5pdKtLDYj+kHp5pxUVkryuEj+Srlc=
github.com/jinzhu/now v1.0.1/go.mod h1:d3SSVoowX0Lcu0IBviAWJpolVfI5UJVZZ7cO71lE/z8=
github.com/lib/pq v1.1.1 h1:sJZmqHoEaY7f+NPP8pgLB/WxulyR3fewgCM2qaSlBb4=
github.com/lib/pq v1.1.1/go.mod h1:5WUZQaWbwv1U+lTReE5YruASi9Al49XbQIvNi/34Woo=
github.com/mattn/go-sqlite3 v1.14.0/go.mod h1:JIl7NbARA7phWnGvh0LKTyg7S9BA+6gx71ShQilpsus=
github.com/pauljamm/geekbrains-conteinerization v0.0.0-20200718135033-222e8d0ee23a h1:JlT+jb1bJMZAU0MRcc4XMYUyVz8eo6qwhCET5gdl+r0=
golang.org/x/crypto v0.0.0-20190308221718-c2843e01d9a2/go.mod h1:djNgcEr1/C05ACkg1iLfiJU5Ep61QUkGW8qpdssI0+w=
golang.org/x/crypto v0.0.0-20190325154230-a5d413f7728c/go.mod h1:djNgcEr1/C05ACkg1iLfiJU5Ep61QUkGW8qpdssI0+w=
golang.org/x/crypto v0.0.0-20191205180655-e7c4368fe9dd/go.mod h1:LzIPMQfyMNhhGPhUkYOs5KpL4U8rLKemX1yGLhDgUto=
golang.org/x/net v0.0.0-20180218175443-cbe0f9307d01/go.mod h1:mL1N/T3taQHkDXs73rZJwtUhF3w3ftmwwsq0BUmARs4=
golang.org/x/net v0.0.0-20190404232315-eb5bcb51f2a3/go.mod h1:t9HGtf8HONx5eT2rtn7q6eTqICYqUVnKs3thJo3Qplg=
golang.org/x/net v0.0.0-20200202094626-16171245cfb2/go.mod h1:z5CRVTTTmAJ677TzLLGU+0bjPO0LkuOLi4/5GtJWs/s=
golang.org/x/net v0.0.0-20200324143707-d3edc9973b7e/go.mod h1:qpuaurCH72eLCgpAm/N6yyVIVM9cpaDIP3A8BGJEC5A=
golang.org/x/sys v0.0.0-20190215142949-d0b11bdaac8a/go.mod h1:STP8DvDyc/dI5b8T5hshtkjS+E42TnysNCUPdjciGhY=
golang.org/x/sys v0.0.0-20190412213103-97732733099d/go.mod h1:h1NjWce9XRLGQEsW7wpKNCjG9DtNlClVuFLEZdDNbEs=
golang.org/x/sys v0.0.0-20200323222414-85ca7c5b95cd/go.mod h1:h1NjWce9XRLGQEsW7wpKNCjG9DtNlClVuFLEZdDNbEs=
golang.org/x/text v0.3.0/go.mod h1:NqM8EUOU14njkJ3fqMW+pc6Ldnwhi/IjpwHt7yyuwOQ=
~
:wq
```

vi common.go

igor@ubuntu-server: ~/app/handler - Терминал

Файл Правка Вид Терминал Вкладки Справка

```
package handler

import (
        "encoding/json"
        "net/http"
)

// respondJSON makes the response with payload as json format
func respondJSON(w http.ResponseWriter, status int, payload interface{}) {
        response, err := json.Marshal(payload)
        if err != nil {
                w.WriteHeader(http.StatusInternalServerError)
                w.Write([]byte(err.Error()))
                return
        }
        w.Header().Set("Content-Type", "application/json")
        w.WriteHeader(status)
        w.Write([]byte(response))
}

// respondError makes the error response with payload as json format
func respondError(w http.ResponseWriter, code int, message string) {
        respondJSON(w, code, map[string]string{"error": message})
}

~
:wq
```

vi users.go

igor@ubuntu-server: ~/app/handler - Терминал

Файл Правка Вид Терминал Вкладки Справка

```
                return
        }

        decoder := json.NewDecoder(r.Body)
        if err := decoder.Decode(&user); err != nil {
                respondError(w, http.StatusBadRequest, err.Error())
                return
        }
        defer r.Body.Close()

        if err := db.Save(&user).Error; err != nil {
                respondError(w, http.StatusInternalServerError, err.Error())
                return
        }
        respondJSON(w, http.StatusOK, user)
}

func DeleteUser(db *gorm.DB, w http.ResponseWriter, r *http.Request) {
        vars := mux.Vars(r)

        name := vars["name"]
        user := getUserOr404(db, name, w, r)
        if user == nil {
                return
        }
        if err := db.Delete(&user).Error; err != nil {
                respondError(w, http.StatusInternalServerError, err.Error())
                return
        }
        respondJSON(w, http.StatusNoContent, nil)
}

func DisableUser(db *gorm.DB, w http.ResponseWriter, r *http.Request) {
        vars := mux.Vars(r)

        name := vars["name"]
        user := getUserOr404(db, name, w, r)
        if user == nil {
                return
        }
        user.Disable()
        if err := db.Save(&user).Error; err != nil {
                respondError(w, http.StatusInternalServerError, err.Error())
                return
        }
        respondJSON(w, http.StatusOK, user)
}

func EnableUser(db *gorm.DB, w http.ResponseWriter, r *http.Request) {
        vars := mux.Vars(r)

        name := vars["name"]
        user := getUserOr404(db, name, w, r)
        if user == nil {
                return
        }
        user.Enable()
        if err := db.Save(&user).Error; err != nil {
                respondError(w, http.StatusInternalServerError, err.Error())
                return
        }
        respondJSON(w, http.StatusOK, user)
}

// getUserOr404 gets a user instance if exists, or respond the 404 error otherwise
func getUserOr404(db *gorm.DB, name string, w http.ResponseWriter, r *http.Request) *model.User {
        user := model.User{}
        if err := db.First(&user, model.User{Name: name}).Error; err != nil {
                respondError(w, http.StatusNotFound, err.Error())
                return nil
        }
        return &user
}
:wq
```

vi deployment.yaml

igor@ubuntu-server: ~/app/kube - Терминал

Файл Правка Вид Терминал Вкладки Справка

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: geekbrains
spec:
  progressDeadlineSeconds: 300
  replicas: 2
  selector:
    matchLabels:
      app: app
  template:
    metadata:
      labels:
        app: app
    spec:
      containers:
        - name: app
          image: nginx:1.12 # это просто плэйсхолдер
          env:
            - name: DB_HOST
              value: database
            - name: DB_PORT
              value: "5432"
            - name: DB_USER
              value: app
            - name: DB_PASSWORD
              valueFrom:
                secretKeyRef:
                  key: db-password
                  name: app
            - name: DB_NAME
              value: users
          resources:
            limits:
              memory: "128Mi"
              cpu: "100m"
          ports:
            - containerPort: 8000
~
~
:wq
```

vi ingress.yaml

igor@ubuntu-server: ~/app/kube - Терминал

Файл Правка Вид Терминал Вкладки Справка

```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: geekbrains
spec:
  rules:
    - host: <CHANGE ME>
      http:
        paths:
          - backend:
              serviceName: geekbrains
              servicePort: 8000
            path: /users
:wq
```

vi service.yaml

```
igor@ubuntu-server: ~/app/kube - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
apiVersion: v1
kind: Service
metadata:
  name: geekbrains
spec:
  selector:
    app: app
  ports:
    - port: 8000
      targetPort: 8000
~
~
~
:wq
```

vi secret.yaml

```
igor@ubuntu-server: ~/app/kube/postgres - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
apiVersion: v1
stringData:
  db-password: supersecretpassword
kind: Secret
metadata:
  name: app
type: Opaque
~
:wq
```

vi service.yaml

```
igor@ubuntu-server: ~/app/kube/postgres - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
apiVersion: v1
kind: Service
metadata:
  name: database
spec:
  ports:
  - port: 5432
    targetPort: 5432
  selector:
    app: database
~
-- ВСТАВКА --                                  1,15          Весь
```

vi statefulset.yaml

igor@ubuntu-server: ~/app/kube/postgres - Терминал

Файл Правка Вид Терминал Вкладки Справка

```yaml
---
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: database
spec:
  replicas: 1
  serviceName: database
  selector:
    matchLabels:
      app: database
  template:
    metadata:
      labels:
        app: database
    spec:
      containers:
        - image: postgres:10.13
          name: postgres
          env:
            - name: POSTGRES_USER
              value: app
            - name: POSTGRES_DB
              value: users
            - name: PGDATA
              value: /var/lib/postgresql/data/pgdata
            - name: POSTGRES_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: app
                  key: db-password
          ports:
          - containerPort: 5432
            protocol: TCP
          volumeMounts:
            - name: data
              mountPath: /var/lib/postgresql/data
  volumeClaimTemplates:
    - metadata:
        name: data
      spec:
        accessModes: ["ReadWriteOnce"]
        resources:
          requests:
            storage: 2Gi
        storageClassName: csi-ceph-hdd-dp1
~
:wq
```

vi main.go

```
igor@ubuntu-server: ~/app - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
package main

import (
        "github.com/pauljamm/geekbrains-conteinerization/practice/8.ci-cd/app/app"
        "github.com/pauljamm/geekbrains-conteinerization/practice/8.ci-cd/app/config"
)

func main() {
        config := config.GetConfig()

        app := &app.App{}
        app.Initialize(config)
        app.Run(":8000")
}
:wq
```

vi model.go

```
igor@ubuntu-server: ~/app/model - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
package model

import (
        "github.com/jinzhu/gorm"
        _ "github.com/jinzhu/gorm/dialects/postgres"
)

type User struct {
        gorm.Model
        Name   string `gorm:"unique" json:"name"`
        City   string `json:"city"`
        Age    int    `json:"age"`
        Status bool   `json:"status"`
}

func (e *User) Disable() {
        e.Status = false
}

func (p *User) Enable() {
        p.Status = true
}

// DBMigrate will create and migrate the tables, and then make the some relationships if necessary
func DBMigrate(db *gorm.DB) *gorm.DB {
        db.AutoMigrate(&User{})
        return db
}
~
~
:wq
```

kubectl apply --namespace stage -f kube/postgres/

```
igor@ubuntu-server: ~/app - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
igor@ubuntu-server:~/app$ kubectl apply --namespace stage -f kube/postgres/
secret/app created
service/database created
statefulset.apps/database created
igor@ubuntu-server:~/app$
```

kubectl apply --namespace prod -f kube/postgres/

```
igor@ubuntu-server: ~/app - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
igor@ubuntu-server:~/app$ kubectl apply --namespace prod -f kube/postgres/
secret/app created
service/database created
statefulset.apps/database created
igor@ubuntu-server:~/app$
```

vi kube/ingress.yaml

```
igor@ubuntu-server: ~/app - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: geekbrains
annotations:
        kubernetes.io/ingress.class: nginx-external
spec:
   host: postgres.stage.info
   http:
     paths:
       - path: /users
         pathType: Prefix
         backend:
           service:
             name: geekbrains
             port:
               number: 8000
~
~
:wq
```

vi **.gitlab-ci.yml**

```
igor@ubuntu-server: ~/app - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
variables:
  K8S_API_URL: https://kubernetes.default

stages:
  - test
  - build
  - deploy

test:
  stage: test
  image: golang:1.14
  script:
    - echo OK

build:
  stage: build
  image: docker:19.03.12
  services:
    - docker:19.03.12-dind
  variables:
    DOCKER_DRIVER: overlay
    DOCKER_HOST: tcp://docker:2375
    DOCKER_TLS_CERTDIR: ""
  before_script:
    - docker login -u $CI_REGISTRY_USER -p $CI_REGISTRY_PASSWORD $CI_REGISTRY
  script:
    - docker build . -t $CI_REGISTRY_IMAGE:$CI_COMMIT_REF_SLUG.$CI_PIPELINE_ID
    - docker push $CI_REGISTRY_IMAGE:$CI_COMMIT_REF_SLUG.$CI_PIPELINE_ID

.deploy: &deploy
  stage: deploy
  image: bitnami/kubectl:1.16
  before_script:
    - export KUBECONFIG=/tmp/.kubeconfig
    - kubectl config set-cluster k8s --insecure-skip-tls-verify=true --server=$K8S_API_URL
    - kubectl config set-credentials ci --token=$(echo $K8S_CI_TOKEN | base64 --decode)
    - kubectl config set-context ci --cluster=k8s --user=ci
    - kubectl config use-context ci
  script:
    - kubectl set image deployment/$CI_PROJECT_NAME *=$CI_REGISTRY_IMAGE:$CI_COMMIT_REF_SLUG.$CI_PIPELINE_ID --namespace
 $CI_ENVIRONMENT_NAME
    - kubectl rollout status deployment/$CI_PROJECT_NAME --namespace $CI_ENVIRONMENT_NAME || (kubectl rollout undo deplo
yment/$CI_PROJECT_NAME --namespace $CI_ENVIRONMENT_NAME && exit 1)

deploy:stage:
  <<: *deploy
  environment:
    name: stage
  variables:
    K8S_CI_TOKEN: $K8S_STAGE_CI_TOKEN
  only:
    - master

deploy:prod:
  <<: *deploy
  environment:
    name: prod
  variables:
    K8S_CI_TOKEN: $K8S_PROD_CI_TOKEN
  only:
    - master
  when: manual
~
:wq
```

vi .dockerignore

```
igor@ubuntu-server: ~/app - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
.gitlab-ci.yml
.git/
kube/
~
~
~
:wq
```

kubectl apply --namespace stage -f kube/

error: error validating "kube/ingress.yaml": error validating data: ValidationError(Ingress): unknown field "annotations" in io.k8s.api.networking.v1.Ingress; if you choose to ignore these errors, turn validation off with —validate=false